REFERENCE MANUAL FOR

MOSES

Phone (713) 975–8146 Fax (713) 975–8179

Contents

I. INTRODUCTION

Our primary objective with MOSES is to provide engineers with the tools necessary to realistically design and analyze marine structures and operations. The increasing sophistication of the offshore industry, coupled with the rapid evolution of the computational power available, has made it obvious that to achieve this goal, a major departure from the traditional approaches would be necessary.

In the past, a problem was analyzed in several distinct parts – each requiring a different view of reality and subsequent model. While this approach is suited to existing organizational structure, it is highly inefficient and error prone. Different models are *ipso facto* different. A substantial quality assurance effort has been required to reconcile these differences, and more importantly, one could not hope to obtain a proper analysis of the complete picture by simply viewing selected parts of it. Obviously, what was really necessary was something that integrated all aspects of the problem.

To bridge this gap, we have created MOSES, a new language for modeling, simulating, and analyzing the stresses which arise in marine situations. This new language offers the necessary flexibility along with the rigor of a programming language. Now, one can easily create new models, document them, and assess their validity – all with a single program.

In addition to specialized capabilities, the MOSES language is rich in general utilities to make one's life easier. Most results of a MOSES simulation are available for interactive reporting, graphing, viewing in three dimensions, and statistical interpretation. *Instead* of manually repeating blocks of data, MOSES provides for loops. *Instead* of having different sets of data for slightly different situations, MOSES provides for conditional execution. *Instead* of having the same data defined in different places, MOSES allows one to define variables and use them later. *Instead* of repeating commands with minor alterations, MOSES allows the user to create his own commands called macros.

The MOSES language is built upon a proprietary database manager specifically designed for its purpose – the storage and retrieval of scientific models and the results of their simulations. By storing all data in a database, MOSES is totally restartable. One can perform some tasks interactively, stop, then seamlessly restart the program to perform other tasks in the background. The database even allows different *types* of simulation with the same model and a stress analysis to be performed for all types concurrently.

Before MOSES, most marine problems were considered in two steps: a simulation followed by a stress analysis. Two different programs were required. Since MOSES performs both of these analyses, one needs only a single program to investigate all aspects of the problem. Also, with MOSES, one is spared the agony of transferring files and of learning the idiosyncrasies of several programs.

Since it must cope with the demands of both simulation and stress analysis, the MOSES modeling language is richer than the norm. From a stress analysis point of view, a MOSES model consists of a set of beams, generalized plates, and connectors. Here, however, these structural elements can also model load generating attributes. To allow for other types of loads, one can define areas and masses, along with constructs called "hulls". This gives MOSES the ability to compute hydrodynamic forces on a system via three hydrodynamic theories: Morison's Equation, Two Dimensional Diffraction theory, or Three Dimensional Diffraction theory.

With MOSES, connectors are not simply "restraints", but the way one connects different bodies. One can select from catenary mooring lines, tension–only and compression–only nonlinear springs, rigid connectors such as pins and launchways, and even true nonlinear rod elements. These connectors are automatically applied during a stress analysis so that one can *correctly* perform a stress analysis of several connected bodies.

The MOSES modeling language is rich enough so that models suitable for other programs can be converted to MOSES models with minimal effort. In fact, interfaces are available for several programs, and others can be quickly developed.

Not being content with simply analyzing a given situation, MOSES provides a menu which aids in the design of mooring lines and lifting slings. Commands are also available which will automatically alter connectors so that different scenarios can be assessed with minimal effort. With a rod connector, the effect of the inertia and damping of the connections may be assessed.

As with connectors, MOSES allows for the basic computations traditionally performed by a naval architect. One can compute the curves of form, the intact or damaged stability, and the longitudinal strength of a vessel. MOSES, however, does not stop here. One can specify interactively, the ballast in any or all of the vessel's tanks and immediately find the resulting condition. If one wishes, he can ask MOSES to compute a ballast plan which will achieve a given condition and then alter it. Finally, if desired, one can ask MOSES to perform a detailed stress analysis of the condition. The program will take care of all of the details of computing the correct inertia, loads, and restraints.

Once a suitable condition has been found, a traditional seakeeping study can be performed with MOSES by issuing a single command. MOSES will then use the hydrodynamic theory selected from the three available to compute the response operators of both the motions of each body and the connector forces. An entire menu of commands is available to post–process these response operators. One can easily find the statistical results for specified sea conditions and create time domain samples of the results to assess phasing. All results can be graphed or reported. Only four additional commands are necessary to produce a detailed stress analysis of the system in the frequency domain.

At any point, one may perform a time domain simulation of the current system. This is accomplished by issuing a command to define the environment, and a second to initiate the time domain simulation. MOSES then takes the hydrodynamic forces computed via the proper hydrodynamic theory, combines them with the other forces which act on the system, and integrates the nonlinear equations of motion in the time domain. At the conclusion, again a menu of post–processing commands are available to assist the analyst in deciphering the results – trajectories of points, forces on elements, connector forces, etc. As before, a stress analysis at events during the simulation requires only a few additional commands.

To simulate the process of lifting a structure off of a barge, lowering it into the water, and bringing it upright, MOSES offers a menu of alternatives. One can interactively ballast compartments and move the hook up or down to assess the results of any field action. These results are stored by event so that they can be reviewed and the action changed, until the desired outcome is attained. As with other simulations, at the conclusion, the results can be post–processed and used for a stress analysis.

A specialized type of time domain simulation is a jacket launch. Here, a single body is moved until it comes free of other bodies upon which it was towed to location. Traditionally, a jacket was launched from a single barge. In anticipation of such an operation, MOSES can simulate a launch from several barges which may be connected.

By combining a nonlinear rod element with other connectors, one can simulate the laying of pipe either from a stinger or from davits. With MOSES, all aspects of the problem can be modeled. The lay vessel and the stinger can be modeled as separate bodies connected via the pipe, hinges, tensioners, and rollers. Once the system is assembled, one can perform static, time, or frequency domain simulations of the laying process.

MOSES can perform a detailed stress analysis for events during a time domain simulation, a static process, or a frequency domain process. There are no essential limits on either the model size, the number of bodies which can be analyzed, or the number of load cases. The solution algorithms are state of the art and the structural post–processing is superior. MOSES can consider not only linear but also spectral combinations of the basic load cases. Thus, if one performs a stress analysis in the frequency domain, he can then consider member and joint checks spectrally. In addition, spectral fatigue can be considered in beams, generalized plates, and tubular joints.

II. ANALYSIS OVERVIEW

MOSES is a simulation language. Thus, the commands which are available are all designed to either describe a system or to perform a simulation. The primary strength here is that the user is free to issue the commands in any order that makes sense. In other words, once a basic system has been defined, the user can alter it in many different ways to change the initial conditions for similar simulations or perform different types of simulations, without altering the basic definition of the system. Also, after a simulation has been performed, he can analyze the deflections, stresses, etc. at different phases of the simulation.

In general, the things with which MOSES performs simulations are called bodies. During a simulation, bodies have N degrees of freedom. The first six of these are the traditional rigid body degrees of freedom, and any others represent deformation of the body. Bodies are composed of smaller pieces called parts, with each part having all of the characteristics of a body itself. MOSES is capable of considering four types of forces which act on bodies: those which arise from water, wind, inertia, and those which are applied. Thus, to MOSES, a body is a collection of attributes which tell it how to compute loads and how to compute deflections. MOSES can deal with up to 50 bodies.

In computing the forces on a body due to its interaction with the water, the user can choose from three hydrodynamic theories: Morison's Equation, Three Dimensional Diffraction, or Two Dimension Diffraction, the particular method used being controlled by the manner in which the body is modeled. A single body can be composed of any combination of hydrodynamic elements. The structure of a body can be defined by any combination of beam and generalized plate elements, and the user has control over whether or not a given structural element will attract load from either wind, water, or inertia.

A second primitive element of the MOSES system is the connector. These elements, in general, attract no loads from the environment and serve to constrain the motion of the bodies. There are five types of connectors: flexible connectors, rigid constraints, launchways, pipes, and slings. Here, flexible connectors can be used to model mooring lines, hawsers, etc., while rigid constraints are used for pins. The user is free to define any combination of connections. Connections are defined separately from the definition of the bodies, and thus, can be altered interactively to simulate different aspects of a particular situation.

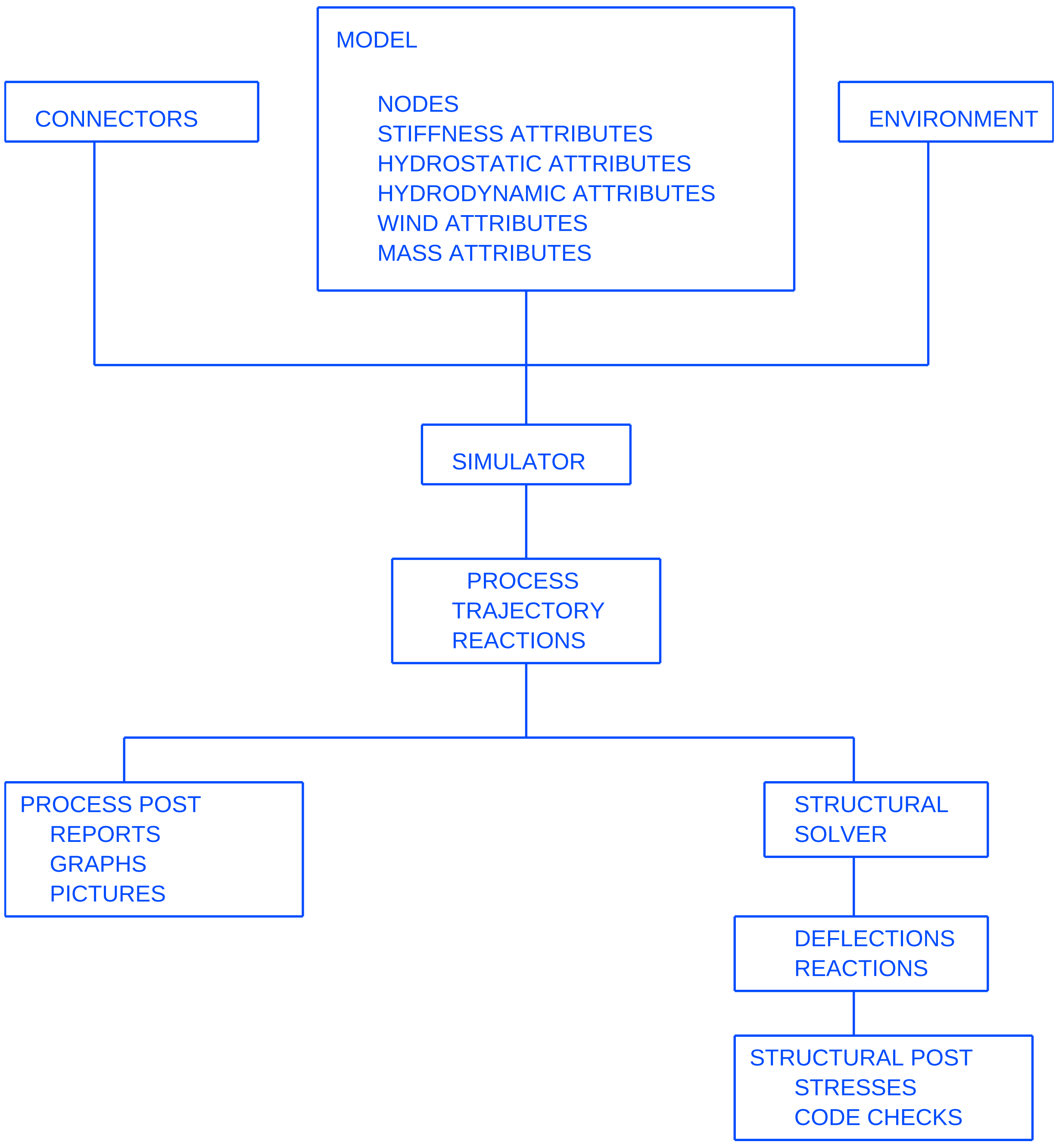
Once a system (bodies and connections) has been defined, the user is free to perform static, frequency domain, or time domain simulations. There are also specialized sets of commands which provide information on the hydrostatics of one of the bodies, the behavior of the mooring system, or the upending of a body. The results of each simulation are stored in a database so that they can be recalled for post–processing, restarting, or for use in a stress analysis.

After a simulation has been performed, the user can perform a stress analysis for selected parts of the system at selected events during the simulation. Here, MOSES will compute all of the loads on the selected part at the event in question and convert these into nodal and member loads for use by the structural solver. If the body has more than six degrees of freedom, then the loads applied include the deformation inertia. The restraints which correspond to the connectors will be added to the structural model. The resulting structural system will be solved for the deflections at the nodes, and the deflections and corresponding element internal loads will be stored in the database.

The post–processing of MOSES is one of its strongest points. Virtually all of the results produced from either a simulation, a mooring command, or a hydrostatic command can be viewed at the terminal, graphed, or written to a hardcopy device. In addition, many results based on the simulations can be computed in the post–processors. In the structural analysis post–processor, code checks, joint checks, deflections, elements loads, and stochastic fatigue can be reported. These reports can be restricted to a small subset at the request of the user. A flow chart of the procedure just outlined is shown in Figure 1.

With the generality provided within MOSES, it is virtually impossible to delimit the tasks which can be accomplished. There are certain things, however, which can be done simply:

- Jacket launch from one or more barges,
- Time or frequency domain simulation of a structure on a system of vessels,
- Time or frequency domain simulation of moored vessels,
- Time or frequency domain simulation of a tension leg platform,
- Docking simulation of a jacket and a pile,
- Upending of a jacket,
- Ballasting and stability of a vessel and cargo,
- Laying of pipe from a lay vessel,
- Lifting a structure from a barge,
- Lowering a structure into the water,
- Loadout of a structure onto a vessel,
- Stress analysis of any of the above, or
- Inplace analysis of a jacket.



ANALYSIS FLOW

FIGURE 1

III. OVERVIEW OF MOSES

Perhaps the easiest way to describe MOSES is that it is not very smart, but it has a good memory. In other words, MOSES must be told to do everything, but remembers almost everything that it has been told. As definition, the things MOSES is told to do are called commands, while the place the results are stored is called the job database.

While all instructions to MOSES are called commands, it helps to separate instructions into the categories: commands, descriptions, and questions. One issues commands to MOSES to accomplish tasks, issues descriptions to define a system for analysis, and asks questions to find out the results of the commands. The database consists of the union of all descriptions issued and the results of all commands. When one asks a question, the answers are obtained by querying the database. While MOSES can be utilized in both ”batch” and ”interactive” environments, it is perhaps best to view all commands as being issued from a terminal. Since the results of all previous commands are available in the database, it is no more difficult to produce a set of results by making several small runs as it is to make one big one.

The majority of the system which will be analyzed is defined to MOSES by a set of descriptions contained in the ”INPUT” file. These descriptions are commands in what is called MOSES Modeling Language, and are processed by the program whenever instructed by the user. When the Modeling Language is processed, the description is converted into an internal model within the job database so that the model has to be processed only when it has been altered.

After a model database has been generated, the user is free to perform simulations. Before proceeding, however, one may wish to alter the definition of the system from that defined by the modeling language. The type of things which can be altered are: the weight of the bodies, the connections among the bodies, or the environment. When the system is altered, the changes are again remembered until the system is altered again. Thus, one can perform numerous simulations on the same basic system without rereading the model. When one issues a simulation command, the simulation is performed, the results stored in the database, and control is returned to the user. No reports are automatically produced, and no questions are asked, so that simulations can easily be performed in the background. To obtain reports of the results one must enter one of the sections of commands which were designed to answer questions about the results of simulations. These sections of commands are called Post–Processing Menus or Disposition Menus. In these sections of the program, one may be asked questions himself, so it is best if these tasks are performed interactively.

The database structure of MOSES allows for ”seamless” restartability. One can terminate the program at almost any point and resume execution later with no loss of information. This structure and the root file concept discussed later free the user from having to worry about naming, reconnecting, and remembering the names of ”restart files”, and provide superior performance to previous systems. Since nothing is necessary to restart the program, nothing further will be said about it, but the capability is one of the primary features of MOSES.

While MOSES is initially not very smart, it can learn. In other words, the user can teach the program how to perform many commands when a single one is issued. This capability is implemented by allowing users to define ”macros”. These are really sets of commands which the program associates with a single name, and any time the name is issued as a command, the entire set will be executed, thus freeing the user from the tedious task of issuing many commands.

The flexibility of MOSES may, at first, overwhelm a new user, but with a little experience one quickly learns to enjoy the power of the system. In the sections which follow, all of the features of MOSES will be discussed. In many cases, the utility of a feature may not be apparent when it is discussed. The primary reason is that there are many facets of the MOSES language which are not really necessary, but are quite useful once one has mastered the basics. Thus, instead of worrying about how each feature is to be used, one should proceed throughout the manual briefly to get a ”general feel” of what one can accomplish and how to do it. The next step is to carefully look over the samples supplied with this installation to see how typical problems may be attacked. The next step in solving a problem is to create a simple problem which has all of the attributes of the real, complex one to be solved and to use it as a prototype in understanding. It is always easier to use small problems to complete ones understanding than it is to cope with both understanding the program and with the details of a large problem.

IV. MOSES BASICS

In order to perform any task with MOSES, one must be able to communicate with the program in a language understandable by both the user and the program. In this section, the rules of grammar and syntax of the language employed and the general operation of MOSES will be discussed.

In discussing the various types of commands, some of the words are "**set off**". By "**set off**", we mean the words are either underlined or printed in bold type, depending on the method used to print this manual. These words are keywords, either commands or options, and *must* be input exactly as written. The characters not "**set off**" represent the data which takes on the appropriate numeric or alphanumeric value. In some cases, an underline is part of an option or command. For example, **END_DISPOSE** is a command. In keeping with the format of this manual, these commands are "**set off**" (and possibly underlined), and the user needs to remember that the underline exists as part of the command.

MOSES provides many features of a programming language. In MOSES, one can alter the flow of either command or description input, make logical checks, define variables, create macros, etc. All of these features operate on both commands and descriptions, so that with this language one can automate the definition of a model as well as build a set of specific commands he needs to perform repetitive tasks.

IV.A The MOSES Interface

MOSES actually has four different user interfaces. By default, it starts in "GUI" mode. The other three are a terminal, a "silent" interface, and the old GUI mode, which we refer to as the window interface.

Silent Interface

With the silent interface, MOSES produces no terminal output except that directed by the internal command **&S_BACK**. This option is quite useful for running MOSES in a "pipe". While the details discussed here differ slightly between the graphical and terminal interfaces, the operation is much the same. MOSES is, and at heart will remain, a language where commands are input and results produced. The graphical interfaces are simply more efficient at their job.

Terminal Interface With the terminal option, MOSES simply runs in the existing terminal (or console) window. It only has a display area and commands are input directly into it. With a terminal interface, the actual commands listed must be used and none of the keyboard shortcuts are operative. Of course pictures or graphics can only be viewed in one of the graphic interfaces. A WINDOWS command prompt will not properly display a MOSES terminal session. To use this type of interface on WINDOWS machine, you should start a "MinGW sh" and use it to run MOSES. You should navigate to c:/ultra/bin/win32/msys (or wherever MOSES is installed). Here you will find a shortcut to msys. You can drag this to your desktop, double click on it, and a window will open. Be warned that this window is a UNIX shell and here you will need to use / instead of \ for path separators.

GUI Interface

This is the standard interface for interactive MOSES sessions. The interface consists of six parts:

The Menu should be familiar to users of the previous versions of MOSES. New to this version is the Help menu which takes you to the new hyperlinked help system available in this release.

The Top Button Bar is where you'll find features that are used often or are only found in this interface (not in the "–text" or –"win" interfaces).

- *Save* lets you save wire frame or GL images as you see them. The current picture is added to your graphics device file.
- *Copy* will copy either text or graphics to the clipboard for use in other programs.
- *Paste* will paste text from the clipboard to the current cursor position.

- *Help* brings up the new hyperlinked, indexed help system.

The Left Button Bar

These controls are specifically for the graphics windows, and are documented in the Controls section.

The Right Side Bar

The right SideBar is currently used to display information about the model in 3D graphics mode. The user can use the select tool to get information about any part of the model. This SideBar is customizable and scriptable from MOSES macros.

The Main Workspace

The Main workspace has three parts to it: the command line, the tabs, and the data window, which can be either text or graphic. The command line is the standard MOSES command line and accepts standard MOSES commands. The text data window shows the same MOSES output the user is accustomed to. The graphical data windows can show standard MOSES pictures, MOSES graphs, and the new OpenGL pictures. The Tab system allows the user to open and interact with multiple pictures. It should be noted, though, that entering the MEDIT menu will cause all pictures to be closed, as the model data may have changed. When a tab containing a picture is in focus, there will be a "control panel" at the bottom of the picture. This panel contain (from left to right): a window showing the current process, a button setting the speed of an animation to half the normal speed, a button for playing/stopping the animation, a slider showing the events, and a box where you can pick the view. When the picture is created, it will show the last event in the current process.

The Status Bar

The Status Bar currently shows whether MOSES is Busy or Ready for another command, as well as showing what command will actually be executed when the mouse moves over an item in the menu.

Window Interface

The window interface presents the user with a window containing four basic areas: a tool bar, a display area, a command line box, and a scroll bar. The display area is used to give you information during a program session. This information is also written to a file, the "log file" so that you can review it later. The information is scrollable so that you can look at any portion of it any time a command is expected. As you will see below, the display can be toggled back and forth between the log, the user manual, and for a window interface, a picture.

The tools bar can be used to change settings and obtain reports without actually inputing the commands to do so. At the moment, it does not entirely suffice in place of the command line but in the future it will. When you push one of the buttons on the tool bar, a menu will "drop down". In this new menu, buttons which are "plane" will immediately do what the title says. Those which end in a > will drop down another level of menu. To clear a menu, you should push the top area which is blank except for a <. A large tree of menus can be cleared by simply hitting an "Enter".

Keyboard Shortcuts

There are special keys that are mapped to commands, or which move one about in the display. They are in either display:

&FINISH	– Ctl F
&PICTURE	– Ctl P
&PICTURE –RENDER GL	– Ctl G
&PICTURE –RENDER WF	– Ctl W
@TOP	– the home key
@BOTTOM	– the end key
+P	– the page up key
–P	– the page down key
!–	– the up cursor key
!+	– the down cursor key

The first of these command terminates MOSES. The next three change to "picture mode" with the first using the type of picture last rendered as the type, and the other two using the render mode specified (G for a GL picture, W for a wire frame). See the section on Pictures for details. These four shortcuts are *always* available. The remainder of them discussed here work only when one is focused on the text display or the command line.

The next four commands simply move the user's reading position. **@TOP** put one a the beginning of the display and **@BOTTOM** at the end. **+P** command moves up a page and **–P** moves down one.

The last two commands move up and down one command in the command history. In addition to storing the entire terminal input/output history, MOSES also saves a record of the commands which have been issued. While there are numerous uses of this command log, the primary one is to allow the user to see "where he is". This is accomplished by the !P command, the form of which is:

!P, NUMB(1), NUMB(2)

When this command is issued, a portion of the command log will be printed to the terminal. If no option is specified, then the commands printed will be preceded by the command number, and if **–N** is specified as an option, then the numbers will not be printed. If neither NUMB(1) nor NUMB(2) is specified, then the last command issued will be printed, and if NUMB(1) is a negative number then the commands from the current number plus NUMB(1) to the current number will be printed. If NUMB(1) is an *, then the entire log will be printed, and if both NUMB(1) and NUMB(2) are positive the commands between the two numbers will be printed. For example, **!P** 10 20 lists all the commands between command number 10 and 20. **!P** 10 20 –**N** does the same thing, except no command number will be printed. **!P** –10 will show the last 10 commands.

Another benefit of the command log is that previous commands can be re–executed by either the command:

!, PHRASE

or by the method discussed below. Here, PHRASE can be either nothing, a number, or a string. If it is nothing, the last command executed will be place in the command box so that it can be edited and executed. This is equivalent to the cursor up key on a window interface. Repeated use of the cursor up key will move up the command history. If it is a number, MOSES will simply execute the specified command. If it is a string, then MOSES will search up the command log until it finds a command containing STRING and will then execute that command.

IV.B Commands, Menus, and Numbers

Each input record can contain three types of data. The first word on the record is called the command or description name and it conveys to the program the type of data being communicated with this record. The format of all records does not require that the data be in any particular column, but instead, the various data is separated by a comma, or by as many blanks as desired. The remainder of the information on the record is of two types: DATA or –OPTIONS. DATA must be in the order specified, while –OPTIONS may be in any order. So that the program can distinguish between data and options, all options begin with a −. If the last word of a record image is a \, then the following record is a continuation of the current record. Also, the option lists consist of data which may or may not be needed. As many pieces of this data as required can be specified in any order, and usually consist of an alphanumeric "option keyword" followed by the corresponding alphanumeric or numeric data. Alphanumeric names may consist of up to eight characters, and numeric values may contain up to twenty characters. The general form of a command line is:

```
COMMAND, DAT1, DAT2, ... –OPTION1 OD11, OD12, ....., \
                    –OPTIONn ODn1, ODn2, ......
```

While the options can be input in any order, sometimes different results may be obtained with a different order of the options. This will occur when the data used by one option is altered by another one. MOSES parses options from left to right, so options which change data that another option will use should be placed first.

When it comes to the actual task of defining a number to MOSES one can accomplish the task in many ways. The flexibility is due to the fact that the command interpreter will perform a conversion of numerical data in accordance with FORTRAN conventions for arithmetic. In other words, a number can be defined as a series of numbers combined by primitive numerical operations.

As an example, consider the number 64. The following representations would all yield the same value:

```
61+3
8**2
(6+2)*8
((35–1)–2)*4/2
```

While this ability may appear to be of limited utility, it proves to be quite powerful when combined with the more advanced language features.

To simplify the operation and documentation of the program, MOSES employs the concept of menus. A menu, as used here, is an available list of commands which can currently be executed. If an attempt is made to execute a command which is not contained in the current list, a message to that effect will be reported, and a prompt for another command will be made. There are several menus in MOSES. When an **END** command is issued, MOSES will return to the next higher menu. To terminate execution of MOSES one simply inputs an **&FINISH** command, which is a valid command in any menu. In a window environment, the key Alt F can be used instead of typing in &FINISH.

There are several commands within MOSES which can be executed regardless of the current menu. These commands are called Internal Commands. In general, they control the operation of the program, set basic variables which effect the analysis, and can be distinguished by the fact that they all begin with the character **&**. An example is the **&FINISH** command, which is valid regardless of the current menu. The primary importance of an internal command is that it can be issued from either the INPUT or COMMAND channel, therefore, it can be set once in the INPUT channel, and later reset interactively.

MOSES uses minimum uniqueness to identify a command in the current list. By this, we mean that only enough of the command need be specified so that the program can *uniquely* define the intended command. If the command issued is not unique, all valid commands which match the one issued will be printed, and a prompt for a unique response will be given. If one issues a null command (a simple carriage return) MOSES will print a list of all the currently valid commands. For Internal Commands, minimum uniqueness in not employed. Instead, one need only specify the first five characters of the command name. Also, notice that MOSES uses minimum uniqueness for commands, but not for modeling language commands. In other words, commands which enter through the INPUT channel must be specified completely with the exception of internal commands.

MOSES has the notion of an escape character. This character is used to "remove" any special meaning associated with the following character. Here, the escape character is the \. An example of the use of this character was shown previously with the continuation of a command line. In this context, the character is used to escape the end of the line. If one actually wishes to input a \, he must use \\. This is particularly important on a PC when this character is used in defining directory paths.

In addition to the special characters discussed above, MOSES employs several others. A full list of the special characters are:

- \ used to remove the special meaning of the following character, and to provide for command line continuation.
- **$** used to denote the end–of–record. Any data which follows this character is ignored (hence it is useful for adding comments.)
- / used to denote the wild character. One or more of these may be placed anywhere in an alphanumeric name.
- **@** used to denote some number of wild characters. If this character is placed

in a name, it acts like some number of wild characters, /.

'	used to delimit a name which contains blanks and/or commas. The name must be enclosed by a pair of 's.
"	used to repeat command names or for a second level of quoting. If this is the first word of a record, the first word of the previous record will be used. If it is encountered in a position other than the command position, it act the same as a '. Here, it allows for a double level of including blanks.
–	The first character of an option name. The option is usually followed by a list of parameters in order to specify some desired action.
&	The first character of an internal command.
:	The first character of a selection criteria.
*	The first character of a point name.
#	The first character of a load attribute.
~	The first character of an element stiffness attribute name.

IV.C Files and the ROOT Concept

To perform an analysis, the user must input a complete description of the state of the system, plus commands defining the type of analysis to be performed. The data communicated to the program will enter through one of two input channels: 1) the "INPUT" channel or 2) the "COMMAND" channel. Generally, commands enter through the COMMAND channel while descriptions normally enter through the INPUT channel. In other words, the database is defined to the program via the INPUT channel while the COMMAND channel is used to tell the program what to do with the data in the database. It helps to think of the database as being defined by an input file while the commands are issued interactively at a terminal. Even though the program can be executed in a batch mode as well as interactively, it is best to think of all execution as being interactive.

While the details may vary with the installation you have, these two channels are files. MOSES organizes files according to a "root name" concept. In other words, the files associated with a job have the same prefix, or root, and the suffix defines the type of file associated with the root. When MOSES is executed, one normally furnishes a root on the command line. MOSES will then first look for commands to be executed in a file ROOT.**CIF**. It will execute any commands found in this file, and when the file is exhausted, it will look for commands from the terminal.

In addition, one execution of the program will result in two subdirectories, ROOT.DBA and ROOT.ANS, being created. ROOT.DBA contains the MOSES database for the root. One should *never* delete any of these files while a given job is being done. The ROOT.ANS directory contains the "answers" associated with the root. The files:

- which begin with OUT are the "output" files,
- which begin with LOG are the "log" files,
- which begin with DOC are the files written during document formatting,
- which begin with GRA are the files containing graphics,
- which begin with PPO are the files containing data for further post–processing,
- which begin with TABLE are the files generated which a STORE command is issues in the Disposition Menu, and
- which begin with MOD are the files containing models which can be used in MOSES.

The initial three characters are followed by a five digit number which signifies the order in which the files of the various files were created; e.g. LOG00001.TXT is the first log file that created and LOG00003.TXT is the third. The suffix of the file denotes the formatting of the file; e.g. TXT is a text file, htm is a HTML file, EPS is a postscript file, etc.

The OUT files contains all of the hardcopy reports you requested and the LOG files contain the commands issued and and MOSES responses to the commands.

IV.D Customizing Your Environment

In the directory where the software is installed, there is a subdirectory named **data**. This subdirectory stores data required for the execution of the software and files that allow the user to customize an installation. The **data** directory is further divided into subdirectories. The ones of interest here are named **local**, **progm** and **site**. The files **moses.aux**, **moses.mac**, **moses.man** and **moses.pgm** are stored in the **progm** directory. These files contain auxiliary shapes data, program macros, the on line reference manual and program parameters and default settings, respectively. Also at this directory level is the original **moses.cus** file provided with the installation.

The files in the **progm** directory are read each time the program is executed as part of program initialization, and should *not* be altered by the user. The **local** directory is provided for user customization. When the program is executed, it checks for the existence of a local database. If these do not exist, then it builds them. During the building of these databases, the program will attempt to read files **moses.mac** and **moses.aux** from the **local** directory. This allows one to add a set of site specific macros and structural shapes to those which are normally available. You should simply create files with the above names and then **delete** the file **moses.sit** on a UNIX machine or **moses.dsi** on a PC. The next time the program is executed, the databases will be recreated with your data included.

Most customization that one needs is available with the **moses.cus** file. This process is even easier than that described above. There can be many different copies of **moses.cus**, and they are read in order. First, the copy in the **data/progm** directory is read, next, the one in **data/local**. These are basically used to set variables for the entire network. After these two, MOSES looks for two more: first in location defined with the environment variable $HOME (%HOME% in WINDOWS), and then in the current working directory. The last two of these allow for customization at the user and job level. If you are homeless (do not know your home), you can find it by typing in a command prompt:

```
echo %home%  – on WINDOWS, or
echo $home   – on anything else
```

The "cus" file contains MOSES commands that localize MOSES for your situation. In addition, there is another set of files which contain user preferences. MOSES looks for **moses.ini** or **.moses.ini** in each of the location it looks for **moses.cus**. When looking in the MOSES install directories, the name without the . is used and in the home and local directories, the name with the . is used. The ini files are again simple text files that you can edit with a text editor, but you can also maintain the .moses.ini file in your home directory directly in MOSES. Simply use the Customize menu and select Preferences.

It is possible to select preferences you actually did not want. For instance, if you choose a proportional font for the log file, none of the reports coming to the screen will look correct, the titles and columns will be misaligned. The easiest way to solve this is to select a fixed pitch font, using Customize/Edit Preferences from the MOSES tool bar. If you want to return to the original defaults shipped with MOSES, delete moses.ini from your $HOME and **data/local** directories, use Customize/Edit Preferences, and click OK without changing anything.

Two other selections are available in this menu: REGISTER and UNREGISTER. REGISTER will register the software with the operating system. This provides the ability to click the MOSES icon to run the software, as well as associate certain file types with particular software. For instance, clicking on root.ans/gdv00001.eps can open the GhostView Postscript viewer. Clicking on root.cif, root.dat, root.ans/log00001.txt or root.ans/out00001.txt can open the VI text editor. Also, a right click on root.log can invoke the TIDY utility, which cleans up a MOSES database. Of course, if one prefers the command prompt method of starting MOSES, less typing is required if the path is set to where the software is installed. Using UNREGISTER will remove all these file associations.

V. MOSES DIMENSIONS

One of the interesting features of MOSES is that all of the data is stored in the database in a neutral format. When one inputs data, the units are converted to the "computational" units of the program, and when results are reported, they are converted to an output set of units. MOSES is informed of the units the user wishes to employ via the internal command &DIMEN. Since this is an internal command, it takes effect immediately. Thus, one can input a model using different units for various portions. He can also receive reports of results in a unit system different from that which defined the model. The form of this command is:

&DIMEN, –OPTIONS

and the available options are:

–DIMEN, LEN, FOR
–SAVE
–REMEMBER

The **–DIMEN** option controls the current program dimensions. Here, LEN is the length unit, which must be either **FEET** or **METERS**, and FOR is the force unit. If LEN = **FEET**, then FOR must be either **KIPS**, **L–TONS**, or **S–TONS**; and if LEN = **METERS**, then FOR must be either **K–NTS** or **M–TONS**. When a **&DIMEN** command with a **–DIMEN** option has been issued, MOSES will expect any subsequent input to be consistent with the units specified on the command, and all output will also be consistent. Notice that this scheme allows for the input of the data in a system of units different from the output with the insertion of a new **&DIMEN, –DIMEN** command.

The last two options allow for temporarily altering the dimensions and returning to the previous ones. In particular, **–SAVE** instructs the program to save the current dimensions so that when **–REMEMBER** is used, the ones previously saved will be recalled.

In documenting the use of units, confusion can arise due to the user's choice of force units. To ameliorate this difficulty, when a force which can be either kips, long tons, short tons, metric tons, or kilo–newtons can be used, we denote the force unit as **BFORCE** and the length unit as either **BLENGTH** or **llength** depending on the type of length unit required. Here, when blength is meters, llength will be mm, and when blength is feet, llength will be inches. This notation of blength, llength and bforce will be used throughout the remainder of the manual. In most places, the units required for a given quantity are specified. If they are omitted, they are:

Quantity		Units
time	–	seconds
angles	–	degrees
temperature	–	degrees F or degrees C
length	–	feet or meters
area	–	ft**2 or m**2
volume	–	ft**3 or m**3
velocity	–	ft/sec or m/sec
acceleration	–	ft/sec**2 or m/sec**2
stress or pressure	–	ksi or mpa
force	–	bforce
moment	–	bforce–blength
weight per unit length	–	bforce/blength
weight per unit area	–	bforce/blength**2

VI. DEVICES AND PROGRAM BEHAVIOR

Perhaps one of the most confusing things which MOSES does is dealing with devices. By device, we mean either a physical piece of hardware such as a printer, pen plotter, terminal screen, or logical device such as a file. In MOSES there are two concepts: a channel and a logical device. Basically, channels can be thought of as either files or as directly connected physical devices. Logical devices are different classes of output which are "connected" to a channel. As data is written to a logical device, it is formatted according to a set of instructions collectively called a style. In the following pages, each of these concepts will be defined in detail. For most of the things defining device attributes, the units required are points. A point is 1/72 of an inch or .3527 mm. The *exception* to the above rule is when one defines the "pitch" of a fixed pitch font which is defined in characters per inch.

VI.A Colors

Colors in MOSES are treated by first defining a set of colors which may be used and then assigning individual colors to a "color scheme". Color schemes are then used by assigning them to styles (which are discussed below), or by special names which are used to draw the user interface window.

Both colors and color schemes are defined with the command:

&COLOR NAME –OPTIONS

To define a color, NAME should be omitted and the option:

–COLOR_ADD, C_NAME, RVAL, GVAL, BVAL

should be used. Here, C_NAME is the name of the color to be defined and RVAL, GVAL, and BVAL are the intensities of red, green, and blue respectively. These values are 0 for none of this color and 255 for the maximum of this color. Thus, to define yellow, one should use

&COLOR –COLOR_ADD YELLOW 255 255 0

By default, there are over 400 colors already defined (more than the colors in the X11 file rgb.txt). To see what colors are predefined, one can use the command:

&COLOR –NAMES

The **&COLOR** command is also used to define the color maps used for pictures; i.e. the map which maps ratios and intensities to colors. This is accomplished with the options:

–S_MAP, C_NAME(1), C_NAME(2), ... C_NAME(n)

and

–R_MAP, C_NAME(1), C_NAME(2), ... C_NAME(n)

Here, **–S_MAP** defines the color map used for stress coloring and **–R_MAP** defines the map for ratio coloring. C_NAME(i) are the names of colors and there can be from two to 128 names. If d = 1. / (N – 1) where N is the number of names specified, then C_NAME1 is used for ratios between 0 and d, C_NAME(2) between d and 2d, ... C_NAME(n–1) between (n–2)d and 1, and C_NAME(n) for ratios greater than 1.

The final thing accomplished with the **&COLOR** command is the definition of color schemes. A color scheme is two sets of colors, one "normal" the other "special". Each of these has: a background color, a foreground color, box edge colors, selected colors, and line colors. The normal colors are those used except when a special effect is desired: Reverse video for something selected in a menu, or a line that is below the water. The option:

–**USE**, C_NAME

causes MOSES to load up the color scheme C_NAME and then what follows will only change what is in C_NAME. Thus, if one uses this option he need only specify those things that he wishes to be different from C_NAME. Likewise, if one issues:

&COLOR NAME .....

and the color scheme NAME exists, then he is actually editing NAME. There are several color schemes predefined in MOSES: DEFAULT, BASIC_FRAME, BASIC, MENU, WIDG_FRAME, WIDGET, and H_COPY. The first of these is the basis for most things. BASIC, BASIC_FRAME and MENU are used for the components of the MOSES User Interface Window and WIDG_FRAME and WIDGET are used for things that "pop up" in this window. H_COPY is used for things which are written to hardcopy devices. In general, the only difference between these is the color of the background.

All of these colors themselves are defined with the options:

–**N_BACKGROUND**, C_NAME
–**S_BACKGROUND**, C_NAME
–**N_FORGROUND**, C_NAME
–**S_FORGROUND**, C_NAME
–**N_BEDGE**, C_NAME(1), C_NAME(2)
–**S_BEDGE**, C_NAME(1), C_NAME(2)
–**N_SELECT**, C_NAME(1), C_NAME(2)
–**S_SELECT**, C_NAME(1), C_NAME(2)
–**N_LINES**, C_NAME(1), C_NAME(2), ... C_NAME(6)
–**S_LINES**, C_NAME(1), C_NAME(2), ... C_NAME(6)

The options –**N_BACKGROUND** and –**S_BACKGROUND** define the background colors. Here (and what follows), the prefix –**N_** defines "normal" colors and –**S_** defines "special" ones. The options –**N_FORGROUND** and –**S_FORGROUND** define the foreground colors. The options –**N_BEDGE** and –**S_BEDGE** each define two colors. The first color is for a sunken box and the last for a raised box. The options –**N_SELECT** and –**S_SELECT** also define two colors, the first for quantities selected and the second for those not selected. Finally, the options –**N_LINES** and –**S_LINES** define six colors for "logical lines 1 – 6". These are used in both graphs and pictures. For example when drawing a graph, the foreground color is used for the border and the axes and line color 1 is used for the first curve, line color 2 for the second curve, etc.

VI.B Defining Styles

Whenever MOSES writes a report or draws a picture, it is done using a set of attributes which are called a style. Precisely which style is used when will be discussed later, but styles are defined to MOSES by a command:

&STYLE, ST_NAME, –OPTIONS

where ST_NAME is the style name and the available options are:

–**COLOR_SCHEME**, CS_NAME
–**PITCH**, CPI
–**POINTS**, CHAR_HEIGHT
–**LEDDING**, LED
–**TABS**, TBCHAR, S(1), S(2), ...
–**R_INDENT**, RI
–**L_INDENT**, LI
–**F_INDENT**, FI
–**BEFORE**, BEF_POINTS
–**AFTER**, AFT_POINTS
–**JUSTIFY**, YES/NO
–**FONT**, FONT_NAME
–**FACE**, FACE_NAME
–**USE** OLD_NAME
–**CSYM_HEIGHT**, CSY_HEIGHT
–**LINE_WIDTH**, LIN_WIDTH

The –**COLOR_SCHEME** option here defines the color scheme which will be used when displaying this style. The options –**PITCH**, –**POINTS**, –**LEDDING**, and –**TABS** are used to define the text size and spacing. Here, CPI is the number of characters per inch which will be printed, CHAR_HEIGHT defines the height of the characters being printed, and LED defines the amount of space between lines of type as a fraction of the type size. Standard type has a ledding of 1.2, so that for a default device, if one wants to double space text, he should specify LED to be 2.2. The –**TABS** option is used to define a tab character, TBCHAR, and a set of tab stops in characters.

The next three options define any indentation from the margins for this style. Here, RI is the indentation from the right margin, LI is the indentation from the left margin, and FI is the indentation for the first line of a paragraph. Both RI and LI must be positive, but FI may be a negative number of a value up to the value of RI. Again, RI, LI, and FI are measured in points.

The –**BEFORE** and –**AFTER** options define the number of points of blank space which will be set before or after a paragraph, while the –**JUSTIFY** option is used to define whether or not the right margin of the text will be justified. If

YES/NO is **NO**, the margin will not be justified.

The –**FONT** option is used to define the type style which will be used, and the value for FONT_NAME must be either **LGOTHIC** or **COURIER**. The –**FACE** option defines the character of the type, and FACE_NAME must be either: **NORMAL**, **UNDERLINE**, **BOLD**, **ITALICS**, or **BITALICS**. Here, NORMAL will be standard type, UNDERLINE will be underlined, etc.

The –**USE** option is used to create a new style based on a previously defined one, OLD_NAME. If used, this should be the first option specified, and it instructs MOSES to use the previous style as the default values for the current style.

The last two options are used when pictures or graphs are being produced. The –**CSYM_HEIGHT** option defines the height of any centered symbols on a graph. These are normally used to differentiate between two curves, or to denote points. The –**LINE_WIDTH** option defines the width of any lines being drawn.

VI.C Logical Devices and Channels

In MOSES the concepts of channel and logical device are quite similar. A logical device is simply an additional layer of abstraction which allows one to achieve precisely the results he wishes. We will begin with the process of defining a logical device which is accomplished by the command:

&LOGDEVICE, LDVNAM, –OPTIONS

and the available options are:

–**CHANNEL**, CHANAM
–**STYLE**, T_STYLE, N_STYLE, A_STYLE
–**MARGIN**, IM, OM, TM, BM
–**LANDSCAPE**, YES/NO
–**PSOURCE**, TRAY

Here, LDVNAM is the logical device name which must be either: **LOG**, **OUTPUT**, **SCREEN**, **GRA_DEVI**, **DOCUMENT**, **TABLE**, **PPOUT**, or **MODEL**. These logical devices serve to provide output for screen reports, hard copy reports, screen graphics, hard copy graphics, hard copy documents, stored tables, post–processing information, and MOSES model data respectively. The options serve to define the appearance of the results. The –**CHANNEL** option associates a channel with the logical device. More will be said about channels later, but the valid values for CHANAM are exactly the same as for LDVNAM.

The –**STYLE** option defines a style which will normally be used when writing things to this logical device. For graphics devices, three styles can be specified with T_STYLE defining the style for text on the graphics, N_STYLE the numbers, and A_STYLE the axes.

The –**MARGIN** option defines the margins for a page in points. IM and OM define the "inside" and "outside" margins, and TM and BM define the top and bottom margin. The –**LANDSCAPE** option can alter the orientation of the print on the paper. If YES/NO is **NO**, the results are placed on the paper so that they should be read with the "long edge of the paper" on the left. If YES/NO is **YES**, then the page will be rotated 90 degrees. The margins are a property of the paper itself with the program taking care of the details when landscape and double sided printing are performed. The –**PSOURCE** option selects the current paper tray. Here, TRAY must be either **UPPER** or **LOWER**, and this option only works for certain physical devices.

Channels are defined with the command:

&CHANNEL, CHANAM, –OPTIONS

where the available options are:

–**PAGE_DIMEN**, WIDTH, HEIGHT
–**DOUSIDE**, YES/NO
–**P_DEVICE**, PDVNAM, LEVEL
–**FILE**, FILE
–**SINGLE**, YES/NO

Basically, this command associates a true physical device with the channel, CHANAM, and the valid values for CHANAM are those given for LDVNAM. Here the –**PAGE_DIMEN** option defines the size of a page on the channel in points,and the –**DOUSIDE** option instructs the printer to print on both sides of the paper, and currently works only on PCL devices.

The physical devices which can be connected to channels are defined with the –**P_DEVICE** option. Here valid values of PDVNAM are: **SCREEN**, **DEFAULT**, **POSTSCRIPT**, **TEX**, **PCL**, **JPG**, **PNG**, **UGX**, **DXF**, **HTML**, or **CSV**. If the physical device is **PCL**, then LEVEL defines the PCL level for the device, which must be either 3, 4, or 5. Here, the physical device SCREEN does double duty in that it is specified for both the LOG and graphics screen channels, and the particular behavior of the SCREEN depends upon the user interface one is using.

Normally, logical devices of the same name are associated with these channels, but it is not necessary. More will be said about this later. Not all physical devices can be connected to all channels. In particular:

- LOG can only be connected to DEFAULT or SCREEN physical devices.
- OUTPUT can only be connected to DEFAULT, TEX, POSTSCRIPT, or PCL physical devices.
- SCREEN can only be connected to DEFAULT or SCREEN physical devices.
- GRA_DEVI can only be connected to UGX, POSTSCRIPT, DXF, JPG, or PNG physical devices.
- DOCUMENT can only be connected to DEFAULT, TEX, POSTSCRIPT, or PCL physical devices.
- TABLE can only be connected to HTML or CSV physical devices.
- PPOUT can only be connected to DEFAULT physical devices.
- MODEL can only be connected to DEFAULT physical devices.

With the exception of the SCREEN, channels are connected to files. For example, the results for channel OUTPUT will be written to the file OUTXXXXX.TXT in the ROOT.ANS directory. In general, the file name is the first three characters of the channel names followed by a five character number and having a suffix which corresponds to the physical device. Here, the physical device/suffixes are:

DEFAULT	txt
POSTSCRIPT	eps
TEX	tex
PCL	pcl
DXF	dxf
JPG	jpg
PNG	png
UGX	ugx
HTML	htm
CSV	csv

If you want to change the directory where the "answer files" are stored, you can use the command **&FILE USE** discussed below. If you want to change the location of the file for a given channel, you can use the **–FILE** can be used. Normally all of the graphics for a given run will be stored in a single file (except for types of JPG or PNG). This can be changed with the **–SINGLE** option where a YES/NO of **YES** will result in each frame of graphics being written to a separate file.

Each time a **&CHANNEL** is issued, the file currently connected to the channel will be closed and a new file will be used. For example, suppose GRA00001.eps is the file currently connected to the GRA_DEV channel. Then the command

&CHANNEL GRA_DEV

will result in GRA00001.eps being closed and the next graphics will be written to GRA00002.eps. If GRA00001.eps is empty, then nothing will happen.

It may seem odd that the available channels and logical devices are the same, but it offers quite a bit of flexibility. For example if one has a postscript printer, then he only needs one of the channels OUTPUT, DOCUMENT, or GRA_DEV, and he can connect all three of these logical devices to it. This makes the results of all three logical devices appear in order when printed. A drawback of this approach, however, is that once the OUTPUT logical device is formatted for a particular device it may become unreadable.

The **TEX** device is actually a file written in a format that can be processes by the popular text formatting program LaTeX. When this device is connected to the DOCUMENT channel, it generates a stand alone file for input into LaTeX. Included in this file is a set of macros which "make it work". When connected to the OUTPUT channel, the file is not complete in that it is missing the macros, the prologue, the "begin document" and the "end document" statements. This occurs since one normally wants to input the output file into the document file to produce a complete document.

The **DXF** and **UGX** physical devices are for saving graphical results. The **DXF** device is used to store MOSES graphics into **DXF** format. The **UGX** device was designed as a page description language, and is more completely defined later. When this device is used, the file can be read by MOSES and converted to a graph on the specified logical device. In particular, graphics saved in this format can be viewed on the screen by simply issuing the command:

```
&INSERT GRA00001.UGX
```

Also, if the primary graphics device is the **GRA_DEV** logical device, then the above command will format the graphics for a hard copy device.

For any channel which has a physical device of PCL, the results in the file contain escape sequences which control the device. Thus, files representing these channels should be sent to the device in *raw* form. If MOSES is being run on a PC, these files should be copied to the printer using the /B option on the DOS COPY command. This avoids the DOS spooler treating the printer escape sequences as end of line or end of file characters.

VI.D Controlling Execution

The user also has the ability to control the various parameters which effect the execution of MOSES and the various files which will be used. Perhaps the most useful of these commands is:

&DEVICE, –OPTIONS

and the available options are

–**BATCH**, YES/NO
–**LIMERROR**, ERLIM
–**US_DATE**, YES/NO
–**SET_DATE**, DATE
–**CONT_ENTRY**, The Entry You Want
–**NAME_FIGURE**, NAME
–**FIG_NUM**, YES/NO, NUMBER
–**G_DEFAULT**, GLDEVICE
–**OECHO**, YES/NO
–**MECHO**, YES/NO
–**IDISPLAY**, YES/NO
–**SILENT**, YES/NO
–**FN_LOWER**, YES/NO
–**COMIN**, FILE_NAME
–**ICOMIN**, FILE_NAME
–**AUXIN**, FILE_NAME
–**IAUXIN**, FILE_NAME

These options can be roughly divided into several classes. The –**BATCH** option defines when the program will terminate abnormally. Here, YES/NO should be **YES** if one wishes to set the mode to batch, and **NO** if one wishes the execution mode to be interactive. The –**LIMERROR** option defines the error limit. If the program is in the "interactive" mode, it will terminate when the error limit, ERLIM, is reached. This limit is defined by the –**LIMERROR** option. In the "batch" mode, MOSES will terminate when any error is encountered.

The next set of options define the way dates and figures are printed. The –**US_DATE** controls the style of the date which will be printed on the output reports. If YES/NO is **YES**, the date will be the month, followed by the day, followed by the year. If YES/NO is **NO**, then the day will be printed first, followed by the month, followed by the year. The –**SET_DATE** options allows one to actually set the date string to the string which follows. The –**CONT_ENTRY** option allows one to define the next entry which will be added to the table of contents. If the string following the option key word is blank, then MOSES will revert to the default behavior. The –**NAME_FIGURE** option allows one to define the string which will be put on graphics along with a figure number. The default here is "FIGURE". The –**FIG_NUM** option instructs MOSES whether or not to put figure numbers on plots, and perhaps what number to use. If YES/NO is **YES**, then "Figure XX" will be placed in the lower left corner of all plots not directed to the screen. Here, XX is a number which will be 1 for the first plot, 2 for the second, etc. If YES/NO is **NO**, no figure numbers will be plotted. If "NUMBER" is specified, then the *next* figure plotted will have the number specified.

The next option is used to alter the destination of graphic output. There are two places where graphics can be deposited: the primary place and the secondary one. When a PICTURE or PLOT command is issued, the results are automatically written to the primary place and when a SAVE is issued, the results are written to the secondary place. The –**G_DEFAULT** option defines the default logical device for graphics. Here GLDEVICE must be either **SCREEN**, or **FILE**, which will define either the **SCREEN**, or **GRA_DEVICE** logical device to be the device where default graphics is written.

The next class of options controls the type of output which is received. The –**OECHO** option is used to control the listing of output. If the YES/NO following this option is **YES**, then each record from the input file is written to the output file as the record is read. Conversely, if YES/NO is **NO**, the echo will not occur. The –**MECHO** option will instruct MOSES to echo commands which are being executed as part of a macro to be echoed to either the terminal (and therefore the log file), or to the output file. Macros being used in the input data will be echoed in the output file if the –**OECHO** YES/NO is **YES**. The options –**IDISPLAY**, and –**SILENT** control the default displays. The option –**IDISPLAY** controls whether or not the valid internal commands will be displayed whenever valid commands are listed. The –**SILENT** option suppresses most of the terminal output, and is useful in macros. For all of these options, the action will be taken if YES/NO is **YES**, and not if YES/NO is **NO**.

The –**FN_LOWER** option controls the way MOSES looks at the directory structure. If YES/NO is **NO** then the directory structure is viewed as case sensitive. In some operating systems it really does not matter even if then results appear in upper and lower case. In other words a file COW.TXT is a different file from cow.txt. One may not want this behavior (transferring files from a WINDOWS machine to a UNIX one for example). In this case one can use the option with a YES/NO of **YES**. Now the complete path of the file will be treated as being lower case.

The final class of options controls where the program obtains its information. In general, commands enter through the "command channel", TERM, and data enters through the "input channel", INPUT. With the –**COMIN** option MOSES gets command data from the file, FILE_NAME, until either an end of file is encountered, MOSES finishes, or another –**COMIN** command is specified. The –**AUXIN** option functions as the –**COMIN** option except that it redirects the flow of information on the INPUT channel. With both of these options, if FILE_NAME is **&E** then MOSES returns to the default channel. With these two options, a fatal error occurs if the file specified does not exist. If one does not desire an error if the file does not exist, then he can use either **–ICOMIN** or **–IAUXIN**. With these two options, the command is simply ignored if the files fail to exist. In some cases, one does not know whether he wishes to redirect the input or the command channels, but instead he wishes to redirect the "current" channel. This can be accomplished by using the command:

&INSERT, FILE_NAME

Here, the current information channel will obtain its data from the file FILE_NAME as discussed above.

VI.E Message Commands

MOSES allows the user to issue messages to both the "output" and "command" channels via internal commands. For the output channel, the messages are limited to two title lines of data which are printed at the top of each page of output. These can be defined via:

> **&TITLE**, MAIN_TITLE
> **&SUBTITLE**, SUBTITLE

Here, MAIN_TITLE is used for the first title line, and SUBTITLE is used for the second title line.

To issue messages to the command channel, one uses the commands:

> **&TYPE**, MESSAGE
> **&CTYPE**, MESSAGE
> **&CUTYPE**, MESSAGE

These commands will write "MESSAGE" to the terminal whenever they are executed. The difference between the commands is that the **&TYPE** command left justifies the output line, while the other two center the line on the screen. The **&CUTYPE** command not only centers the line, but also underlines it.

If one is writing macros, it is often necessary to report to the user an error or warning. This is accomplished with the command:

> **&ERROR**, CLASS ,MESSAGE

Here, CLASS is the class of warning and can be either **WARNING**, **ERROR**, or **FATAL**, and MESSAGE is the message you wish to have printed along with the class. If CLASS is **FATAL** then the program will terminate after printing the message.

In addition, there is a menu which can be used to define reports:

> **&REPORT**, HEAD(1), HEAD(2) –OPTIONS

and the available options are

> **–HARD**
> **–BOTH**

The report generated will have a primary heading of HEAD(1) and a secondary heading of HEAD(2). If no options are specified, it will be written to the terminal. If **–HARD** is specified, it will be written to the output file, and if **–BOTH** then it will be written to both devices. The menu is exited with an **END_REPORT** command.

Inside this menu, the commands

TYPE, MESSAGE
CTYPE, MESSAGE
CUTYPE, MESSAGE

are available. These commands are exactly the same as those discussed above.

VII. GENERAL PURPOSE INTERNAL MENUS

MOSES has several features of a general nature which are quite useful. These will be discussed in this section before the more specific abilities are considered.

VII.A The &SELECT Menu – The Selection Process

When communicating with MOSES, one must often select data from a set. For example, when one issues a command, he is really selecting one command for processing from a set of valid commands. In contrast, there are cases where one may select more than one item from the set. In either case, data must be given to MOSES so that a selection can be made. In most cases, the data will consist of the names of the items to be selected. Since simply issuing the names can become cumbersome, MOSES uses a more general method, the Selector.

A Selector can be a name, a name containing either @'s or /'s, or a Selection Criteria. Here, a / is a "wild character" which stands for any character, and an @ stands for an arbitrary number of characters. In many cases, the word **MATCH** will be used. By a match, we mean that two names are equivalent to within the wild characters. As an example, each of the following strings match the other:

ABCDEFGH
A/CDEFGH
A@H

A Selection Criteria is a more general method of selecting data. In essence, it is a name (:SEL_NAME, which must begin with a **:**) with which two sets of selectors are associated. The first set of selectors (S_NAME(i)) define a set of names which will be selected. The second set (E_NAME(i)) define a set of names which will be excluded.

The objective of the selection process is to take a set of values and to apply a selection rule which will result in a "selected" subset. The selector :SEL_NAME operates on the admissible set in two steps. The first is to search the entire list of values for a match with any of the S_NAMEs. This results in a subset which is then subject to exclusion by the second step. In other words, the results of the first step are checked for a match against any of the E_NAMEs. If a match is found, that item is removed from the selected set.

A special menu is provided to define and examine selection criteria. This menu is entered via the command:

&SELECT

and the valid commands are:

END_&SEL
NAME, :SEL_NAME, –OPTION
SELECT, S_NAME(1), S_NAME(2), ..., S_NAME(n)
EXCEPT, E_NAME(1), E_NAME(2), ..., E_NAME(n)
INFO_SEL, :SEL_NAME, –OPTION

LIST_SEL, –OPTION

The first of these commands define the "current" selection criteria name. All of the commands which follow will deal with the "current" name until it is redefined. If no option is included, then the action which follows will be added to the existing definition of the selection criteria. To "start over" one should use the option **–DELETE** on the NAME command. This will remove any previous data defined for the selection criteria. The two commands **SELECT** and **EXCEPT** are used to define the names for selection and exception respectively. The **LIST_SEL** command gives a list of selection criteria which have been defined, and the **INFO_SEL** command gives the specific S_NAMEs and E_NAMEs for the selection criteria :SEL_NAME. Both **LIST_SEL** and **INFO_SEL** accept the option **–HARD**. If this option is omitted, the reports will be written to the terminal, if it is included, they will be written to the output file.

If no S_NAMEs are defined for a given :SEL_NAME, then all of the available list will be selected in the first step, and if no E_NAMEs are defined, then all values selected by the S_NAMEs will be selected. Thus, a :SEL_NAME which has not been defined will select everything.

As an example of how to assemble these commands, consider:

```
&SELECT
   NAME :COW
   SELECT 1, 2, 3, 4, M@
   EXCEPT M1, M2, M3
   NAME :DOG
   SELECT D@
END_&SEL
```

Here, two selection criteria are defined, :COW and :DOG. The first one selects 1, 2, 3, 4 and everything beginning with M except M1, M2, and M3. :DOG simply selects everything beginning with a D.

In many cases, one can define a selection criteria quickly by using the abbreviated command:

&SELECT, **:**SEL_NAME, –OPTIONS

where the options are:

–SELECT, S_NAME(1), ..., S_NAME(n)
–EXCEPT, E_NAME(1), ..., E_NAME(n)

With the abbreviated form of the command, the selectors defined in the example above can be quickly defined by:

```
    &SELECT :COW –SELE 1, 2, 3, 4, M@ \
              –EXCEPT M1, M2, M3
    &SELECT :DOG –SELE D@
```

VII.B The &UGX Menu

As was mentioned previously, MOSES has a language for describing a picture on a page. There are two uses of this format: A way to ”save” graphics for later viewing, and a way to define pictures. The **&UGX** Menu allows pictures to be converted from the UGX format into some other format. Alternately, one can use the commands in this menu to generate pictures of his own creation.

The UGX Menu is different from most menus in that it is accessible *only* though a macro! Each trip into the **&UGX** Menu is used to produce a single page of graphics. To enter this menu, one simply issues:

&UGX, WHAT, WHERE

where WHAT is the name of a previously defined macro and WHERE can be used to alter the definition of the logical device to which the picture will be written. WHERE must be either: **SCREEN**, **DEVICE**, or **SAVE**, and if it is omitted, then the picture will be written to the primary graphics device.

A valid UGX macro is like any other macro except that it contains the commands listed below and it has the command

END_&UGX

as its last command. For example,

```
&MACRO LINE
MOVETO 0 0
LINE   20 20
END_&UGX
&ENDMACRO
&UGX LINE
```

will cause a picture to appear on the screen consisting of a single line.

Several commands are available to create pictures. The origin of the coordinate system is in the upper left hand corner of the page, the x axis is parallel to the top of the page, and the y axis is parallel to the left hand side of the page. All dimensions given here are in points. To allow for a **UGX** picture to be correctly mapped onto any device, the command:

BBOX, X_MAX, Y_MAX

will automatically establish a non–distorting scale so that the picture will fit on the current physical page. Here, X_MAX and Y_MAX are the maximum X and Y values which are used in the picture. Notice that by the choice of origin, there are no negative X or Y coordinates in this system.

To allow one flexibility in defining a picture, one can change the frame of reference at will by issuing the command:

> **TRANSFORM**, XO, YO, –OPTIONS

and the available options are:

> –**SCALE**, X_SCALE, Y_SCALE
> –**ANGLE**, ANGLE
> –**MATRIX**, Q11, Q21, Q12, Q22

After this command is issued, a transformation from the input coordinates to the frame described above is established which works as follows:

> o = t + Q*S*i

Where the two components of t are defined by XO and YO, the Q matrix is defined by either ANGLE or the components of Q, and the S matrix is a diagonal matrix with components given by X_SCALE and Y_SCALE. Normally, one does not need the **–MATRIX** option and instead defines the transformation via the **–ANGLE** option. Here, ANGLE is the angle in degrees through which the original x axis must be rotated to yield the new x axis. This rotation is positive toward the original y axis.

As with all output operations in MOSES, the **&UGX** commands are performed according to a style. To establish the current style here, one issues the command:

> **STYLE**, C_STYLE

Where C_STYLE is the style which will be used for all subsequent drawing. One can also change the color by issuing the command:

> **COLOR**, COLOR_NUM

where COLOR_NUM is a line color number. The association of line color numbers and colors was discussed previously.

Pictures are created by combining lines and text, and for each primitive, drawing begins at the "current cursor position". The current cursor position can be defined by the command:

> **MOVETO**, X, Y

To draw a generalized line, one issues:

LINE, X(2), Y(2), ..., X(n), Y(n)

This command will draw n–1 straight line segments beginning at the current cursor location and ending at the coordinates X(n), Y(n). At the conclusion of the drawing, the current location will be at X(n) and Y(n). To fill a polygon in the current color, one uses:

FILL, X(2), Y(2), ..., X(n), Y(n)

which works the same way as **LINE** does.

Three other primitives are available for drawing lines. To draw a rectangle, one can issue:

BOX, X_DIM, Y_DIM

The resulting rectangle will have the upper left corner at the current location and it will have a width of X_DIM and a height of Y_DIM. At the conclusion, the current location will remain unchanged. To draw a circular arc, one issues:

ARC, RADIUS, ANG(1), ANG(2)

The center of the arc is at the current location and the arc will have a radius of RADIUS. The arc will be drawn from the angles ANG(1) to the angle ANG(2). These angles are measured from the x axis positive toward y. If the two angles are omitted, a circle will be drawn. The current location is not changed by this command. The final line primitive draws a line with an arrow at the end of it, and is defined by:

DLINE, X2, Y2

The line is drawn from the current location to the point X2, Y2 and the arrow is placed at the second end. At the conclusion, the current location is at X2, Y2.

Four commands are available to annotate the picture. The command:

CSYMBOL, CSYM_NUM

will produce a "centered symbol" of a type defined by CSYM_NUM at the current location. CSYM_NUM is an integer from 1 to 9, where each number produces a different type of symbol. To write text, one has three commands available, none of which moves the current location. The command:

TEXT, ANGLE, TXTSTR

will write the string of text defined by TXTSTR beginning at the current location. The text will be at an angle, ANGLE, from the x axis. If ANGLE is omitted, zero will be used. The current location is not effected by this command. Another way of drawing text is with the command:

CTEXT, X2, Y2, TXTSTR

Here, the text string defined by TXTSTR is centered between the two points defined by the current location and X2, Y2. The text is drawn with the beginning of the string toward the current location so that the bottom of the text is parallel to the line between the two points. A similar command is:

DIMENSION, X2, Y2, TXTSTR

This command draws text the same way as CTEXT, but it also draws dimension lines between the two points.

Finally, if one is merging a page in **UGX** into other graphics generated by MOSES with figure numbers, the **UGX** figure should also be numbered. This is accomplished by issuing the command:

FIG_NUM, X, Y

which will place the figure number at the coordinates specified by X and Y.

VII.C The &D_GENERATE Menu – Document Formatting

To allow one to fully utilize MOSES's database structure, a text formatting capability is provided. This formatter basically takes input text and formats it according to a set of given instructions for a specified output device. To simplify matters, this process occurs within an internal menu, entered via the command:

&D_GENERATE, –OPTIONS

The available options are:

–**CMD**, CMD_FILL_CHAR
–**SAVE**
–**NO_CONT**

The **–CMD** option is used to alter the "internal command character" which will be used in this menu. Normally, this character is the **&**. The other two options control what is done with the results when the document formatter is exited. The –**SAVE** option instructs MOSES to save the current settings for page numbers, etc., and the **–NO_CONT** option says not to write a table of contents. To exit this menu, one issues a command:

CMD_FILL_CHAR**END**

Here CMD_FILL_CHAR is the same character specified with the –CMD option; i.e. if no –CMD option were issued, the menu would be exited with

&END_&D_GENERATE.

The basic job of the document formatter is to take a paragraph of text and rearrange it according to a user defined "style" which was discussed earlier and a set of formatting commands. Formatting commands are enclosed within a pair of { and }. To associate a given style with a paragraph, one should use the formatting command

{STYLE, ST_NAME**}**

The style ST_NAME will be used for all paragraphs until another style name is selected. Notice that if one uses a style before it is defined, an error will result. Notice that this is the general form of a formatting command; a { immediately followed by a command name, followed by a token delimiter, followed by data for the command, and finished with a }.

Until now, paragraphs have been discussed, but not defined. MOSES uses a method of implicit paragraph generation. In other words, a paragraph is a collection of lines delimited by either a line beginning with a blank, or certain formatting commands. This scheme works well for most text, but on occasion, one wishes to format the text himself. To allow for this, MOSES can operate in the ”pre–formatted” mode, where text is simply transmitted to the output device without rearranging it. In the pre–formatted mode, each line is a paragraph. To enter this mode, one issues the formatting command:

{PRE}

and to exit:

{!PRE}

In addition to the general commands described above, there are several others available which may expedite special tasks. In particular

{SKIP, POI_REL**}**
{MOVE, POI_ABS**}**
{EJECT}
{EODD}

allow for positioning text on a page. Here, POI_REL is a number of points relative to the current position at which the next text will be printed, and POI_ABS is a absolute position on a page where the next text will be printed. The **{EJECT}** and **{EODD}** commands cause next text to be printed on a new page. The **!EODD** command will create a blank page if the current page is an odd numbered one, and the document is double sided.

To control the general appearance of the pages, one can use

{RPAGNUM, NUMB**}**
{HEAD, TEXT**}**
{FOOT, TEXT**}**

The first of these commands sets the ”current” page number to be NUMB, and **{HEAD}** and **{FOOT}** are used to define headers and footers. The styles used for headers and footers are HEAD and FOOT respectively. The TEXT on a header or footer can be any one line of text containing positioning instructions, and it can contain the special symbol #, which will be replaced with the current page number when the text is written. For both footers and headers, the –BEFORE option of the style defines how far down from the bottom margin or top of the page the header or footer will be printed.

In many instances, one is producing a document which requires a table of contents and which has sections. This is facilitated by the commands:

{SECTION, TEXT**}**
{PART, TEXT**}**
{SUBPART, TEXT**}**

{BPAGE, TEXT**}**
{CONTENTS, TEXT**}**
{INDEX, TEXT**}**

For the first three of these commands, a section label will be created, the section heading will be printed using the style **SECTION**, and a table of contents entry will be generated. The next command will simply create a blank page with a single line of text in the center of it. The next command will either place the table of contents at this position in the document or create a place for it. The last command will place the index at this position in the document.

In addition to the general formatting via styles, MOSES has the ability to temporarily override styles within a paragraph. This is accomplished by enclosing parts of the text within a set of delimiters. To set some text bold, one should use

{b}

at the beginning of the bolding and

{dc}

at the conclusion. Likewise for underlining one uses

{U}

at the beginning and

{!U}

at the end. For italics, one uses the pairs **{I}** and **{!I}**.

Often on wishes to format a "list" of things. This is accomplished by entering the "LIST" mode with:

{LIST, LIST_TYPE –OPTIONS**}**

Here LIST_TYPE is the type of list and must be chosen from **ITEMIZE** or **USE_FIRST**; and if omitted, ITEMIZE will be used. A LIST_TYPE of ITEMIZE will put a "bullet" as the beginning of each list item while USE_FIRST will use the first non–blank character in the list as the item delimiter. Now, the list is composed of "items" which are simply paragraphs. To exit LIST mode, one uses:

{!LIST}

There is one option:

–SPACE, SPACE

which defines the vertical space in points at the beginning and end of the list.

The final formatting construct is "TABLE" mode which is entered with

{TABLE, TABLE_TITLE, –OPTIONS**}**

and exited with

{!TABLE}

A table consists of a centered title, TABLE_TITLE, and a matrix. The matrix is defined as each paragraph being a row. The general definition of a table row is

|column(1)|column(2) ..... |

where the beginning of the row is indented (rows are viewed as paragraphs). The | delimit the definition of the elements within the row. By default, the element will be centered within the column width. If one does not wish the elements to be centered, he can use < or > as the first character of a part to either left or right justify the part within the column. There is one option:

–MARK, SEP_CHAR

which is used to change the column separation character from | to SEP_CHAR. A degenerate list of a single row can be simply defined with

{CENTER |part(1)|part(2) ..... |**}**

This table will have no title.

VII.D The &BUILDG Menu

A menu is provided so that the user can simply input data and use it in the Disposition Menu. To exercise this capability, one should enter the command:

&BUILDG –OPTION

Here, the data which must be entered can be thought of as a matrix. The rows of the matrix correspond to the variables, and each row of the matrix contains the values of the variables at a common event. The details depend on what –OPTION is selected.

If one does not specify an option, MOSES will ask for the values of the legends. The legends correspond to labels of the columns of the matrix. Next the data is input. One simply inputs the matrix a row at a time. A null line terminates the input, and puts the user into the Disposition Menu.

If one specifies the option **–BRIEF**, then MOSES will neither prompt for data nor query for correctness. This enables one to read a file automatically which has been generated via a **STORE** command in the disposition menu, or any file written with the same structure. An example of this command is:

```
    &BUILDG –BRIEF
    DATA_1
    DATA_2
    DATA_3
$    DATA_1      DATA_2       DATA_3
    1          2            3
    4          5            6
    7          8            9
```

Here, DATA_1, DATA_2 and DATA_3 are the labels that will be used in the graph. The blank line between DATA_3 and the comment line is required. The remainder defines the data points. Each line defines potential points on a graph. The Disposition Menu is entered when a blank line of data is encountered.

Finally if one specifies the option **–CSV**, then MOSES will read the following as a csv file. structure. An example of this command is:

```
    &BUILDG –CSV
    DATA_1 DATA_2 DATA_3
$    DATA_1      DATA_2       DATA_3
    1          2            3
    4          5            6
    7          8            9
    END
```

VII.E The &TABLE Menu

A menu is provided so that the user can simply input data and have it written as either a CSV (comma separated value) or a HTML file. To exercise this capability, one should enter the command:

&TABLE –OPTION

followed by as many records as necessary to define the rows of the table. You should have the same number of tokens in each row (input record) and if you want to have spaces in a token, you should delimit the token with either a " or a '. To have a cell blank, you can use either " " or NA or NOT_USED. The following options control the appearance of the table:

–**HEADING**, HEAD
–**TITLE**, NCOL(1), CT(1), ... NCOL(n), CT(n)
–**BOLD**, YES_NO
–**H_SKIP**, YES_NO
–**ROW_SHADE**, YES_NO
–**FIGURES**, COL_SEL, RIGHT
–**EXTR_SHADE**, COL_SEL(1), COL_SEL(2), ...
–**V_LINES**, COL_SEL(1), COL_SEL(2), ...

After defining the rows, you should end the table with

END_&TABLE

The –**HEADING** and –**TITLE** options define the text at the top of the table and you probably want more than one of each. The headings and titles are emitted in the order the options are specified and the headings are emitted before the title. If the –**BOLD** option is specified with YES_NO if **YES** then the titles and headings will be set in "bold" type, and if the –**H_SKIP** option is specified with YES_NO of **YES** then a line will be skipped between each heading. In actuality, headings are simply a title which spans all of the columns of the table. Titles, however, can span NCOL(i) columns.

The –**ROW_SHADE** option instructs MOSES to shade the rows two at a time. If this option is selected with YES_NO of **YES** you will get two white rows followed by two rows shaded in light green. The –**FIGURES** option offers a way to change the display of the numbers. It says to change the number of figures after the decimal point for columns selected by COL_SEL to be RIGHT figures. You can specify more that one –**FIGURES** option. The –**EXTR_SHADE** option is used to define the columns for which the maximum and minimum values will be shaded in a different color. Here, COL_SEL(i) are selectors for the columns which will be shaded. COL_SEL(i) can be a single number which selects that mode number, or a pair of numbers A:B which selects all modes from A to B. The –**V_LINES** option defines the columns where vertical lines will be drawn at the right. Vertical lines are always drawn at the left of column 1 and to the right of the last column. If you have a column selector of 1, then a vertical line will be drawn between columns 1 and 2.

For CSV tables, the **–H_SKIP**, **–BOLD**, **–ROW_SHADE**, **–EXTR_SHADE**, and **–V_LINES** options have no effect.

VIII. PICTURES

As mentioned above, one can obtain a picture of the current situation by issuing the command the **&PICTURE** command. This command is used to get pictures of everything, and as a result, it has options to control:

- The type of data used in the picture,
- The title of the picture,
- The portion of global space to paint,
- The "annotation" of the picture,
- The production of a deflected shape picture,
- The use of color in the picture, and
- The names of things included in the picture.

Once something is specified on a **&PICTURE** command, it is remembered through out a MOSES session, unless however, it is changed. There are Tool Bar controls for all of the things on the picture command as well as a set of keyboard shortcuts to make life easier.

There is quite a bit of data which can be defined to create a picture which is precisely what one wants. To make it easier to reproduce these pictures, MOSES employs the concept of a "user defined view". One defines a view with the command:

&PI_VIEW –OPTION, V_NAME, ...

Here, V_NAME is the name which will be given to the view, and the data which follows is **any** data which can be placed on the **&PICTURE** command. The –OPTION may be either **–ADD** or **–DELETE**. For –DELETE, only V_NAME is required. Now, once a view has been defined, a picture can be produced with

&PICTURE, V_NAME, ...

Here, V_NAME is the name of a view defined via **&PI_VIEW**, and what follows can be nothing or any valid **&PICTURE** data. If additional data is specified, it modifies the picture defined in V_NAME. To make it easy to plot all of "your" views, you can use the string function:

&NAMES(PI_VIEW)

which returns all of the user defined views.

The form of the command is:

&PICTURE, VIEW_DATA –OPTIONS

With so much going on here, it is easy to get a picture which is worthless. The option:

–RESET

will reset the defaults so that something appears.

By default, the current title and subtitle are placed at the top of the picture. The options **–TITLE** and **–SUBTITLE** can be used to alter these values.

–TITLE, M_TITLE
–SUBTITLE, S_TITLE

The

–RENDER R_TYPE

option defines the manner in which the picture will be rendered. If you are in the GUI user interface, you can specify a R_TYPE of **SOLID** to have the picture rendered as a picture of solid objects. Alternative, a R_TYPE of **WF** will render the pictures as a "wire frame". WF rendering is the only thing available with a WIN interface. When rendering a solid picture, connectors are rendered as lines so that they will appear regardless of their diameter and/or distance. If, however, you exercise the option

–CON_SOLID YES/NO

with a value of YES, then they will be rendered as round elements with the proper diameter.

Finally, a picture that has been created may be written to a file for later processing by using the option

–SAVE_PIC

The file used by this command is specified on the **–SECONDARY** option of the **&DEVICE** command.

VIII.A Types of Pictures

The type of data used to construct a picture is specified with the option

–**TYPE**, TYPE

Here, TYPE must be selected from **DEFAULT**, **STRUCTURE**, **MESH**, or **COMPARTMENT**. When this option is used, the selected data appropriate to the TYPE specified will be extracted from the database, converted into "strings" which can be plotted, and stored in a different portion of the database. If an **&PICTURE** command is issued within the &SURFACE MENU, then blocks will be used instead of the current TYPE. The TYPE **DEFAULT** will show all exterior compartments and any beams which have load attributes.

There are two levels of control over the data to be plotted. First, one can select parts of the model by using the options for a **&REP_SEL** command issued *prior* to selecting a TYPE. Only data selected will then be passed to the picture painter. Additional control is available in the picture painter itself. Each string is assigned names for: **NAME**, **BODY**, **PART**, **ENDS**. **PARENT**, and **PIECE**. The "NAME" of a string is the element names if it is a structural element or connector, the panel name if it is a panel, or the load group name if it is a load group attribute. The "BODY" and "PART" are the obvious choices except for connectors where their BODY and PART is **GROUND**. The "ENDS" of a string are simply the points at the ends of the string.

"PARENT" and "PIECE" are a bit more abstract and the definition depends on the type of string. For structural elements with load attributes the "PIECE" is **SLAT** and for those without load attributes, it is **SELE**. The "PARENT" for all structural elements is the element class. For connectors, the piece is **CONNECTOR** and the parent is the element class. For panels, the piece is the piece name and the parent is the compartment name. For load group attributes the piece is either: **#BUOY**, **#TUBE**, **#AREA**, **#PLATE**, **#WEIGHT**, or **#LSET** depending upon how it was defined, and the parent is the load group name. If one is viewing pictures from the &SURFACE_MENU, then block names replace compartment and piece names in the above scheme.

To obtain pictures of the panels which will be used in a three dimensional diffraction analysis, one should use the TYPE **MESH**. With this TYPE, one can use the option –**DETAIL** and MOSES will generate a refined mesh for plotting which corresponds to current settings of the –**M_DIST** option of the **&PARAMETER** command.

In addition to these "names", the strings have numbers associated with them. In general, these numbers are a ratio, a cdr, a stress, and a deflection. Only nodes will have a deflection associated with them. These numbers can be used in the picture painter to alter the color mapping so that one can pictorially represent something other than the configuration. Only joints, beams, and plates have a cdr or cumulative damage ratio. Connectors have a ratio which corresponds to their current unity ratio. Compartments have a ratio which is their current percent full. The numbers associated with the structural elements are assigned in the Structural Post–Processing Menu. Until one enters this menu, all numbers will be zero. For nodes, the ratio is the maximum punching shear unity ratio for all load cases, the deflection is the last deflection reported with a JOINT DISPLACEMENT command, and the stress is zero. For beams, the ratio is the maximum code check ratio for all load cases, the stress is the largest Von Mises stress divided by is yield that is reported by a BEAM command, and the deflection is zero. These numbers are stored in the database so that at any time after the Structural Post–Processing, one can issue **&PICTURE** and see the results.

VIII.B Picture Views

As mentioned above, the "data" for an **&PICTURE** command is data describing the view. This data is

VIEW_DATA = VIEW, VAX, VAY, VAZ

Here, the values of VIEW and VA(i) define the projection which will be plotted. There are seven valid values for VIEW: **TOP**, **BOTTOM**, **STARBOARD**, **PORT**, **BOW**, **STERN**, and **ISO**. The first six of these produce projections in the global XY, YZ, and XZ planes respectively, while the **ISO** view is an isometric of the selected portion. The three angles, VA(i) are angles (deg.) which move the structure from its initial position to the one for viewing. The view produced is a projection of the rotated structure onto the plane specified by VIEW, and if no VA(i) are specified, then the bodies will not be rotated prior to projection.

For some types of data, an alternative method of defining the view is provided with the option

–**PLANE**, POI(1), POI(2), POI(3), TOL

Here, POI(i) are names of three points which will define a plane to be plotted. Here, the X axis is defined by a line connecting the first two points, and the Z axis is perpendicular to the X axis in the direction of the third point. When the plane is projected, the X axis will point toward the "left" and the Z axis will point "up". Finally, TOL is a tolerance (feet or meters) for members in the plane. If it is omitted a default will be used.

To simplify viewing, there is an option to incrementally change the view:

–**INC_VIEW**, WHAT, AMOUNT

Here, WHAT describes the action one wishes to perform and optionally, AMOUNT specifies an amount which depends on WHAT.

If WHAT is **ROTATE**, then AMOUNT specifies the rotation increment in degrees. If WHAT is **TRANSLATE**, then AMOUNT specifies the translation increment as a fraction of scene size. Two values of WHAT control the action of the mouse for GL pictures. A value of **SELECT** instructs MOSES to use the mouse for selection while a value of **ROTATE** says to use it for rotation. The final two unusual values here are **OBSERVER** and **MODEL** which defines what move, the observer or the model.

The remaining values of WHAT are of the form AB_DIR. Where A can be either a blank, **F** or **S**, B can be either **R** or **T**, and DIR can be **UP**, **DOWN**, **IN**, **OUT**, **LEFT**, **RIGHT**, **PORT**, or **STARBOARD**. The A part of WHAT defines the change of the move. A blank says to take the normal increment, **F** says to take 4 times the normal, and **S** to take 1/4. The B part of WHAT defines the what will change. A **R** says the move will be a rotation and a **T** says it is a translation. **UP** and **DOWN** define translations up and down vertically and roll rotations of the model or pitch rotations of the observer. **PORT** and **STARBOARD** define yaw rotations of both the observer and the model. There are keyboard shortcuts for all of these actions:

OBSERVER	O
MODEL	M
SELECT	S
ROTATE	R
T_IN	Up Arrow
T_OUT	Down Arrow
T_LEFT	Left Arrow
T_RIGHT	Right Arrow
T_UP	Page Up
T_DOWN	Page Down
R_UP	Home
R_DOWN	End
R_PORT	Insert
R_STARBOARD	Delete

Here the first column is the value of WHAT and the second is the key that maps to it. In addition, holding the CTL key down and then using a mapped key is the same as adding a **F** and holding the MOD key down is the same as adding a **S**.

The Left Button Bar duplicates some of the above functionality and adds two other features. A Click on the button resets the scene to the initial view; in other words, it is the same as **&PICTURE –RESET**. The rotate button is the same as **–INC_VIEW ROTATE** and the select button is the same as **–INC_VIEW SELECT**. After selecting this tool, press and hold down a mouse button in the picture window and move the mouse to rotate the scene. Finally, the GL button is used to change the render mode. This is useful for getting higher quality rendering or to change to a faster render mode when dealing with large models. Changing to Line Mode, navigating to the location of interest, and then changing to Normal Mode is a useful technique. The mode changes in this order: Normal–>Detailed–>Line–>Point–> (repeat)

With a GUI interface, each picture is placed in a separate frame so that you can look at them again. If you wish, you can delete some of the frames with the option:

–DELETE, N:M

Where N and M are numbers. With this option, MOSES will delete frames N through M. If M is omitted, all frames greater than and equal to N will be deleted. If only a single number is specified, then only that frame will be deleted.

VIII.C Picture Selection

There are quite a few options which can be used to paint only a portion of the data available. They are:

–**XG_WIND**, X_MIN, X_MAX
–**YG_WIND**, Y_MIN, Y_MAX
–**ZG_WIND**, Z_MIN, Z_MAX
–**CONNECTORS**, :CONE_SEL
–**POINTS**, :PNT_SEL, D_MIN, D_MAX
–**ONE_VERTEX**, :1V_SEL
–**RATIO**, BEG_RATIO, END_RATIO
–**STRESS**, BEG_RATIO, END_RATIO
–**CDR**, BEG_RATIO, END_RATIO
–**NAMES**, :NAME_SEL
–**BODY**, :BODY_SEL
–**PART**, :PART_SEL
–**PIECE**, :PIECE_SEL
–**PARENT**, :PART_SEL
–**CATEGORY**, :CAT_SEL
–**ENDS**, :END_SEL(1), ..., :END_SEL(4)

The three options –**XG_WIND**, –**YG_WIND**, and –**ZG_WIND** provide the ability to select portions of the model for viewing. MOSES checks the coordinates of each element of the picture against a "window" in space and scales the plot so that only those strings which are within the window will be plotted. Here, X_MIN and X_MAX define the limits of the window in the global X direction. Likewise, Y_MIN and Y_MAX define the window in the global Y direction and Z_MIN and Z_MAX do so in the global Z direction.

One may select only a portion of the strings for viewing by using some of the remaining options listed above. The –**CONNECTORS**, –**POINTS**, and –**ONE_VERTEX** options define a selector defining the connectors, points, or one vertex elements (strings with a single vertex) which will be plotted. If the selector is @ then all all will be shown, if it is blank, then none will be plotted, etc. The values D_MIN and D_MAX define the minimum and maximum diameter (inches or mm) of the shape used to represent the points. The diameter used will be based on the diameter of the elements in the model. The –**POINTS** option can be used in conjunction with –ANNOTATE POINTS to show specific points in the model.

The –**RATIO**, –**CDR**, and –**STRESS** options selects strings for plotting based on the value of their "ratio" "cdr" or "stress". Only strings with a ratio, cdr, or stress between BEG_RATIO and END_RATIO will be plotted. With the –**NAMES**, –**BODY**, –**PART**, –**PIECE**, –**PARENT**, –**CATEGORY**, or –**ENDS** options the selectors: :NAME_SEL, :BODY_SEL, :PART_SEL, :PART_SEL,

:PIECE_SEL, :CAT_SEL, and :END_SEL(1), ..., :END_SEL(4) will select strings based on the values of the appropriate quantities associated with the string. To qualify for being in a picture, a string must be selected by all five selection criteria.

VIII.D Picture Special Effects

In addition to the other things, the options

–**WATER_COLOR**, YES/NO
–**DEFLECT**, DFLMAG
–**ANOTATE**, WWHAT
–**COLOR**, CRITERIA
–**BACK_COLOR**, YES/NO
–**CULL_BACK**, YES/NO
–**CROP_FOR_TITLE**, YES/NO
–**SHRINK**, AMOUNT
–**T_SIZE**, TITLE_SIZE
–**A_SIZE**, ANO_SIZE
–**WG_MIN**, SIZE

can be used for special effects. The –**WATER_COLOR** option with a YES/NO of YES will use a different color (slightly darker) for things under the water than the color for those above. It will also result in the intersection of a panel with the waterplane being drawn. A YES/NO of NO will not produce the intersection and all lines of a panel will be drawn in the same color.

The –**DEFLECT** option instructs MOSES to make a plot of the deflected shape of the strings. Here, DFLMAG is the magnification which will be applied to the deflection when making the plot. To return to normal plots, one should use the –DEFLECT option with a magnitude of zero or less.

The –**ANOTATE** option defines whether or not the strings will be annotated with text when they are plotted. If WWHAT is **NO**, no annotation will be made. If WWHAT is **NAMES**, the names of the strings will be plotted. If WWHAT is **POINTS**, only the names of the points (strings with one vertex) will be plotted. Conversely, if WWHAT is **STRINGS**, only the names of strings with more than one vertex will be plotted. To plot only class names of strings, one should use **PARENT**. If WWHAT is **RATIO** or **STRESS**, then ratio or stress associated with the string will be used for annotation. To plot only the ratio or stress of points, one should use **P_RATIO** or **P_STRESS**. To plot only the ratio or stress of strings with more than one vertex, one should use **S_RATIO** or **S_STRESS**.

The next set of options control the meaning of the color used for the plot. By default, different strings are assigned different colors based on familial relationships. This mapping of color to string can be altered with the –**COLOR** option. Here, CRITERIA must be either **MODELED**, **BODY**, **PART**, **RATIO**, **CDR**, **STRESS**, **FLOODED**, or **SELECTED**. With a value of **MODELED** the color of a string is determined by the color defined by either **&DEFAULT** or the string attribute itself. With values of **BODY** or **PART**, the color of a string will be determined by its body or part. Each string has, as attributes, a ratio, a cdr, and stress, and each point has a deflection. By specifying **RATIO** or**CDR**, a color will be associated with each string based on the value of its ratio or cdr. Likewise, **STRESS** associates a color based on the ratio of the string's stress to its yield stress. For a CRITERIA of **FLOODED** or **SELECTED** the entire picture will be plotted in a "weak" color, while also honoring any window options. For **FLOODED** the flooded beams will be plotted in a bright color. Likewise, for **SELECTED** strings selected by the **–RATIO**,**–CDR**, **–NAMES**, **–BODY**, **–PART**, **–PARENT**, or **–ENDS** options are then plotted in a bright color. This is an excellent way to identify where these selected strings are located in an overall view of a model. Notice that using **SELECTED** with an **–ANOTATE** option and any selection option will yield a picture that only annotates the selected portion yet allows you to see the complete picture.

The option **–CULL_BACK** with a YES/NO of YES will omit the plotting of any polygon which faces toward the back of the view. Use of this option with a YES/NO of NO will revert to plotting all polygons. The **–BACK_COLOR** option with a YES/NO of YES will use a different color for back facing polygons. Of course, a YES/NO of NO will paint all polygons of the same class in the same color. The **–CROP_FOR_TITLE** option allows one to set the upper boundary for a picture to be below the title if YES/NO is YES. If YES/NO is NO, then the picture will be drawn over the title.

The **–SHRINK** option will move the outline of a string inward toward its center an amount AMOUNT. This is quite useful when using color to select things in a mesh model. Without shrinking, the color of a panel can be overwritten by the color of its neighbor. Also, this option is useful when looking for "missing" panels or in conjunction with **–BACK_COLOR** when looking for incorrectly ordered panels. The **–T_SIZE** and **–A_SIZE** options allow one to change the size of titles and annotation characters respectively. The defaults are 10 pixels for both.

Finally, the **–WG_MIN** options defines the minimum wave grid size to be "SIZE".

VIII.E Picture Animation

Normally, one receives a single picture in response to a &PICTURE command. If, however, this command is issued from either the Static Process Menu or the Process Post–Processing Menu, an animation of the process will be plotted. For an animation, the number of frames can be controlled with the option:

–**EVENTS**, EVE_BEGIN, EVE_END, EVE_INC

Here, EVE_BEGIN is the first event for which a picture will be drawn, ESTOP is the last event for which a picture will be drawn, and EVE_INC is the increment at which pictures will be drawn between EVE_BEGIN and ESTOP. One can specify multiple sets of these three values, and pictures will be drawn according to each set. By default, all events in the process will be plotted. If this "goes by too fast", use this option with

–EVENTS 0 9000 .1

which plots frames at .1 increments. If this is still too fast, change .1 to something smaller. Of course, increasing the increment will speed up the picture.

If one one has an animation, the option

–**MOVIE** M_TYPE, F_RATE, P_MULT

will produce a movie. When this option is used, the movie will be written to the a file and then operation will return to normal. Here, M_TYPE is the format of the movie and it must be either: **AVI** or **MPG**. F_RATE, is the frame rate, and P_MULT is the "play multiple". This will result in a movie with frames between EVE_BEGIN and ESTOP (defined with the –EVENTS option discussed below) at

P_MULT / F_RATE

second intervals. Notice that if P_MULT is .5, then the movie will be played at one half of the default speed. The defaults are to use a frame rate of 5, to play the entire process, and to use a play multiple of 1. If this is not smooth enough, you may want to increase the frame rate, but this will increase the time required to generate the movie. Finally, if you use M_TYPE of AVI, then Windows Media Player will not play it *unless* you have DviX codecs installed.

VIII.F Picture Ray Tracing

For achieving an even higher image quality or generating high resolution images of your model, MOSES supports exporting to a ray–tracing program. Ray tracing calculates the path of every beam of light in your scene, making it a very computationally intensive process, but one that can handle reflections, refraction, and transparency correctly, as opposed to the approximations we make in the normal 3D display in order to give users a smooth, interactive experience.

To export a scene for ray–tracing, it is recommended (but not required) that users first visualize the scene in MOSES's 3D viewer. Once the scene is positioned correctly, issuing the command:

```
&PICTURE –RENDER RAYTRACE –SAVE
```

will generate a file in the .ans directory called povxxxxx.pov which can be loaded and run by POV–Ray by double–clicking on it. If double–clicking on the file does not open POV–Ray, you need to download and install it from here The downloads are about half–way down the page.

Once you have it installed, double–clicking the .pov file should open it. Hit the Run button in POV–Ray to begin the render. For better quality, we recommend going to the menu and selecting "Render–>Edit Settings/Render" and selecting the last option under "Section:" [1280x1024, AA 0.3].

For those looking for an even more realistic rendering, you can use the &PICTURE option

–W_FANCY YES/NO

This option with YES/NO of **YES** will render realistic small waves when there is no sea specified.

After installing POV–Ray, command line users can directly operate on the pov00001.pov file by executing something like the following:

```
povray –ipov00001.pov –w1024 –h768
```

This would take input file pov00001.pov and render it at 1024x768. For further options, see the POV–Ray documentation.

IX. ADVANCED LANGUAGE FEATURES

In addition to the basic command structure outlined above, MOSES provides many features of a programming language. In MOSES, one can alter the flow of either command or description input, make logical checks, define variables, create macros, etc. All of these features operate on both commands and descriptions, so that with this language, one can automate the definition of a model as well as build a set of specific commands he needs to perform repetitive tasks. The language described here is an interpreted, string based language. In other words, all of the functions described here are performed before control is passed to the command interpreter.

IX.A Variables

MOSES has two types of variables, global and local. The difference between the two is the extent of their lives. A global variable lives forever (as long as the database for a given root exists), while a local variable vanishes once the procedure which defined it ceases to exist. The scheme employed here is similar to that used elsewhere in that if one defines a local variable with the same name as a global one, then the global variable will be "over loaded" as long as the local variable is active. In other words, if one defines a local variable with the same name as a local variable in a higher macro or with the same name as a global variable, the previous one will not be used so long as the new local variable remains defined. The "dummy variables" in macros are a special case of local variables.

By a variable, we mean a string which is stored in the database and which may be referred to by its name. Before one may reference a global variable, it must be defined via a **&SET** command. The form of this command is:

&SET, VAR, =, value

Notice that the equal sign *must* be followed by a space or comma before the value. All variables (including arguments) are processed as string replacements. In other words, the above command defines a string of characters. These characters will be replaced whenever the string %VAR is used in an input record. There are actually several ways of instructing MOSES to substitute a variable. In *all* cases, the variable must be referenced with a % as the beginning character. The next character may be a (, in which case, the reference must end with a). If the (is omitted, the reference can end with either a %, a comma or a blank. The only real reason to enclose the name in parentheses is that MOSES will honor one level of recursive use of variables.

For example:

```
&SET DOG = 20
&SET CAT = %DOG%*2000
&SET COW = %DOG%A
```

or

```
&SET COW = %(DOG)A
```

```
&TYPE %COW%
```

or

```
&TYPE %COW
```

or

```
&TYPE %(COW)
```

will set variable DOG to 20, CAT to 20*2000 (i.e. 40000), COW to 20A, and will type 20A to the terminal. For the recursive use of variables, consider:

```
&SET DOG1 = 20
&SET DOG2 = 40
&SET INTE = 1
&SET ANS  = 30*%(DOG%(INTE))
&TYPE %ANS
```

will result in 30*20 being written to the terminal. Here, the variable INTE will be evaluated first, and then the variable DOG1 will be substituted.

Often, one wishes to have variables which are local to a given macro. This can be accomplished by "typing" a set of variables to be local as follows:

&LOCAL, LVAR(1), LVAR(2) = VAL, ...

When local variables are typed, their values are set to blank, unless one follows the variable name with a token of = and another token containing the value to which the variable is to be set. Once a local variable is typed, it may be reset via an **&SET** command and used as if it were a global variable.

IX.B Loops and IF's

Two elements which must exist before a language can be useful are some means of altering execution and some method of repetition. Here, the execution control is offered by a standard block IF construct:

> **&IF**, LPHRASE(1), **&THEN**
> **&ELSEIF**, LPHRASE(2), **&THEN**
> **&ELSE**
> **&ENDIF**

Here, LPHRASE(1) and LPHRASE(2) are "logical phrases", and if LPHRASE(1) is .TRUE., then the commands which follow will be executed until the **&ELSE** is encountered. If LPHRASE(1) is .FALSE., then MOSES will evaluate LPHRASE(2). If it is .TRUE., then the commands following **&ELSEIF** will be executed until **&ELSE** is encountered, otherwise the commands between **&ELSE**, and **&ENDIF** will be executed. Both **&ELSEIF** and **&ELSE** may be omitted, and more than one **&ELSEIF** can be placed between an **&IF** and **&ELSE**, but **&IF** and **&ENDIF** must always be present. There is virtually no limit on the nesting of **&IF** blocks.

In MOSES, the concept of repetition is implemented via the constructs:

> **&LOOP**, INDEX, BEGVAL, ENDVAL, INCR
> **&LOOP**, VAR, (LIST(1), ... LIST(n))
> **&NEXT**, LPHRASE
> **&EXIT**, LPHRASE
> **&ENDLOOP**

The **&LOOP** command marks the beginning of a set of commands which will be repeated and the **&ENDLOOP** command marks the end. When this construct is encountered, MOSES will continue to execute the commands delimited until a termination criteria is satisfied. Termination can occur in one of two ways depending upon the form of the **&LOOP** command itself. The parameters on the first form of the **&LOOP** command allow for an "indexed" loop. In other words, INDEX is the name of a local variable which will be set to the *integer* BEGVAL at the beginning of the loop, and it will be updated to its current value plus INCR at each repetition. Loops of this type will terminate when INDEX is greater than ENDVAL. The second form of **&LOOP** is a "while" on the values in parenthesis. In other words, with this form, the variable VAR will be set to the value LIST(1) for the first trip through the loop, LIST(2) for the second trip, etc. The loop will terminate after LIST(n) has been used. Alternately, a loop will terminate whenever an **&EXIT** command is encountered with a logical phrase of .TRUE.. The commands between the **&LOOP** and the **&ENDLOOP** are executed in order. A "jump" to the bottom of a loop occurs whenever an **&NEXT** command is encountered with a logical phrase of .TRUE.. An example of the first form of the loop command is shown below:

```
&LOOP I 1 &TOKEN(N %NODES%) 1
  &SET J = %I%
  &NEXT &LOGICAL(%J% .EQ. 9)
  &EXIT &LOGICAL(%I% .GT. 11)
  &TYPE This is Loop Number %I%
&ENDLOOP
```

Here, I is the INDEX, and the loop continues from 1 to the number of tokens in the variable NODES, in increments of 1. The variable J is set to the value of I each pass through the loop, and a message is typed to the screen. If the J is equal to 9, the message is not printed. If I is greater than 11, or if I is the number of tokens in the variable NODES, the loop is terminated.

To terminate execution of either a macro or a command file, one can use the command:

&EOFILE, LPHRASE

When this command is encountered, and LPHRASE is .TRUE., the command file or macro will terminate execution. Similarly, one can conditionally terminate execution of the program by:

&FINISH, LPHRASE

which terminates the program if LPHRASE is .TRUE..

IX.C Macros

MOSES allows a user to execute a series of commands with the input of a single name. This is accomplished by allowing the user to define a "macro". Macros allow the user to conditionally execute commands, repeat blocks of commands, and set both local and global variables. These capabilities provide the user with a true "high level" language in which he can actually write a program to accomplish common tasks.

One defines a macro as follows:

```
&MACRO, NAME, ARG(1), ARG(2), ..., ARG(n),  \
        –OPTION(1), OPTVAR(1), CARG(1) = DEFAULT(1), ... \
        –OPTION(2), OPTVAR(2), CARG(2) = DEFAULT(2), ...
COMMAND(1)
COMMAND(2)
.
.
.
&ENDMACRO
```

Here, COMMAND(i) is any command which is either an internal command or a valid program command. The data following &MACRO is called the "picture" of the macro and NAME is the name of the macro and ARG(i) are the valid arguments to the macro. While there are no restrictions on the names for macros, one should be careful not to choose a name which will conflict with a valid program command. If the valid arguments are specified as above, then MOSES will parse the command for you and check for syntax errors. In some cases, however, one neither knows nor cares precisely the number or type of the arguments which will be specified. In this case, only the macro name should be specified when defining the macro. In all cases, the arguments actually supplied to the macro by the user will be available in the local variable **ARGV**. Thus, one can parse the values himself using **&TOKEN** and **ARGV**.

The arguments are "local variables" to the macro. When the macro is invoked (or executed), the user defines the values of the variables on the command. In other words, to execute the above macro, one would issue the command:

```
NAME, ARG1, ARG2, ...., ARGn
```

Now, suppose that for some command in the macro, the user had used the value for ARG1 which was accomplished by including the line:

```
COMMAND, %(ARG1)
```

When the macro is executed, the string %(ARG1) will be replaced by the string which is passed to the macro.

The options in macro arguments allow one to construct macros which have optional arguments. Any string on the macro definition line which begins with a − is considered to be a macro option. The string following the option name OPTVAR(i) is the name of the option variable which is set to .FALSE. if the option is not exercised when the macro is executed and to .TRUE. if the option was exercised. The other strings following an option (ARG(i)) are "optional data". The values of the optional data are all blank if they were not specified when the macro was executed. Additionally, one can specify a default value for this data, using the " = DEFAULT " syntax after the optional data.

When MOSES parses the picture, it will associate any "excess tokens" with the last variable in the picture. As an example, suppose that a macro was defined as:

```
&MACRO COW A B –OPT OPT C
&ENDMACRO
```

and that it was executed by:

```
COW C D E –OPT THIS IS AN OPTION
```

The results of this would be that the variable OPT would be set to .TRUE., A would be set to C, B would be set to D E, and C would be set to THIS IS AN OPTION. The same result can be obtained through the use of defaults. Consider the following macro defined as:

```
&MACRO COW A = C B = 'D E'  –OPT OPT C = 'THIS IS AN OPTION'
&ENDMACRO
```

and executed by:

```
COW  –OPT
```

This operation would behave the same as the previous example, but with fewer keystrokes. Notice the use of the single quotes to delimit data containing blanks.

By definition, a macro will be executed whenever the user specifies the macro name as a command. One may restrict when a macro can be executed by the following command:

&M_ACTIVE, NAME, COMNAM

After this command has been issued, macro NAME will only be executed from menus which have COMNAM as a valid command. This command should be used when one creates a macro containing commands which must be executed from a specific menu.

Sometimes, one will define a macro with a name which conflicts with a command name. If this occurs, it can be rectified by changing the name of the macro. This is accomplished via the **&M_CNAME** command. Its form is:

&M_CNAME, OLD_NAME, NEW_NAME

Also, if one defines a macro which does not work properly and he wishes to change it, he can delete the macro and start over. This is accomplished as:

&M_DELETE, MACRO_NAME

IX.D String Functions

Perhaps one of the more useful concepts MOSES employs is the "String Function." A string function converts an input string into another string, and the general syntax of a string function is:

&FUNCTION(ARG(1), ARG(2), ...)

There are quite a few things which can be accomplished with string functions. In the next several sections specific string functions will be discussed, but some examples are a string function is o check if a variable has been defined. The function:

&V_EXIST(VARNAM)

will return the string, **.TRUE.** if the variable VARNAM has been defined. If VARNAM has not been defined, a value of **.FALSE.** is returned. Another useful string function is **&FORMAT** which is used for formatting, and its form is:

&FORMAT(FMT, STRING)

Here FMT is a formatting instruction and STRING is the string to be formatted. If STRING is a number, then FMT can be a FORTRAN format (e.g. F10.2, I3, etc.). Otherwise, FMT must be either: **UPPER**, **LOWER**, **FIRST**, or **COMMA**. For **UPPER**, the entire string will be made into upper case characters, and for **LOWER**, the converse will occur. **FIRST** transforms only the first character in the string into upper case. Finally, **COMMA** adds a comma after each token in the string and an "and" between the last two tokens. FMT can have at most 8 characters.

To compress a logical phrase, one can use the function:

&LOGICAL(LPHRASE)

Here, a logical phrase is a string of numbers, characters, and logical variables combined by logical operators. Here, numbers are any string which can be converted to a number and logical variables are strings with values of either **.TRUE.** or **.FALSE.**. Any two tokens can be compared by the logical operators: **.EQ.** and **.NE.**. For numbers, four additional operators are available: **.LT.**, **.LE.**, **.GT.**, and **.GE.**. These six operators denote respectively: equal, not equal, less than, less than or equal, greater than, and greater than or equal.

A single logical variable can be modified by the operator **.NOT.**, and two logical variables can be combined by the two operators **.AND.** and **.OR.**. As examples, consider:

&LOGICAL(A .EQ. A)

```
&LOGICAL(.NOT. A .EQ. A .AND. B .EQ. B)
&LOGICAL(5 .LE. 4)
&LOGICAL(5 .GT. 4 .OR. A .EQ. B)
```

The strings resulting from these functions would be respectively: .TRUE., .FALSE., .FALSE., and .TRUE..

IX.D.1 The &INFO String Function

Often, one wishes to know the current settings. To accomplish this task, MOSES provides the string function:

&INFO(NAME, DATA)

In general, there are four classes of information available. Information about:

- Error information,
- Run information,
- Unit information,
- File information,
- Program information,
- Current output device information, and
- Graphics information.

Error information is obtained with a NAME of **SEVERITY**. When using macros or loops. it is sometimes useful to know if an error or warning occurred during execution. If so, appropriate action can be taken. The string function

&INFO(SEVERITY FLAG)

is used to determine if an error or warning has occurred. Here, FLAG is either **ERROR** or **WARNING**. The function will return a value of **.TRUE.** if FLAG is set to **WARNING**, and either an error or a warning occurred during the *previous* command. Likewise, a value of **.TRUE.** is also returned if FLAG is **ERROR**, and an error occurred during the previous command. Otherwise, a value of **.FALSE.** is returned. Consider the following example:

```
&IF &INFO(SEVERITY ERROR) &THEN
  &TYPE LOST BEYOND HOPE
  &FINISH
&ELSE &INFO(SEVERITY WARNING) &THEN
  &TYPE FIXING SITUATION
  FIXUP
&ENDIF
```

In this example, a message is typed to the screen and a fixup macro is executed if a warning resulted from the previous command. If an error occurred, a different message is typed, and the program is exited.

Run information is obtained with either: **MENU DATE**, **T_OF_DAY**, **TITLE**, **SUBTITLE**, **ROOT**, or **CWD**. The first of these returns the current menu while the second returns the date and the third the twenty four hour time when the function was called. TITLE and SUBTITLE return the title and subtitle respectively. ROOT returns the root file being executed as a file alone and CWD returns the current working directory of the program. Thus, to have a fully qualified path name of a file with the same path and primary name as the root, but with an extension of NEW, one would use:

&INFO(CWD)&INFO(ROOT).NEW

Unit information is obtained with either: **BFORCE**, **LFORCE**, **BLENGTH**, or **llength**. If **BFORCE** is used, then a string describing the unit used for "big force" (kips, l–tons, s–tons, kn, or tonnes) is returned. The other admissible values return the names for "little force" (pounds or newtons), "big length" (feet or meters), and "little length" (inches or mm).

File information is obtained with either: **C_FILE**, **C_PATH**, **C_IFILE**, **C_IPATH**, **C_CFILE**, **C_CPATH**, **FILE_EXISTS**, or **CHA_FILE**. Here, **C_FILE** and **C_PATH** return the names of the last file and path used by MOSES. **C_IFILE** and **C_IPATH** return the last used input file, meaning names ending in .dat and input file path. Types of **C_CFILE** and **C_CPATH** return the last used command file, meaning names ending .cif and the command file path. **FILE_EXISTS** returns .TRUE. or .FALSE. based on where DATA is a file which exists. **CHA_FILE** returns the file currently associated with the channel specified.

Program information is obtained with: **VERSION**, **REVNUM**. **HOME**, **PGMPAT**, or **OS_CLASS**. VERSION returns the general version being executed; e.g. REV 5.10, while REVNUM returns the total revision number; e.g. REV 5.10.010. HOME returns the path to the user's home directory. PGMPAT returns the path to where the program being executed is stored; e.g. /ultra. A type of **OS_CLASS** will return the type of operating system currently running the program.

For information about the current output device, one should use a NAME of either: **TPGLEN**, **OPGLEN**, **FONT**, **PWIDTH**, **PHEIGHT IMARGIN**, **OMARGIN**, **TMARGIN**, **BMARGIN**, **DOUSIDE LPI**, **CPI**, or **DEVICE**. The DEVICE value will return the physical device name of the current output device. All dimensions here returned here are in points except for CPI and LPI which are characters per inch and lines per inch respectively. Here TPGLEN returns the length of the screen. All of the others provided information about the current output channel.

Information about the current graphics is obtained with either: **FIG_NUM**, or **PICT_TYPE**. **FIG_NUM**, returns the number of the *next* figure plotted. PICT_TYPE returns the type of data currently being used for pictures.

IX.D.2 The &NUMBER String Function

Perhaps one of the most useful string functions is the function:

&NUMBER(TYPE, NUM(1), NUM(2), ...)

This function has an option keyword, TYPE, and a set of strings which are normally numbers. Here, the valid forms of the function are:

&NUMBER(?, STRING)
&NUMBER(REAL, RN)
&NUMBER(INTEGER, RN)
&NUMBER(SIN, RN)
&NUMBER(SIND, RN)
&NUMBER(COS, RN)
&NUMBER(COSD, RN)
&NUMBER(TAN, RN)
&NUMBER(TAND, RN)
&NUMBER(ACOS, RN)
&NUMBER(ASIN, RN)
&NUMBER(ATAN, RN)
&NUMBER(ATAN2, X, Y)
&NUMBER(SQRT, RN)
&NUMBER(LN, RN)
&NUMBER(EXP, RN)
&NUMBER(ABS, RN)
&NUMBER(MIN, RN(1), RN(2), ...)
&NUMBER(MAX, RN(1), RN(2), ...)
&NUMBER(MEAN, RN(1), RN(2), ...)
&NUMBER(SORT, RN(1), RN(2), ...)
&NUMBER(INTERPOLATE, X, X(1), ... Y(n))
&NUMBER(NORM, RN(1), RN(2), ...)
&NUMBER(DOT, RN(1), RN(2), ...)
&NUMBER(UNIT_VEC, RN(1), RN(2), ...)
&NUMBER(CROSS, RN(1), RN(2), ...)
&NUMBER(SCALE, SF, RN(1), ...)
&NUMBER(ADDV, SF, RN(1), ...)
&NUMBER(3PTS2Q, P1(1), P1(2), ... P3(3))
&NUMBER(VECG2L, Q VG(1), VG(2), VG(3))
&NUMBER(VECL2G, Q VL(1), VL(2), VL(3))

The result produced depends, of course, upon the value of TYPE. The first option, **?**, is different from the others, in that it returns a value of **.TRUE.** if the string is a valid number or **.FALSE.** if it is not. All of the others return a set of numbers.

As an example of how these functions operate, consider the following:

&NUMBER(REAL, (2+3)**8/2.)

This function will read the string "(2+3)**8/2." and convert it to a number, and then convert this number back into a string with eight significant figures. It is this final string which will be passed to the command interpreter. The function with a type of **INTEGER** operates in a similar manner, except that here, the result will be an integer. The idea behind these two is that in some cases a string representing a number may become too long to convert; therefore, strings can be "compressed" with this function.

The first several TYPEs take a single number as input and return a single number. The particular conversion which occurs with most values of TYPE is rather obvious from the name. In particular, **SIN**, **COS**, **TAN** produce strings with the value of the trigonometric functions of the same name and with the argument in radians. The trigonometric functions which end in **D** assume that the argument will be in degrees. The functions **ATAN**, **ATAN2**, **ACOS**, and **ASIN** are inverse trigonometric functions, and ATAN2 returns the angle who's tan is X/Y. The angles returned here are in radians. The types of **LN**, **EXP**, **SQRT**, and **ABS** produce the natural logarithm, the exponential, the square root, and the absolute value respectively.

The remainder of TYPEs take more than a single number as arguments. The **MAX** and **MIN** return the extremes of the arguments. A type of **MEAN** returns the mean of a set of numbers, while a a type of **SORT** sorts a set of numbers in ascending order. **INTERPOLATE** takes the first number as the desired "X" value, the next N/2 numbers as an array of Xs and the last N/2 numbers as an array of Ys. It returns the Y values which corresponds to the first number input. Of course, the X values must be in increasing order.

The next remainder of TYPEs allow one to treat strings as vectors of numbers. For a type of **NORM**, MOSES will return a string which is the square root of the sum of the squares of the numerical arguments. For a TYPE of **DOT**, the number of numerical arguments must be divisible by two, and it returns the inner product of two vectors represented by the 2*n numbers. With **UNIT_VEC** the value returned is the input vector scaled by its length. With **SCALE**, the first number is a multiplier and the function will return N numbers which represent the remaining numbers multiplied by the first. **ADDV** takes the first number and uses it as a scalar multiplier for the first N/2 numbers and it vectorially adds the result to the second N/2 numbers.

The last three functions deal with matrices and vectors. The TYPE **3PTS2Q** returns a direction cosine matrix. Here P1, P2, and P3 are the coordinates of three points (You need nine numbers here). The "origin" of the local system is at P1. The vector from P1 to P2 points "in the direction of the local X axis, and the vector from P1 to P3 points in the direction of the local Z axis. This direction cosine matrix will transform local vectors into global ones. The TYPE **VECG2L** transforms a vector in global coordinates to local ones and **VECL2G** performs the inverse. Here Q is a direction cosine matrix (there are nine numbers for Q) VG is a vector in global coordinates and VL is a vector in local coordinates.

IX.D.3 The &STRING String Function

Another function for dealing with strings is:

&STRING(ACTION, STRING(1), STRING(2), ... STRING(n)**)**

Here ACTION defines what you want the function to do, and STRING(i) are the strings. STRING(1) is the "basic string" and the others may be used for additional operational data, depending on ACTION. Here, the valid forms of the function are:

```
&STRING(SUBSTR,    STRING(1), BCN, ECN)
&STRING(BEFORE,    STRING(1), DELIM )
&STRING(AFTER,     STRING(1), DELIM )
&STRING(MATCH,     STRING(1), STRING(2) )
&STRING(NULL,      STRING(1) )
&STRING(O_NUMBER,  STRING(1), NUMBER )
&STRING(OVERLAY,   STRING(1), STRING(2) )
&STRING(REVERSE,   STRING(1), STRING(2), .....  )
&STRING(N_EXTRACT, WORD, NUMBER, STRING(3), ....  )
&STRING(REPEAT,    WORD, NUMBER )
```

For **SUBSTR**, the result will be the characters BCN through ECN of the input string, STRING(1). For **BEFORE**, the result will be the characters of STRING(1) starting at the beginning of the string through the character before the character DELIM. For **AFTER**, the result is the last portion of STRING(1), beginning with the character after the last occurrence of DELIM. **MATCH** returns a value of **.TRUE.**, if STRING(1) matches STRING(2), and **.FALSE.** if it does not. The function **NULL** returns a value of **.TRUE.** if STRING1 is null, and a value of **.FALSE.** otherwise. The function **O_NUMBER** "overlays" the integer, NUMBER, on the string, STRING(1), starting at the end. For example if STRING(1) is N00000000 and number is 3333, then the returned string will be N00003333. The function **OVERLAY** works in a similar fashion, except the values here are characters. The command &STRING(OVERLAY 12345678 ABCDE) will return a value of 123ABCDE, for instance. Both of these functions are useful for building names, which can be used to automate the creation of various analytical models. The function **REVERSE** will reverse the tokens of STRING1. **N_EXTRACT** will return NUMBER numbers from STRING(3), starting at the token immediately following the word WORD. If there are not NUMBER numbers, zeros are returned; i.e. first, WORD is found in the string, and the next token is checked. If it is not a number or if WORD is not found, then NUMBER zeros are returned. If this token is a number, then the first value returned is set to the number and the next token is checked. This process is repeated until either NUMBER numbers have been found, or a non–number terminates the process. Finally, the function **REPEAT** will return a string with NUMBER occurrences of STRING.

IX.D.4 The &TOKEN String Function

One useful string function is:

&TOKEN(ACTION, STRING2**)**

which returns a substring of STRING2 based upon ACTION. If ACTION equals **L**, then the last token of STRING2 is returned, and if ACTION equals **N** the number of tokens in STRING2 will be returned. In other cases, ACTION will be a number, a number followed by a **:**, or a pair of numbers separated by a **:**. Here, the number, N, will return the Nth token of STRING2, N:M will return Nth through Mth tokens and N: will return the Nth through the last token. If N is greater than the number of tokens in the string, a null token is returned. For example,

```
&SET STRING = A B C D E F G H I J K
&TYPE &TOKEN(L %STRING )
&TYPE &TOKEN(N %STRING )
&TYPE &TOKEN(3 %STRING )
&TYPE &TOKEN(3:5 %STRING )
&TYPE &TOKEN(8: %STRING )
```

will result in the following being written to the terminal: K, 11, C, C D E, and H I J K.

IX.D.5 The &GET String Function

When writing macros for MOSES, it is often desirable to prompt the user for information. This can be accomplished by using the string function:

&GET(WAY, DATA **)**

This string function returns a string, and here WAY is the manner in which the response is obtained and DATA is the other information necessary to make the request. WAY must be either **FILE**, **RESPONSE**, **YES/NO**, **PICK**, **N_PICK**, **GET_LIST**, or **GET_NAME** and DATA depends upon WAY. The details of how the information is requested depend on the current user interface. For a terminal interface, it is all done with text and typing. With a window interface, different dialog boxes appear.

The first three types of WAY request for a file name, a general string input, and a YES or NO response respectively. Here, DATA is simply a description of the data requested. For example,

&INSERT &GET(FILE Select your MOSES custom file)

and

&INSERT &GET(RESPONSE Input your MOSES custom file name)

will both return a string, but the first will pop up a file selection box and allow the user to browse and click on the desired file. The second will pop up an input box and the string returned may or may not be a valid file name. YES/NO places DATA in a box with YES and NO buttons and returns either YES or NO depending on which button is pushed.

The last four types of WAY pop up a list and the user is allowed to pick from the list. For PICK the form of DATA is:

MAX_PICK, TITLE, ITEM(1), ITEM(2), ...

Here MAX_PICK is the maximum number of items which can be chosen, TITLE is the title of the dialog box, and ITEM(i) are the items which can be chosen. If more than one item is chosen the string will be composed of the name of each item chosen.

The type of WAY of N_PICK stands for "name pick" and here the form of DATA is:

MAX_TO_PICK, TYPE_OF_NAME, TITLE

Here TYPE_OF_NAME is a valid value of the &NAMES command (issue the command &NAME NAMES to see what is available or see the documentation for &NAMES), TITLE is again the title of the pick box, and MAX_TO_PICK is the maximum number of items which may be chosen. With this type, the list depends on the current state of MOSES, in that the list is obtained from the current database. This removes the macro writer from having to know the state. As an example, consider

&REP_SELECT –BODY &GET(N_PICK 1 BODIES 'Select A Body')

which will present the user with a list of the bodies defined and ask him to select one.

The last two types of WAY GET_LIST and GET_NAME have a form of DATA exactly the same as PICK and N_PICK respectively. The difference is that with these types, a combination EDIT/PICK box is popped up. Here one can type data into the edit box or select from the available list, or do both.

IX.E Getting User Input

On can use the **&GET** string function to simply get a response from the user, but for many situations this leads to a boring, modal dialogs. A more flexible method is using **&STRING** macros and WIZARDS. A wizard is a set of commands which are defined inside a &STRING macro and A **&STRING** macro is the same as an ordinary macro *except* that it can contain only internal commands, and it can be executed only with an **&E_STRING** command. Let us look at an example, suppose that one wanted to have a command which prompts for the units to use. This can be accomplished with

```
&STRING D2M
   &DIMEN –DIMEN &GET(PICK 1 'PICK UNITS'    \
                        'Feet,Kips     ' \
                        'Feet,S–tons   ' \
                        'Feet,L–tons   ' \
                        'Meters,Tonnes' \
                        'Meters,KN     ' )
   END
&ENDSTRING
&E_STRING D2M
```

Notice that there is an "END" command at the bottom of the definition of the &STRING macro D2M. This command signals the end of the execution of the macro. At first glance this looks like a concept which is of no value of all, but this is not true. First, as we will see later, &STRING macros can be associated with buttons on the menu bar. Also, WIZARDS must be defined with a &STRING macro. For example, the above could be accomplished with a WIZARD as:

```
&STRING D2M
WIZARD Dimensions –COMMAND  &DIMENSION –SIZE 450 410
   TAB_ADD To Use
      WIDGET_ADD YES–NO –REMEMBER "Remember Previous"
      WIDGET_ADD YES–NO –SAVE     "Save Current"
      WIDGET_ADD RADIO  –DIMEN    "Settings"          \
                          "Feet, Kips     "  \
                          "Feet, L–tons   "  \
                          "Feet, S–tons   "  \
                          "Meters, Tonnes "  \
                          "Meters, KN     "
END
&ENDSTRING
&E_STRING D2M
```

Basically what happens here is the the wizard will build a command from the user's actions. The beginning of the command will be &DIMENSION and this is defined with the –COMMAND option. This wizard has one tab labeled "To Use" (all wizards must have a tab) and three things one can choose: two check boxes and one pick from a set of "radio buttons". A WIDGET command defines a single widget which will be placed in the tab and the first token after WIDGET defines the type of widget. The second token defines a prefix which will be added before the main data selected by the widget. The fourth token is a title The YES–NO widgets select whether of not the prefix is added to the command and the RADIO widget will add on of the prefix –DIMEN and the text selected.

Now, let's be more precise. The command

WIZARD Main Title –OPTIONS

instructs MOSES to enter the Wizard Menu and the options available here are:

–**COMMAND** First Part of Command
–**SIZE** SHOR, SVER, NCOL
–**BUTTON_TEXT** Text On Button

Here, the –**COMMAND** option was discussed above, and the –**SIZE** allows the user to specify the horizontal and vertical size of the wizard window in pixels, and the number of columns. Every wizard has a button that the user must push to exit the wizard and execute the command. The –**BUTTON_TEXT** option defines the text on the button. If this option is omitted, "OK" will be used.

As mentioned above, every wizard must have at least on tab, and it can have many of them. Each one is defined with the command

TAB_ADD STRING

where "STRING" defines the text which will be painted on the top of the tab. Once a tab has been defined, all text and widgets defined will be added to the tab until a new **TAB_ADD** command is issued. The command

TEXT_ADD TEXT

will simply add the text "TEXT" to the current tab and the current location.

WIDGET_ADD WIDGET_TYPE, W_PREFIX, W_DESC, W_LIST, –OPTIONS

and the available options are

–**INITIALIZE** W_INITIAL_VALUE
–**SUFFIX** W_SUFFIX
–**L_DESCRIPTION** L_DESCRIPTION

–**ACTIVATE** KEY_TO

Here, WIDGET_TYPE defines the type of widget which will be created and it must be chosen from either **YES–NO**, **BOX**, **RADIO**, **SEL_ONE**, or **SEL_MULTIPLE**. The YES–NO widget has a check box. If it is checked, then the prefix will be emitted, if it is not checked, then the prefix will not be emitted. The BOX widget has an input box in which the user can put input. If he enters data, then it and the prefix will be omitted. The RADIO widget has a list with a circle. If one clicks on the circle, the the text shown will be emitted with the prefix. Only a single value can be selected. The SEL_ONE and SEL_MULTIPLE widgets are drop down selection boxed. Only a single item can be selected with a SEL_ONE widget, but multiple ones can be selected with SEL_MULTIPLE. W_PREFIX defines the prefix which will be emitted when the user selects the widget. It can be blank. W_DESC is the brief text which will be placed to the left of the selection part of the widget, and W_LIST is a list of things which can be selected. The text defined here is what will be emitted after the prefix. The order in which widget results are emitted is the order in which they were input. Thus, if you use things without a prefix, they probably should be defined first.

The –**INITIALIZE** and –**SUFFIX** options defines initial values and suffixes for the widget. If an initial value is specified, then this will be the value displayed in the box as if the user input it. A suffix will not normally be needed, but if it is specified, then what is emitted by this widget will be the concatenation of the prefix, the value, and the suffix.

Ideally the words in the list should suffice to tell the user what he is selecting. Sometime this is not the case and you need to add a –**L_DESCRIPTION** option which defines a "long description. This description is written in a pop up window when on "mouses over" the description.

The –**ACTIVATE** option is complicated. When this option is used, the widget is initially hidden. Here KEY_TO is an option on the previous widget and when the previous widget selects KEY_TO, the hidden widget is shown giving choices which only make sense for the chosen value. For example:

```
WIDGET BOX –DEFLECTION "Show Deflected Shape"   YES NO
WIDGET BOX –DEFLECTION "Deflection Multiplier" –ACTIVATE YES
```

Here, the second widget is hidden unless YES is selected by the first one.

IX.F Programming the Tool Bar

The MOSES Tool Bar is a set of buttons at the top of the display area which, when pushed, drop down another set of buttons. Each of these buttons represent either another drop down set of buttons, or a command. The entire system is user programmable with a command and "STRING" macros. The basic command here is:

&MENU, ACTION, NAME, TITLE(1), RETURN(1), ...

Here, ACTION is the action which must be either **CREATE**, **DELETE**, or **ADD**, NAME is the name of the menu, RETURN is the "return string". Of course, the last two items are required for each line in the menu.

The NAMEs associated with menus are up to the user except the one **T_BAR** which is the name of the Tool Bar itself. Normally the only action one needs is CREATE, but with T_BAR, one can ADD a menu to it, or DELETE one he has added. The TITLE(i) are the strings which are printed on the buttons. For the Tool Bar itself, these are best left short as the real estate is limited. Remember here if you have spaces in any thing, you need to enclose them in either " or ' pairs.

Now, RETURN(i) can be either: the name of another menu, the name of a &STRING macro, or a true return string. Let us look at an example. Suppose that one wanted to add a button to the Tool Bar to set the dimensions to meters. This can be accomplished with

```
&MENU, ADD , T_BAR DIM2M '&DIMEN –D METERS TONNES'
```

Now, when you push this button, the dimensions will be changed to meters and tonnes. Suppose, however, that you sometimes wish to use kilo–newtons? To allow for this, you must interact with the user and a &STRING macro is required. The following:

```
&STRING D2M
  &DIMEN –D METERS &GET(PICK 1 'FORCE UNIT' TONNES KN)
  END
&ENDSTRING
&MENU ADD T_BAR DIM2M D2M
```

Now, when one pushes the DIM2M button, it will execute the &STRING macro D2M. In here, the user is asked to choose the force unit he wants to use.

The entire Tool Bar structure used in MOSES can be investigated in the file /ultra/data/progm/moses.mac.

IX.G Using Files

Normally the user does not have to worry about the files that MOSES is using. In some circumstances, however, the user wants to change the file associated with a given type or to write information to a file of his choice. These things can be accomplished with the string function

&F_READ(TYPE VAR)

and command

&FILE, ACTION, DATA, –OPTIONS

The string function **&F_READ** is used to read a file and the command **&FILE** is used for a variety of tasks. When reading or writing, files are referred by type and TYPE is this type. A TYPE is associated with a file when the file is opened. The string function will read a record from the file and store it in the variable VAR. The value returned by the function is .TRUE. if there is no more data to read and .FALSE. if there is more to read. To check to see if a file exists, you can use the string function:

&INFO(FILE_EXISTS filename)

which returns **TRUE** if the file exists or **FALSE** if it does not.

With **&FILE** the DATA and –OPTIONS depend on the value of ACTION, and the valid values of ACTION are: **MKDIR**, **RM**, **MV**, **CP**, **USE**, **ANS_DIRECTORY**, **OPEN**, **CLOSE**, and **WRITE**. The first four ACTIONS do not have any options:

&FILE MKDIR, A
&FILE RM, A
&FILE MV, A, B
&FILE CP, A, B

where the MKDIR ACTION creates the directory A, the RM ACTION deletes the file A, the MV ACTION renames file A to B, and the CP ACTION copies file A to B.

The **USE** action is used to alter the default association of files and file TYPES. The syntax here is:

&FILE, **USE**, TYPE, FILE_NAME

and TYPE is the type of file which will be altered. It is a either a user defined file type or type "PREFIX" of a channel. Also, FILE_NAME is the new file name which will be associated with the file type. When this option is exercised, the old file will be closed and the new one used until further notice.

The **ANS_DIRECTORY** action is used to alter the default association of files and file TYPES. The syntax here is:

&FILE, **ANS_DIRECTORY**, DIR

where DIR is a directory in which the "answers" will be stored.

Before one can write to a file, it must be opened with the command:

&FILE, **OPEN**, **–TYPE**, TYPE, **–NAME**, FILE_NAME

Here, TYPE is a handle you can associate with the file and FILE_NAME is the name of the file. The handle can be any name of up to eight characters and is used to read, write and close the file. Once a file has been opened, you can write to it with the command:

&FILE, **WRITE**, TYPE, STRING

which writes the line STRING to the file with type of TYPE. After writing a file, you should close it with

&FILE, **CLOSE**, TYPE, –OPTIONS

If you specify **–DELETE** then the file will be deleted.

You cannot read and write to a file at the same time. Instead it must be written, closed, reopened, and then read. As an example, consider

```
&FILE OPEN –TYPE COW –NAME MILK_COWS
&FILE WRITE COW Brown
&FILE WRITE COW Black and White
&FILE WRITE COW Holstein
&FILE CLOSE COW
&FILE OPEN –TYPE COW –NAME MILK_COWS
&LOOP
   &EXIT &F_READ(COW VAR)
   &TYPE %VAR
&ENDLOOP
&FILE CLOSE COW
```

which writes three lines to a file named MILK_COWS and then reads them.

IX.H Functions

While one can accomplish almost anything with the tools that have been discussed previously, one finds that if this is all he has, then things get convoluted quickly. MOSES has a concept called "functions" which can be used to organize data more effectively. This concept is based on the true mathematical definition of a function. Suppose that A and B are two sets, and for each element of A there is associated an element of B. This association of elements is called a function from A to B. The set A is called the domain of the function, the set B is called the range of the function, and the elements in B are called the values of the function. What we are going to discuss here are functions who's domain is either the set of positive integers or a "variable" and the range of these functions is a set of strings.

In this sense, a "variable" is not a simple global or local variable. It is, in fact, a set of names, and a name is a string of up to eight characters. The variable itself has a name which is a string of up to twenty four characters. The name must be of the form

```
<DBFILE>VAR_NAME
```

Here DBFILE is the name of the database file where the variable will be stored. Now,we could spend a good bit of time talking about this, but lets just say that there are two that are always available: MACDBF and SYSDBF, so you should use one of these (preferably MACDBF). VAR_NAME is simply the name of the variable. To create a variable, one uses the internal command:

&VARIABLE, VAR_NAME, –ADD NAME

Which adds the name "NAME" to the variable VAR_NAME. Now, once a variable has names, you can find out what they are with the string function

&VARIABLE(–WHILE VAR_NAME SEL WHAT)

extracts all the names in "VAR_NAME" that match SEL and sets the (normal global or local) variable WHAT to the value of the name. It also returns a string .FALSE. if there are more names to be output or .TRUE. if it is finished.

Let's look at an example:

```
&VARIABLE <MACDBF>COWS –ADD JERSEY
&VARIABLE <MACDBF>COWS –ADD HOLSTEIN
&VARIABLE <MACDBF>COWS –ADD ANGUS
```

Will build a variable <MACDBF>COWS which has three names in it: JERSEY, HOLSTEIN, and ANGUS. Here we will be using the command &LOOP without any entries. This advanced method of using &LOOP creates an infinite loop.

Now,

```
&LOCAL LIST  =
&LOOP
  &EXIT &VARIABLE(–WHILE  <MACDBF>COWS @  A_COW)
  &SET LIST   = %LIST %A_COW
&ENDLOOP
```

will produce the local variable "LIST" which contains the three names of cows. Finally, there is the string function

&NUMVAR(VAR_NAME)

that returns the number of names in the variable "VAR_NAME"; e.g.

```
&NUMVAR(<MACDBF>COWS)
```

will return 3.

Often one wishes to "while" items which are in the database. This can be accomplished with

```
&LOOP N ( &NAMES(NODES) )
&ENDLOOP
```

Unfortunately, for large models, this will produce a "buffer overflow". When this happens you need to change the above to

```
&SET NOD_VAR = &NAMES(NODES –V)
&LOOP
  &EXIT &VARIABLE(–WHILE  %NODE_VAR    @  A_COW)
&ENDLOOP
```

The first line above uses the &NAMES string function and the option **–V** to get the names that MOSES uses to store the data which allows} you to use the MOSES data directly.

As mentioned above, "functions" are build on top of variables. Before one can add values to a function, he must first "define" it with the internal command:

&DEFINE, FUNNAM, –OPTIONS

Here, FUNNAM is the name of the function, and the available options are:

–DOMAIN, DOMNAM,

–MEMORY

Here FUNNAM is the name of the function and it follows the same rules as those for a variable name. DOMNAM define the domain of the function. If the option –DOMAIN is omitted, then the domain of the function will be the set of positive integers otherwise it is the variable DOMNAM. The option **–MEMORY** instructs MOSES that the primary residence for values of the function will be in memory rather than on a file. If omitted, the primary residence will be on the database file. Normally, memory resident functions are accessed more quickly, but not always.

Once a function has been defined you should define its values. This is accomplished with the internal command:

&FVPUT, FUNNAM, VAR, STRING

Here, FUNNAM is the name of the function, VAR is the value (name) in the variable DOMAIN associated with STRING. For example:

```
&DEFINE <MACDBF>COW_DATA –DOMAIN <MACDBF>COWS
&FVPUT <MACDBF>COW_DATA JERSEY Milk
&FVPUT <MACDBF>COW_DATA HOLSTEIN Milk
&FVPUT <MACDBF>COW_DATA ANGUS Beef
```

defines the function <MACDBF>COW_DATA and associates data for three types of cows. To get the values from a function, one uses the string function

&FVGET(FUNNAM, VAR)

The following

```
&LOOP
  &EXIT &VARIABLE(–WHILE  <MACDBF>COWS @  A_COW)
    &TYPE The %A_COW cow is a &FVGET(<MACDBF>COW_DATA
%A_COW).
&ENDLOOP
```

Will type a line defining the class for each type of cow.

Functions are similar to files. You open and close files but you &READY and &SUSPEND functions. The internal command

&READY, FUNNAM

makes the function FUNNAM available for extracting values. When you are through with a function, you can use the internal command:

&SUSPEND, FUNNAM, –OPTIONS

The options must be either: **–DISCARD**, **–SAVE**, or **–CLOSE**. If the function is discarded when it is suspended, all data for the function is deleted from memory. If the –SAVE option is used, all data for the function will be written to the data base file if it has changed and the function will remain active. If the –CLOSE option is used, then the function will first be saved and then discarded. In both the subroutine call and the internal command, FUNNAM can include wild characters. Thus, one can suspend more than one function with one command.

One thing to know is that at the end of an **INMODEL**, MOSES closes all function (yours included). Thus you may need to &READY you functions. Actually MOSES function are more general that what we have discussed, but this is all that is available from the command interface.

X. THE DISPOSITION MENU

At the conclusion of many commands, the user is placed into the "Disposition Menu". In this menu, data may be processed and written to either an output file, a post–processing file, the terminal, a graphics device, or to a global variable.

When dealing with data in the Disposition Menu, it is best to think of it as being a matrix. The columns of the matrix are called variables and the rows are called records. Each variable is identified by its column number or name , and one can obtain a list of the names of the variables and their column numbers by issuing the command,

VLIST.

Throughout this menu, one selects columns with column selectors. These selectors can be a single number, a colon separated pair of numbers, or a name selector; e.g. 5, 7:8, @x:@. The colon pair selects all columns from the first number to the second one. In many cases, only a limited number of columns can be selected. In this case, the first one selected will be used. For example, suppose that you can select six values and you use 1:12 for the column selector, then only 1:6 will be used.

Sometimes, one may wish to alter the values of the data available. This can be accomplished by using either of the commands:

C_SCALE, SCALE_F, CS(1), CS(2), ...
C_SHIFT, SHIRT_F, CS(1), CS(2), ...

The **C_SCALE** command defines a multiplier by which a variable will be scaled, and the **C_SHIFT** defines a constant which will be added to the variable. Once of these factors are established, they will stay in effect until it is changed with a new **C_SCALE** or **C_SHIFT** command, or until the Disposition Menu is exited.

X.A Reporting, Viewing and Storing Data

Three commands are available which allow one to select portions of the data and write it to a post–processing file, an output file, or to the screen.

There are several options which can be used on more than one command:

–**HARD**
–**BOTH**
–**HEADING**, HEAD
–**RECORD**, BEG_RNUM, END_RNUM
–**VALUES**, CV, VAL_MIN, VAL_MAX
–**MAG_USE**
–**FIGURES**, COL_SEL, RIGHT

They will be defined here and then listed for the commands for which they are applicable.

By default, the results for commands that produce reports (except for the REPORT command discussed above) is to write the results to the terminal. The –**HARD** option instructs MOSES to produce a report on the OUTPUT channel and the –**BOTH** option writes the results to both the OUTPUT channel and the terminal. When these reports are written, they have a single line generic heading. The –**HEADING** allows one to replace the generic heading with one you specify. You can specify as many of these options as you wish. The will appear on the page in the order you specify them.

The –**RECORD** and –**VALUES** options defines the records which will be considered. Here, a "RECORD" is simply a row of the matrix of data. With the –**RECORD** option the beginning and end record numbers are simply specified. With –**VALUES** the records considered are defined with the values of a column of data. Here CV is the column number for which the values will be obtained and VAL_MIN and VAL_MAX are two numbers (VAL_MIN is less than VAL_MAX). BEG_REC is then the largest record number which the values of column CV is less than or equal VAL_MIN and END_REC is the greatest record number where the value of this column is greater than VAL_MAX. If neither –**VALUES** nor –**RECORD** are specified, all records will be considered.

The –**MAG_USE** option instructs MOSES to add a second heading line based on the definition of magnitude defined with the –MAG_DEFINE option. The –**FIGURES** option offers a way to change the display of the numbers. It says to change the number of figures after the decimal point for columns selected by COL_SEL to be RIGHT figures. You can specify more that one –**FIGURES** option.

Perhaps the easiest command to explain is the **REPORT** command, which produces a formatted output file report. The format of this command is:

REPORT, DATA, –OPTIONS

The form of the **REPORT** command depends upon the original command that placed the user here. Often only **REPORT** is necessary. Some original commands allow for data and options to be specified on the **REPORT** command. These details will be discussed with the original command.

The next of these is:

VIEW, CS(1), CS(2), ... –OPTIONS

and the available options are:

–**HARD**
–**BOTH**
–**HEADING**, HEAD
–**RECORD**, BEG_RNUM, END_RNUM
–**VALUES**, CV, VAL_MIN, VAL_MAX
–**MAG_USE**

This command offers the user the opportunity to view the selected data at the terminal, or, if the **–HARD** option is specified, the results will be written to the output file. The data viewed is defined by the column selectors CS(1), CS(2),.. If no column selectors are specified, MOSES will prompt the user for all required data.

The **STORE** command is used to store the selected data in a "table". The result here is much the same as what you get in the **&TABLE** menu, i.e. either a CSV or HTML table. The format of the command is:

STORE, CS(1), CS(2), ... –OPTIONS

and the available options are:

–**HEADING**, HEAD
–**RECORD**, BEG_RNUM, END_RNUM
–**VALUES**, CV, VAL_MIN, VAL_MAX
–**MAG_USE**
–**TITLE**, NCOL(1), CT(1), ... NCOL(n), CT(n)
–**H_SKIP**, YES_NO
–**BOLD**, YES_NO
–**ROW_SHADE**, YES_NO
–**EXTR_SHADE**, COL_SEL(1), COL_SEL(2), ...

−**V_LINES**, COL_SEL(1), COL_SEL(2), ...

with the exception of −**RECORD** and −**MAG_USE** these option work exactly as documented with **&TABLE**.

X.B Adding Columns

Occasionally, it is desirable to create a new column of data based on the existing columns, or create a completely new column of data. This can be performed with the command:

ADD_COLUMN, NAME, –OPTIONS

where the available options are:

–**COLUMN**, C(1), C(2), ... C(n)
–**INPUT**, CS, X(1), Y(1), X(2), Y(2), ...
–**COMBINE**, CS(1), F(1), CS(2), F(2), ...
–**NORM**, CS, NCOL
–**RMS**, CS(1), CS(2), ...
–**POWER**, P_N, P_C, CS(1), F(1), CS(2), F(2), ...
–**DERIVATIVE**, CS(1), CS(2)
–**INTEGRAL**, CS(1), CS(2)
–**FILTER**, R_TYPE CS(1), CS(2), RL(1), RU(1), ... RL(n), RU(n)
–**SMOOTH**, CS, NL, NR, ORDER

The first two options create completely new columns. The **–COLUMN** option allows the user to simply input a column of data. Here, the number of values input must be the same as the number of rows of the existing columns. A good source of this data may be from SET_VARIABLE –COLUMN. On the other hand, the **–INPUT** option allows more flexibility in the number of points input. Here, CS is the column selector of an existing column which is "like" the X values. For instance, if the X values are times, then CS should be the column selector for event number. MOSES will fit a spline to the input data and then create the new column by interpolating a value from the spline for every value of CS(i).

The remainder of the options create a column based on the values of the existing ones. The **–COMBINE** option will combine two or more columns of data, using the factors specified. For instance,

ADD_COLUMN NEW –COMBINE 1 1 2 –1

will make the new column the difference of columns of 1 and 2. Here, CS selects the column, and F is the combine factor. To find the norm of two or more columns, the **–NORM** option is used. CS again selects the column, and NCOL is the number of columns to consider, including the first one. The command:

ADD_COLUMN NEW –NORM 3 3

will find the norm of columns 3, 4 and 5. The **–RMS** option is similar to **–NORM**, except that here, one specified the columns to be combined directly. factor. The **–POWER** option is a generalization of the above. Here, P_C is the power each column is raised before summing and P_N is the power to which the sum will be raised; e.g.

```
ADD_COLUMN NEW –POWER .5 2 3 1 4 1 5 1
```

Gives the same result as the –NORM example and

```
ADD_COLUMN NEW –POWER 1 1 3 3
```

Gives the same result as the –COMBINE example.

The **–DERIVATIVE** and **–INTEGRAL** adds new columns which are the derivative or integral of column CS(2) with respect to column CS(1), and the **–LN** and **–EXP** options add columns which are the natural log and the exponential of the original column.

The **–FILTER** option filters a column of temporal data using a Fourier Transform. Here, R_TYPE defines the type of ranges to be input, **FREQUENCY** for angular frequency, or **PERIOD** for periods. MOSES will then compute the Fourier Transform of the function with domain in column CS(1) and range in column CS(2). After the transform, all values within any of the ranges defined will be discarded and an inverse transform will be computed for the new column. For example,

```
ADD_COLUMN NEW –FILTER PERIOD 1 4 0 30
```

will produce a new column named NEW from the old curve defined by columns 1 and 4. Here, column 1 is TIME or EVENT. All of the contribution from periods between 0 and 30 will be discarded to form the new function.

The **–SMOOTH** option creates a new column by smoothing an old one, but here, a Savitzky–Golay filter accomplishes the smoothing. CS defines the column to be smoothed, and NL, NR, and M define the smoother. NL is the number of points to the left, NR is the number of points to the right and ORDER is the order of polynomial used in constructing the filter. The defaults are 8 for NL and NR and 4 for ORDER. Since this algorithm does not use Fourier analysis, it can be used to smooth spectral or FFT data.

X.C Recasting Data

There are three *special* commands in this menu: **SPECTRUM**, **FFT** and **CULL**. All of these commands take a subset of the original data and transform it into a new set of data to be disposed. This has the interesting effect of one being in the Disposition Menu from the Disposition Menu. The first time and **END** is encountered, one leaves the latest Disposition Menu, but is still in the Disposition Menu. Now, however, the original data is again available.

There are several options which can be used on more than one command:

–**RECORD**, BEG_RNUM, END_RNUM
–**VALUES**, CV, VAL_MIN, VAL_MAX

The –**RECORD** and –**VALUES** options defines the records which will be considered. Here, a "RECORD" is simply a row of the matrix of data. With the –**RECORD** option the beginning and end record numbers are simply specified. With –**VALUES** the records considered are defined with the values of a column of data. Here CV is the column number for which the values will be obtained and VAL_MIN and VAL_MAX are two numbers (VAL_MIN is less than VAL_MAX). BEG_REC is then the largest record number which the values of column CV is less than or equal VAL_MIN and END_REC is the greatest record number where the value of this column is greater than VAL_MAX. If neither –**VALUES** nor –**RECORD** are specified, all records will be considered.

The form of the first of these commands is:

SPECTRUM, CS(1), CS(2), ... –OPTIONS

and the available options are:

–**RECORD**, BEG_RNUM, END_RNUM
–**VALUES**, CV, VAL_MIN, VAL_MAX

When issued, MOSES will compute a spectrum for the columns selected by CS(2), ... CS(N), assuming that CS(1) is the independent variable. The form of the second one is:

FFT, CS(1), CS(2), ... –OPTIONS

and the available options are:

–**RECORD**, BEG_RNUM, END_RNUM
–**VALUES**, CV, VAL_MIN, VAL_MAX

These two commands are very similar, the only difference is that with **FFT**, a

Fourier transform of the data is produced instead of a spectrum. The last of these special commands is:

CULL, CS, EXL(1), EXU(1), ... EXL(n), EXU(n)

This command creates a new set of data from the original by excluding specified parts. Here, CS defines a column, and other values define ranges of values of the selected column which will be excluded in the new data. For example, consider:

CULL, 1, 0 50 200 300

The new data will contain all of the original except those rows where column 1 had values either between 0 and 50 or 200 and 300. One obvious use for this command is to extract starting transients from a time sample. Another use is to partition frequency domain data to compute independent statistics on each part.

X.D Extremes and Statistics

There are several options which can be used on more than one command:

–HARD
–BOTH
–HEADING, HEAD
–RECORD, BEG_RNUM, END_RNUM
–VALUES, CV, VAL_MIN, VAL_MAX
–MAG_USE

They will be defined here and then listed for the commands for which they are applicable.

By default, the results for commands that produce reports (except for the REPORT command discussed above) is to write the results to the terminal. The **–HARD** option instructs MOSES to produce a report on the OUTPUT channel and the **–BOTH** option writes the results to both the OUTPUT channel and the terminal. When these reports are written, they have a single line generic heading. The **–HEADING** allows one to replace the generic heading with one you specify. You can specify as many of these options as you wish. The will appear on the page in the order you specify them.

The **–RECORD** and **–VALUES** options defines the records which will be considered. Here, a "RECORD" is simply a row of the matrix of data. With the **–RECORD** option the beginning and end record numbers are simply specified. With **–VALUES** the records considered are defined with the values of a column of data. Here CV is the column number for which the values will be obtained and VAL_MIN and VAL_MAX are two numbers (VAL_MIN is less than VAL_MAX). BEG_REC is then the largest record number which the values of column CV is less than or equal VAL_MIN and END_REC is the greatest record number where the value of this column is greater than VAL_MAX. If neither **–VALUES** nor **–RECORD** are specified, all records will be considered.

The **–MAG_USE** option instructs MOSES to add a second heading line based on the definition of magnitude defined with the –MAG_DEFINE option.

The **EXTREME** command offers the user the opportunity of obtaining a report on the extremes of the data. The form of this command is:

EXTREME, CS(1), CS(2), ... –OPTIONS

and the available options are:

–HARD
–BOTH

–**HEADING**, "HEAD(1)", "HEAD(2)"
–**RECORD**, BEG_RNUM, END_RNUM
–**VALUES**, CV, VAL_MIN, VAL_MAX
–**MAG_USE**

With this command, one will obtain a report of the extremes of the data selected. Here, the first value entered will become the "independent" variable, and the remainder the dependent ones. MOSES will search through the results from BEG_RNUM to END_RNUM to find the minimum and maximum value of each type of data selected. It will then issue a report for each value of the independent variable at which an extreme occurred. This report will contain the values of all of the variables and a remark as to which variables have suffered an extreme. The report will be written to the terminal unless the **–HARD** option was used, in which case it will be written to the output file.

The **STATISTIC** command generates a report on the statistics of the data. It produces statistics for the results from BEG_RNUM to END_RNUM for each type of data selected. The specific form of this command is:

STATISTIC, CS(1), CS(2), ... –OPTIONS

and the available options are:

–**HARD**
–**BOTH**
–**RECORD**, BEG_RNUM, END_RNUM
–**VALUES**, CV, VAL_MIN, VAL_MAX
–**MAG_USE**
–**HEADING**, "HEAD(1)", "HEAD(2)"
–**TYPE**, STYPE
–**EXTREMES**, TIME, DEVIATION, MULTIPLIER

Where the report is written depends on the use of the **–HARD** and **–BOTH** options. Here, CS(1) is the independent variable against which the statistics will be computed. Normally, it is "event" so that the remaining columns of data can be considered to be time samples. If this is the case, MOSES will compute the following quantities:

Mean
Variance
RMS
Std. Deviation
Skewness
Kurtosis
Av of 1/3 Highest
Av of 1/3 Lowest

Av of 1/100 Highest
Av of 1/100 Lowest
Av of 1/1000 Highest
Av of 1/1000 Lowest
Maximum
Minimum
Pred. Max
Pred. Min
Av of 1/3 Highest–Mean
Av of 1/3 Lowest–Mean
Av of 1/100 Highest–Mean
Av of 1/100 Lowest–Mean
Av of 1/1000 Highest–Mean
Av of 1/1000 Lowest–Mean
Maximum – Mean
Minimum – Mean
Pred. Max – Mean
Pred. Min – Mean

of the variables selected. These quantities are calculated for the It will also compute averages for the peaks encountered. *Notice* these peaks are computed from the samples themselves and not by assuming any type of probability distribution. Extreme values of the maximum and minimum are also predicted. This prediction is controlled by the **–EXTREMES** option. Here TIME is the time in seconds for the extreme. If, for example, TIME is 3600, then the predicted value will be the probable maximum in one hour. The default is three hours.

In general, the predicted extreme is of the form

PE = MEAN +– DEVIAT * FACTOR

here MEAN is the mean and the plus is used for the maximum and the minus for the minimum. Traditionally, the standard deviation is used for DEVIAT and FACTOR is given by

FACTOR = sqrt { 2 Log (r * Np) }

where Np is the number of peaks in the sample, and r is the ratio of the length of the sample to TIME. The values of DEVIATION and MULTIPLIER can be used to change this behavior. In particular, the value of DEVIATION is used to change DEVIAT. Here, a value of **STANDARD** will use the standard deviation while a value of **PEAKS** will use the largest peak and smallest peak values minus the mean. PEAKS is the default and normally gives better predictions than the traditional method. The final value, MULTIPLIER can be either **GAUSSIAN** or **WINTERSTEIN**. GAUSSIAN is the default. If WINTERSTEIN is used, then FACTOR will be computed according to the paper "Nonlinear Vibration Models for Extremes and Fatigue" by S.R. Winterstein. If one is using both PEAKS and GAUSSIAN, then factor is different than that given above. Here it is

FACTOR = sqrt { 2 Log (r * Np) } / sqrt { 2 Log (Np) }

In other words, here the given peak is scaled up based on the ratio of the predicted extreme in three hours to that predicted by the current sample.

In some cases, however, the independent variable is not time but frequency, and the other columns are either Fourier Coefficients or spectral ordinates. For frequency data different statistics are computed. If the frequency data resulted from either a FFT or SPECTRUM command in the Disposition Menu, then MOSES automatically knows how to treat the data. If, however, the original data was frequency type, then one must use the **–TYPE** option to define how to treat it. STYPE can be either **Fourier** or **SPECTRUM**. With frequency data, the report consists of:

0th Moment
1st Moment
2nd Moment
3rd Moment
4th Moment
Root Mean Square
Significant
Ave of 1/10 Peaks
Ave of 1/100 Peaks
Ave of 1/1000 Peaks
3 Hour Max.
TP Peak Period
TV Visual Period
TZ Zero Up–Crossing
TC Crest Period

moments of the spectrum, averages of the peaks, and several periods of the data. Here, in contrast to the time statistics, the statistics are derived assuming a Raleigh distribution.

X.E Plotting

In MOSES, a graph consists of one or more dependent variables plotted on one or two ordinate (vertical) axes, all against an independent variable which is plotted using the abscissa (horizontal) axis. Two ordinate axes (a left and an optional right axis) are available in order to graph variables which differ greatly in magnitude, but which the user wishes to present on the same graph. If two or more dependent variables are graphed, MOSES differentiates the curves by adding symbols to several points in each curve, and a legend is placed on the graph which defines these symbols. In addition, the title for the abscissa is set to the name of the independent variable, and for the left and right axes, each title is set to the name of the first dependent variable defined for that axis.

To produce a graph, one issues the command:

> **PLOT**, IVAR, L(1), L(2), .., –OPTIONS

and the available options are:

> –**RAX**, R(1), R(2), ..
> –**LIMITS**, X(1), X(2), ..
> –**SMOOTH**, SM_TOL
> –**ADD**, NUM_ADD
> –**POINTS** LEGEND, X(1), Y(1), X(2), Y(2), ..
> –**CLEAN_LINE**
> –**CROP_FOR_LEGEND**
> –**NO_EDIT**
> –**T_MAIN**, TITLE
> –**T_SUB**, TITLE
> –**T_X**, TITLE
> –**T_LEFT**, TITLE
> –**T_RIGHT**, TITLE
> –**LEGEND**, NUMBER, TITLE

Here, IVAR is the column selector of the independent variable, and L(1), L(2), ..., L(n) are the column selectors of the dependent variables to be plotted using the left axis. If two ordinate axes are needed then the variables to be plotted on the right axis are added by using the option: **–RAX**. The **–LIMITS** option is used to limit the range of the X axis of a plot. Here, X1 and X2 define the allowable range of the independent variable.

Normally, each curve is drawn by merely connecting the data points with a series of straight lines. If, however, one uses **–SMOOTH**, the program will fit a set of cubic splines to the original data so that the root mean square of the fit is less than SM_TOL. Additionally, one can use the option **–ADD** so that NUM_ADD points are added between the original points, and the result plotted. Notice that a quite small value of SM_TOL (1E–7) with NUM_ADD = 0 will essentially reproduce the original plot, while the same value of SM_TOL with NUM_ADD greater than zero will produce a graph which passes through the original points with the additional ones added using the spline fit.

The **–POINTS** option adds the points specified as centered symbols to the plot. No lines will be drawn connecting these points. Here, LEGEND is the name given for the legend of the new points. The X values will be scaled the same as the independent variable, and the Y values will be scaled with the left axis. This option is useful for adding information to the plot from external sources, such as model test data.

The option **–CLEAN_LINE** instructs MOSES to use color only to distinguish between curves on the plot. Normally, the legend box is drawn in the domain of the plot, and it is possible that some of the curves will be drawn in the legend box. The **–CROP_FOR_LEGEND** option tells MOSES to change the behavior so that a curve will never enter into the legend box.

The behavior of MOSES after the **PLOT** command is issued depends on the options **–NO_EDIT**, **–T_MAIN**, **–T_SUB**, **–T_X**, **–T_LEFT**, **–T_RIGHT**, and **–LEGEND**. If none of these options were specified, MOSES will go into edit mode and will ask the user which legend (or title) he wishes changed. An empty line (a simple carriage return) will take MOSES out of edit mode. Note that each legend is limited to twenty characters, while each title may be up to seventy–two characters in length. With these options, TITLE is a string (probably delimited by ' to include blanks), and the option keyword defines where the title will be placed. For **–LEGEND**, NUMBER is the legend number where title will be placed.

After a graph has been made, the user may elect to make another graph with the same variables, but change some of the options. The **AGAIN** command is provided for this purpose.

AGAIN, –OPTIONS

Here, any of the options valid for the **PLOT** command may be specified.

After a graph has been made, a copy may be written to a file for later processing by using the SAVE_GRAPH command,

SAVE_GRAPH

The file used by this command is specified on the **–SECONDARY** option of the **&DEVICE** command.

MOSES will, in general, compute a scale for each axis of a graph so that the functions will "fill" up the page. In some circumstances, one may wish to establish the same scale for several different plots. This can be accomplished via the **RANGE** command, the form of which is:

RANGE, –OPTIONS

and the available options are:

–**X**, MIN_VALUE, MAX_VALUE
–**LEFT**, MIN_VALUE, MAX_VALUE
–**RIGHT**, MIN_VALUE, MAX_VALUE

If one of the options is issued and no data follows, then the axis specified by the keyword will be automatically scaled. To control the scaling of an axis, one should follow the axis keyword with two numbers: the minimum value for the axis and the maximum number for the axis. These extremes will then be used to establish the scale. Notice that when the automatic scaling is not in operation, it is possible that some portion of the curves may be off of the page.

X.F Getting Data

The final command in this menu is used to extract information from the database and place it in a "variable" where it is available to the advanced user. Its form is:

SET_VARIABLE, VAR_NAME, –OPTIONS

and the available options are:

–**RECORD**, BEG_RNUM, END_RNUM
–**VALUES**, CV, VAL_MIN, VAL_MAX
–**NUM_COLUMNS**
–**NUM_ROWS**
–**NAMES**, CS(1), CS(2), ...
–**COLUMN** CS(1), CS(2), ...
–**STATISTICS**, CS(1), CS(2)
–**MINIMUM**, CS_PUT, CS_GET
–**MAXIMUM**, CS_PUT, CS_GET
–**SELECT**, CS_PUT, CS_GET, VAL(1), VAL(2), ...

The results of this command are stored as a string in the variable VAR_NAME. The first of theses options is different from the others in that it simply changes the records that will be searched by any *following* option. All of these others actually produce results. The simplest of the remainder of the options are –**NUM_COLUMNS** and –**NUM_ROWS** which simply writes a string defining the number of columns or rows of data which exist. The –**NAMES** option will write the variable names for the columns CS(1), CS(2), etc into the variable. The –**COLUMN** option simply copies the values of the columns CS(i) into the string. Here, the values of each column are copies for each record.

The –**STATISTICS** option operates in the same manner as the **STATISTICS** command. Here CS(1) is the column number of the independent variable, and CS(2) is the column number of the data for which statistics will be returned. This will result is numerous tokens being defined in VAR_NAME, one for each row obtained when the **STATISTICS** command is issued. Also the order of the tokens is the same as the order of the row.

The remainder of the options deal with two columns of data, and write numbers into the global variable based on the values of the two columns. The first column specified is called the "put" column and the second the "get" column. The –**MINIMUM** and –**MAXIMUM** options will write the values of the "put" column into the global variable for the record at which the "get" column is either a minimum or maximum.

The –**SELECT** option operates in a similar manner, except that here the values written into the global variable will be the values of the "put" column which correspond to specified values of the "get" column. For example, suppose that the "get" column had values of 1 2 3 4 5 and 6, and the "put" column had values of 10 11 12 13 14 15 and 16. If one now used the **–SELECT** option with values of 1.5 and 5.5, then the strings stored in the global variable would be 10.5 and 15.5.

XI. REPORT CONTROL & INFORMATION

When obtaining written information about the system, the user has control over the extent of information provided. This control is exercised by using a set of selectors which can be defined as options on the report generating command, or on an internal command, **&REP_SELECT**. Once one of these options has been specified, it *remains* in effect until it is *explicitly* redefined on either an **&REP_SELECT** command or on a reporting command.

The form of the command is:

&REP_SELECT, –OPTIONS

and the available options are:

–**BODY**, :BODY_SEL
–**PART**, :PART_SEL
–**LGROUP**, :LG_SEL
–**COMPARTMENT**, :CMP_SEL
–**CLASS**, :CLS_SEL
–**NODE**, :NODE_SEL(1), :NODE_SEL(2), :NODE_SEL(3), :NODE_SEL(4)
–**TAG**, :TAG_SEL
–**ELEMENT**, :ELE_SEL
–**PANEL**, :PAN_SEL
–**MAP**, :MAP_SEL
–**DATA**, :DATA_SEL
–**SELALL**

Here, :BODY_SEL, :PART_SEL, :LG_SEL, :CMP_SEL, :CLASS_SEL, :NODE_SEL(i), :ELE_SEL, :MAP_SEL, :PAN_SEL, :DATA_SEL and :TAG_SEL define the manner in which quantities are selected. They can be either selection criteria, simple names, or names containing wild characters. If one of these options is not selected, then the selector defined by the last use of the corresponding option will govern.

Node selection is more complicated than that of class or element. Suppose that an element has two vertices. Then for the element to be selected, both nodes must match either :NODE_SEL(1) or :NODE_SEL(2). Thus, to select all elements connected to a given node, one should set :NODE_SEL(1) to be the specified node and the remaining node selectors to be @. Alternately, to select a single element, one can set :NODE_SEL(1) and :NODE_SEL(2) to be the end nodes of the element.

In general, the selectors define the things which will be reported. However, when one asks for a report of elements, the report will be restricted as follows:

- The class name of the element must match the class selector defined by the last –**CLASS** option,

- The nodes which form the vertices of the element all must match a node selector defined by the last –**NODE** option,
- The element tag must match the tag selector defined by the last –**TAG** option, and
- The element name must match the element selector defined by the last –**ELEMENT** option.

The –**TAG** option restricts only on the class of item being reported; e.g. if one is reporting properties of elements, then only elements whose "tag" matches :TAG_SEL will be reported or if one is reporting classes then only those tag matches :TAG_SEL will be reported.

Occasionally, one does not want the last selection criteria to remain active. This is achieved with the –**SELALL** option, which resets all previous selection criteria, and selects everything in the database.

XI.A Obtaining the Names of Quantities

Often it is useful to be able to obtain a list of names which are available in the database. In MOSES, one can obtain such a list by issuing the command:

&NAMES, NAME, –OPTIONS

where the available options are the options of the **&REP_SELECT** command and **–HARD**, or using the string function

&NAMES(NAME, :SELECTOR**)**

The value of NAME defines the category for which names will be listed and it must be one of:

BODIES	Active Bodies
CATEGORIES	Categories
CLASSES	Classes
COMPARTMENTS	Compartments
CONNECTORS	Connectors
DMARKS	Draft Marks
DURATIONS	Durations
ELEMENTS	Active Elements
ENVIRONMENTS	Environments
GRIDS	Grids
HOLES	Holes
INTEREST	Interest Points
IN_BODIES	Inactive Bodies
IN_ELEMENTS	Inactive Elements
IN_PARTS	Inactive Parts
I_SPECTRA	Input Spectra
LOADGROUPS	Load Groups
LSETS	Load Sets
MACROS	Macros
MAPS	Load Maps
M_GROWTH	Marine Growth
NAMES	Names
NODES	Nodes
PANELS	Panels
PARTS	Active Parts
PIECES	Pieces
PI_VIEWS	Pi_views
POINTS	Points
PROCESSES	Processes
PROFILES	Profiles
SELECTORS	Selection Criteria
SHAPES	Shapes
SN	SN Curves
SOILS	Soils
TVARS	Time Variations
VARIABLES	Global Variables
S_CASES	Structural Solution Cases
R_CASES	Structural Report Cases
NG_S_CASES	Non Converged Structural Solution Cases

When the **&NAMES** command is issued, the resulting list will be displayed at the terminal unless the option **–HARD** is specified. Also, when using the string function the results will be limited to only those names which match :SELECTOR.

XI.B Obtaining the Status of the System

The **&STATUS** command can be used to obtain a report on various quantities at the *current event*. In some menus, the current event is not completely defined until the menu is exited, so **&STATUS** is not always available. In general, these reports can be divided into ten categories: General Information, System Information, Connector Information, Compartment Information, Compartment Hole Information, Load Group Information, Categories and Load Sets, Element Information, Map Information, and Structural Solution Information. The form of this command is:

&STATUS, REP_TYPE, :SELE, –OPTIONS

Here, REP_TYPE specifies the type of information one wishes to report, and it must be either:

- For General Information: **NOTE**, **NAMES**, **CURVES**, **T_CONVOLUTION**, **F_CONVOLUTION**, **PARAMETER**, **SIZE**, **PROCESS** or **SN**
- For System Information: **B_W**, **FORCE**, **CONFIGURATION**, **BODY**, **DRAFT**, **B_MATRIX**, **A_MATRIX**, **D_MATRIX** or **MOTION**
- For Environmental Information: **ENVIRONMENT**, **SEA**, **SEA_SPECTRUM**, **SEA_TSERIES**, or **WIND_TSERIES**,
- For Connector Information: **F_LWAY**, **G_LWAY**, **F_CONNECTOR**, **G_CONNECTOR**, **DG_CONNECTOR**, **S_ROD**, **F_ROD**, **CL_FLEX**, **SPREAD**, **LINES**, **PIPE**, **PILE**, **TIP–HOOK** or **ALIAS_NO**
- For Compartment Information: **PIECE**, **COMPARTMENT**, **CG_COMPARTMENT** or **S_COMPARTMENT**
- For Compartment Hole Information: **V_HOLE**, **P_HOLE**, **WT_DOWN** or **NWT_DOWN**.
- For Load Group Information: **M_LOADG** or **F_LOADG**
- For Categories and Load Sets: **M_LSET**, **CATEGORY**, **M_CATEGORY** or **D_CATEGORY**.
- For Element Information: **ELEMENT** or **F_ELEMENT**
- For Map Information: **MAP** or **N_MAP**
- For Structural Solution Information: **S_CASE**, **R_CASE** or **AMOD**

The amount of information obtained is controlled by the selection criteria, :SELE. The available options, in addition to those on the **&REP_SELECT** command are:

–HARD
–BRIEF
–PLOT
–FORCE, FORCE_NAME(1), ..., FORCE_NAME(n)

Here, FORCE_NAME(i) is a selector which selects forces from the list: **WEIGHT**,

CONTENTS, **BUOYANCY**, **WIND**, **WIND**, **V_DRAG**, **WAVE**, **R_DRAG**, **SLAM**, **CORIOLIS**, **W_DRIFT**, **DEFORMATION**, **EXTRA**, **APPLIED**, **INERTIA**, **A_INERTIA**, **C_INERTIA**, **FLEX_CONNECTORS**, **RIGID_CONNECTO** and **TOTAL**. The meaning of these forces can be found in the section on FORCES.

The reports one receives with this command are usually written to the terminal. If however, one specifies the **–HARD** option, they will be written to the output file and no information will be received at the terminal. The **–BRIEF** option limits the scope of a report. The precise effect of **–BRIEF** depends upon the report being generated. The **–PLOT** option can only be used for the **DG_CONNECTOR**, **S_ROD**, **F_ROD**, **SEA**, **SEA_SPECTRUM**, **SEA_TSERIES**, or **WIND_TSERIES** reports. When this option is used the user will be placed in the Disposition Menu so that he can plot the results.

The General Information reports are given via a REP_TYPE of **NAMES**, **NOTE**, **CURVES**, **T_CONVOLUTION**, **F_CONVOLUTION**, **PARAMETER**, **SIZE**, **PROCESS** or **SN**. The first of these will produce a list of things names which can be used with &NAMES and a description of them. The second of these produces a list of database names and the notes one has associated with them. For this type, one can use the option

–NAMES, :NODE_SEL

to select only names in selected categories. For example:

&STATUS NOTE –NAMES COMPARTMENT

will produce only a list of compartments and their notes. **CURVES**, **T_CONVOLUTION**, and **F_CONVOLUTION**, produce information about curves or convolutions. All three of these types honor the :SELE variable which defines the names for which a status will be produced. All three honor the **–PLOT** option so that the data results can be plotted. **T_CONVOLUTION**, will give the status of the time convolution for selected names and **F_CONVOLUTION** will produce information about the Fourier Transform of the time convolution. If :SELE is a body name, the results are for the convolution associated with that body. **PARAMETER** simply reports some of the values set with the **&PARAMETER** command. **SIZE** produces a report of the number of various things in the model. **PROCESS** produces the name of the current process and a list of all process names. A REP_TYPE of **SN** provides a report of the defined SN curves.

System Information status is obtained via a REP_TYPE of **B_W**, **FORCE**, **CONFIGURATION**, **BODY**, **DRAFT**, **B_MATRIX**, **A_MATRIX**, **D_MATRIX**, or **MOTION**. The first of these, **B_W**, contains the weights acting on each body, the buoyancy, the radii of gyration, and perhaps the metacentric heights. The last of these are reported only if they are meaningful; i.e. the weight is within one percent of the buoyancy. Detailed information about the forces on bodies can be obtained with **FORCE** which reports a breakdown of the forces acting on the body due to each class of environment and constraint. **CONFIGURATION** contains the location of each body in the system, and the total force acting on each body. **BODY** will produce a report of the body properties set with: the options –D_DMARK, –FM_MORISON, –SPE_MULTIPLIER, –FACT_CONVOLUTION, –PERI_USE, and –WAVE_RUNUP on the &DESCRIBE BODY command. It also lists the current Mean Drift and Pressure names associated with each body. **DRAFT** will produce a report of the draft readings at the defined draft marks. The **B_MATRIX**, **A_MATRIX**, and **D_MATRIX**, actions produce reports of the weight, apparent, and defined weight matrices respectively. The report **MOTION** produces the global location of the Interest Points and the "motion" of these points since the last time the **&DESCRIBE INTEREST** was issued.

Environmental Information status is obtained via a REP_TYPE of **ENVIRONMENT**, **SEA**, **SEA_SPECTRUM**, **SEA_TSERIES**, or **WIND_TSERIES**. All of these except the first two accept the **–PLOT** option so that the data results can be plotted. The first of these produces various information about the current environment. The type **SEA** gives statistics and maxima information for the sea and time of the current environment. This is quite useful in checking that the sea has a peak of the desired height within the time sample. In addition to the raw peaks found, the most probable peak for this number of cycles is reported, the ratio of the peak found to that predicted and the number of cycles normally required to produce a peak of this size are reported. The information available for plotting is the sea elevation as a function of time. The **SEA_SPECTURM** type reports (and allows for the plotting of) the frequency, period and spectral value of the sea spectrum. The value reported here is the sum of the values over all headings. The value of **SEA_TSERIES** reports information about the Fourier Coefficients that will be used to generate the sea in the time domain, and **WIND_TSERIES** reports similar information for the wind.

Connector Information reports are obtained via a REP_TYPE of **F_CONNECTOR**, **G_CONNECTOR**, **DG_CONNECTOR**, **S_ROD**, **F_ROD**, **CL_FLEX**, **F_LWAY**, **G_LWAY**, **SPREAD**, **LINES**, **PIPE**, **TIP–HOOK**, or **ALIAS_NO**. The scheme here is that things which begin with a F_ produce reports of forces, those with a G_ produce geometry, those with a S_ produce stresses, and those with a DG_ produce detailed geometry. What follows the _ defines the type of connector for which results will be reported: CONNECTOR – normal connectors, ROD – rod connectors, LWAY – launch way connectors. Thus **F_LWAY** and **F_CONNECTOR** produce reports of the forces which currently act in the launchways and connectors respectively, while **G_LWAY** and **G_CONNECTOR** produce reports on the geometries of the same quantities. The commands: **DG_CONNECTOR** **S_ROD**, and **F_ROD** for detailed geometry honor the value :SELE and the –**PLOT** option. Here, :SELE should select only a single connector. If more are selected, an error will be reported and only the first will be used.

The last of these reports do not follow the convention. A type of **SPREAD** provides a report of the flexible connector types **ROD**, **B_CAT**, **H_CAT**, **SL_ELEM**, and **TUG_BOAT**. This report includes a basic summary for a mooring spread, including connector forces and local and global headings of the connectors. The **LINES** type produces a report about the B_CAT connectors which include: the horizontal distance between the fairlead and anchor, the length of the first segment, the line on bottom, the tension and ratio at the top, and the horizontal and vertical pull on the anchor. The **DG_CONNECTOR** type produces a detailed report of the geometry of the connectors selected by :SELE. A type of **CL_FLEX** produces a report of the data for the classes of the flexible connectors, and a type of **TIP–HOOK** produces a report detailing the geometry of the boom tip and hook during an upending simulation. **PIPE** reports the active length, tension in the tensioner, and tensioner limits for a pipe assembly, **PILE** produces a summary of the soil and the multipliers associated with each pile, and finally the report obtained with **ALIAS_NODE** is a list of all pairs of nodes and their alias.

Compartment Information reports are obtained via a REP_TYPE of **PIECES**, **COMPARTMENT**, **CG_COMPARTMENT**, or **S_COMPARTMENT**. The first of these produces information about the pieces which comprise the compartment. The remainder deal with exterior compartments. The first of these produces: the type of filling, the specific gravity of the contents, the maximum and current amounts of ballast, and the minimum, maximum, current percentages full, and the sounding. The next produces the type of filling, the current weight, current percent full, sounding, the CG, and the CG derivative with respect to angle change. Finally, **S_COMPARTMENT** produces a report of the location of the sounding tube.

Compartment Hole Information reports are obtained via a REP_TYPE of **V_HOLE**, **P_HOLE**, **WT_DOWN**, or **NWT_DOWN**. The **V_HOLE** produces a report of the valve data for the compartment, while **P_HOLE** produces differential head and pressure data. The **WT_DOWN** and **NWT_DOWN** REP_TYPEs produce a report of the current heights of the weather tight or non weather tight down–flooding points.

Load Group Information is obtained via a REP_TYPE of **M_LOADG** or **F_LOADG**. The reports obtained with **M_LOADG** give the values of the multipliers currently being used for each load group. **F_LOADG** gives more localized information. Here a breakdown of the force acting on the element or load groups selected by :SELE is reported. The option **–FORCE** can be used to select the types of force reported. If **–FORCE** is not specified, then only the total will be reported.

Category and Load Set Information is obtained via a REP_TYPE of **M_CATEGORY**, **M_LSET**, **CATEGORY**, or **D_CATEGORY**. The reports obtained with the first two of these simply list the current value of multipliers for Categories and Load Sets respectively. **CATEGORY** yields a report of the weight and buoyancy multipliers, the weight, center of gravity, and buoyancy for each category. Finally, **D_CATEGORY** gives the weight and buoyancy multipliers, the weight and the description of each category.

Element Information is obtained via a REP_TYPE of **ELEMENT** or **F_ELEMENT**. The report obtained with REP_TYPE of **ELEMENT** gives a list of the currently selected elements, and consists of the name of the element, its class, and a list of the nodes at its vertices. The list of currently selected elements is controlled by the options of the **&REP_SELECT** command. Here, for an element to be selected, its class must match the class selector, the element name itself must match the element selector, and the nodes at the vertices must match the node selectors. If an option of **–BRIEF** is used for elements, then only the element names will be displayed. **F_ELEMENT** gives a breakdown of the force acting on the elements selected by :SELE. The option **–FORCE** can be used to select the types of force reported. If **–FORCE** is not specified, then only the total will be reported.

Map Information is obtained via a REP_TYPE of **MAP** or **N_MAP**. **MAP** produces a list of the load map names, the part to which the map applies, and the point selectors of the map. **N_MAP** provides the same information except that the point selectors are replaced with the structural nodes actually selected. Both of these types accept the **–MAP** option of the **&REP_SELECT** command. If this option is used, only the maps selected will be printed.

Structural Solution Information is obtained via a REP_TYPE of **S_CASE**, **R_CASE**, or **AMOD**. **AMOD** yields a status of the current allowable stress modifiers, **S_CASE** yields all available cases to post–process, and **R_CASE** yields a status of the currently defined R_CASE cases. The report obtained with **R_CASE** lists the names of the cases, and the constituents of each case. If the **–BRIEF** option is used, then only the names of the cases will be displayed.

XI.C Obtaining Summaries of the Model

To obtain summaries of the database, one must enter a sub–menu devoted to this purpose. To enter this menu, one need only issue the command

&SUMMARY

At this point, numerous commands are available. When finished with summaries, the user must exit the report sub–menu by issuing the **END_&SUMMARY** command, which returns the user in the menu where he entered **&SUMMARY**. To restrict the quantity of information received, one can employ the options for the previously discussed &REP_SELECT command thereby limiting the reports to subsets based on the selectors. In general, all reports are separated by body and part, an exception being summaries based on connectors and classes which have no body or part. Thus, most reports can be obtained for only selected bodies and parts. For all of the commands in this menu, one can request different types of data to be reported. If no type selection is made, reports for all of the available types will be printed.

Summaries of the properties of compartments are obtained with the command:

COMPART_SUM, TYPE(1), TYPE(2), ..., –OPTIONS

Here, TYPE must be chosen from **PROPERTIES**, **PIECES**, **E_PIECES**, **TUBTANK**, **PANELS**, **MESH**, **STRIP**, or **LONG_STRENGTH**, and the available options are those of the **&REP_SELECT** command. A TYPE of **PROPERTIES** produces a report for each compartment giving the name, description, specific gravity, volume, weight, full CG, and the maximum derivative of the CG. A TYPE of **PIECES** produces a report for each piece giving the compartment, the piece, the permeability, the diffraction type, the projected area, and the integral of the normal over the area (this should be zero). Similarly, a TYPE of **E_PIECES** gives a report of each piece which forms the exterior of the vessel. The report gives: the diffraction type, the permeability, the wind and drag coefficients, and the water depth / draft current force multipliers. A TYPE of **TUBTANK** produces a report on the tubtanks defined for each compartment. A TYPE of **PANELS** produces a list of the panels defining each piece of each selected compartment for the selected bodies and parts. With this type, the −**PANEL** option of **&REP_SELECT** is honored, and if it is used, only the panels selected will be reported. A TYPE of **MESH** will produce a report of the diffraction mesh for all bodies which match the body selector. A TYPE of **STRIP** will yield a report of the planes to be used for strip theory computations. A TYPE of **LONG_STRENGTH** produces a summary of the vessel longitudinal strength properties.

To produce reports for selected bodies and parts for load groups selected by the load group selector, one issues the command:

LOADG_SUM, TYPE(1), TYPE(2), ..., –OPTIONS

where TYPE(i) must be chosen from **ATTRIBUTE**, **UD_FORCE**, or **MATRICES**, and the available options are those of the **&REP_SELECT** command. A TYPE of **MATRICES** is used for reporting the mass, added mass, and damping matrices for the group. A TYPE of **ATTRIBUTES** is used for the buoyancy and area contributions to the group. Finally, a TYPE of **UD_FORCE** is used for reporting user defined loads for the group.

The next command is used to obtain summaries of the properties of beams. It returns reports for elements selected with the element selector, for classes selected by the class selector, and with end nodes which match the node selectors. The form of the command is:

BEAM_SUM, TYPE(1), TYPE(2), ..., –OPTIONS

where TYPE(i) must be chosen from **LOADS**, **PROPERTIES**, **UD_FORCE**, **CLEARANCE**, **SCF**, **TUBE**, **ENDS**, or **VORTEX** and the available options are those of the **&REP_SELECT** command, and, for a type of **VORTEX**,

–**W_VELOCITY**, WIND_VELOCITY
–**C_VELOCITY**, CURRENT_VELOCITY
–**BRIEF**

A TYPE of **PROPERTIES** reports the location, offsets, length, etc. of beams. A TYPE of **LOADS** reports the intrinsic load attributes (weight and diameters) for elements, and **UD_FORCE** reports the user defined load sets applied to beams. A type of **SECTION** is used for reporting beam section properties. A type of **CLEARANCE** produces a report giving the distance from the extremities of a member to the extremities of all of the other members not connected to the given one. A type of **SCF** gives the stress concentration factors for the selected beams along the beam. In addition to the SCF, the type of connection, the thickness the SN curve also reported. A type of **TUBE** produces a report of the diameter, thickness, yield stress, and length of each segment of the selected tubular members. A type of **ENDS** reports the part coordinates of the ends of the selected beams.

Finally, a type of **VORTEX** is used to obtain information about vortex shedding on the selected beams. It will produce two reports: one for beams out of the water where wind provides the excitation, and one for beams in the water where current provides the excitation. Either of these reports can be limited to only those beams which should be "checked" by using the –**BRIEF** option. Here, the first three natural frequencies of vibration of the beam both inplane and out of plane are computed. These frequencies are used to compute the critical velocities (velocities at which vortices will be shed at the same frequency as the natural frequency of the beam). In addition, the wind speeds that mark the beginning and end of Region II of vortex shedding are reported. Conceptually, no vortex shedding occurs within Region II. Finally, a comment is added to "check" beams which may be subjected to vortex shedding. This comment occurs whenever the smallest critical speed is below the beginning of Region II, and whenever a critical speed is greater than the end of Region II and less than the specified velocity. The velocities which are used in the "checking" criteria are WIND_VELOCITY (knots) and CURRENT_VELOCITY (ft/sec or m/sec) specified with the two options **–W_VELOCITY** and **–C_VELOCITY**.

In computing natural frequencies, several assumptions have been made which may prove to be inapplicable to the situation. In particular, it is assumed that the mode of vibration is given by

mode = sin n * pi * x / (k * L)

where x is measured from the left end of the beam. The diameter used in computing both the Reynolds number and the critical wind speed is a length average of the wind diameter for all element attributes.

To obtain a summary for generalized plates one should issue:

PLATE_SUM, TYPE(1), TYPE(2), ..., –OPTIONS

where TYPE(i) must be chosen from **PROPERTIES**, **FACE**, **SUBELEMENT**, or **VERTEX** and the available options are those of the **&REP_SELECT** command. The first of these produces a report similar to PROPERTIES for beams, The second one reports the faces for the generalized plate, the third on reports the subelement names area and centroids for each generalized plate, and the last one reports the nodes, releases, and offsets for each vertex.

To obtain a summary for RESTRAINTS, one should issue:

RESTRAINT_SUM, –OPTIONS

and the available options are those of the **&REP_SELECT** command. This command will produce a report of the location of the ends of each element which is not neither a beam nor a plate and is not a connector.

The next command produces a report of the weight, center of gravity, buoyancy, and center of buoyancy of each class and each load group by category. Only those selected elements and load groups belonging to selected parts will be considered. The form of the command is:

CATEG_SUM, –OPTIONS

and the available options are those of the **&REP_SELECT** command. In particular, if the option **–BRIEF** is used then only the total for each category will be reported.

To obtain information about the property classes selected by the class selector, one issues the command:

CLASS_SUM, TYPE(1), TYPE(2), ..., –OPTIONS

where TYPE(i) must be chosen from **DIMENSION SECTION**, **MATERIAL**, or **SOIL** and the available options are those of the **&REP_SELECT** command. A TYPE of **DIMENSION** reports the information about the section type and size. **SECTION** reports section and stiffener properties. **MATERIAL** provides information about the redesign and material properties of the class, such as yield strength and Young's Modulus. Finally, **SOIL** will produce a report of the soil properties for the selected classes that have soils defined.

Information about points selected by the node selector is generated with the command:

POINT_SUM, TYPE(1), TYPE(2), ..., –OPTIONS

where TYPE(i) must be chosen from **NOSES**, **POINTS**, **N_COINCIDENT**, **SCF** or **PROPERTIES** and the available options are those of the **&REP_SELECT** command. A TYPE of **NODES** will produce information about POINTS which are have structural elements connected to them; i.e. NODES which are selected by the node selector. This TYPE reports the node name, the node type, the X, Y, Z coordinates in the part system, the X, Y, Z coordinates in the global system, and the degrees of freedom of the node which are fixed. Here the type is either: TUBE–JOINT, TUBE/TUBE, TUBE/CONE, JOINT, NODE or EXTREMITY. The second two types are used for nodes that have only two beams in a line connected. JOINT is used for nodes that have more than one element connected but are not TUBE–JOINTS, A TYPE of NODE is used for two connected elements that are not TUBE/TUBE or TUBE/CONE types, and EXTREMITY is used for a node with only one element connected. A TYPE of **POINTS** will list all points which are not nodes, their coordinates, and the node associated with the point. A type **N_COINCIDENT** will produce a report of the nodes in each part which are coincident. Information about tubular joints are obtained with the last two TYPEs. A TYPE of **SCF** will produce a report on the stress concentration factors and the API strength unity ratio. A type of **PROPERTIES** will produce a report of the diameter, thickness, yield stress, and angles for each brace for the selected joints, and the API strength unity ratio.

A final command is used to produce a summary of the wave elevation, wave velocity, and wave acceleration for all grids selected, and its form is:

GRID_SUM, –OPTIONS

Here, the available options are those of the **&REP_SELECT** command. The selection criteria here is the one defined via **–DATA**.

XII. THE MOSES MODEL

MOSES operates using a database philosophy. In other words, the commands to the program are instructions to perform some operation upon the information which currently resides within the database. Before one can obtain meaningful results, he must have a database upon which to operate, i.e. a model of the basic system must be defined for the program. This model is defined via commands in a modeling language described below. Although the distinction is somewhat arbitrary, it helps to think of the model as being composed of two different types of data: some which remain constant and some which change as the analysis proceeds. The constant part is the "basic" model and the remainder are settings which normally evolve. For example, the physical description of a barge will normally remain constant throughout an analysis, but the ballast configuration, the draft, trim, and cargo normally will not. Another way of viewing this difference is that the "constant" model normally consists of a relatively large amount of data, while the variable part is much smaller.

MOSES has two different menus for defining the constant part of the model: the **INMODEL** menu and the **MEDIT** menu. It was originally intended for the user to "read" in a model, and then "edit" to add some additional features or fix some mistakes. The situation has evolved so that now it is not necessary to first read in the model. One can define it in its entirety from the MEDIT menu. A small price must be paid, however. In the **INMODEL** menu, it is assumed that one starts with nothing and ends with a complete model. This allows MOSES to defer some associations until it completes reading the model. In the MEDIT menu, the assumption is that a model already exists and thus, it must be kept up to date. The difference in philosophy requires that in MEDIT, a piece of data must have been previously defined before it can be referenced. Also, defining a model with **INMODEL** is computationally more efficient.

Data defined via the **INMODEL** menu is input from the "INPUT" channel (a file root.dat) and the process is initiated with the command:

INMODEL, –OPTIONS

There are two options available here:

–OFFSET
–PROCESS, PRC_NAME

Here, **–OFFSET** will cause local axial offsets to be computed at the ends of any tubular member connected to a tubular joint, and **–PROCESS** allows setting the current process to be PRC_NAME after the **INMODEL** is complete. After an **INMODEL** command has been issued, it should be re–issued only if one wishes to substantially alter the model. Doing so will result in **all** previous results being deleted from the database. If you want to have a second **INMODEL**, a

&DEVICE –AUXIN command *must* be re–issued.

During the **INMODEL** process, you can use two commands:

> **USE_VES**, VES_NAME

or

> **USE_MAC**, MAC_NAME

to load a predefined vessel model or macro. When one of these is issued, MOSES will look in a set of predefined places to see if it can find the specified name. By default, the order will be the current directory, the /ultra/data/local directory, and then the places where MOSES supplied vessel models and macros are located. You can use the command,

> **&PATH**, TYPE, ADDITIONS

to add places to be checked. Here, TYPE must be either **VESSEL** or **MACRO** and it defines which path is being altered. ADDITIONS is simply a list of new places to check. When this is done, the search order will be the current directory, the places specified by ADDITIONS, and then the previous places.

Defining or altering a model with **MEDIT** is initiated with the command:

> **MEDIT**,

and is exited via an

> **END_MEDIT**

command. All of the valid **INMODEL** commands are valid during **MEDIT** as well as several additional ones. In general, the extra commands are those which delete things, change things, and define connections. The restriction to **MEDIT** will be made explicit when these commands are defined.

Within either of these menus, the model is defined with a set of commands defining the primitives upon which MOSES will operate. Each of these will be discussed in detail below. During this discussion, we will also define the commands which alter the settings of some of the primitives. It should be remembered that even though one can issue these commands within a model defining menu (they are internal commands), it normally does not make sense to alter something until it has been completely defined. Also, remember that MOSES was designed to be an integrated program for both simulating a process and performing a stress analysis of the system during the process. To accomplish this integration, the model which one prepares for MOSES is conceptually different than one would prepare for a program designed for either of the two tasks alone. Thus, a model for MOSES is not simply a model of the structure of the system. Instead, it is a model from which MOSES can compute not only the stiffness of the system, but also the loads which act on the system.

The basic idea behind the modeling language is to convey as much of the needed information as possible with the minimum amount of description. Thus we have modeling commands which define the "physical" components of the system instead of defining only some aspects of it. Of course, one must have a method of overriding the built–in assumptions, and in some cases, there is no easy way to model what one desires.

The basic ingredients for performing a simulation are bodies which are considered rigid, and composed of parts. Bodies are connected with special elements called connectors. A part is the smallest entity upon which a structural analysis can be performed. In essence, a part is simply a named, connected subset of the model.

Most properties of the system are described by the attributes of the parts. In other words, every attribute of the system must be an attribute of some part of the system. Therefore, everything except bodies belong to some part. There is a special part which does not belong to any body, and it is in this part that elements which connect bodies reside. These elements, called connectors, are quite important. They define both the boundary conditions for a stress analysis, and the constraints on the bodies for a simulation. To avoid confusion, elements which connect parts (elements which are connected to nodes in different parts) must belong to a part with a type of PCONNECT. Thus, the only elements which can span parts are restraints, connectors or PCONNECTors. A PCONNECT part should not have any nodes which belong to it.

The geometry of the model is defined by quantities called points. Points all have names beginning with a *. The subset of the points to which structural elements are connected are called nodes. All other points are associated with some node. Both points and nodes are defined in a coordinate system which belongs to the part.

The structural attributes of the model are defined by a set of beam and generalized plate elements which connect the nodes. Since there are many elements in a structure which have common properties, MOSES allows one to associate a name with a set of properties. This set of properties is then associated with the applicable elements by specifying the name on the element definition command. The name associated with a set of properties is called the class name of the properties, and it must begin with the character ∼. The concept of class is important in MOSES since it is used not only for defining properties, but as a way of associating elements for post–processing and redesign.

The remainder of the attributes of a part are used to compute the loads which act upon it. In general, there are four sources of loads which MOSES can consider:

applied, inertia, wind, and water. Notice that there is a conceptual difference between the applied loads and the others in that the other loads arise due to the interaction of the system with its environment. Thus, for applied loads, one models the loads themselves, while for the other classes, he must model the physical attributes which give rise to the loads.

Of these sources of loads, the system/sea interaction loads are by far the most complicated. In fact, we will distinguish six different types of structure/sea loads: buoyancy, added mass, radiation damping, viscous drag, linear wave excitation, and wave drift force. To allow flexibility, MOSES employs three hydrodynamic theories: Strip Theory, Three Dimensional Diffraction Theory, and Morison's Equation. When Morison's Equation is used, viscous drag is computed, but no radiation damping. The other hydrodynamic theories, however, consider no viscous damping, but only radiation damping. Thus, if one wishes to add some viscous drag to a system using a diffraction hydrodynamic theory, he needs to define some Morison's Equation load attributes.

The primary objective of the MOSES modeling language is to minimize the number of additional load attributes which need to be defined. To ease the definition process MOSES employs two additional concepts: the load group, and the compartment. Compartments are used to define the loads associated with "vessel like" attributes, and load groups are used to define loads associated with "structural like" attributes. Compartments serve to define the loads which arise due to the interaction of the bodies with the water for things which cannot be adequately modeled with Morison's Equation. There are two classes of compartments: interior and exterior. The exterior compartments define the exterior of a body and the interior ones define subdivisions which may or may not be open to the sea. Interior compartments can also be ballasted, in which case, compartments define a part of the inertia of the system.

Since all of the model information in MOSES is stored by name, the names are of importance in conveying to others what type of data is associated with the name. Unfortunately, the limited space of a name often does not allow one to adequately describe things. Thus, **all** of the modeling commands accept two options **–NOTE** and **–TAG**. The first of these allows one to attach a 60 character description of the data. This is a peculiar option *in that it must be specified by all five characters.* One can then use an **&STATUS** command to report the name and the notes for selected types of data. No quotes are needed for this option to include blank spaces, but if you have a –, it must be escaped if it is not followed by a blank. The **–TAG** options allows one to define an eight character "tag" to the name. Tags can be used to select reports for a given class of data to only that which matches its tag. If a tag is not specified, then one which is the same as the name will be given.

There are several commands in MOSES which exports portions of the model: **&EMIT**, **&EXPORT**, **&CONVERT**, **E_TOTAL**, and **E_PRESSURE**. Each of these commands also accepts the **–NOTE** option. When the exported file is written, the TITLE, SUBTITLE, and the NOTE will be included in the exported file if they are not blank.

One important aspect of the MOSES language is the string function. Most of the string functions discussed previously performed simple tasks such as arithmetic. There is another class of function which "returns" information about the current model or configuration. These are extremely useful in automating certain tasks. Each of these functions will be discussed later, but one which is useful in generating loops is:

> **&NAMES(**QUANT, SELE**)**

which returns the names of database quantities. Here, QUANT is the category for which names is desired. This valid categories is defined in the section "Obtaining the Names of Quantities", or you can obtain it by issuing either:

```
&NAMES NAMES
```

or

```
&STATUS NAMES
```

The &STATUS gives not only the list of names, but also a brief description. The behavior here depends on if SELE is omitted. If specified, then all names of QUANT which match SELE will be returned. If SELE is not specified, then most of the time, all of the names for QUANT will be returned to the command line. For PARTS, ELEMENTS, COMPARTMENTS, LOAD GROUPS, POINTS, and NODES however, the results are returned only for the current body and part.

With the power of the MOSES language, users often want to emit the commands used to build a model to a separate file for later use. This can be achieved elegantly using **&EMIT**, which has the following syntax:

> **&EMIT**, –OPTIONS

where the options are:

> –**BEGIN**
> –**BOX**, COMMENTS
> –**LINE**, COMMENTS
> –**END**

Perhaps the best way to illustrate this command is with the following example:

```
&EMIT –BEGIN
&DESCRIBE BODY TEST
```

```
&EMIT –BOX This is the main hull model
PGEN HULL –CS_WIND 1 1 1 –CS_CURRENT 1 1 1
   PLANE 0 50 100 150 200 250 300 –RECT 0 20 90
END
&EMIT –LINE Cargo Wind Area
PGEN CARGO –LOCATION 0 0 20 –CS_WIND 1 1 1
   PLANE 100 150 200 –RECT 0 40 100
END
&EMIT –END
```

The **–BEGIN** option causes MOSES to declare the units being used. The **–BOX** and **–LINE** options simply add comments to the resulting file, which is useful for documentation of a model. The –BOX options places a box around the comments, while the **–LINE** option places the comments at the end of a line. The **–END** option stops the emit process, and returns MOSES to parsing modeling commands as normal. If the program finishes without using **&EMIT –END**, MOSES will automatically include this command as part of closing all files.

The commands to be emitted must come through the input channel, typically the root.dat file. None of the modeling commands following **&EMIT** will be parsed as they normally would. Instead, the modeling commands, complete with comments, are placed in the post processing output file, typically root.ppo.

XII.A Converting Models

Often, one is given a model in a format which is not suitable for use in MOSES. In certain cases, this model may be converted into a useful one by a special purpose menu. This menu may be entered via the command:

> **&CONVERT**, MODEL_TYPE, –OPTIONS

Here MODEL_TYPE defines the type of model to be converted and must be either: **SACS**, **STRUCAD**, **DAMS**, **STRUDL**, **HULL**, **OSCAR** or **PLY**. The available options are:

> –**JPREFIX**, JP
> –**CPREFIX**, CP
> –**JRIGHT**, XXXXXXXX
> –**CRIGHT**, XXXXXXXX
> –**LOADS**, FLAG
> –**CNS_DIA**, FLAG
> –**IG_DOFS**, FLAG

When the **&CONVERT** command is issued, MOSES will read data from the current channel and write a MOSES model to the file MOD00001.TXT file in the answers (.ans) directory. This process will continue until the command **END** is encountered. The conversion for **SACS** and **STRUCAD** models are relatively complete, but only a subset of all of the dialects of **STRUDL** are honored. A MODEL_TYPE of **HULL** will convert a hull description using the outdated STAT and OSET commands to the current description using the PLANE command. One can also convert **OSCAR** "miscellaneous additions" such as LJNT, LMEMT, LMEMD, LMEMS, and JPLATE commands into the proper format by using the type **OSCAR**. When an input is found which MOSES does not convert, it is emitted as a comment. Thus, the file resulting from a conversion may contain quite a few comments before the converted data. A MODEL_TYPE of **PLY** will convert a PLY polygon file into a MOSES mesh. The name of the piece generated will be the value of the variable "PIECE" if it exists. If this variable has not been defined, then the name will be "PLY".

During the conversion process, MOSES "adds" something to each node and class name. It will *always* add a * to a node name and a ∼ to the class name; but, if instructed, it will add more. What is added is controlled by the options. The –**JPREFIX** option defines a set of characters, JP, which will be added to the beginning of each node name, and the –**CPREFIX** options defines characters which will be added to the beginning of each class name. The default for JP is **J** and the default for CP is to add nothing. If one wants to add only a "*", he should use "*" for JP.

Normally, the beginning character (either a * or ∼), the prefix, and the original name are simply concatenated without blanks to form the new name. If one wishes, he can "overlay" the new name on a template. This is accomplished by the **–CRIGHT** and **–JRIGHT** options. As an example, suppose that one issued the command:

```
SACS, –JPREFIX J –JRIGHT, 0000
```

and that one has a joint name 1. Then MOSES will first construct the template *J0000 and the right justified name " 1". The right justified name will then be overlaid on the template to produce the new name *J0001.

Care should be used with the name altering options, since it is possible for two original names to be mapped into the same new name. This can occur when the original names have more characters than 7 minus the number of characters in the prefix when appending, or more than the number of characters in the template plus the number of characters in the prefix when overlaying.

Normally, MOSES will convert all "load" data in the existing model into weights. In the process, the weight is assumed to be the weight of something "in air". The converted load will use a macro that has an option –SGRAV which specifies the specific gravity of the material (used as steel). If, the loads were coded on a different basis, the conversion can be changed via the **–LOADS** option. If "FLAG" is **WATER**, then a variable will be set so that the weight input will be appropriate for air, and again the specific gravity will be used to compute a buoyancy. If "FLAG" is **SIGN**, then it is assumed that the sign of the Z component of the load defines a weight if it is negative and a buoyancy if it is positive. Here, no specific gravity is used. This option works only for a SACS or STRUCAD model. If "FLAG" is **LOAD**, then the loads are really converted as loads.

In general, the "load" data is divided into load cases and often commented as to the origin of the load. In MOSES, one can use either of these two schemes to create Categories of load. A **&DEFAULT** command is inserted before each load case to set the default Category to be the load case name. Also, if the loads are commented, the comment is available to further separate the loads. The use of the comment is governed by the variable USE_COMMENTS which is set to TRUE by default. If you want to ignore the comments, change the setting to FALSE. Also, the load macros allow for a factor to be used, and the factor is set to a variable with the load case name. In this way, one can reproduce the weight of a structure with several load cases combined with different factors. The values of the factors are defined before the load data itself.

The option **–CNS_DIA** instructs MOSES to include the redesign option, based on a constant diameter, in the class definition.

The final option **–IG_DOFS** allows one to pass the information on deleted degrees freedom to the MOSES model. Normally this information is not transmitted since it almost certainly will be incorrect. If you use this option, you must be quite careful that the model is properly restrained.

In general, the conversion process works quite well. With a STRUDL model, however, one must be careful. STRUDL is not only a modeling language, but also a command language. The "action" commands are ignored when a model is converted and hence, the conversion may not be what occurs in the original file. In particular, if the file "ignores" members, then generates dead loads, the MOSES model will have loads on the ignored members. Another difficulty occurs with units. Some versions of STRUDL will actually allow the abbreviation of "T" for TONNES. The MOSES converter requires at least "TONN". Also, in STRUDL, it is possible to continue a line by ending it with a "–". MOSES will not accept this, and all line ending "–" should be changed to either a "+" or a "\". One thing to notice is that in STRUDL, the strong axis of a beam is the Z axis while in MOSES, it is the Y axis. In the conversion process, Y will be changed with Z. The only problem we know about with local axis conversions is with a vertical member which "points down" where local properties will perhaps be incorrect. STRUDL finite elements are handled by placing the element name, class and nodes in the converted file. This allows the user to build a macro to represent the element. The simplest macro is shown below:

```
&macro SBCT class nodes
plate %class% %nodes%
&endmacro
```

XII.B Defaults

To allow flexibility in using MOSES, the user is free to set many of his own defaults. When defining a model or altering its definition, there are a number of things which normally have the same value. In other words, many things have a "default" value. The default values used are defined with the command:

&DEFAULT, –OPTIONS

Two basic options:

–SAVE
–REMEMBER

allow the user to "push and pop" the default stack. Suppose that one wishes to alter some of the defaults temporarily and then return to the initial set. This can be accomplished by issuing **&DEFAULT –SAVE** which "saves" the current settings on the default stack. Now, one can alter the defaults at will, and, upon issuing **&DEFAULT –REMEMBER**, the initial set will again become active.

Most of the options here are again used on some command. As a result, the documentation here may be brief so that a more detailed discussion may follow. If an option is used when one of these commands is issued, then values specified by the option will be used. Otherwise, the default (the values specified via **&DEFAULT**) will be used.

Defaults are also used to define the "current" coordinate system and frame of reference for defining: nodes, interest points, and diffraction vertices. The options defining these defaults are:

–RECT
–CYLINDER
–SPHERICAL
–LOCATION, XO, YO, ZO, RX, RY, RZ
–LOCATION, XO, YO, ZO, *PT(1), *PT(2), *PT(3), *PT(4)

The first three options define the meaning of the three numbers which define the local coordinates. If **–RECT** is the last of these specified, then the numbers are rectangular coordinates in the current frame. Likewise they can be either cylindrical if one specifies **–CYLINDER**, or spherical coordinates when **–SPHERICAL** is used. Cylindrical coordinates require a radius, angle and Z coordinate, while spherical coordinates require a radius, angle in the XY plane and an azimuth angle.

The **–LOCATION** option defines a new frame of reference. Here, XO, YO, and ZO are the coordinates of the new frame of reference in the part system. The orientation of the new frame is defined by either three Euler angles RX, RY, and RZ, or by up to four nodes. When using the Euler angles, the new frame orientation is determined by three successive rotations about the Z, Y, and X axes respectively. When using nodes, the location and orientation both depend upon the number of nodes specified. For one node, the orientation of the frame is the same as the part frame, but the new origin is at the specified node. For two nodes, the new origin is at the first node and the orientation is the same as that of the element system of a beam between the two nodes. For three nodes, the new origin is at the midpoint between the first two nodes, the new X axis is perpendicular to the plane formed by the nodes, and the new Y axis points from the first node to the second one specified. Finally, for four nodes, the origin is at the midpoint between the second and the fourth nodes, and the Y axis points from the fourth node toward the second one, and the X axis goes from the origin to the midpoint of the line segment connecting the first and third nodes.

The options:

–**BBC_MUL**, MULT
–**CO_SCF**, SCF_TYPE
–**LEN_FACTOR**, FRACHOL
–**MAX_CHD_LEN**, MAXCHOL
–**CHD_FIXITY**, CHD_FIX
–**MIN_SCF**, MIN_SCF

establish defaults for the same options on the point definition command and are discussed there.

The options:

–**COLOR**, NAME_COL
–**TEXTURE**, NAME_TEX, X_SCALE, Y_SCALE

define the default color and texture for the model. Here, NAME_COL is any color which has been previously defined. See the section on Colors for a discussion on defining colors. The NAME_TEX value for −**TEXTURE** is the name of a file in either /X/data/textures or /X/data/local/textures (here MOSES is store in /X). The X_SCALE and Y_SCALE are scale factors which will be applied to the texture. The NAME_TEX of **NONE** will yield a null default texture.

The options:

–**BAS_CATEGORY**, NAME_BAS
–**EXT_CATEGORY**, NAME_EXT

define the default Category for structural element and additional load attributes respectively. NAME_BAS will be the category for all load attributes directly associated with a structural elements and NAME_EXT will be the category for all additional load attributes unless they are specifically defined on the element or load attribute command.

The default scheme is also used for defining default properties of element classes. The material properties are set via the options:

–**SPGRAVITY**, SPGR
–**DENSITY**, RHO
–**EMODULUS**, EMOD
–**POI_RAT**, POIRAT
–**ALPHA**, ALPHA
–**FYIELD**, FYIELD
–**SN**, TYPE(1), SN1_A, SN1_B, SN1_R, \
TYPE(2), SN2_A, SN2_B, SN2_R, ...

Here, SPGR is used to define the material density by the ratio of its density to that of standard water, RHO is the material density (pounds/ft**3 or newtons/m**3), EMOD is the Young's Modulus (ksi or mpa), POIRAT is the Poisson's Ratio, ALPHA is the coefficient of thermal expansion (1/Deg F or 1/deg C), and FYIELD is the yield stress (ksi or mpa). For a discussion on the –SN option, see the section on associating SN curves.

Three options define the default resize properties for classes. These are:

–**RDE_SELE**, TYPE(1), RD(1), TYPE(2), RD(2), ...
–**KL/R_LIMIT**, KLR
–**D/T_LIMIT**, DOT

Here, TYPE(i) defines a section type and must be either: **TUBE**, **BOX**, **PRI**, **BU**, **W**, **M**, **S**, **HP**, **WT**, **MT**, **ST**, **L**, **C**, **MC**, **WBOX**, **DL**, **LLEG**, **CONE**, or **PLATE**. The values RD(i) are a selector which defines the default redesign selector for the section type TYPE(i). KLR and DOT are the default KL/R and D/T limits on shape selection for tubes.

Several options are available define element defaults:

–**USE**, USE(1), USE(2), ..., USE(i)
–**NUSE**, NOT_USE(1), NOT_USE(2), ..., NOT_USE(i)
–**FLOOD**, YES/NO
–**STW_USE**, YES/NO
–**KFAC**, KY, KZ
–**CMFAC**, CMY, CMZ
–**DIR_BEAM**, SAV1(1), SAV1(2), SAV1(3), SAV2(1), SAV2(2), SAV2(3)
–**DIR_PLATE**, SAV1(1), SAV1(2), SAV1(3), SAV2(1), SAV2(2), SAV2(3)

–**SCF**, TYPE(1), SCF(1), ...

All but the last of these will be discussed in detail along with the discussion of modeling elements. The last two define true default behavior. For a discussion of the –**SCF** option look in the section on associating SCFs with fatigue points.

The options:

–**MD_FORCE**, MD_FORCE, MD_RADIATION, MD_CORIOLIS
–**MD_PHASE**, MD_PHASE
–**SPE_MULTIPLIER**, SPEMUL
–**FM_MORISON**, FM_FACTOR
–**SP_ORIENT**, VX, VY, VZ, HX, HY, HZ
–**SP_HEIGHT**, X, Y, Z
–**DT_CONVOLUTION**, DT_CONV
–**WAVE_RUNUP**, YES/NO

define defaults which can be overridden with **&DESCRIBE BODY** commands.

The options:

–**CS_WIND**, CSW_X, CSW_Y, CSW_Z
–**CS_CURRENT**, CSC_X, CSC_Y, CSC_Z
–**AMASS**, AMA_MULT
–**TANAKA**, TANAKA_FACTOR
–**CS_WIND**, CSW_X, CSW_Y, CSW_Z

define defaults which can be overridden with **&DESCRIBE PIECE** or **PGEN** commands.

While the previous options set defaults for modeling commands, those that follow set defaults for other types of options. The option:

–**FILL_TYPE**, FTYPE

defines the default type of filling for compartments. Here, FTYPE must be either **CORRECT**, **APPROXIMATE**, **APP_NONE**, **APP_WORST**, **FULL_CG**, **FCG_NONE**, or **FCG_WORST**. The meaning of these types are discussed in the section on filling compartments.

The options:

–**WATER**, RHOWAT
–**SPGWATER**, SPGWAT
–**RAMP**, RAMP_TIME
–**DEPTH**, WATDEP
–**SP_TYPE**, TYPE

–**W_PROFILE**, WP_TYPE
–**W_PERIOD**, TW(1), TW(2), ..., TW(n)
–**W_DESIGN**, DTYPE
–**W_SPECTRUM**, STYPE
–**W_MD_CORRELATION**, FACTOR
–**MD_PERIOD**, TD(1), TD(2), ..., TD(n)
–**PROBABILITY**, STAT, PDATA
–**T_REINFORCE**, TB
–**HEADING**, H(1), H(2), ..., H(n)
–**PERIOD**, T(1), T(2), ..., T(n)

define default values used by the **&ENV** command and they are discussed there. The last two options allow one to define the default headings and wave periods which will be used for several different commands. They are discussed in detail later.

XII.C Parameters

The &PARAMETER command is used to define parameters used in various computations. One of these commands is included in the file **moses.cus** so that one can alter these settings to suit their particular purposes. This file should be consulted to ascertain what settings are being used. This command functions in a manner similar to the **&DEFAULT** command in that any option specified on some other command with the same name as that here will override the default. The form of this command is:

&PARAMETER, –OPTIONS

Again as with **&DEFAULT**, there are two basic options:

–SAVE
–REMEMBER

which allow for temporarily altering the parameters and returning to the previous ones. In particular, the **–SAVE** option instructs the program to save the current dimensions so that when **–REMEMBER** is used, the ones previously saved will then be used.

The options:

–DRGTUB, RE(1), DC(1), RE(2), DC(2), ...
–F_CD_TUBE, CDTFREQ
–FM_ROD, ROD_FACTOR
–DRGPLA, DCP
–AMCTUB, AMT

define the hydrodynamic properties of generalized plates and tubular members. The added mass coefficients of tubular members and generalized plates are taken to be constant. The drag coefficient for tubular members is a function of the Reynold's Number in the time domain and constant in the frequency domain, and are defined with the options **–DRGTUB** and **–F_CD_TUBE**. The **–FM_ROD** option performs for rod elements the same task as **–F_CD_TUBE** does for tubes. The drag coefficient for generalized plates is defined with the **–DRGPLA** option. The added mass for generalized plates and panels, is computed as described in the section on Forces and for tubes it is defined with the **–AMCTUB** option.

The options:

–WCSTUBE, CSHAPE
–REL_WIND, YES/NO

control computation for wind forces. The wind shape coefficient for tubular members is defined by the option **–WCSTUBE** and here CSHAPE is the new value for the coefficient. The **–REL_WIND** option defined whether or not the wind force will be computed based on the relative wind velocity or the wind velocity itself.

Normally MOSES computes slam loads on plates and tubes by computing the derivative of the added mass. This can be a numerically sensitive computation and the two options:

> **–SL_TUBE**, SCT
> **–SL_PLATE**, SCP

can be used to define a slamming coefficient that is independent of time. A slam force will only be computed when the element has a waterplane intersection. To return to the normal way, one should specify **AUTOMATIC** for SCT or SCP. Theoretically, there should be slamming for both the element entering the water and when it is exiting. The option

> **–SLAM_BOTH**, YES/NO

controls this. If YES/NO is **YES** then slams occur for both cases. If it is **NO** then slams only occur when the element is entering the water.

The option

> **–T_AVERAGE**, TYPE

defines the way that the "average period" is computed when doing fatigue or cycle counting in the frequency domain. If TYPE is **DNV** then the average period is the associated with the "average zero up crossing frequency" defined in DNV RP–C206 (Section 6.9.4). Otherwise, it will be computed with the traditional formulae:

> Tav = 2 pi sqrt { M2 / [M0 (1 – eps)] }
> eps = sqrt { [M0 * M4 – M2**2] / (M0 M4) }

The options:

> **–API_TDRAG**, YES/NO
> **–AF_ENVIRONMENT**, YES/NO

are used in computing velocity square forces. The first one controls the relative velocity for tubes. If YES/NO is **YES** then the relative velocity is the component normal to the tube as in API RP2A. If YES/NO is **NO** then it is the true relative velocity. The second one controls the way wind and drag are computed on areas. If YES/NO is **YES** then the drag force is in the direction of the environment. If YES/NO is **NO** then it is perpendicular to the area. One should use **YES** to have the force depend on the projected area.

In many cases, MOSES will perform a numerical integration over either an area or a length. The precision of this integration can be controlled via the options:

–**MAXLEN**, LENGTH
–**MAXAREA**, AREA
–**MAXREFINE**, REFINE_NUMBER

Here, MOSES will divide an element into pieces such that each length or area of each piece will be less than LENGTH (feet or meters) or AREA (ft**2 or m**2). The maximum number of pieces any one element will be broken into is REFINE_NUMBER.

The option:

–**M_DISTANCE**, DISTANCE

is used to define the amount of refinement which will be performed on a diffraction mesh when hydrodynamic properties are computed. Here, DISTANCE (feet or meters) defines a maximum distance which the side of a panel or the length of a strip can have. Use of this option allows one to define a quite crude mesh and have MOSES automatically refine it to achieve any desired degree of precision.

The options:

–**STRETCH_SEA**, YES/NO
–**NONL_SEA**, YES/NO

control how the wave kinematics are computed. If YES/NO is **YES** for the first option, then the sea will be "stretched" above the mean water level. If not, then the linear kinematics equation will be used directly above the mean water level. If YES/NO is **YES** for the second option, then the wave profile will be computed using an estimate of the nonlinear pressure term. This results in the wave crest being higher than a trough is low. If it is **NO**, then a linear wave profile will be generated.

The option:

–**IN_SCF**, TYPE

is used to define the method used to compute the "inline" stress concentration factors between two tubular sections. The section on SCF Binding discusses this in greater detail.

XII.D Convolutions

Convolutions are powerful mathematical tools. Simply, a convolution is the integral from 0 to infinity of a function K times a history; i.e.

C = integral_0îinfinity [K(s) X(t–s)]

K is called the kernel and x is the motion, velocity, etc. While the convolution is really the integral, we will sometimes call the kernel the convolution.

Convolutions arise in MOSES in two ways:

- MOSES computes hydrodynamics and from the frequency domain results, MOSES computes an inverse Fourier Transform to obtain a kernel, or
- the user defines a kernel.

The &DATA command is used to define "kernel" which used are various computations. Here, one associates a name with a function (set of data) and then uses that name to refer to the function. The form of this command is:

&DATA CONVOLUTION, TYPE, NAME, DATA –OPTIONS

Here, TYPE is the type of data which is being defined either FREQUENCY or TIME. NAME is the name you wish to give to the convolution, and DATA is the numbers used to define the convolution. DATA is a set of numbers T(1), K(1,1), K(2,1), ... T(n), K(1,n), ... K(m,n). Here, T is either the time or frequency and K is the corresponding value of the kernel. Normally you define a curve with an independent variable and a single dependent variable, but if you specify the option

–NDOF, M

Then you can have M values of K for each value of T.

The behavior of any of these can be obtained with

&STATUS TYPE NAME –PLOT

Where NAME is the name of the curve about which you want information and TYPE is either T_CONVOLUTION or F_CONVOLUTION depending on what type of data you wish to view.

XII.E Curves

The &DATA command is used to define "curves" which used in various computations. Here, one associates a name with a function (set of data) and then uses that name to refer to the function. The form of this command is:

&DATA CURVES, TYPE, NAME, DATA –OPTIONS

Here, TYPE is the type of data which is being defined, NAME is the name you wish to give to the curve, and DATA is the numbers used to define the curve. DATA is an n–tuple where normally n is two; i.e. normally you define a curve with an independent variable and a single dependent variable. TYPE must be either **C_PROFILE**, **P_SPECTRUM**, **F_SPECTRUM**, **M_GROWTH**, **W_HISTORY**. **LT_MULTIPLIER**, **CT_LENGTH**, **EFFICIENCY**, **CS_VELOCITY**, or, **AM_PRESSURE**. The behavior of any of these can be obtained with

&STATUS CURVES NAME –PLOT

Where NAME is the name of the curve about which you want information.

The first five of these define curves which are used in defining the environment.

- **C_PROFILE** defines a current profile. The DATA is Z(1), V(1), ..., Z(n), V(n) where Z(i) are depths (feet or meters) and V(i) are current velocities (ft/sec or m/sec). at the corresponding depth.
- **P_SPECTRUM** defines either a wind or wave spectrum as a function of period. Here DATA is P(1), S(1), ..., P(n), S(n) P(i) is a period (sec.), and S(i) is the spectral value. Since the spectral values will later be scaled to get the proper zeroth moment, you can use any units you wish.
- **F_SPECTRUM**,defines either a wind or wave spectrum as a function of frequency. Here DATA is F(1), S(1), ..., F(n), S(n) F(i) is a Period (sec.), and S(i) is the spectral value. Since the spectral values will later be scaled to get the proper zeroth moment, you can use any units you wish.
- **M_GROWTH** defines the "marine growth" for elements. Here, the DATA is Z(1), ADD(1), ..., Z(n), ADD(n) where Z(i) is the depth and ADD(i) is the increase in element outside diameter (inches or mm) due to marine growth.
- **W_HISTORY**. defines a "wind history". Here DATA is a set of three "n" numbers T(1), V(1), ANG(1), ..., T(n), V(n), ANG(n). Here T(i) is the time, V(i) is the wind speed (knots) and ANG(i) is the direction from which the wind comes (degrees). Now, what MOSES does is to compute the mean wind speed of the history you input and subtract the mean from the input values. Now at each computation step, the deviation history speed is added to the mean. This speed and the history wind heading are then used to compute a wind force.

LT_MULTIPLIER is used to define load multipliers which vary with time. Here

DATA is T(1), V(1), ... T(n), V(n) where T(i) is the time and V(i) is the multiplier at that time. The TYPE of curve accepts the option:

–PERIODIC

If this option is specified, then the defined values will be repeated with a period of the last time in the list. If the option is not used, the last value in the list will be used for times larger than the last.

The next two curve TYPEs are used with connectors. **CT_LENGTH** is used to define the rate of change of the length of a connector. Here DATA is T(1), V(1), ... T(n), V(n) where T(i) is the time and V(i) is the rate of change of length (ft/sec or m/sec) at T(i). **EFFICIENCY** is used to define the propeller efficiency as a function of water particle velocity. Here DATA is V(1), E(1), ... V(n), E(n) where V is the water particle velocity (ft/sec or m/sec) and E is the efficiency.

The type **CS_VELOCITY** is used to define a drag coefficient which varies with the relative speed. Here the DATA is pairs of velocities and drag coefficients.

The last curve TYPE, **AM_PRESSURE**, is used to define added mass pressures as a function of submergence. Here DATA is a set of four "n" numbers S(1), AP_SURGE(1), AP_SWAY(1), AP_HEAVE(1), ..., S(n), AP_SURGE(n), AP_SWAY(n), AP_HEAVE(n). Here S(i) is the submergence (feet or meters) and AP_SURGE, AP_SWAY, and AP_HEAVE are the surge, sway, and heave added mass pressures (feet or meters) of water.

XII.F Sensors

Sensors are things which monitor conditions and when certain values are exceeded, alarms are set. Sensors are defined with the command:

&DESCRIBE SENSOR, SENSOR_NAME, –OPTIONS

and the available options are:

–**ON**, :SEL(1), :SEL(2), ...
–**OFF**, :SEL(1), :SEL(2), ...
–**DELETE**, :SEL(1), :SEL(2), ...
–**SIGNAL**, S_TYPE, S_SOURCE, S_DESIRED, S_VAL, S_B, S_N
–**CONVOLUTION**, CVL_NAME
–**DERIVATIVE**, YES/NO
–**LIMITS**, LIM_L, LIM_U
–**ACTION**, A_TYPE, RECEIVER

With the first three options, one does not need a SENSOR_NAME. These are used to turn on, turn off, or delete the sensors selected by :SEL(i). The other options are used to define a single sensor, SENSOR_NAME.

–**SIGNAL** is used to define the "signal" which the SENSOR will monitor. Here, S_TYPE the type of signal and it must be chosen from **TIME**, **POINT**, **VECTOR**, **C_LENGTH**, **C_FORCE**, **MIN_WT_DOWN**, **MIN_NWT_DOWN**, or **BODY_ANG**. S_SOURCE is the source of the signal. For signal types of time, no source is necessary. For a type of **POINT**, source is the name of the point you wish to monitor, for **VECTOR** it is the name of the two points which define the vector, for types of **BODY_ANG**, **MIN_WT_DOWN BODY_ANG**, and **MIN_NWT_DOWN** it is the name of the body to check, for either **C_LENGTH** or **C_FORCE** it is the name of the connector. For **BODY_ANG** there are three nominal values: a roll, a pitch, and a yaw. These are not the three Euler angles, but for small angles, they are a good approximation. Types of **MIN_WT_DOWN BODY_ANG**, and **MIN_NWT_DOWN** produce the minimum of either WT or NWT downflooding points for the body specified.

For types of TIME, MIN_WT_DOWN, and MIN_NWT_DOWN, this is all the data you need. For other types you need to define how to map the data into a single value. This is done with the remainder of the data. S_DESIRED is the desired value of the signal and may be omitted. It is used when sensors are connected to a control system. Here S_VAL is either **NORM**, or **VALUES** and S_B and S_N are integers. What the last three things do is to tell MOSES how to take the general data and transform it into a single number to monitor. For example

NORM 1 3

says to take a vector signal and to take the norm of the first three values as the signal. This may be useful when monitoring C_FORCE (connector force). Likewise,

```
    VALUE 3
```

Says to select the third component of the vector signal. This could be useful for monitoring the height of a point. If the **–CONVOLUTION** option is selected, then the raw signal is processed by the convolution before it is monitored. Also, the **–DERIVATIVE** option can be used to define a signal which is the velocity, relative velocity, or change in length of a connector instead of the default position and length.

–LIMITS is used to define a lower and upper bound of the signal. If These bounds are exceeded, then an alarm is sounded. During a time domain simulation, action is then taken based on the **–ACTION** option. Here A_TYPE must be chosen from **NONE**, **STOP_SIMULATION**, **STOP_WINCH**, **DEACTIVATE**, or **CHANGE_PROP**. and the action will be applied to RECEIVER. For example suppose one specified

```
    &DESCRIBE SENSOR M_LINE1 –SIGNAL C_FORCE LINE1 NORM 1 3 \
                        –LIMITS 0 50                \
                        –ACTION DEACTIVATE LINE1
```

Here MOSES would monitor the tension is connector LINE1 and when the tension exceeded 50, then the line would be deactivated.

Sensors can also be used statically with the **&INFO** string function. Here, there are two types available: **ALARM_SENSOR** and **VALUE_SENSOR** the first of these returns .TRUE. or .FALSE. depending on the alarm setting and the second results the value of the signal itself.

XII.G The Environment

The definition of the environment is a two step process. First, one defines any "auxiliary" data necessary, then he defines the environment referencing the auxiliary data by name. Most of the auxiliary data was discussed in the section on CURVES, Here, we will discuss the definition of the environment itself first, and follow immediately with a definition of how to define the auxiliary data not previously discussed. The environment is defined with the command:

&ENV, ENV_NAME, –OPTIONS

and the available options are:

–**DURATION**, DURATION
–**WATER**, RHOWAT
–**SPGWATER**, SPGWAT
–**PROBABILITY**, STAT, PDATA
–**SEA**, SEA_NAME, SEA_DIRECTION, HS, PERIOD, GAMMA
–**A_SEA**, SEA_NAME, SEA_DIRECTION, HS, PERIOD, GAMMA
–**SP_TYPE**, TYPE
–**SPREAD**, EXP
–**S_PERIOD**, TW(1), TW(2), ..., TW(n)
–**MD_PERIOD**, TD(1), TD(2), ..., TD(n)
–**TIME**, TOBSERV, DELTA_TIME, TTRA_SET, NCYCLES
–**RAMP**, RAMP_TIME
–**T_REINFORCE**, TB
–**WIND**, WIND_SPEED, WIND_DIRECTION
–**W_PROFILE**, WP_TYPE, EXP
–**W_DESIGN**, DTYPE, DURATION
–**W_SPECTRUM**, STYPE
–**W_HISTORY**, HISTORY_NAME
–**W_PERIOD**, TW(1), TW(2), ..., TW(n)
–**W_MD_CORRELATION**, FACTOR
–**CURRENT**, VC, CURRENT_DIRECTION
–**CURRENT**, PRO_NAME(1), CURRENT_DIRECTION(1), PRO_NAME(2), CURRENT_DIRECTION(2)
–**TIDE**, CHANGE
–**DEPTH**, WATDEP
–**M_GROWTH**, MG_NAME
–**T_PRESSURE**, TMP_NAME
–**USE_MEAN**, YES/NO

Here, ENV_NAME is the name of the environment. If it is omitted, then one is working with the "default" environment, and the precise action that MOSES takes will be different than if a name is specified. In either case, the system will be subjected to the environment specified by this command until another **&ENV**

command is issued.

If no ENV_NAME is specified, all of the environmental data is set to its default value. In essence, the default environment is a null environment with the density of water, the water depth, the ramp time, and the wind profile taken from those defined by an **&DEFAULT** command. The options are then processed which are used to alter these defaults.

If an ENV_NAME is specified, the action is a bit more complicated. In this case, MOSES will check the database to see if an entry corresponding to this name exists. If it does not exist, then a new entry is created which corresponds to the name, and the environmental data is set to the defaults. If an entry does exist, then the environmental data corresponding to that name is retrieved. MOSES then proceeds to process the options, which are used to alter the existing values. *Notice* that if this is a new name, you are altering the defaults, and if it is an existing name, you are altering the existing values. When the options have been processed, the new data is stored in the database for further use and modification.

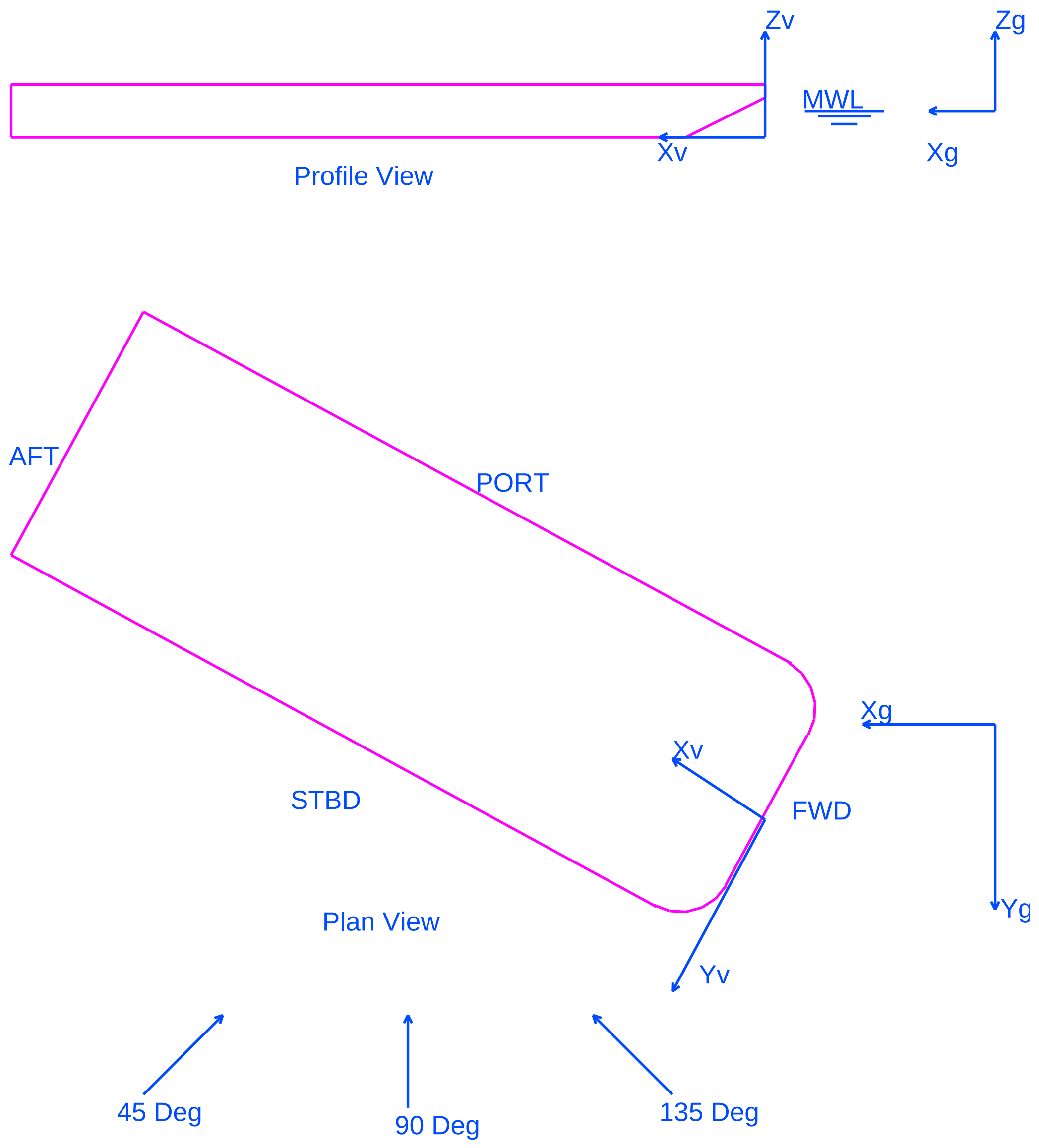
With the use of names, a user can save a considerable amount of input in certain situations. In fact, during Frequency Domain Post–Processing, several of the commands also act as **&ENV** commands. For this reason, there is some "environmental" data which may not appear to be environmental, unless one views environmental in a rather general sense.

For the options requiring an environment direction, SEA_DIRECTION, WIND_DIRECTION or CURRENT_DIRECTION, the direction is positive from global X towards global Y, as shown in Figure 2.

The –**DURATION** option is used for computing fatigue, and simply defines the length of time in days that the environment will act. Either of the two options –**WATER** or –**SPGWATER** serve to define the mass properties of water. Here, RHOWAT is the specific weight of water (pounds/ft**3 or newtons/m**3) and SPGRWAT is the specific gravity of water (density of water divided by density of standard water).

The –**PROBABILITY** option is used to control the statistics which will be defined when computing statistics of quantities in an irregular sea. STAT must be either: **RMS**, **SIGNIFICANT**, **1/10**, **MAXIMUM**, or **DURATION**. If **MAXIMUM** is selected, then PDATA is used to define the number by which the root mean square (RMS) will be multiplied to obtain the reported maximum. Here, if PDATA is omitted, a value of 3.72 will be used. If **DURATION** is selected, the probable maximum based on a specified duration will be reported. Here, one specifies the duration in seconds with PDATA.

To define the sea for a given environmental condition, one employs –**SEA**, –**A_SEA**, –**SPREAD**, –**SP_TYPE**, –**SPREAD**, –**TIME**, –**MD_PERIOD**, –



ENVIRONMENTAL HEADINGS
FIGURE 2

S_PERIOD, –**RAMP** and –**T_REINFORCE** options. In essence, the first two options defines the basic sea condition and the others serve to instruct MOSES how to treat the details. In MOSES, one can have several different seas which will be added together to form the sea state to which the model will be subjected. –**SEA** is used to define the first sea, and –**A_SEA** is used to define subsequent ones. Other than this distinction, the two options are identical. For either –**SEA** or –**A_SEA**, SEA_NAME is the name of the sea. It must be either **REGULAR**, **ISSC**, **JONSWAP**, **2JONSWAP** or a name defined as a "curve" with **&DATA CURVE P_SPECTRUM**, **&DATA CURVE F_SPECTRUM commands, in the &DATA** with **GRID**. Here, HS is the significant height (feet or meters), PERIOD is the period (sec.), SEA_DIRECTION is the mean heading (deg.), and for a name of **JONSWAP**, GAMMA is the "peakedness factor". The precise definition of PERIOD depends upon the value of TYPE specified with the –**SP_TYPE** option or which was specified on a **&DEFAULT** command. If TYPE is **PEAK** then period is the period at which the spectrum has a maximum. If it is **MEAN**, then the period at which the spectrum peaks is given by

Tp = 1.2958*PERIOD

Notice that for an **ISSC** spectrum (GAMMA = 1), PERIOD is the mean period. *For values of GAMMA other than one, PERIOD has no meaning other than that given above.*

If one is using a sea defined by a wave grid, then the period and height are ignored, and for anything other than **JONSWAP**, GAMMA is ignored. When an input spectra is used, the input values are scaled so that the sea will have the specified significant height. Also with an input spectrum, one can omit the PERIOD, and the spectrum will be as input. If one does add a period, the spectrum will be scaled so that the peak frequency will move according to an **ISSC** spectrum.

All of the **ISSC**, **JONSWAP** and **2JONSWAP** spectra can be represented by the equation:

S(f) = alpha * 172.8 * HS**2 * z * exp [–1948.184*z] * p / f

where f is the frequency in rad/sec.,

z = 1. / (f * Tp) **4
p = GAMMA ** exp [(beta – 1) **2 / (2 * sigma**2)]
beta = Tp * f / (2 * pi)

sigma is .07 for frequencies less than the peak and .09 for those greater than the peak, and alpha is a parameter computed to make the total area under the spectrum equal to HS**2 / 16. With these spectra, the significant height is twice the square root of the area. For **2JONSWAP**, the value of GAMMA is not input, instead, it is obtained from the DNV guidelines. Unfortunately, their relationship between GAMMA and the "slope" is dimensional. What we have is

```
s = Tp * sqrt( g/(2*pi*Hs) )

GAMMA = 5                for s < 4.5
GAMMA = 1                for s > 6.2
GAMMA = exp( 5.86 – .94 s) for 4.5 <= s <= 6.2
```

There is almost never a need to define a spectrum by its ordinates. Many users think that because someone gives them a Bretschneider or a PM spectrum, that they need to resort to such measures, but this is not necessary. The problem comes from the lack of consistency in the manner in which sea data is provided. All of the sea spectra we know of have the same form as the one above. The difference is in the manner in which the period variable is specified. Sometimes, a peak period is specified, sometimes a significant, sometimes a mean, other times none at all. The easiest case is when one is given a PM or Pierson Moskowitz spectrum and no period is specified. Here, one can use

```
PERIOD = sqrt ( ( 23.27 * 2 * pi * HS ) / g )
```

which says that the ratio of mean wave length to height is 23.27. In other cases, one should find where the specified spectrum has a peak and use the result above to obtain PERIOD.

The –**SPREAD** option controls the spreading of the seas, and EXP is the exponent which defines the wave spreading function, F:

```
F(THET) = COS(H – THET)**EXP  for –pi/2 < THET < pi/2

F(THET) = 0                   Otherwise
```

Here H is the mean wave heading. If the –**SPREAD** option is not used, a value of 200 will be used for EXP. The –**MD_PERIOD** option defines the periods, TD(I), at which wave drift forces will be generated when performing a time domain simulation. If they are omitted, the values specified via **&DEFAULT** will be used. Similarly, the –**S_PERIOD** option defines a set of periods which will be used to synthesize wave frequency excitation. If they are omitted, the values specified via **&DEFAULT –PERIOD** will be used. In the case when a spectrum will be used to synthesize a time domain sample of a sea, the –**RAMP**, –**TIME**, and –**T_REINFORCE** options are used to control the synthesis.

The variable RAMP_TIME of the –**RAMP** option defines the time interval over which the sea will be linearly ramped from zero to its full value. If either option –**RAMP** or –**T_REINFORCE** are omitted, the values defined via **&DEFAULT**

will be used.

The **–TIME** option sets the observation time and time increment of the synthesis. Here, TOBSERV, is the end time of the synthesis and DELTA_TIME is the time increment. TTRA_SET can be used to set a time translation. If TTRA_SET is a number, then this number translates time equal zero to this time. The simulation will still begin at zero, but the environment at time equal zero will have the same as that at the specified time. If TTRA_SET and CYCLES are omitted, a translation of zero will be used. If, however, TTRA_SET is **FIND** then MOSES will attempt to find a time so that during the synthesis a peak will be found which corresponds to the probable maximum during a simulation with NCYCLES. In essence, the following algorithm is used:

- Compute the wave amplitudes,
- If the amplitude is within 85 percent of the desired value, keep it as a viable peak,
- Keep finding peaks until there are 10 candidates.
- Pick the candidate which is closest to the desired value.

It is possible that a ”good” solution to this problem will not be found. In particular, if NCYCLES is too large then no peak greater than that specified may be found. The other extreme is if NCYCLES is too small in comparison to the observation time. Here, it may not be possible to get a time interval in which a peak will not be too large. The above scheme will, however, normally yield a result which is larger than that requested.

The option **–T_REINFORCE** is used to pick the phases of the frequency components. If the sea was defined as a regular wave, then the phase is zero. If, however, the sea was specified via a spectrum another method is used. Here, the phases are chosen as

phi(i) = –TB * w(i)

Here, phi(i) is the phase for the ith component, w(i) is the frequency (rad/sec) of the ith component, and TB is a time. By using this scheme, you always get the same sea, and two time domain simulations will yield identical results. The time, TB, has an interesting interpretation – it is a time at which all of the components reinforce, i.e. at this time the amplitude of the sea is the sum of all of the Fourier coefficients. Depending on the number of components you use, this sum is ”unreasonably” large and this time is to be avoided during the simulation. Thus by default, it is set to be a large number. If you need to change it to perform Monte Carlo simulations, use **–T_REINFORCE**. Finally, if TB is negative, then the phases will be chosen randomly using the absolute value of TB as the seed.

The wind is defined with the options **–WIND**, **–W_PROFILE**, **–W_PERIOD**, **–W_DESIGN** **–W_SPECTRUM** **–W_HISTORY**, and **–W_MD_CORRELATION** options. The **–WIND** option defines the wind speed, WIND_SPEED, which is the mean wind speed in knots, and WIND_DIRECTION is the direction from which the wind blows. The **–W_PROFILE** option defines how the wind will vary with height. Here, WP_TYPE can be **ABS**, **API**, **NPD** or **POWER**. If TYPE is **ABS**, then height coefficients will be computed according to the ABS rules. If WP_TYPE is **API** or **NPD**, the wind variation with height will be according to the specified code. If, however, WP_TYPE is **POWER**, then EXP is the power which defines the power law for wind variation. In other words, if EXP is equal 1/7, then the wind will vary with height according to a 1/7 power law. To specify the type and duration of the design wind speed, the **–W_DESIGN** option is used. Here, DTYPE can be either **API** or **NPD**, and DURATION is the design wind speed average in seconds.

To define the temporal or frequency behavior of the wind, one uses either the **–W_SPECTRUM** or **–W_HISTORY** options. The first of these defines the type of wind spectrum to be used. For this option, STYPE can be **API**, **NPD**, **HARRIS**, **DAVENPORT OCHI**, or the name of a spectrum previously defined with **&DATA CURVE** with a TYPE of either **F_SPECTRUM** or **P_SPECTRUM**. Most of these standard spectra can be found in a paper "Wind Turbulent Spectra for Design Consideration Of Offshore Structures", OTC 5736, by Ochi and Shin. The API spectrum is taken directly from the API–RP2a 21st edition and the NPD spectrum is taken from NORSK STANDARD N–003, "ACTIONS and ACTION EFFECTS" Rev1, February 1999. It should be noted that there is a contradiction between the turbulence factor and the wind spectrum for the **NPD** spectrum. For this case, MOSES uses the spectrum for the shape and the intensity for the variance.

Instead of using a spectrum, the time variation of the wind can be defined with the , **–W_HISTORY** option, This option instructs MOSES to use a previously defined wind history of HISTORY_NAME. This history is defined with **&DATA CURVE W_HISTORY**.

The **–W_PERIOD** option defines the periods, TW(I), at which wind forces will be generated when performing a time domain simulation. If they are omitted, the values specified via **&DEFAULT** will be used. Finally, **–W_MD_CORRELATION** defines the relative phase of the wind velocity components with the wave drift components. If FACTOR is 0, then the two have phases 90 degrees apart, if FACTOR is one, then the two phases are the same. Intermediate values of FACTOR produce relative phases between these two extremes.

The current is defined via the **–CURRENT** option. There are two styles for this option. With the first, one simply specifies a current speed, VC (ft/sec or m/sec), constant with depth and the direction from which it comes, CURRENT_DIRECTION (deg.). The second method is used to define more complicated currents. Here, PRO_NAME(i) is the name of a current profile defined in the **&DATA CURVE C_PROFILE** and CURRENT_DIRECTION is again the direction from which the current comes. Notice that one can define several different profiles from different directions. MOSES will combine each of the profiles to yield the true current velocity at any depth.

The tide is defined by the **–TIDE** option where CHANGE is the change in (feet or meters) of the water level. The water depth is set with the –**DEPTH** option, using units of feet or meters.

The marine growth on the structure is specified by the –**M_GROWTH** option, where MG_NAME is a name which has been defined via the **&DATA CURVE M_GROWTH** command.

If one wishes to include internal pressure or temperature effects during a structural analysis, he must specify the internal pressure and temperature distribution. This is accomplished with the **–T_PRESSURE** option with TMP_NAME being a name which has been previously defined via the **&DATA** Menu with a **T_PRESSURE** command.

When an environment is used for computing forces or stresses spectrally, one can either combine the deviation with the mean, or omit the mean by using the –**USE_MEAN** option depending upon whether or not YES/NO is **YES** or **NO**. If this option is omitted, then the mean will be used.

As mentioned previously, some of the environmental data must be defined in a separate menu. This is accomplished by issuing

&DATA ENVIRONMENT

One can then define basic environmental data. At the conclusion of defining environmental data, one should then issue **END_&DATA** to exit the menu.

The command

T_PRESSURE, TMP_NAME, OBJECT(1), TMP(1), INP(1), GH(1), SC(1), \
OBJECT(2), TMP(2), INP(2), GH(2), SC(2), ..

defines a temperature and internal pressure distribution within the elements which can be referenced by the name TMP_NAME. Here, OBJECT is the name of the object to which the temperature TMP(i) (degrees F or degrees C) and internal pressure distribution will be applied. The pressure distribution is defined by a given gage pressure (ksi or mpa), INP(i), measured at a global height GH(i) (feet or meters), and SC(i) is the specific gravity of any fluid contained inside the element. If OBJECT(i) is two node names (they may include wild characters, but must begin with a *), the attributes will apply to all beams between those two nodes. If OBJECT(i) is an attribute class name (begins with a ∼), then the attributes will apply to all elements which belong to classes which match

OBJECT(i). If OBJECT(i) does not begin with either an * or a ∼, then the attribute will be applied to all members whose names match OBJECT(i).

The command

ENVIRONMENT, ENV_NAME, –OPTIONS

stores a complete environment in the database. The options for this command are the same as those for **&ENV**, described earlier.

The command

S_GRID, GRID_NAME, GRID_TYPE, DEPTH, HEIGHT, PERIOD

is used to define a wave grid. Here, GRID_TYPE is the type of wave used to generate the grid and it must be either: **REGULAR**, **STOKES**, **STREAM**, or **INPUT**. For this command, GRID_NAME is the name of the wave grid, and DEPTH, HEIGHT, and PERIOD are the water depth (feet or meters), wave height (feet or meters), and period (sec.) used in generating the wave as shown in Figure 3. If one of the first three values is specified, MOSES will automatically generate a wave grid using either a regular wave, a Stokes fifth order wave, or a wave generated by a stream function algorithm. If **INPUT** is specified for GRID_TYPE, then the user is placed into another sub–menu where the grid can be defined. This is accomplished via the commands:

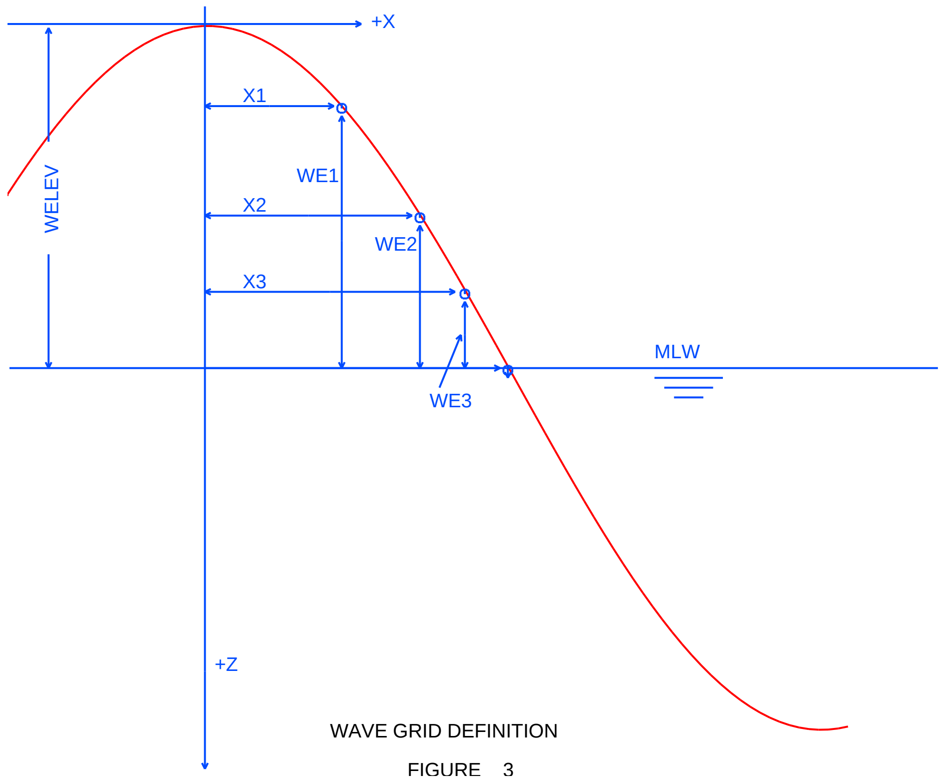
HORIZONTAL, X, WELEV
VERTICAL, Z(1), VX(1), VZ(1), AX(1), AZ(1), ...
END_INPG

The HORIZONTAL command defines the locations along the wave where the grid is to be defined. X is the distance from the wave crest (feet or meters), and WELEV is the elevation of the wave above mean water level (feet or meters). Once the horizontal location has been defined, one should input as many VERT commands as necessary to define the wave velocity and acceleration. Here, Z(i) is the depth of the point below the mean water surface, negative for points above the mean water surface, VX(i) and VZ(i) are the horizontal and vertical wave velocities (ft/sec or m/sec), and AX(i) and AZ(i) are the horizontal and vertical wave accelerations (ft/sec**2 or m/sec**2). When the grid has been completely defined, one should issue END_INPG to return to the **ENVIRONMENT** sub–menu.

There is a string function:

&ENV(INFO)

Which returns information about the current environment. Here INFO must be **S_HEIGHT**, **S_PERIOD**, **S_GAMMA**, **S_CREST**, **S_HEADING**, **S_SP_TYPE**,



WAVE GRID DEFINITION

FIGURE 3

S_PERIOD, **W_SPEED**, **W_DIRECTION**, **W_SPECTRUM**, **W_DESIGN**, **W_PROFILE**, **W_PERIOD**, or **C_SPEED**, **C_DIRECTION**, **WAT_DEPT**, **TIDE**, **WATER**, **MD_PERIOD**. **T_OBSERVE**, **T_INCREMENT**, **RAMP**, **T_REINFORCE**, The values beginning with S_ return information about the sea, the ones beginning with C_ return information about the surface current, the ones beginning with T_ and RAMP return information about the time, the ones beginning with W_ return information about the wind, and the ones beginning with S_ return information about the sea. **WAT_DEPT** returns the water depth and **WATER** returns the specific gravity of water.

XII.G.1 Durations

The purpose of a DURATION is to associate a time of exposure to an environment. With mooring line fatigue this is done directly on the &ENV command itself. For all other cases it is done in the **&DATA DURATION Menu**. This is done in one of two ways depending on the type of analysis performed.

For either type of analysis, the following command is used.

STRPOST
    &DATA DURATION, DURATION_NAME, PRC_NAME, –OPTION

where the only available option is:

    –TDOM, DUR_TIME

Here DURATION_NAME is the "duration name". When asking for fatigue results one selects the durations to be applied, and PRC_NAME is the process name for which the duration will be applied. The **–TDOM** option signifies that one wishes to perform time domain fatigue on the process PRC_NAME and that the total duration of this process is DUR_TIME in days. This is all that is required for a time domain process.

When **DURATION** is used without the **–TDOM** option, a sub–menu is entered where frequency domain fatigue data will be defined. In other words, for frequency domain the DURATION command is followed by a series of records of the form:

    WTIME(i), –OPTIONS

Where the available options are:

    –SEA, SEA_NAME, THET, HS, PERIOD, GAMMA
    –A_SEA, SEA_NAME, THET, HS, PERIOD, GAMMA

Here, WTIME(i) is the time (days) which the system will be exposed to the seastate defined by the remainder of the record. Here the parameters SEA_NAME, THET, HS, PERIOD, and GAMMA are the same as those used to define a sea with the **&ENV** command. If a duration environment is defined with more than one spectrum then MOSES provides two ways to compute the "average period". The choice is governed with the **–T_AVERAGE** option of the **&PARAMETER** command.

When all of the duration has been defined, one exits the sub–menu with an

    END_DURATION

command. Notice that one can perform both time domain fatigue and frequency domain fatigue in the same run, and both types of fatigue can be computed for the same process.

XII.H Fatigue and Cycle Counting

Fatigue is a particularly complex topic, as there are several components to consider:

- The SN curve, which defines the number of cycles of a given maximum (hot spot) stress the material can withstand without breaking,
- The Stress Concentration Factors (SCF) which define maximum stresses in terms of the nominal ones, and
- The duration which defines lifetime history of the environment to which the structure has been exposed.

Normally the result of interest is a Cumulative Damage Ratio (CDR). This is simply a sum of ratio of the fraction of the life used for all stress ranges. Sometimes, however, one wishes to view the loading history independently of the SN curve. Thus, MOSES has the capability for counting either load cycles or stress cycles sorted into bins. One can compute fatigue (or cycles) for BEAMs, PLATES, tubular JOINTs and mooring lines and the details will vary with type. In general, one can compute fatigue in either the time or frequency domains.

One is really interested in fatigue at all points in an element, but in reality we only consider it at a set of "fatigue points": the ends, at points along a beam where the section changes, or at welds. Generalized plates are special in that fatigue is computed at the centroid of each subelement.. To really complicate these matters, some fatigue points have automatic methods to compute stress concentration factor and others do not. In particular, MOSES has automated methods for computing stress concentration factors for: tubular connections, tube/cone connections, and tubular joints. Stress concentration factors must be manually associated with all other fatigue points.

In MOSES there are two ways to compute fatigue damage, on an element by element basis, or on a tubular joint basis. The reason for the two methods is that for tubular joints, there is a body of knowledge for automatically computing the SCFs and the associated hot spot stresses. For non tubular joints, the information is much less extensive. Thus, for tubular members, one can do joint fatigue to capture the damage at the ends, but one must do element (beam or generalized plate) fatigue to get CDRs at intermediate locations. Also, element fatigue must be used to get the CDRs at the ends elements which are not part of tubular joints.

As with stress concentration factors, an SN curve must be associated with each fatigue point. Again, tubular joints are special in that one normally has only two choices for SN for a tubular joint and the association of SN is different for doing JOINT fatigue than it is for doing BEAM fatigue.

The definition of the environmental history depends on whether a time domain or a frequency domain simulation is being used. For frequency domain, a set of RAOs are computed and they are used along with a scatter diagram of environments which act for a specified time. In the time domain, one time domain simulation is performed per process. Load cases are defined at a reasonable number of times during this simulation and the system solved for the time traces of the stresses. A Rainflow Counting technique as outlined in ASTM E–1049, "Standard Practices for Cycle Counting in Fatigue Analysis" is then used to compute the stress cycles and perhaps the cumulative damage. These results are "scaled" by the ratio of duration time to simulation time. The results are summed over the selected durations, so one can compute fatigue in both domains and over all lifetime situations.

If a duration environment is defined with more than one spectrum then MOSES provides two ways to compute the "average period". The choice is governed with the **–T_AVERAGE** option of the **&PARAMETER** command.

In the following we specifically discuss the role of the **&REP_SELECT** command, but any of the options discussed for this command can be issued on the option which requests fatigue or cycle information. In other words:

```
&REP_SELECT –SN X
JOINT_POST FATIGUE
```

produces the same result as

```
JOINT_POST FATIGUE –SN X
```

The same can be said of the **BEAM_POST** or **PLATE_POST** commands.

XII.H.1 Defining SN Curves

Before any fatigue can be computed, an SN curve must be defined. This is accomplished with the command:

&REP_SELECT –OPTIONS

and the available options are:

–**SN**, CURVE, TYPE, S(1), N(1), ... S(n), N(n)
–**THICK_SN**, TO, POWER, MAXCOR, FLAG
–**S_IMP_FACTOR**, SFACTOR
–**F_STRESS**, S_MULT

Once a curve has been defined, one need only specify

–SN, CURVE

to select, CURVE, for use in computing cumulative damage ratios. MOSES has a set of curves automatically generated and hence these do not need to be defined to be used. The automatic ones are: the

- **API_X** and **API_XP** curves from API–RP2A 21st edition, the
- **API_WJ**, **API_WJG** **API_WJP**, **API_WJ_WET**, **API_WJG__WET**, **API_WJP_WET**, **API_CJ**, **API_CJG**, and **API_CJP** from API_RP2A 2nd supplement to the 21st edition, the
- **API_CHAIN**,and **API_WIRE** curves from API–RP2F,
- **HSE_TP** curve from HSE, and the
- **DNV_B1**, **DNV_B2**, **DNV_C**, **DNV_C1**, **DNV_C2**, **DNV_D**, **DNV_E**, **DNV_F**, **DNV_F1**, **DNV_F3**, **DNV_G**, **DNV_W1**, **DNV_W2**, **DNV_W3**, **DNV_T**, **DNV_B1_WET**, **DNV_B2_WET**, **DNV_C_WET**, **DNV_C1_WET**, **DNV_C2_WET**, **DNV_D_WET**, **DNV_E_WET**, **DNV_F_WET**, **DNV_F1_WET**, **DNV_F3_WET**, **DNV_G_WET**, **DNV_W1_WET**, **DNV_W2_WET**, **DNV_W3_WET**, and **DNV_T_WET** curves from DNV RP–C203 (2008).

For the WJ and CJ curves, the ones with "_G" include weld improvement due to grinding and the ones with "_P" include improvement due to hammer peening.

To define a curve, one first specifies TYPE which defines the type of curve. TYPE should be **STRESS** for a normal curve, or **TENSION** for a curve like the WIRE and CHAIN curves. The values S(i) and N(i) define the stress (or tension / break tension) and the corresponding number of cycles. *After* the curve is defined, one uses the **–THICK_SN** option to define a correction which depends on thickness. Most documents say that this is a reduction of the SN curve, but in MOSES, it is viewed as an increase in SCF. Thus, our factor is greater than one and is the inverse of what would be a reduction of SN. The only real difference is that POWER here is *positive* where with a reduction it is negative. Here, TO, is the nominal thickness (inches or mm) and POWER is the power of the correction; i.e. the curve will be corrected by a factor, FACT, where:

FACT = MIN ((THICK/TO)**POWER, MAXCOR)

and THICK is the thickness. The details here are governed by FLAG. If FLAG is **USE_BRACE**, then FACT computed as above with THICK equal to the brace thickness is used for both the brace and the chord. Otherwise FACT is computed for both the brace and the chord using their respective thicknesses. Here, MAXCOR can be omitted and it will be set to "infinity".

The **–S_IMP_FACTOR** option is used to define a "stress improvement factor". This factor simply reduces the SN curve by SFACTOR; i.e. the stress in the SN curve is divided by SFACTOR.

Neither the API X nor XP curves built into MOSES have any thickness correction! The reason is that there is quite a bit of ambiguity in API–RP2A about how to treat this question. In particular, depending on the details, no correction may be needed. If you do need it, however, it can be included with:

```
&REP_SELECT  –SN X –THICK_SN  1 .25 1.5 USE_BRACE
```

This tells MOSES to use a correction on any brace having a thickness greater than 1 inch with a power of .25 up to a maximum factor of 1.5 and to use the brace factor for the chord. The 1.5 maximum comes from the fact that RP2A says that the X curve need not be reduced below the XP curve, and the X curve reduced by a factor of 1./1.5 has the same endurance limit as the XP curve.

The **–F_STRESS** option is a simple way of overcoming a deficiency in most of the published SN curves; they are only defined over a limited range of cycles. Most of the time, this does not matter, but the algorithm MOSES uses strictly considers only damage which occurs in the specified range. Without any further action, a beam with high stress might actually accumulate *less* damage if the stress were *increased*! This occurs because increasing the stress moves more of the harmful cycles outside the range of the SN curve. The value S_MULT is a multiplier which MOSES multiplies by the RMS of the stress spectrum to check for this bad behavior. If the product S_MULT * RHS is greater than the stress at the first point of the SN curve, then the stochastic integration is *not* performed, but damage is accumulated based on the first point in the SN curve. This is simply an estimate to keep one out of trouble. By default, S_MULT is 3.72, but you can set it to 0 to strictly obey the standard algorithm.

XII.H.2 Associating SN Curves with Points

An SN curve must be associated with each fatigue point. The details differ based on the type of fatigue being computed, but in all cases it is accomplished with an **–SN** option. For mooring line and tubular joint fatigue, the "current" SN name will be used; i.e. the last name specified on either a &REP_SELECT command or its equivalent.

For elements (other than mooring lines) there are three ways to define them: on a &DEFAULT command, on the ~CLASS command, or on the element definition command. In any case failure to include a definition results in the last definition being used; i.e.

- The –SN specified on the element command is used,
- If no –SN is specified on the element command, than specified on the class definition is used,
- If no –SN is specified on the class definition, then that defined on the &DEFAULT is used.

For the &DEFAULT command, the default SN curves at various points on an element are defined with the option

```
–SN, TYPE(1), SN1_A, SN1_B, SN1_R, \
     TYPE(2), SN2_A, SN2_B, SN2_R, ......
```

Here, TYPE(i) defines a section type and must be either: **TUBE**, **CONE**, **BOX**, **WBOX**, **PRI**, **IBEAM**, **G_IBEAM**, **TEE**, **CHANNEL**, **ANGLE**, **D_ANGLE**, **LLEG**, or **PLATE**. SN1_A is the SN curve name for the A end of the element, SN1_B is for the B end, and SN1_R is for all remaining segments. This sequence can be repeated for as many section types as desired. Also, if only one SN curve name is provided, the same SN name is used for all locations along the element. If two SN names are specified, the first name applies to the "A" end of the element, and the second SN names applies to the "B" end and all segments in between. Of course, the SN curve specified here must be defined to MOSES using the appropriate commands, such as **&REP_SELECT**.

For the element and class definition commands, the situation is a bit different. Here, we know how many fatigue points we have so we can directly define an SN curve for each one of them. Thus, for classes, we have the option for each segment of

```
–SN CURVEA, CURVEB
```

Here, CURVEA is the SN curve which will be used at the beginning of the segment and CURVEB is the curve used at the end of the segment. If one uses two names for each segment, the one in the middle will be defined twice. If this is done, the second definition will actually be used; i.e. the first curve specified will, in fact, be used for the beginning of the segment. Also, CURVEB needs to be specified only on the last segment.

For elements, we have

–**SN**i, CURVE

which defines the SN curve for the "ith" fatigue point of the element to be "CURVE". Here, –**SNA** defines curve at the A end, –**SNB** defines curve at the B end, –**SN1** defines curve at the end of the first segment, etc.

XII.H.3 Associating SCFs with Tubular Joints

Like SN curves a set of stress concentration factors (SCFs) must be associated with each fatigue point. The details differ based on the type of fatigue being computed. For tubular joint fatigue, the SCFs will normally be computed based on the joint geometry and the load path and is controlled by a set of options on the point definition or **ED_POINT** command, and the default for all of these parameters is specified on similar options of the **&DEFAULT** command. The applicable options are:

–**CO_SCF**, SCF_TYPE
–**LEN_FACTOR**, FRACHOL
–**MAX_CHD_LEN**, MAXCHOL
–**CHD_FIXITY**, CHD_FIX
–**SCF_BOUNDS**, MIN_SCF, MAX_SF

This computation is controlled by the –**CO_SCF** option. The valid values for SCF_TYPE are **K&S**, **API**, **MARSHALL**, and **EFTHYMIOU**. With the automatic computation of SCFs, two values are computed: one for the root of the weld on the brace side, and one for the root of the weld on the chord side. The –**LEN_FACTOR**, –**MAX_CHD_LEN**, and –**CHD_FIXITY** options define parameters which are used by some of the methods in computing the SCFs. For some methods, the SCFs depend upon the length of the chord. FRACHOL is a fraction of the actual chord length which will be used. The default value of FRACHOL is one. MAXCHOL is the maximum length which will be used and its default value is "infinite". Suppose that the chord length was 200. If FRACHOL is set to .5 and MAXCHOL is 50, the program will first set the length to be used to .5*200. It will then take the maximum of 100 and 50 and use this value in the computation. The –**CHD_FIXITY**, option defines the "chord flexibility" CHD_FIX. This is a measure of the bending support at the ends of the chord. It should be a number between .5 and 1. The first of these corresponds to the chord ends being pinned and the later fixed. The –**SCF_BOUNDS** option defines minimum and maximum values for SCFs. Any computed SCF smaller than MIN_SCF or greater than MAX_SCF will be replaced with the limit.

If stiffeners are associated with either of the two chord classes, then they will be used in computing the "crushing" of the joint, in computing the code check for the joint, and in computing the stress concentration factor for the joint. The number of stiffeners will be the sum of the number for both chord segments. For situations where two classes define the chord with different stiffener attributes, then the properties of the last one encountered will be used. Here, the stiffeners are "smeared" over the effective length (or true brace footprint of the joint). The SCFs computed by whichever method are reduced according to "Lloyd's Register of Shipping Recommended Parametric Stress Concentration Factors for Ring_Stiffened Tubular Joints".

XII.H.4 Associating SCFs with Element Points

In computing fatigue in elements, the element SCFs are used. For elements there are four ways to define them: on a **&DEFAULT** command, on the class definition command, on the element definition command, or in some cases from the tubular joint SCFs. In any case failure to include a definition results in the last definition being used; i.e.

- For intermediate fatigue points between tubular sections, an SCF will be computed based on the algorithm specified on the –IN_SCF option on the &PARAMETER command,
- For intermediate fatigue points between tubular and conical sections, an SCF will be computed as specified in API RP2A or DNV (they are the same). MOSES will also check for two tubes that have no braces and will compute an inline SCF as if the change in section was internal to the beam itself. For other cases, the –SCF specified on the element command is used, If no –SCF is specified on the element command, then that specified on the class definition is used, If no –SCF is specified on the class definition, then that defined on the &DEFAULT is used.
- In the case where one of the ends of a tubular element is part of a tubular joint, then a part of the joint SCF will be used for the SCF of the element at that end.

To define the "inline" SCF between two tubular sections, then the

–IN_SCF, TYPE

option of **&PARAMETER** is used. There type must be either **DNV** or a number. The number is the exponent, e, used in computing the SCF according to:

SCF = 1 + A * B

A = 3 (T2 – T1)/ T1

B = T1**e / (T1**e + T2**e)

Where T1 and T2 are the tubular thickness. If DNV is specified a value of 2.5 will be used for e. Here T2 is greater than T1.

For the **&DEFAULT** command, the default SN curves at various points on and element are defined with the option

–SCF, TYPE(1), SCF(1), ...

The **–SCF** option defines default stress concentration factors for different types of section. Here, TYPE(i) defines a section type and must be either: **BOX**,

PRI, **BU**, **W**, **M**, **S**, **HP**, **WT**, **MT**, **ST**, **L**, **C**, **MC**, **WBOX**, **DL**, **LLEG**, or **PLATE**. The values SCF(i) are the stress concentration factor for the corresponding type. Please notice that one cannot define a default SCF for a tube. This would defeat the normal computation of SCF's for joint fatigue. If you have a tube, want to do beam fatigue, and want an SCF other than 1, you need to manually do it on the command which defines the beam.

For class definition commands, we have the option:

–**SCF** SCF_BEG, SCF_END

Here, SCF_BEG is the SCF which will be used at the beginning of the segment and SCF_END is the curve used at the end of the segment. If one uses two SCFs for each segment, the one in the middle will be defined twice. If this is done, the second definition will actually be used; i.e. the first SCF specified will, in fact, be used for the beginning of the segment. Also, SCF_END needs to be specified only on the last segment.

The SCF definition for elements is accomplished with the –**SCF** option which is followed by from one to eight numbers. If there is no SCF option, then the default SCF defined for this class will be used. Here, –**SCFA** will change the values at the first end of the beam –**SCFB** will change them at the other end, and –**SCF1** will change them at the intersection between the first segment and the second, etc. If –**SCF** is used, then the stress concentration factor will be the same at all fatigue points. For the ends of the beam, the form of this option is:

–**SCF**i, VALUES(1), ... VALUES(n)

One can input 0, 1, 3, 4, 8, or 24 values. Zero values sets the SCFs to the default values. If 1 values is input, then it will be used as the SCF for axial and for strong and weak axis bending. If three values are input then they are the axial, strong axis, and weak axis binding SCFs.

The remaining numbers (4, 8, or 24) are used to override the computed SCFs when the beam is part of a tubular joint. The first four values here define the SCFs at the brace at the saddle, crown, in plane bending and out of plane bending. If only four values are given, then these will be repeated for all three load classifications and for the chord. With eight values, the second four values define the SCFs for the chord and the brace/chord values will be used for all joint classifications. If twenty four values are specified, the first twelve are for the chord in K joints, T&Y joints and X joints, and the last twelve are the corresponding values for the chord. K, T&Y, and X. (See section on Associating SCFs with Tubular Joints for more information).

Also, the "joint SCFs" are used when computing beam fatigue. Here, the maximum of the crown and saddle SCFs and the maximum inplane and out of plane

SCFs for all classifications will be used at the end when computing beam fatigue. This will result in beam fatigue which will be a bit less that that computed at the same end by joint fatigue, but it is not excessively conservative for slamming fatigue.

XII.H.5 Beam Fatigue Due to Slamming

If one is computing fatigue in the time domain, then the damage due to slamming is correctly included. When fatigue is considered in the frequency domain, however, the situation is quite different. Here, an element which is out of the water in the mean position occasionally enters the water and a slam event occurs. MOSES has a special algorithm to compute fatigue due to these slamming events.

This computation depends on the parameters specified on either the **BEAM_POST** command or the

> **&REP_SELECT** –OPTIONS

and the options applicable here are:

> –**SLA_COEFFICIENT**, S_COE
> –**SLA_FIXITY**, S_FIXITY
> –**SLA_DAF**, S_DAF
> –**SLA_CDAMP**, S_CDAMP
> –**SLA_MULTIPLIER** S_VEL(1), S_MUL(1), ... S_VEL(n), S_MUL(n)

The data will be discussed below, and only beams that are out of the water in the mean position and of which are allowed to have forces due to added inertia will be considered for beam fatigue.

First, notice that here we do not have the normal frequency domain phenomenon. Instead, we have an occasional impulsive load which acts on the beam, and when the load is removed, then beam experiences a free vibration decay. Thus, according to DNV, the force per unit length on a beam due to slamming is:

> W = .5 * rho * D * S_COE * abs (V * Vv)

Where rho is the water density, D is the tube diameter, and the option –**SLA_COEFFICIENT** defines S_COE is the slam coefficient (nominally 3), and V is the relative velocity between the beam and the water, and Vv is the relative velocity vertically. Now, the maximum bending moment in the beam, assuming that the load is uniformly applied over the length, can be written as

> M = S_FIXITY * W * L * L

Here the –**SLA_FIXITY** is used to define S_FIXITY which is a number that depends on the fixity of the beam. It is 1/8 if the beam is simply supported and 1/12 if it is built in. This, in turn gives a maximum stress in the beam of

> s = S_DAF * SCF * R * M / I

Where R is the beam radius, I is the section inertia, SCF is the stress concentration factor, and the option –**SLA_DAF** defines S_DAF is the dynamic amplification factor which is nominally 2.

After the impulse is applied and released, the beam will freely vibrate with decreasing amplitude

s(k) = s(k–1) * exp (–200*pi/S_CDAMP)

Here S_CDAMP is the percentage of critical damping defined with the **–SLA_CDAMP** option. Now, the total damage due to a single impulse is

CDR = SUM [1./ N(s(k))]

Here, N is the number of allowable cycles at the stress s(k), and the sum continues until there is no further damage.

There remain two unanswered questions: the first is how many slam events do we have and what are the velocities associated with each slam event. The first question is easily answered if one assumes that the slam events are Raleigh distributed. In this case, one simply computes the number of times in a given time that the probable motion will exceed the mean. This gives us a slam velocity or number of slams per hour. The second question is not so clear. Suppose that we write the velocity as a multiple of the RMS.

V = f * Fmax * Vrms

Here V is the velocity which will be used, Fmax is the multiplier which gives the "maximum" event from the RMS, and f is a factor. If the slam velocity is "low" then the slam events are associated with extreme events and the multiplier f should be 1 so that the velocity used is the "maximum" velocity. If on the other hand the slam velocity is high, then slams are not rare events and the multiplier should be one appropriate to a normal event such as 1/Fmax. MOSES uses an linear interpolation for f with the points in the table defined by the **–SLA_MULTIPLIER** option. This option defines a table of slam velocities, S_VEL(k), vs. multipliers, S_MUL(k). By default, the table consists of three points: If the slams per hour are <= 1 then f is 1, if the slams per hour are 10, then f = 1/1.86, and if they are >= 360, then f = 1/3.72.

XII.I Forces

In MOSES, forces on the system are separated into nineteen named categories:

- **WEIGHT**: This is the weight of a portion of the system.
- **CONTENTS**: The weight of ballast in compartments or fluid in an element.
- **BUOYANCY**: The buoyancy of a portion of the system.
- **WIND**: The wind force acting on a portion of the system,
- **V_DRAG**: The viscous drag acting on a portion of the system. This is the velocity squared term in Morison's equation, the viscous roll damping, or the viscous drag on a piece with the –CS_CURRENT option. This can be either an excitation due to wave or current, or a damping in still water.
- **WAVE**: This is the linear exciting force on a portion of the system.
- **R_DRAG**: The radiation damping due to hydrodynamics or that due to a #DRAG load attribute.
- **SLAM**: This is the force due to mass transfer into the system; i.e. it is the velocity time the mass flow rate term in the equations of motion.
- **CORIOLIS**: This is the force due to Coriolis acceleration. It also produces a "slowly varying" force.
- **W_DRIFT**: This is the nonlinear part of the wave force, or the slowly varying "wave drift force". In MOSES, it does not contain an approximation of the Coriolis force. This is different than most other programs.
- **DEFORMATION**: This is the force on a body due to deformation of the body. This only occurs when a body has generalized degrees of freedom.
- **EXTRA**: This is a "extra" force that can be added to produce equilibrium in a given configuration. This is useful to "cover up" modeling errors or errors in the environment.
- **APPLIED**: This is a true force applied to a portion of the body.
- **INERTIA**: This is the mass of the body times the acceleration.
- **A_INERTIA**: This is the added inertia times the acceleration.
- **C_INERTIA**: This is the inertia of the contents times the acceleration.
- **FLEX_CONNECTORS**: This is the force due to flexible connectors.
- **RIGID_CONNECTORS**: This is the force due to rigid connectors.
- **TOTAL**: This is the sum of all of the other contributions.

The accuracy of the computation of forces on a plate or panel depends on the shape of the panel. In particular, the results for more "compact" panels is superior to those which are "long and skinny". As a guide to the "badness" of panels, a badness measure is reported for the &SUMMARIES of panels and plates. Now, often this measure is taken as the "aspect ratio" of the panel; i.e.

$$R = W/H$$

Where W is the width, H is the height and $W > H$. This is all fine and good for rectangular plates, but when these enter the water the submerged portion is rarely a rectangle.

To generalize this notion, in MOSES, we define badness as

B = P / [2 sqrt(pi*A)] − 1.

Here P is the perimeter and A is the area. Notice that for a circle, B is zero and that for rectangular plates it increases with the increase of aspect ratio.

When added mass is computed for a panel or plate, the results from DNV–RP–C205 are used. In particular, the force computations presented there for added mass forces are tabulated as a function of B and interpolated based on the current values of B for the submerged portion of the plate.

XII.J Categories and Load Types

As mentioned above, the basic idea behind MOSES is that one defines attributes which the program then uses to compute loads. Now, an attribute may create load from different sources. To distinguish these classes of loads, MOSES employs the concept of the "load type", which is simply a name given to each of the sources of loads, and is either: **#DEAD**, **#WIND**, **#BUOY**, **#AMASS**, or **#DRAG**, for the intrinsic loads, or a user supplied name which begins with a # for the applied loads. These names are used to control which type of load is applied to each load attractor. Here, **#DEAD** is the load due to weight, **#WIND** to wind, **#BUOY** is hydrostatic pressure, **#AMASS** is hydrodynamic pressure, and **#DRAG** is viscous water force.

For structural elements, one can select which of these forces will be applied. This is accomplished with two options on either **&DEFAULT**, **BEAM**, or **PLATE** commands:

> –**USE**, USE(1), USE(2), ..., USE(i)
> –**NUSE**, NOT_USE(1), NOT_USE(2), ..., NOT_USE(i)

Here, the values of USE(i) and NOT_USE(i) are the selectors for the names of the load types defined above. For example,

> –USE @ –NUSE #WIND

Instructs MOSES to consider all of the load types except wind when computing the force on an element. When a structural element is defined, MOSES first sets the load type flags to those defined with **&DEFAULT**. Then either of these options encountered on the definition itself serve to alter the settings defined by those from **&DEFAULT**.

The options defined above allow one to effectively define which elements are exposed to different environments. To give the user even more control, there is an additional concept, the Category. Each load attribute in MOSES has a category name associated with it and each category name has a set of multipliers for each load type. The category name is associated with a load attribute when it is defined in a manner similar to the load type flags. The **&DEFAULT** command accepts two options:

> –**BAS_CAT**, BASE_CAT_NAME
> –**EXT_CAT**, EXTRA_CAT_NAME

the –**BAS_CAT** option defines a Category name, BASE_CAT_NAME, to the load attributes defined with the structural elements themselves. –**EXT_CAT** defines a Category name, EXTRA_CAT_NAME, for any additional load attributes (those which belong to Load Groups or those defined with a #ELAT command). Now, when a element or load attribute is defined, the default Category (either

BASE_CAT_NAME or EXTRA_CAT_NAME) will be associated with it. This default association can be changed with the option

–CATEGORY, CAT_NAME

which associates CAT_NAME with the attribute. This option is available on **BEAM**, or **PLATE** commands and any command which begins with a #. Reports are available which produce sums of weight and buoyancy by Category. As an example, consider

BEAM BODY_NAME –CATEGORY INSIDE –NOTE Interior Beams ∼CLASS ..

This defines a beam, associates it with a Category, INSIDE, and defines a description of the Category "Interior Beams".

Load Groups have a multiplier for the total force of the group. Initially this multiplier is set to one.

Control of all of these multipliers is available with the command:

&APPLY, –OPTIONS

where the available options are:

–PERCENT
–FRACTION
–FORCE :NAME(1), VAL(1), ... :NAME(n), VAL(n)
–LOAD_GROUP :NAME(1), VAL(1), ... :NAME(n), VAL(n)
–TIME NAME, C_NAME
–CATEGORY :CAT(1), :NAME(1), VAL(1), ... :NAME(n), VAL(n)
–MARGIN :CAT(1), VAL_INC(1), ... :CAT(n), VAL_INC(n)

The first two options define the way the VAL(i) are interpreted. If a VAL is preceded by a **–PERCENT** option, it is interpreted as a percent. If it is preceded by a **–FRACTION**, it is a multiplier. If neither of these options precedes a value, it is interpreted as a percent. In essence, percents are divided by 100 to convert them into multipliers.

The **–FORCE** option defines multipliers for user defined load sets. Here, :NAME(i) are selectors for these sets and VAL(i) are the multipliers one wishes to associate with the set. By default, all user defined load sets *have a multiplier of zero.* Thus, they will not be applied unless one uses this option.

The **–LOAD_GROUP** option defines multipliers for load groups. By default, these multipliers are one. Thus, one normally does not need this option. One can, however, turn off the force due to a load group using this option. For example,

&APPLY –PERCENT –LOAD_GROUP A_NAME 50 –FRACTION B_NAME 0.50

will apply 50 percent of load group A_NAME, and one half of load group B_NAME.

The **–TIME** option define multipliers which vary with time. Here, NAME is the name of either a LOAD_GROUP or a user defined load set and C_NAME is the name of a curve which has been previously defined with a **&DATA CURVE TIME** command. At each event during a time domain simulation, MOSES will interpolate a multiplier from the specified curve. If NAME is later specified with –LOAD_GROUP or –FORCE options, the time variation will be "turned off".

The last two options control the load type multipliers for Categories. The **–CATEGORY** option allows one to control the multipliers for all load types while the **–MARGIN** option sets only the #DEAD, or weight, multiplier. Here, VAL(i) are the same as for the load group and load set options, while VAL_INC(i) are *increases*; i.e. the multiplier is really 100 or 1 plus VAL_INC depending upon –PERCENT or –FRACTION option in effect. Here, :CAT(i) is a selector for the Categories for which multipliers will be set and :NAME(i) are selectors for the load types. For example:

&APPLY –PERCENT –CATEGORY STR_MODEL @ 0 #DEAD 105

will set the multipliers for category STR_MODEL to all zero except for the weight which will be applied with a multiplier of 1.05.

There are three actions of the &INFO string function that can be used to obtain information about categories:

&INFO(ACTION, :SELCAT)

Here ACTION must be either **WT_CATEGORY**, **BU_CATEGORY**, or **MU_CATEGORY** which will return the weights. buoyancies, or multipliers for the categories selected by :SELCAT. For the first two, five tokens will be returned for each selected category: the name of the category, either its weight or buoyancy, and either its CG or CB. These are for either the current part or the last part selected with the –PART option of a &REL_SEL command. The CG or CB will be in the part system. The MU_CATEGORY action returns the name of the category and the category multiplier for each selected category.

XII.K Bodies and Parts

The basic ingredients for performing a simulation are bodies. For simulation purposes, bodies are considered rigid, and composed of parts. A part is the smallest entity upon which a structural analysis can be performed. Bodies are connected by parts which contain special elements called connectors. In essence, a part is simply a named portion of the model.

All properties of the system are described by the attributes of the parts. In other words, every attribute of the system must be an attribute of some part of the system. Thus, everything except bodies belong to some part.

In MOSES, each body, and each part will have a separate coordinate system associated with it. Since MOSES can consider bodies which move in space, we also employ a global coordinate system. The global system is fixed in time, and is used to locate positions in space. Its origin is arbitrarily located on the water surface, and its Z axis points upward.

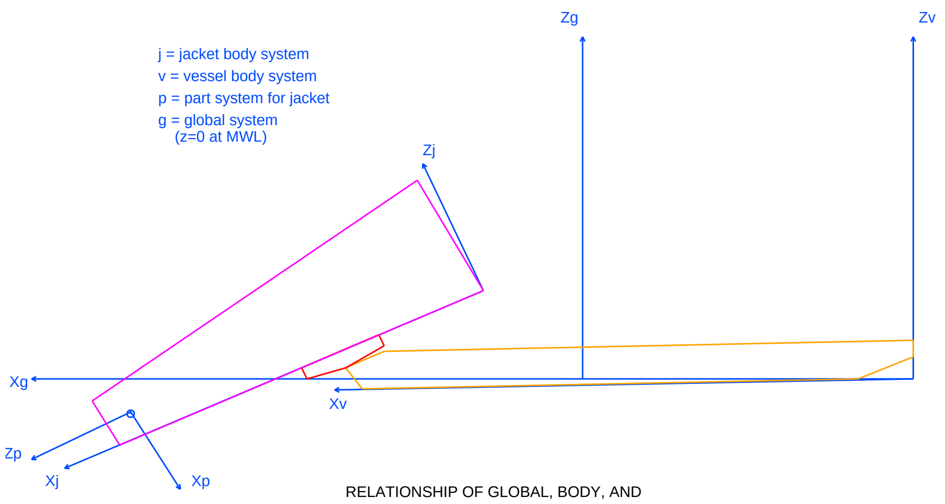
The part coordinate system is used to define the geometry of a part. In other words,

- The location of the points,
- The load attributes applied to the part, and
- The resulting structural displacements of the elements.

Thus, the part system is used in defining the model and in the definition of the structural deflections.

The body system is used for simulations. In most cases, the body system will be identical to the part system of the part which has the same name as the body. The axes of the body system are parallel to the global system when roll, yaw, and pitch are zero. The orientation of the body system defines the orientation of the body by three Euler angles, yaw, followed by a pitch, followed by a roll. An illustration of the global and part coordinate system along with a jacket and vessel body system during a launch is shown in Figure 4.

To distinguish between different bodies and parts, a block concept is used; i.e., all of the data for a part and body is defined contiguously in the input. One defines the different parts and bodies by **&DESCRIBE** commands. All of the data which follows one of these commands will be associated with this body or part until another **&DESCRIBE** command is encountered. This command also allows the user to alter the attributes of the body as the analysis proceeds. The format of this command is:



RELATIONSHIP OF GLOBAL, BODY, AND
PART COORDINATE SYSTEMS - ALL Y-AXES OUT OF PLANE

FIGURE 4

&DESCRIBE BODY, BODY_NAME, –OPTIONS

where the available options are:

–**IGNORE**, DOF(1), DOF(2), ...
–**GEN_DOF**, MODE_SEL(1), MODE_SEL(2), ...
–**S_DAMPING**, CFRACTION
–**SECTION**, EI, X(1), SM(1), ..., X(n), SM(n)
–**LOCATION**, X(1), X(2), ...
–**DMARK**, DM_NAME, *DPT(1), *DPT(2)
–**D_DMARK**, :DM_NAME(1), :DM_NAME(2), ...
–**PR_NAME**, PR_NAME
–**MD_NAME**, MD_NAME
–**MD_FORCE**, MD_FORCE, MD_RADIATION, MD_CORIOLIS
–**MD_PHASE**, MD_PHASE
–**FM_MORISON**, FM_FACTOR
–**SPE_MULTIPLIER**, SPEMUL
–**SP_ORIENT**, VX, VY, VZ, HX, HY, HZ
–**SP_HEIGHT**, X, Y, Z
–**DT_CONVOLUTION**, DT_CONV
–**FACT_CONVOLUTION**, CONV_FACTOR
–**PERI_USE**, PER
–**WAVE_RUNUP**, YES/NO

To ease the difficulty of defining a model, whenever a body is defined, a part with the same name is automatically defined. Any attribute defined before a part is specifically defined becomes the property of the "body" part. *NOTICE* that since everything must belong to some part of some body, a **&DESCRIBE BODY** should be at the beginning of each set of data.

Normally, all bodies have six degrees of freedom. Sometimes, one wants to "ignore" some of these degrees of freedom. This is accomplished with the **–IGNORE** option. Here, DOF(i) is the name of the degree of freedom to be ignored, and must be chosen from the list **X**, **Y**, **Z**, **RX**, **RY**, and **RZ**. When a degree of freedom is ignored, the force in this degree of freedom is set to zero, and the inertia is set to a very large number.

Normally, bodies have six degrees of freedom. The option **–GEN_DOF**, however, allows one to use previously computed vibration modes as generalized degrees of freedom for a body. For a discussion on computing modes for a body, see the section on Extracting Modes. Here, MODE_SEL(i) are selectors for the mode numbers one wishes to use. MODE_SEL(i) can be a single number which selects that mode number, or a pair of numbers A:B which selects all modes from A to B. Once a body has more that 6 degrees of freedom, some measure of its deformation and the deformation inertia is accounted for in any equilibrium, time domain, or frequency domain analysis. Thus, this is an excellent way to study the effect of flexibility on the results. To the user, however, the number of degrees of freedom make no difference in how to accomplish various tasks. The only effect is the computer resources required. The **–S_DAMPING** option defines the modal damping which will be used for the generalized degrees of freedom. Here, CFRACTION is the fraction of critical damping to be used. If this option is omitted, .01 will be used.

Two options of **&DESCRIBE BODY** serve to override defaults when computing longitudinal strength. The **–SECTION** option is used to define the "section properties". Here, EI is the stiffness of the body. The X(i)s are the X locations along the body where one wishes to compute longitudinal strength results, and SM(i) is the body section modulus at X(i). If only a single location is specified, then MOSES will redefine this data so that results are computed at each station, and the section modulus will be constant along the length of the body. For SM the units will be either ft**3 or meters**3, and for EI, force–ft**2 or force–meters**2. If one simply wants to define the locations at which the results are reported, he can use the **–LOCATION** command.

The option **–DMARK** is used to define draft marks and **–D_DMARK** is used to delete draft marks. Draft marks are rays along which one measures draft. Each mark is defined by two points. The first point is the "origin" of the draft mark and the second is used to define the direction of the mark. Draft is the distance along this ray from the first point to where the waterplane intersects the line. Here, DM_NAME is a name given to the draft mark, *DPT(1) a point defining the origin of the mark, and *DPT(2) is a point defining the direction of the mark. On should use a –DMARK option for each draft mark defined. When deleting marks, :DM_NAME(i) are selectors which select the marks to delete.

The next two options: **–PR_NAME** and **–MD_NAME**, allow one to change the pressure packet name and the drift packet name associated with the body. Here, PR_NAME and MD_NAME are the new pressure or mean drift names. In most cases, the mean drift database is created as a consequence of creating the pressure database, and a discussion of the mean forces can be found in the section of the HYDRODYNAMICS MENU and in the subsection on mean drift.

The next three options allow one to change the multipliers used with the mean drift force. The **–MD_FORCE** option defines three scale factors, MD_FORCE, MD_RADIATION, and MD_CORIOLIS which are used in computing the total mean drift force. The last two of these are multipliers of the radiation and Coriolis contributions to the mean force as they are added to the diffraction contribution. The MD_FORCE factor multiplies the total mean force before it is accumulated. The current response operators for the current process are used when computing the radiation and Coriolis contributions. Finally, the **–MD_PHASE** defines a phase angle (deg.) of the wave drift force in the frequency domain. Here, each drift force component will have a phase, PHASE, relative to the wave crest being at the origin. The default for these options is set with **&DEFAULT**.

The next two options of **&DESCRIBE BODY** deal with viscous damping, and both of them override defaults set with **&DEFAULT**. The **–FM_MORISON** option defines a factor which will be multiplied by the computed viscous force to yield an applied viscous force in the frequency domain. The option **–SPE_MULTIPLIER** defines the spectral linearization multiplier. When nonlinear quantities are linearized with a spectral linearization in the frequency domain, the RMS of the velocity is multiplied by the factor SPEMUL and used in computing a constant drag coefficient.

When performing a simulation in the Static Process Menu, one needs to define two vectors which are used to measure the "roll and pitch" angles and a point which is used to measure height. This is accomplished with the two options **–SP_ORIENT** and **–SP_HEIGHT**. All of these quantities used with these options are defined in the part system of the body. The height used to terminate a lift is defined by X, Y, and Z. The pitch angle is defined as angle formed by the vector VX, VY, and VZ with the waterplane, and the roll angle is the angle formed by the vector HX, HY, and HZ with the waterplane. Here, VX, VY, and VZ are the components of a vector in the part system which will be vertical when the jacket is in the installed condition. If these options are omitted, the values defined with the same name on an **&DEFAULT** command will be used.

The last four options provide the user ways in which to "tweak" the hydrodynamic data to suit his purposes. All these options use as defaults values defined with an option of the same name on an **&DEFAULT** command. The first three options control the way the frequency dependent nature of the added mass and damping are considered in the time domain. As described in the Hydrodynamics section, mathematically, this results in a convolution. This convolution is, however, often difficult to deal with numerically. One of the leading causes of numerical instability is a badly behaved convolution. In fact, with a convolution, it is possible to take too small a time step! There is a limit to how much velocity history can be stored. For extremely small time steps, this history may not be long enough to adequately describe the damping. The **–DT_CONVOLUTION** option defines the time increment at which the convolution will be integrated. This must be equal or greater than the time step at which the time domain is computed. If you use **TIME_STEP** for DT_CONV, then the convolution will be computed at the time steps used to integrate the equations of motion otherwise it will be computed for times steps of DT_CONV. This option allows one to use a reasonable time step (around 1/4 second) to compute the convolution and a smaller one to integrate the equations of motion which ameliorate the above problem. An alternative way of dealing with this problem is by using the last two options. The **–FACT_CONVOLUTION** option scales the kernel of the convolution by CONV_FACTOR before it is applied. If CONV_FACTOR is zero, the convolution will not be added, and if CONV_FACTOR is one, all of the convolution will be added. The **–PERI_USE** option allows one to replace the convolution terms in the equations of motion with single added mass and damping matrices. The matrices used will be the frequency domain matrices at the specified period, PER. This eliminates any numerical problems with the convolution. The defaults here CONV_FACTOR = 1 and PER = 0, i.e. use all of the convolution in the time domain.

In the time domain, the buoyancy is computed "correctly" at each time step. Since the frequency domain forces already include the effects of the wave elevation, this buoyancy is computed assuming that the water surface is flat. If **–WAVE_RUNUP** is used with a YES/NO of **YES**, then a different algorithm is used. Here, the water surface is assumed to be adequately described by the incident potential and the "correct" Froude–Krylov force computed. When this option is used, the Froude–Krylov force is not included in the frequency domain forces applied.

The string function which returns data for bodies is:

&BODY(ACTION, BODY_NAME, –OPTION)

Here, BODY_NAME is the name of the body for which data is desired and ACTION must be either: **CURRENT**, **E_NODES**, **P_NAME**, **EXTREMES**, **DRAFT**, **LOCATION**, **VELOCITY**, **MXSUBMERGENCE**, **BOTCLEARANCE**, **NWT_DOWN**, **MIN_NWT_DOWN**, **WT_DOWN**, **MIN_WT_DOWN**, **E_PIECES**, **I_PIECES**, **MAX_BUOYANCY**, **MAX_CB**, **DISPLACE**, **CB**, **GM**, **G_ROLL**, **I_VECTOR**, **F_WEIGHT**, **F_CONTENTS**, **F_BUOYANCY** **F_WIND**, **F_V_DRAG**, **F_WAVE**, **F_SLAM**, **F_R_DRAG**, **F_CORIOLIS**, **F_W_DRIFT**, **F_DEFORMATION**, **F_EXTRA**, **F_APPLIED**, **F_INERTIA**,

F_AINERTIA, **F_CINERTIA**, **F_FLEX_CONNECTORS**, **F_RIGID_CONNECTORS**, **F_TOTAL B_CG**, **B_WEIGHT**, **B_RADII**, **B_MATRIX**, **A_CG**, **A_WEIGHT**, **A_RADII**, **A_MATRIX**, **D_CG**, **D_WEIGHT**, **D_RADII**, or **D_MATRIX**. The action **CURRENT** is special in that it should have no other input; i.e. no BODY_NAME and no –OPTIONS. It returns the name of current body. All other action need a body name. If this name is omitted, the current body will be used. Thus if your model contains only a single body the name may be safely omitted.

The action **E_NODES** returns the names of the extreme nodes for the body, and **P_NAMES** returns the names of the parts in the body. The action **EXTREMES** returns the length, width, and height of a body where these quantities are measured along the body X, Y, and Z axes respectively. The action **DRAFT**, the reading of the draft marks will be reported, while the actions **LOCATION**, **VELOCITY**, **MXSUBMERGENCE**, and **BOTCLEAR**, return the requested data on the location of the body. **WT_DOWN** and **NWT_DOWN** will return a list of the down–flooding points of the specified type , and **MIN_WT_DOWN** and **MIN_NWT_DOWN** return the minimum height of the points about the water.

The actions **E_PIECES** and **I_PIECES** return a list of the external or internal pieces in the body. The actions **MAX_BUOYANCY** and **MAX_CB** return the non–compartment buoyancy and its center when the body is completely submerged, while the actions actions **DISPLACE**, **CB** and **GM** return the current value of these things. For the actions **CB**, **MAX_CB**, **LOCATION**, and **VELOCITY** the option **–GLOBAL** can be used to change the returned data from being in the body system to being in the global system. The **G_ROLL** returns an angle which is the angle of inclination of the body from the vertical. More precisely, if v is a vertical vector represented in the body system, then G_ROLL is the arc tangent of the sqrt(v(1)*v(1) + v(2)*v(2)) / v(3). The **I_VECTOR** returns a vector which, if a moment is applied in this direction, will cause the inclination to increase, or I_VECTOR = [v(1), v(2), 0]/sqrt(v(1)*v(1)+v(2)*v(2)).

The actions which begin with a **F_** return forces acting on the body. The data following the _ specifies the type of load which will be returned. The meaning of these forces can be found in the section on FORCES.

If no option is specified, then the results are in the body system. Alternately, one can specify **–GLOBAL** to return the values in the global system.

The remainder of the actions deal with the weight of the body. Those beginning with a **B** are applicable to the "basic" weight of the body, those beginning with an **A** to the "apparent" weight of the body (basic weight plus weight of contents), while those beginning with a **D** deal with the "define" weight for the body. What follows the _ define the type of data returned: WEIGHT for the weight, CG for the centroid of the weight, radii of gyration for RADII, and MATRIX for the 6X6 weight matrix (mass matrix divided by g).

Parts are defined similarly to bodies. In particular, the command to describe a part is:

&DESCRIBE PART, PART_NAME, PART_TYPE, –OPTIONS

and the available options are:

–**MOVE**, NX, NY, NZ, NRX, NRY, NRZ

or

–**MOVE**, NX, NY, NZ, *PT(1), *PT(2), *PT(3), *PT(4)

Here, PART_NAME is the name to be given to the part, and PART_TYPE is the part type. The values one can use for PART_TYPE are somewhat arbitrary, but the two types of **JACKET** and **PCONNECT** have special significance. The part type, **JACKET**, is used to denote the portion of the system which will be treated differently when the special transportation facilities of MOSES are employed. To avoid confusion, elements which connect parts (elements which are connected to nodes in different parts) must belong to a part with a type of **PCONNECT**. Thus, the only elements which can span parts are connectors or **PCONNECT**ors. A **PCONNECT** part should not have any nodes which belong to it. There is a special part, **GROUND**, which does not belong to any body, and it is in this part that elements which connect bodies reside. These elements which are called connectors are quite important, since they define both the boundary conditions for a stress analysis, and the constraints on the bodies for a simulation.

In most cases, the body system of a body will be coincident with the part system of all parts belonging to the body. When certain types of connections (launch legs) are defined, however, the body system will be altered as described later. Also, the user can alter a part system using the **–MOVE** option. Here, NX, NY, and NZ are the location (feet or meters) of the origin of the part system with respect to the reference, and NRX, NRY, and NRZ are a set of Euler angles which defines the new orientation of the part system. Alternately, if the second form of the command is used, the four points define the orientation of the part system. Here, the part X axis will be from the midpoint of the segment connecting *PT(4) and *PT(2) to the midpoint of the segment connecting *PT(3) and *PT(1). The part Z axis is defined by the cross product of the X axis with the vector from *PT(4) to *PT(2), and the Y axis is given by the right hand rule. This option can be used both in defining a model, and when editing one, but one can move a "body part" (a part with the same name as a body) *only* while editing. Moving a body part is equivalent to changing the orientation of all of the parts of the body. When one is moving a part, the above data defines the part system with respect to the "body part" system and when moving a "body part", it defines the body part with respect to the body system. To orient the part system, suppose that the part and the reference systems are aligned, and that the part system is rotated an amount NRZ about the reference Z axis. Next, rotate the part system about the current part Y axis, and amount NRY, and finally rotate an amount NRX about the new part X axis.

The string function:

&PART(ACTION, PART_NAME, –OPTION)

returns the current data about a part. Here, PART_NAME is the name of the part for which data is desired and ACTION must be either: **CURRENT**, **MAX_CB**, **MAX_BUOYANCY**, **CG**, **WEIGHT**, **RADII**, or **E_NODES**. The action **CURRENT** returns the name of current part, while the **WEIGHT**, **CG**, and **RADII** actions simply return the current weight, CG, and radii of gyration of the part. These are the "dry" values of the part, without any compartment contents. The **MAX_BUOYANCY** and **MAX_CB** return the non–compartment buoyancy its center when the part is totally submerged. If no option is specified, then the results are in the part system. Alternately, one can specify **–BODY** or **–GLOBAL** to return the values in another system. The action **E_NODES** returns the names of the extreme nodes for the part.

After a set of bodies have been defined, it is necessary to place them in space as the point of departure for a simulation. This is accomplished via the command:

&INSTATE, –OPTIONS

where the options are:

–**LOCATE**, NAME, X, Y, Z, RX, RY, RZ
–**MOVE**, NAME, DX, DY, DZ, DRX, DRY, DRZ
–**CONDITION**, NAME, DRAFT, ROLL, TRIM
–**POINT**, *PNT(1), H(1), *PNT(2), H(2), ... *PNT(n), H(n)
–**DRAFT**, DMARK(1), D(1), DMARK(2), D(2), ... DMARK(n), D(n)
–**GUESS**, *NODE(1), *NODE(2), *NODE(3)
–**VELOCITY**, NAME, VX, VY, VZ, VRX, VRY, VRZ
–**SL_SET**
–**LINES**, :ACTIVE, :LSEL(1), TEN(1), :LSEL(2), TEN(2), ...
–**EVENT**, EVE_NUM
–**PREVIOUS**
–**C_FORCE**, FLAG

The uses of the option keywords are: **–LOCATE** positions a body in space absolutely; X, Y, and Z are the global coordinates of a point on the body, and RX, RY, and RZ are the new Euler angles. The –**MOVE** option moves a body relative to its last known position; in this case, the parameters are changes in position and orientation. –**CONDITION** is the same as –**LOCATE** except that only 3 degrees of freedom are necessary. Here, NAME is the name of either a body or a report point. If a point name is specified, then the position specified will be for that point. If a body is specified, then the location will be the location of the origin of the body. All of the units for these options are feet or meters and degrees. The three options –**POINT**, –**DRAFT**, and –**GUESS** all set the body according to the positions of a set of points. With the –**GUESS** option, MOSES will change the orientation of the body so that the three nodes *NODE(1), *NODE(2), and *NODE(3) will lie in the waterplane. Here, the three nodes cannot be colinear, and one cannot rotate a body ninety degrees. With –**DRAFT**, one specifies a set of draft marks and the draft readings at the marks and with –**POINTS**, one specifies a set of points and the height of these points above the water. MOSES then finds the draft, trim, and heel of the body that gives a "best fit" to the data specified. The –**VELOCITY** option defines the initial velocities of a body. Here again, the velocities specified are for a point, a node, or the origin of the body in ft/sec or m/sec and degrees/second.

In general, when rigid constraints have been defined between two bodies, it is necessary to locate only one of the bodies. If one attempts to specify both, he may have a difficult time in locating the various bodies without violating a constraint. In other words, if only one body is specified, MOSES will locate this body and locate the other bodies in such a way as to satisfy the constraints. Also, one may not give a body a yaw (RZ) after you have connected launchways to it.

Positioning a system of bodies with connections is not a trivial task. The next two options are designed to minimize this difficulty. With these options, not only is the configuration of the system altered, but the lengths of some of the connectors are also changed. The objective is to have the system be in equilibrium. The –**SL_SET** option simply computes the proper length for the tip–hook element of each tip–hook set so that the lines have no slack in the specified configuration. This will probably not be an equilibrium configuration, but it will be a good starting place for one.

The –**LINES** option instructs MOSES to alter the lengths of the flexible connectors which match :ACTIVE so the system will be as close to equilibrium as possible. The additional data controls the operation of the algorithm so that the connectors which match :LSEL(i) are altered to make the resulting tension (compression) in the connector "as close as possible" to that specified by TENS(i). For connectors which are not selected by any of the selectors, the algorithm will attempt to make the tension as close as possible to the current tension. As mentioned above, the objective is an equilibrium configuration, but this cannot always be achieved. In particular, "anchor lines" cannot be used to alter the vertical force on a body, so that if one has only anchor lines, then there is no guarantee that the vertical forces will balance at the conclusion of the command. Likewise, if one has only "vertical" connectors, the horizontal forces may be out of balance.

The **–EVENT** option sets the initial configuration to be the one at event EVE_NUM of the current process, and **–PREVIOUS** is used to replace the initial state with the "previous" one. Here, "previous" is the state existing before the last command which changed the state before **–EVENT** was issued.

Finally, the **–C_FORCE** option computes an "extra" force on each body which makes the specified configuration an equilibrium configuration. This action occurs if FLAG is **COMPUTE**. If FLAG is **NO**, the extra force will be set to zero.

In addition to the weights which were defined for elements and load groups, MOSES employs an additional weight which can be applied to any part. This is called a "defined" weight and is controlled with the command:

> **&WEIGHT**, –OPTIONS

where the available options are:

> **–COMPUTE**, BODY_NAME, ZCG, KX, KY, KZ
> **–DEFINE**, PART_NAME, WEIGHT, XCG, YCG, ZCG, KX, KY, KZ
> **–TOTAL**, PART_NAME, WEIGHT, XCG, YCG, ZCG, KX, KY, KZ

The first of these options is restricted to bodies while the others can be used with parts. For the first option, all data given should be in the body coordinate system of the body, BODY_NAME. For the others, all data should be in the part system of the part PART_NAME. The **–COMPUTE** option instructs MOSES to compute the weight so that, in the initial condition, the gravitational force will be equal to the sum of the other forces. The local Z center of gravity of the computed weight will be placed at the specified location (ZCG). Alternately, a **–DEFINE** option defines a load with the weight and center of gravity specified. The option **–TOTAL** creates an additional weight so that the total weight, center, and radii of gyration of the part will be that specified. In either case, the weight which is defined will have radii of gyration KX, KY, and KZ with respect to the point of application. Here the location of the cg and the magnitude of the radii of gyration are to be interpreted in the Body System. If bending moments are computed, the load will be represented as a trapezoidal load acting over the entire body length. By default the loads resulting from the defined weight will be distributed to all nodes in the body part of BODY_NAME or the part PART_NAME.

As was implied above, while bodies are the basic ingredients of simulations, they have virtually no properties other than the parts which belong to them, and all of the physical properties of the bodies are inherited from their parts. This is quite important since one can move parts between bodies, create new bodies, etc. While it may seem to be of limited use, this is not really the case. Suppose that one wishes to investigate the behavior of a jacket throughout its installation.

First, one may wish to examine the loadout where the only thing of interest is the jacket itself. Here, a single body of "jacket" is needed. Next, however, one wants to consider the transportation. Here, a new body composed of the two parts: jacket and the barge would be of interest. Finally, during launch, two bodies, the jacket and barge would be used. The activity of bodies and parts is controlled with the command:

&DESCRIBE ACTIVITY, –OPTION1, –OPTION2 :SEL(1), :SEL(2), ..

and the available options are **–BODY**, **–PART**, **–ACTIVE**, and **–INACTIVE**. Here, one first uses either **–BODY** or **–PART** to specify what type of entity will be changed. Next, he specifies either **–ACTIVE** or **–INACTIVE** to specify the activity status. Any body (or part) selected after the second option will have its activity status set according to the option. As many options as one wants can be used on a single command. When a body is set inactive, its parts are not set inactive. There is no reason for this since only active bodies will be considered and this would make it more difficult to reactivate a body.

In addition to the above, the entire system can be redefined with the command:

&DESCRIBE SYSTEM, **–BODY**, BODY_NAME(1), PART(1), ..., PART(i), \
–BODY, BODY_NAME(2), PART(1), ..., PART(i)

When this command is issued, all bodies and parts are set to inactive. Then the bodies named (BODY_NAME(1), ...) are either activated or defined, and the parts specified after the BODY_NAME(i) and before the next **–BODY** are activated and associated with the body BODY_NAME(i).

XII.L Geometry

The geometry of the model is defined by a set of points in the part system. The names of all points begin with a *. The location of the points can be specified with respect to either the part origin, or another point. At least one point in the model, however, *must* be specified with respect to the part system. Every point belongs to a part, and to allow for specifying fixed positions in space, there is a part which does not belong to any body, **GROUND**. Therefore, to define fixed locations in space one can specify that the following points will be fixed by issuing **&DESCRIBE PART GROUND** before the fixed points are defined. The part system of **GROUND** is the global system.

Points to which one attaches structure have a special name, nodes. Each point which is not a node will have a node associated with it. Any load associated with a point is ultimately applied to the structure at the associated node. The associated node is the node closest to the point which is in the "associated node set." Normally, this set will be the set of all nodes. In some situations, however, one wishes to exclude certain nodes for being "associated" and thus from attracting loads. This is accomplished with the command:

&DESCRIBE NODE_NAS, PART_NAME(1), :PNT_SEL(1), ...

where PART_NAME(i) is a part name, and :PNT_SEL(i) is a selection criteria which defines the nodes which will not be associated.

Points which have both a chord, a single tubular member with the same outside diameter and at least one other tubular element are called tubular joints. Tubular joints are not true parts of a model since they inherit the most of their properties from the elements which intersect to form the joint. The concept of a joint is really only used in checking the codes which are applicable to joints and in computing fatigue at the intersection of the elements forming the joint.

Once a point has been defined, its properties can be changed with the **ED_POINT** command, which is available either during an **INMODEL** or in the **MEDIT** menu. Its form is:

ED_POINT, :PSEL, –OPTIONS

Here, :PSEL is a selector (selection criteria, or name which may contain wild characters) of the points for which the options are applicable, and the options can be any option valid for the point definition command.

Often, it is desirable to obtain information about certain points on the bodies during a simulation. The points at which these reports will be issued are defined by the command

&DESCRIBE INTEREST –OPTIONS

and the available options are:

−**DELETE**, :PNT_SEL(1) .... :PNT_SEL(i)
−**ASSOCIATE**, :PNT_SEL(1) .... :PNT_SEL(i)

Here :PNT_SEL(k) are selectors for points. The option −**DELETE** deletes the interest points selected by all of the selectors and −**ASSOCIATE** makes interest points of all points selected by the selectors. Interest points have a bit more information associated with them than the other types of points. In particular, each time the **&DESCRIBE INTEREST**, is issued, the global location of the interest point is saved. These locations are used for a reference in measuring "motion" in several reports.

Once a set of points for a part have been defined, they can be exported to a file for later use with the command:

&EXPORT POINTS

XII.L.1 Defining Points

To define a point, one issues the command:

> *NAM, X, Y, Z, –OPTIONS

and the available options are:

> –**REFERENCE**, *RPA, *RPB, ..
> –**RECT**
> –**CYLINDER**
> –**SPHERICAL**
> –**LOCATION**, XO, YO, ZO, RX, RY, RZ
> –**LOCATION**, XO, YO, ZO, *PT(1), *PT(2), *PT(3), *PT(4)
> –**DEL**, DOF(1), ... DOF(i)
> –**JNTCLASS**, PRCK, PRCT, PRCX
> –**BBC_MUL**, MULT
> –**EFF_CHD_LEN**, ECHD_LEN
> –**CO_SCF**, SCF_TYPE
> –**LEN_FACTOR**, FRACHOL
> –**MAX_CHD_LEN**, MAXCHOL
> –**CHD_FIXITY**, CHD_FIX
> –**MIN_SCF**, MIN_SCF

where *NAM is the name of the point defined, and X, Y, Z are the coordinates (feet or meters) in the "current point system" defined by the **&DEFAULT** command. If the option –**REFERENCE** is employed, then the coordinates are relative to the specified points; otherwise, they are with respect to the part origin.

If one reference point is specified, then X, Y, and Z define a vector from this reference point to point *NNAM. If two points are specified, then X specifies the distance from the point *RPA to point *NNAM, along a line from *RPA to *RPB, and Y and Z are ignored. If three points are specified, then they will define a local coordinate system with the origin being at the first point, the x axis being from point 1 to point 2, the z axis being perpendicular to the plane formed by the three points, and the y axis given by the right hand rule. Again, X, Y, and Z are local coordinates in this system. If four points are specified, then a point is first located at the intersection of the two lines which connect *RPA and *RPB, and *RPC and *RPD. The X, Y, and Z coordinates then define the vector from this point to point *NNAM. Remember that if one changes the coordinates of a point being referenced, then the coordinates of the defined point will also change. Some examples of using reference points are shown below:

> *PT3 10 –REFERENCE *PT1 *PT2

This will create a point 10 feet (or meters) from *PT1 along a line formed by *PT1 and *PT2. This example is handy for defining points along a battered jacket leg.

*PT5 –REFERENCE *PT1 *PT2 *PT3 *PT4

This will create a point at the intersection of the lines formed by points *PT1 and *PT2, and *PT3 and PT4, which is useful for defining points at the middle of X braces.

If one wished to temporarily override the current system, he may do so by specifying any of the options **–RECT**, **–CYLINDER**, **–SPHERICAL**, or **–LOCATION** which were discussed with the **&DEFAULT** command.

The option **–DEL** is used to delete the degrees of freedom of a point which specified by DOF(i) which must be either **X**, **Y**, **Z**, **RX**, **RY**, or **RZ**. If no DOFs are specified, then all degrees of freedom will be fixed. Notice that this option only has meaning if the point is also a node.

All of the remaining options define the "joint" behavior of points. MOSES automatically classes joints based on the load path. In some rare cases, one may wish to override the automatic joint classification. This can be accomplished with the **–JNTCLASS** option. Here, PRCK, PRCT, and PRCX are the percentages of K, T, and X joint types used to classify a joint. When this option is used, the specified classification for *all* load cases.

In Joint Crushing MOSES treats the joint as a two dimensional ring. Two basic assumptions here are that the braces do not alter the stresses in the ring and that an effective length of the chord is used to distribute the load. The **–BBC_MUL** option allows a factor of the bending stress *under* a brace to be used. In other words, if one believes that the brace will prevent any bending stress in the chord under its footprint, then he should specify a value for 0 with the **–BBC_MUL** option. The opposite, conservative, view is that the brace has no effect in the bending of the chord where one specifies a value of 1 for MULT. Of course, one can specify any value between these two. The **–EFF_CHD_LEN** option is used to change the effective chord length for Joint Crushing from the default behavior computed according to API RP2A for both joints with and without rings.

The remainder of the options control the computation of SCFs for tubular joints. These options were discussed in the section Associating SCFs with Tubular Joints.

XII.L.2 Geometry String Functions

The string function for points is one of the most useful, and its form is:

&POINT(ACTION, DATA, –OPTION)

Here, ACTION must be either **COORDINATES**, **D_NODE**, **PART**, **EULER_ANGLES**, **N_2NODES**, **N_BOX**, **RATIO**, **DEFLECTION**, **HOOK_LOC**, **OFFSET**, **CLOSE**, **NEAREST**, or **REL_MOTION**. The details of DATA vary with ACTION, but normally DATA will be one or more point names. If ACTION is **COORDINATES**, this function returns the location of the point specified with DATA. Here, the location returned will be in the part system, unless one of the options **–BODY** or **–GLOBAL** has been specified. If the point has not been defined, a null string will be returned. One can use this function during an **INMODEL**, to define locations which may be required, provided that the point referenced has been previously defined. If ACTION is **D_NODE**, the function will return the distance between the two points specified via DATA. If ACTION is **PART**, the part name of the point specified with DATA is returned.

When changing frames of reference, the ACTION **EULER_ANGLES** may be useful. This action returns a set of Euler angles from three points specified via DATA. These angles will establish the new x axis along the line from first point to second, a new Z axis perpendicular to the plane formed by the three points, and a new Y axis given by the right–hand rule. The Z axis is defined by the cross product of the X axis and the vector from the first to the third point.

The ACTION **N_2NODES** returns all points along the line segment between the first point specified via DATA and the second one. These points will be in order with the first point specified being the first point returned and the second one specified as the last one returned.

The remaining values ACTION are different than those above in that the DATA are not strictly point names. The ACTION **N_BOX** is different from the others in that the data here is a set of global coordinates: X1, X2, Y1, Y2, Z1, and Z2. The function will return all points which are in the "box" defined by the six planes defined by these coordinates. The ACTION **RATIO** will return the last value of joint unity ratio computed for the specified point, and an ACTION of **DEFLECTION** will return the three components of the last computed deflection of the point.

The ACTION **HOOK_LOC** has a form for DATA of: *NN1, LEN1, *NN2, LEN2, ..., *NN4, LEN4, and is used to find the global position of a hook which is connected to points *NN1, *NN2, etc. It assumes that the harness lengths are LEN1, LEN2, ... respectively. This function is useful primarily in setting up a lifting problem when one models the sling as a collection of flexible connectors.

The values of ACTION **OFFSET** and **CLOSE** are used to find offsets. Normally, they are used in the definition of connectors. The form of DATA for **OFFSET** is: *POINT, :SEL, X, Y, Z and for **CLOSE** PART_NAME, :SEL, X, Y, Z. Here, X, Y, and Z are locations (feet or meters). For both values of ACTION, MOSES finds the point closest to a specified location but they differ in detail. For an ACTION of **OFFSET**, the specified point is the location of *POINT plus the X, Y, and Z specified, and any point which matches :SEL in a part different than that of *POINT is a candidate. The string returned is **–GO1**, the x, y and z values of the offset from specified point to the "close" point, and the string *POINT itself. For **CLOSE**, the specified point is simply the location in the part PART_NAME defined by X, Y, and Z, and all points which match :SEL are considered for being "close". In either case, the vector returned is represented in the part system of the "close" point. As an example, consider:

```
CONNECTOR ~CC &POINT(OFFSET *1 @ 0 0 10)
```

This will define a connector of class ~CC. End 1 of the connector is at a physical location of *1 plus 0 0 10 in the part system of *1 and end two is at *1. This connection is actually defined via the closest point and an offset and the command set to MOSES will look like

```
CONNECTOR ~CC –GO1  XX YY ZZ *B22 *1
```

where XX, YY, ZZ, and *B22 will be values computed so that the end of the connector is at the correct location.

The ACTION of **NEAREST** takes a point name and a selector and returns the point which matches the selector closest to the given point. In particular:

```
&POINT(NEAREST *R *R@)
```

will return the name of the point which begins with *R (and which is not *R itself) that is closest the point *R.

Another useful ACTION is **REL_MOTION**. Here, data is two point names *P1, *P2, and six numbers: X–, X+, Y–, Y+, Z–, and Z+. What the function returns is the minimum and maxima of the relative location between the two points and the values input, i.e. the output is part of the input which has been modified. This is best seen by example. Suppose that one issues:

```
&SET ENVEL =
&SET ENVEL = &POINT(REL_MOT *1 *2 %ENVEL )
```

You will find that at this point X– = X+, Y– = Y+, and Z– = Z+, and X–, Y–, Z– is the vector from *1 to *2 in the *1 body system. If you issue the second of these commands several times, the result will be that ENVEL will contain the extremes of the relative position between the two points. From the above example, if this ACTION is used with null data, the accumulation is initialized. Also, the values returned are in feet or meters.

Often, one wants to transform vectors from body to global coordinates and conversely. This is easily accomplished with the string function:

&V_TRANSF(ACTION, BODY_NAME, DATA)

Here, ACTION must be either **V_B2G**, **V_G2B**, **V_P2B**, **V_B2P**, **L_B2G**, **L_G2B**, **L_P2B**, **L_B2P**, **F_B2G**, **F_G2B**, **F_P2B**, or **F_B2P** and DATA is either three or six coordinates (feet or meters) or six force components (bforce, bforce–feet or meters). The function takes the input and transforms it, returning the same number of components as DATA input. If ACTION begins with: a **V** then a simple vector rotation will be performed, with an **L** a rotation and a translation will be performed, and with an **F** a force transformation will be performed. The remainder of ACTION defines the direction of the transformation. The action **B2G** transforms a body representation into a global one, and **G2B** the inverse. The action **P2B** transforms a part representation into a body one, and **B2P** the inverse.

XII.M Element Classes

Since there are many elements in a structure which have common properties, MOSES allows one to associate a name with a set of properties. One then associates the name of the set by specifying the name on the element definition command. This name is called the *class name* of the properties and the elements of the set are called classes. The class name must begin with a ∼. The concept of class is important in MOSES since it is used not only for defining properties, but also because MOSES redesigns by class.

In general, MOSES allows for an element to have different properties along its length. To define such an element, one should have a Class definition command for each segment of the element. When more than one segment is defined, the first set of properties are associated with the beginning or "A" end of the element, and the last set of properties associated with the "B" end of the element. The lengths of the segments are defined by an option, **−LEN**.

As mentioned above, classes are used to define all elements of the model, and as a result, there are many ways to define them. Basically, a class definition consists of at least:

∼CLASS, SEC_TYPE, A, B, ... –OPTION

Here, SEC_TYPE is a name which defines the types of elements which can use this class, A, B, ... are dimensions (inches or mm), and there are many options. In general, certain options are only applicable to certain types of classes, but two groups are available to all; material options and load attribute options.

The material options are:

–**SPGRAVITY**, SPGR
–**DENSITY**, RHO
–**EMODULUS**, EMOD
–**POI_RAT**, POIRAT
–**ALPHA**, ALPHA
–**FYIELD**, FYIELD
–**TENSTR**, TENSTR
–**SN**, CURVE, TYPE, S(1), N(1), ... S(n), N(n)

All but −**TENSTR** can be defaulted via the **&DEFAULT** command. The −**TENSTR** option defines the tensile strength of the segment, TENSTR, ksi or mpa. The default value of TENSTR is 1.66 times the yield stress.

MOSES uses the class properties to compute various loads which will act on the body. The options:

–**WTPLEN**, WTPFT

–**DISPLEN**, DPFT
–**BUOYDIAMETER**, B_DIAMETER
–**DRAGDIAMETER**, D_DIAMETER
–**WINDDIAMETER**, WOD
–**AMASDIAMETER**, AMOD

can be used to alter the properties which will be used to compute the loads. Normally, the weight per length is computed from the true cross–sectional area and the density. If, however, the **–WTPLEN** option is specified, the weight per unit length will become WTPFT (bforce/blength). The buoyancy for an element is computed as the sum of that arising from a diameter and a displacement per unit length. For tubular members, the displacement per length is zero and the diameter is the OD. For non–tubular elements, the diameter is zero and the displacement per unit length is the element cross–sectional area times the density of water. One may alter this with the **–DISPLEN** and **–BUOYDIAMETER** options. Here, DPFT is the new displacement per unit length (bforce/blength) and B_DIAMETER is the "buoyancy diameter" (inches or mm). The wind force, viscous drag, and added mass are all computed based on an equivalent diameter (inches or mm). These can be altered via the options: **–DRAGDIAMETER**, **–WINDDIAMETER**, and **–AMASDIAMETER**.

Once a class has been defined, it can be redefined with the command:

ED_CLASS, ~CLASS_NAME, SEGNO, ...

Here, ~CLASS_NAME and SEGNO define the class and the segment which are being edited, and the remainder of the command is the same as the class command. The only thing that the segment will inherit from the previous definition is the length or percent length. If SEGNO is omitted, the first segment will be redefined, and if SEGNO is greater than the last segment defined, the last segment will be modified. For example,

ED_CLASS ~CLASS_NAME 1 TUBE 60 1.5 –FY 42

will change the diameter, thickness and yield strength of the first segment of the class ~CLASS_NAME to the values specified.

To add a segment, specify a value of zero for SEGNO. The **ED_CLASS** command works in the input channel through **INMODEL**, or under the **MEDIT** menu.

A string function is available that provides information regarding classes, and has the following form:

&CLASS(INFO, CLS_NAM, SEGNO)

where INFO must be either **ELEMENTS**, **N_SEGMENT**, **TYPE**, **DIMENSION**, **FRICTION**, **DENSITY**, **EMODULUS**, **SPGRAVI**, **ALPHA**, **FYIELD**,

POI_RATIO, **TENSTRENGTH**, **B_TENSION**, **WTPLEN**, **DISPLEN**, **BUOYDIAMETER**, **DRAGDIAMETER**, **WINDDIAMETER**, **AMAS-DIAMETER**, **SECTION**, **PERLENG**, **LENGTH**, **REFINE**, **RDES**, **POINTS**, **REFERENCE**, **T_STIFF**, **L_STIFF**, **CFB**, **F_TYPE**, **R_SPACE**, **P_FY**, **M_P**, **ETA**, **P_N**, **SCF**, **SN**, **PLATE_DI**, **NAME_DIM**, **NAME_SEG**, **CLUMP**, **SOIL**, **PYMULT**, **TZMULT**, **QWMULT**, **SEND**, **SLOPE**, **DEPANC**, **IG_STIF**, **PR_M_THRUST**, **PR_EFFICIENCY**, **PR_T_ALIMITS**, **PR_R_ALIMITS**, **RR_ALPAH**, **PR_GAMMA**, or **PR_R_DIST**. Here CLS_NAM is the name of the class used to return information, and SEGNO is the segment number of the class. Most of these return values of that one sets with options of the same name on a class command. Some, however, require more information. When INFO is **TYPE**, the string function returns the type of class, such as H_CAT, B_CAT, ROD, SL_ELEM, GSPR for flexible classes, or any of the various structural section types, such as PLATE or TUBE. A value of **N_SEGMENT** returns the number of segments in a class, while a value of **ELEMENTS** provides the element names using the specified class name. When INFO is **DIMENSION**, the dimensions of the specified segment of the class are returned. These dimensions could be diameter for class types such as H_CAT, or cross section dimensions for structural classes. The value of INFO of **NAME_DIM** returns to name of the dimensions returned with **DIMENSION**, e.g. Diameter and Thickness. Finally, the value of INFO of **NAME_SEG** returns the name of the segment type, e.g. Rod.

XII.M.1 Structural Classes

There are quite a few different section types which one may use in defining a class for a structural beam or plate:

```
~CLASS SHAPE_NAME                    PT PW –OPTIONS
~CLASS TUBE    A B C D               –OPTIONS
~CLASS CONE    A B C           PT PW –OPTIONS
~CLASS BOX     A B C D         PT PW –OPTIONS
~CLASS WBOX    A B C D E       PT PW –OPTIONS
~CLASS PRI     A B             PT PW –OPTIONS
~CLASS IBEAM   A B C D         PT PW –OPTIONS
~CLASS G_IBEAM A B C D E F G H PT PW –OPTIONS
~CLASS TEE     A B C D         PT PW –OPTIONS
~CLASS CHANNEL A B C D         PT PW –OPTIONS
~CLASS ANGLE   A B C D         PT PW –OPTIONS
~CLASS D_ANGLE A B C D         PT PW –OPTIONS
~CLASS LLEG    A B C D E F G H PT PW –OPTIONS
~CLASS PLATE   A B C D               –OPTIONS
```

Here, the letters A through H which follow the section type are dimensions (inches or mm) which describe the size of the section and are defined in Figures 5 and 6 at the end of this section. The remaining two pieces of data, PT and PW define the thickness and width (inches or mm) of any attached plate one wishes to include in the section. Any attached plate is always attached at the side of the section in the beam +Z direction. The area of the plate is not included in the axial area of the section, but the inertia and resulting change in neutral axis is considered. Notice that a **PLATE** can be corrugated if one inputs the three additional lengths which define the corrugation. If only one number is input, the plate is flat.

All of these sections except the **LLEG** are connected to the nodes at the neutral axis *unless* instructed otherwise with the option:

–**REFERENCE**, WHERE

Here, WHERE must be either **TOP** or **BOTTOM**. When this option is used, the node will be attached at the center of the "top" or "bottom" of the section. With symmetric section this is quite clear, but with non symmetric ones it is a bit harder to describe. An angle, for example, will be connected at the center of the flange if BOTTOM is used, but the bottom of the web when TOP is used. A **LLEG** section is special in that by default, the node is connected to the center of the tubular portion.

The shapes are as show below. The option:

–**REFLECT**

can be used on shapes which are not symmetric about the neutral axis to reflect the shape about it. This has the same effect as specifying

–CA 180

as an option on all of the elements which use this class.

In addition to the standard options discussed above, there are several ones specific to *beam* classes, the first group of these are:

–**SCF** SCF_BEG, SCF_END
–**SN** CURVEA, CURVEB

which define fatigue properties for the elements. For a discussion of the –**SCF** and –**SN** options, see the sections on associating SCFs and SN curves with fatigue points.

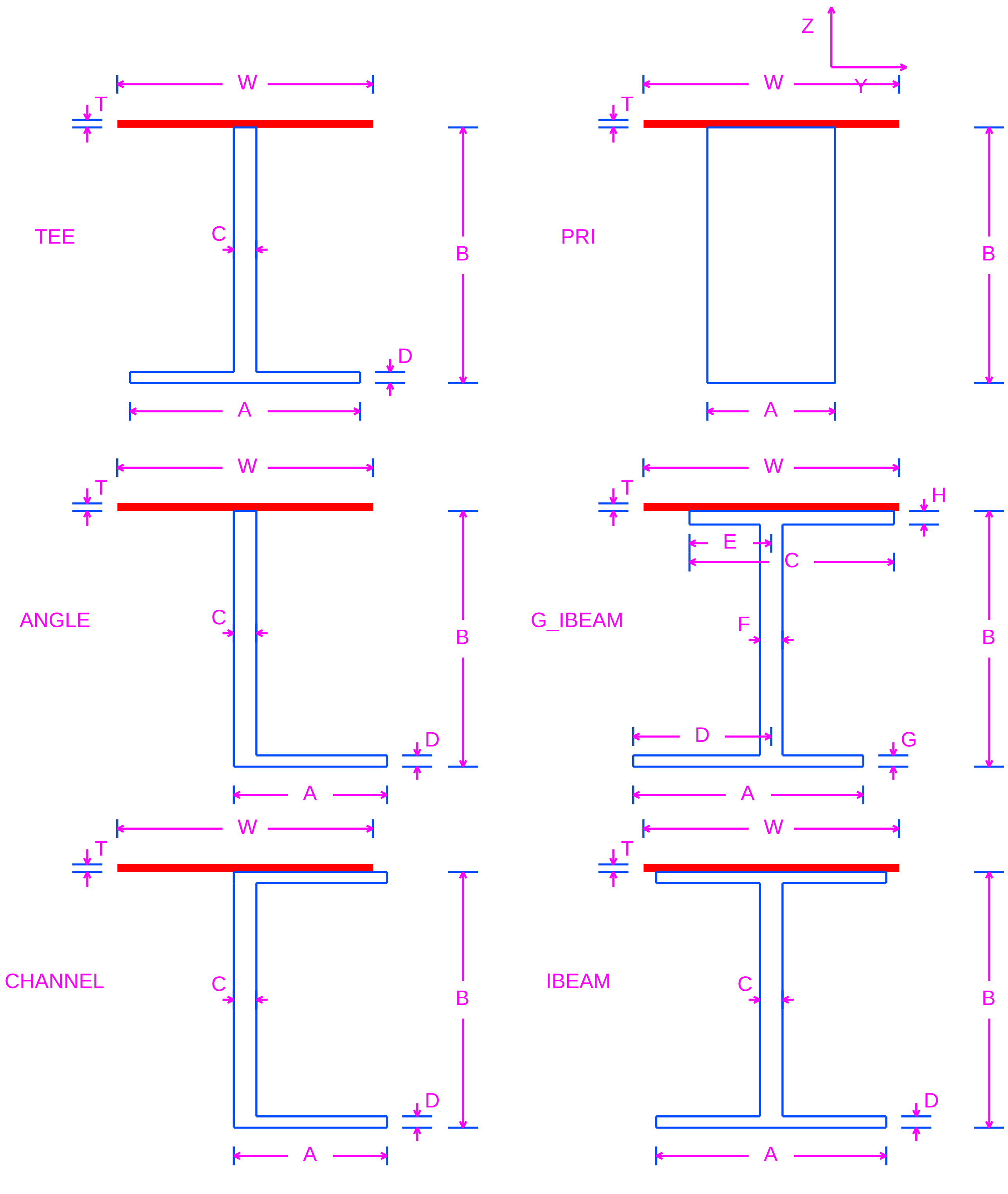
The next group contains the section options:

```
–SECTION, AREA, IY, IZ, J, ALPHAY, ALPHAZ
–POINTS, Y(1), Z(1), AY(1), AZ(1), ..... \
            Y(n), Z(n), AY(n), AZ(n)
–P_FY, FY(1), FY(2), .... FY(n)
–M_P, Zy, Zz
–P_N, Pn
–ETA, ETA
–F_TYPE, TYPE
```

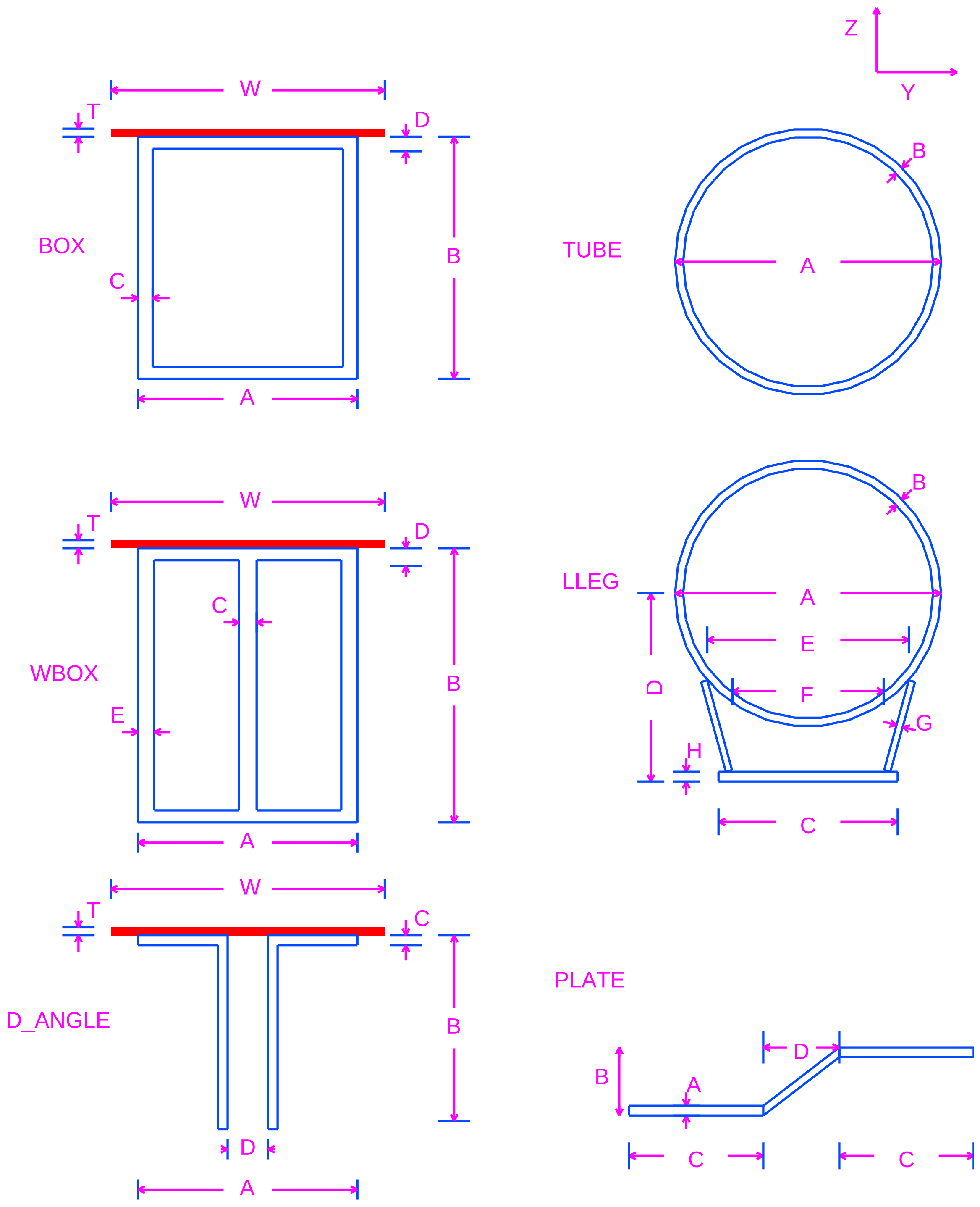
If one wishes to override the section properties computed from the dimensions, then he should use the –**SECTION** option. Here, AREA is the cross–sectional area (inches**2 or mm**2), IY and IZ are the moments of inertia (inches**4 or mm**4), J is the polar moment of inertia (inches**4 or mm**4), and ALPHAY and ALPHAZ are the shear area multipliers. The ALPHAs are numbers which transform average shear stresses into the true shear stresses at the neutral axis, i.e.

TAU(n) = ALPHA(n) * SHEAR_FORCE / AREA

If any of these values are less than or equal to zero, the program computed value



CATALOG OF SECTIONS
FIGURE 5



CATALOG OF SECTIONS
FIGURE 6

will be used.

Normally, MOSES determines "critical" points at which to compute stresses. Sometimes, one wishes to define these points himself. Stress points are defined with the –**POINTS** option. Here, Y(n) and Z(n) are the beam system Y and Z coordinates of the point and AY(n) and AZ(n) are the values of alpha for that point. (These are the same alphas discussed above except that instead of computing the stresses at the neutral axis do it at the nth point.) The option –**P_FY** allows one to define the yield stress at the "critical points". If one uses this option, he really should define his own points so that he is certain of the physical location of each point. The options –**M_P** and –**P_N** define the plastic moments and the nominal axial strength of the section. *NOTICE* If you use the –**SECTION** option to change the properties, you probably will also need to use the –**M_P** and –**P_N** options as well. The option –**ETA** defines the exponent "eta" in the interaction formulae of the AISC LRFD code check. Finally, the option –**F_TYPE** defines the fabrication type of the section. Here, TYPE must be either **FABRICATED** or **COLD_FORGED**.

The next class of Structural Class options are:

–**LEN** , L
–**PERL**, PCLEN
–**REFINE**, NUM_REFINE
–**RDES**, :NAME, KL/R_LIM, D/T_LIM

In general, MOSES allows for an element to have different properties along its length. To define such an element, one should have a Class definition command for each segment of the element. When more than one segment is defined, the first set of properties are associated with the beginning or "A" end of the element, and the last set of properties associated with the "B" end of the element. The lengths of the segments are defined by the –**LEN** option with L (feet or meters) being the length. For BEAM classes, the length of every segment but one should be defined. MOSES will then compute the length of this segment so that the total length of the element is correct. Also, one can define the segment lengths with the –**PERL** option. Here, PERL is the percentage of the total length of the element which will be attributed to a given segment.

When one uses a shape type of **CONE**, MOSES will automatically divide the beam into a number of prismatic segments based on the number specified with the –**REFINE** option. Thus, if –**REFINE** is not specified, then the beam will consist of a single tube with a diameter which gives the correct volume. The thickness of the approximate cylinder is

T = Ti * (R1 + R2) / (2 * Ra)

where Ti is the thickness input, R1 and R2 are the radii at the ends and Ra is the diameter of the approximate cylinder. This is an approximation of the correct thickness for small values of (R1–R2)/L. Here, the first dimension given is the outside diameter (inches or mm) at the "beginning node" of the element, the second dimension is the thickness (inches or mm), and the third dimension is the outside diameter (inches or mm) at the "end node" of the element. The options which alter the load attributes are not honored for cones, and one cannot use a cone section as the one with zero specified length if when defining beams composed of different shape types if the section is refined.

MOSES has an ability to automatically redesign a class of members so that all members within that class have favorable code checks. All resizing is performed on a subset of the *shape table* (more will be said about this in the next section). The subset considered during resizing is defined by a single selector, and two limits define with the **–RDES** option. The program will consider only those shapes which match the selector. For tubes, only sections which satisfy the d/t and kl/r limits specified will be considered. MOSES will consider all shapes selected to produce a shape which will yield a minimum cost and which satisfies the code check criteria.

MOSES allows one to define "stiffeners" for structural elements. Both longitudinal and transverse stiffeners can be defined, and they add both stiffness and weight to a model. In addition to adding weight and stiffness, stiffeners are used in checking some codes. In general, stiffeners are associated with a *previously defined class.* The weight added to elements by stiffeners can be eliminated with the use of the –**ST_USEW** option of either **&DEFAULT** or on the element definition command, or by specifying **–WTPLEN** to be zero when defining the stiffener class. The weight is computed by the weight per length of the stiffener times its length times the number of stiffeners. If one does not specify a class for the stiffeners, then the stiffeners are "magic" in that they are weightless and automatically pass any checks on their properties.

While these are conceptually simple, one can easily become confused by the details. While the form of the options used to define stiffeners is the same for all elements, the details differ. Thus, let us begin by considering stiffeners on generalized plates. Here, the options used to define stiffeners are:

> **–T_STIFF**, SPACE, ~STIF_CLASS, WHERE
> **–L_STIFF**, SPACE, ~STIF_CLASS, WHERE

The option which begins with **–T** defines transverse stiffeners and that beginning with **–L** defines longitudinal ones. Here, longitudinal stiffeners are parallel to the element X axis and transverse ones are perpendicular to the X axis. For both of these, options, ~STIF_CLASS is the class name which will be used to define the stiffener, and *have been defined previously.* WHERE defines the "vertical position" of the stiffener. WHERE may be either **+Z**, or **–Z**. If +Z is used, the stiffeners will be connected to the "top" size of the plate and for –Z, the bottom side. If WHERE is omitted, INTERNAL or +Z will be used.

Longitudinal stiffeners on beams are similar to longitudinal stiffeners on plates, and are defined with the **–L_STIFF** option discussed above and

–LN_STIFF, NUMBER, ~STIF_CLASS, WHERE

Also, here WHERE can also have the additional values of **INTERNAL** of **EXTERNAL** which make sense for closed sections. In reality, +Z and EXTERNAL are the same as are –Z and INTERNAL. The difference between **–L_STIFF** and **–LN_STIFF** is that the first defines the location of the stiffener by a spacing (here SPACE is in inches or mm) and the second by the number of stiffeners. Obviously,

SPACE = DISTANCE / (NUMBER + 1)

Where DISTANCE is the distance over which the stiffeners are applied. For tubes, the distance is the circumference (inner or outer depending on WHERE). All other shapes are composed of rectangles. Here, DISTANCE and NUMBER are used for each rectangle of the section and DISTANCE is the longer dimension. Thus, for a "PRI" section, the DISTANCE is the greater of A and B. Fractional stiffeners are used and they are "smeared" over the full width. Thus, adding stiffeners increases both of the inertias of the section.

Transverse stiffeners on beams are the most complicated and are defined with either of the two options:

–T_STIFF, SPACE, ~STIF_CLASS, WHERE, LENGTH
–TN_STIFF, NUMBER, ~STIF_CLASS, WHERE, LENGTH

Here, the values have the same meaning as those for longitudinal stiffeners on beams. The new value, LENGTH, will be discussed in a minute. One of the problems with transverse stiffeners is that they are used for two purposes: stiffeners against hydrostatic collapse and to stiffen joints. Since transverse stiffeners on beams are normally used on tubes, we will call them rings here. To avoid confusion, rings will be used to stiffen joints **only** if the class has more than one segment and the stiffeners are in the segment closest to the joint}.

Transverse stiffeners suffer from the same problems as does buckling lengths. If an element is not fully supported on both ends, then the longitudinal stiffener spacing may be longer than the element length! This is where the value LENGTH comes in. It provides a "DISTANCE" to be used in the conversion from spacing to number of stiffeners. You can specify three things for LENGTH: a length in feet or meters, the token **LENGTH**, or the token **BLENGTH**. If **LENGTH**, is specified, the length of the element is used, if **BLENGTH** is specified, the minimum of the two buckling lengths is used, a number input will be used directly, and if this parameter is omitted, the segment length will be used.

Let us consider two examples. First, suppose that we have an element which is actually part of a logical beam, and suppose the logical beam has a single hydrostatic ring. If all elements of the logical beam had the same properties, they could be defined with the class:

```
~LBEAM TUBE OD T –TN_STIFF 1 ~SC EXTERNAL BLENGTH
```

This class will place one ring in the center of the logical beam and have a stiffener spacing of half the minimum buckling length. Each element would have the same stiffener spacing, but a stiffer weight of the weight of the ring times the element length divided by the buckling length. As another example, consider a beam for which we need a joint ring at one end. The class

```
~LBEAM TUBE OD T –TN_STIFF 1 ~JR EXTERNAL –LEN 2.5
~LBEAM TUBE OD T –TN_STIFF 1 ~HR EXTERNAL
```

We *must* use a segment to define the joint ring, hence the first segment. The hydrostatic ring defined for the second segment will be the middle of the second segment and the stiffener spacing here is half the length of the second segment.

XII.M.2 Class Shapes

MOSES maintains a table of standard shapes. Here, a ”shape” is nothing more than a partial class definition. If one has an element made of one of these shapes, he can simply specify it by SHAPE_NAME. In other words, one can specify:

~CLASS W14X140

and nothing else is required. Alternatively, one can now add any valid class option to this definition to tailor it for his purposes. One can obtain a list of the currently available shapes by issuing **&NAMES SHAPES**. The basic table supplied with the program contains AISC, British, and French shapes.

One can add to the default shape table by entering a new menu with the command:

&DATA SHAPES

This command should be followed by records of the form:

NAME, TYPE, A, B, ... H –OPTIONS

where the available options are the section options discussed above:

–**SECTION**, AREA, IY, IZ, J, ALPHAY, ALPHAZ
–**POINTS**, Y(1), Z(1), AX(1), AY(1), ... \
            Y(n), Z(n), AX(n), AY(n)
–**P_FY**, FY(1), FY(2), ... FY(n)
–**M_P**, Zy, Zz
–**P_N**, Pn
–**ETA**, ETA
–**F_TYPE**, TYPE
–**T_STIFF**, SPACE, ~STIF_CLASS, WHERE
–**L_STIFF**, SPACE, ~STIF_CLASS, WHERE
–**TN_STIFF**, NUMBER, ~STIF_CLASS, WHERE
–**LN_STIFF**, NUMBER, ~STIF_CLASS, WHERE

Here, NAME is the name which one wishes to give the shape, and TYPE is a valid class section type (**TUBE**, **CONE**, **BOX**, **PRI**, **WBOX**, **IBEAM**, **G_IBEAM**, **TEE**, **CHANNEL**, **ANGLE**, **D_ANGLE**, **PLATE**, or **LLEG**), and A, B, etc. are dimensions (inches or mm) which are appropriate to define the shape.

When the shapes have been completely defined, one should issue **END_&DATA** to exit. Normally, shapes defined via this menu are added to the basic shape table provided with the program, and remain defined only for the duration of a given database. One can permanently add shapes to the basic table. To find out how to accomplish this, look in the section on Customizing Your Environment.

For the AISC shapes, the standard names are used for most of the shapes. The exceptions occur when the standard name exceeds eight characters. Jumbo W shapes are denoted by a **J** suffix. Angles are named **L**ddwwtt where dd is the depth in 1/10s of an inch, ww is the width in 1/10s of an inch, and tt is the thickness in 1/16s of an inch. Double angles are named s**L**ddwwtt where s is the spacing between the two angles in 1/8 of an inch, dd is the depth in 1/10s of an inch, ww is the width in 1/10s of an inch, and tt is the thickness in 1/16s of an inch. Square and rectangular tubes are named **TS**ddwwtt where: dd is the depth in inches, ww is the width in inches, and tt is the thickness in 1/16s of an inch. Here, exceptions are made for dd and ww values for some of the smaller sizes. For these tubes, dd and ww are in 1/10s of an inch.

For British shapes, names **U**ddd**B**mmm are used for Universal Beams with depth ddd (millimeters), and mass mmm (kilograms) per meter. Likewise, **U**ddd**C**mmm is used for Universal Columns, **U**ddd**P**mmm is used for Universal Bearing Piles, and **J**ddd**S**mmm for Joists. The remaining shapes are named **C**ddd**B**www for channels, **SH**ddttt for Square Hollow Sections, **RH**ddwwtt for Rectangular Hollow Sections, **LE**dddttt for Equal Angles, and **LU**ddwwtt for Unequal Angles. Here, dd denotes the depth in centimeters, ww the width in centimeters, and tt the thickness in millimeters.

The French shapes in the table are denoted **HEAZ**ddd, **HEAY**ddd, **HEBZ**gddd, **HEBY**gddd, **HEMZ**ddd, **HEMY**ddd, **IPEZ**ddd, **IPEY**ddd, **IPEZ**ddd, and **IPEY**ddd. Here, the shapes with the Z in their name are defined with the strong axis in the normal direction while those with the Y in their name are rotated 90 degrees. Here, ddd is the depth of the section in millimeters.

The shape type of **TUBE** is special in that no plate can be attached to a tube. This shape also allows for having a tube inside of a tube, with the inside tube being specified by the dimensions C and D. If one has an inside tube, then he should *not* specify contents for this element within this class. This is useful for defining piles inside of legs.

XII.M.3 Pile Classes

A pile class is simply a beam class with a soil defined and some extra options available. The extra options for ~CLASS PILE classes are:

> –**REFINE**, NUM_REFINE
> –**PYMULT**, PMUL, YMUL
> –**TZMULT**, TMUL, ZMUL
> –**QWMULT**, QMUL, WMUL
> –**SOIL**, SOIL_NAME

The –**REFINE** option defines the number of segments into which the pile segment will be broken to solve the nonlinear pile/soil interaction problem. If it is omitted, a single element will be used. The options –**PYMULT**, –**TZMULT**, and –**QWMULT** define multipliers for the basic soil properties. If they are omitted, values of 1 will be used.

The option –**SOIL** defines the name of the soil in which the pile will be embedded. Here, SOIL_NAME is the name of a soil which has been previously defined in the **&DATA** Menu. This is accomplished by first entering the menu with the command:

> **&DATA SOIL**, SOIL_NAME

and when the definition is complete, exit the menu with **END_&DATA**. In the menu, one uses the following:

> **DEPTH**, ZDIS
> **PY**, P(1), Y(1), ..., P(n), Y(n)
> **QW**, Q(1), W(1), ..., Q(n), W(n)
> **TZ**, T(1), Z(1), ..., T(n), Z(n)
> **MPY**, MULP, MULY
> **MQW**, MULQ, MULW
> **MTZ**, MULT, MULZ

The basic rule here is that one first defines a depth. All of the properties then defined, until a new depth is encountered, apply to the specified depth. Here, ZDIS is the positive distance from the mudline to point of interest (feet or meters). For points between two depths, a linear interpolation is performed, and for points outside the table, the last closest point is used. The **PY**, **QW**, and **TZ** commands define the force deformation behavior of the soil. Here,

- **PY**– defines the lateral behavior. P is the lateral force per unit length of pile (bforce/llength) which is required to produce a lateral deflection of Y (inches or mm),
- **TZ**– defines the skin friction. T is the skin friction per unit length of pile (bforce/llength) which is required to produce a deflection on the pile surface

of Z (inches or mm),
- **QW**– defines the end bearing. Q is the force (bforce) at the end of the pile resulting from a vertical deflection of W (inches or mm).

The commands **MPY**, **MQW** and **MTZ** can be used to establish multipliers for the curves. This is particularly important since all of the basic curves depend on pile size. With these commands, the ith set of values will be multiplied by MULi before the data is stored in the database. As an example, if your PY data is given in tonnes and meters and your current units are kilo–newtons and meters, you could issue:

```
MPY, 1./1000, 1000.
```

and input the data as given.

Consider the following example:

```
~PILE TUBE 1066.80 25.4 –DENSITY 77008.5  –SOIL DIRT  \
                    –REFINE 20 –FYIELD 360 –PYMULT 0.001
CONNECTOR PILE1 ~PILE  *J6110 *P110
```

Here, a tubular class is defined, the density is modified, the soil name is specified, and the pile is divided into 20 segments for a structural solution. The yield strength and PY multiplier is also modified. This class definition is then used to describe a connector, where *J6110 belongs to the part jacket, and *P110 belongs to the part ground.

XII.M.4 Flexible Connector Classes

MOSES provides for several ways to flexibly connect bodies to one another or to ground. These range from simple springs to assemblies of elements such as lifting slings and pipe assemblies. MOSES bases the definition of a connection on the type of class defined for it.

To define an element with properties which vary along the length, one should define multiple class commands with the same name. In contrast to beam and pile classes, every segment's length *must* be defined for flexible classes. Also, the order of the segments can be thought of as starting at the fairlead, moving toward the anchor.

There are three classes which define connectors with zero length, FOUNDATIONs, LMUs, and GSPRs. The class definition of these is:

```
~CLASS, FOUNDATION, SENSE, DF(1), SPV(1), AF(1)  ... \
                    DF(n), SPV(n), AF(n),      \
                                   –OPTIONS
~CLASS, GSPR, SENSE, DF(1), SPV(1),  AF(1) ... \
                     DF(n), SPV(n), AF(n),      \
                    –LEN, L, –OPTIONS

~CLASS, LMU,  LEN, OD(1), OD(2),              \
                    DF(1), SPV(1), AF(1) ...  \
                    DF(n), SPV(n), AF(n),     \
                    –OPTIONS
```

In addition to the standard class options discussed above, the following are, in general, available:

–SYMMETRIC, YES/NO
–IG_STIFF
–SEND, KE(1), KE(2)
–CONVOLUTION, CVL_NAME
–X_PY, P(1), Y(1), P(2), Y(2), ..., P(n), Y(n)
–Y_PY, P(1), Y(1), P(2), Y(2), ..., P(n), Y(n)
–Z_PY, P(1), Y(1), P(2), Y(2), ..., P(n), Y(n)
–X_DAMPING, Co, Ex, Fo
–Y_DAMPING, Co, Ex, Fo
–Z_DAMPING, Co, Ex, Fo
–FRICTION, MU

Here DF(i) is the name of the degrees of freedom (in the element system) which will be restrained and must be chosen from the list: **X**, **Y**, **Z**, **RX**, **RY**, **RZ**. By default, the element system is aligned with the body system of the first node defining a connection. More will be said about this when defining elements are discussed.

SPV(i) is the value of the spring (bforce/blength for degrees of freedom X, Y, and Z, and bforce–blength/radian for degrees of freedom RX, RY, and RZ) and AF(i) is the allowable force in this direction. For GSPRs and FOUNDATION elements, SENSE defines the direction in which the element will act; i.e. if SENSE is **TENSION**, then the connector will have no load (in any degree of freedom) when the element X displacement is negative. If SENSE is **COMPRESSION** then it will have no load when the element X displacement is positive. The length here defines an element X offset from the nominal beginning of the element to the spring. In reality, a FOUNDATION is simply a GSPR (generalized spring) which has zero length, and will be checked in a FOUNDATION check. The **–SYMMETRIC** option can be used with connectors which are symmetric about the element X axis. If this option is used, then one should only define the "Y" properties and MOSES will automatically take care of the rest. By default YES/NO is NO for GSPR and FOUNDATION connectors and YES for LMU connectors.

A LMU has geometry – a length and two diameters and is intended to model a pin in a cone. The cone and pin geometry are illustrated in Figure 7. Here, all three numbers are in inches or mm. It too is a GSPR with a SENSE of COMPRESSION, but the deformation is not measured from the two points. It is best to think of the two point for the connector as the tip (bottom) of the pin and the "top" of the cone. Let us define

T = (OD(2) – OD(1)) / (2 LEN)

Which is the tangent of the cone angle. Now, if the pin tip is above the cone top, then there will be no force. Now let D_v be the distance the pin tip is below the cone top and D_h be the horizontal distance that the pin tip is away from the cone center. If D_v is less than LEN, then we will have a horizontal deformation given by:

Delta_h = D_h – T * (LEN – D_v)

If we assume that the force between the sides of the cone and the pin is perpendicular to the surface, then horizontal deformation will produce a vertical deformation

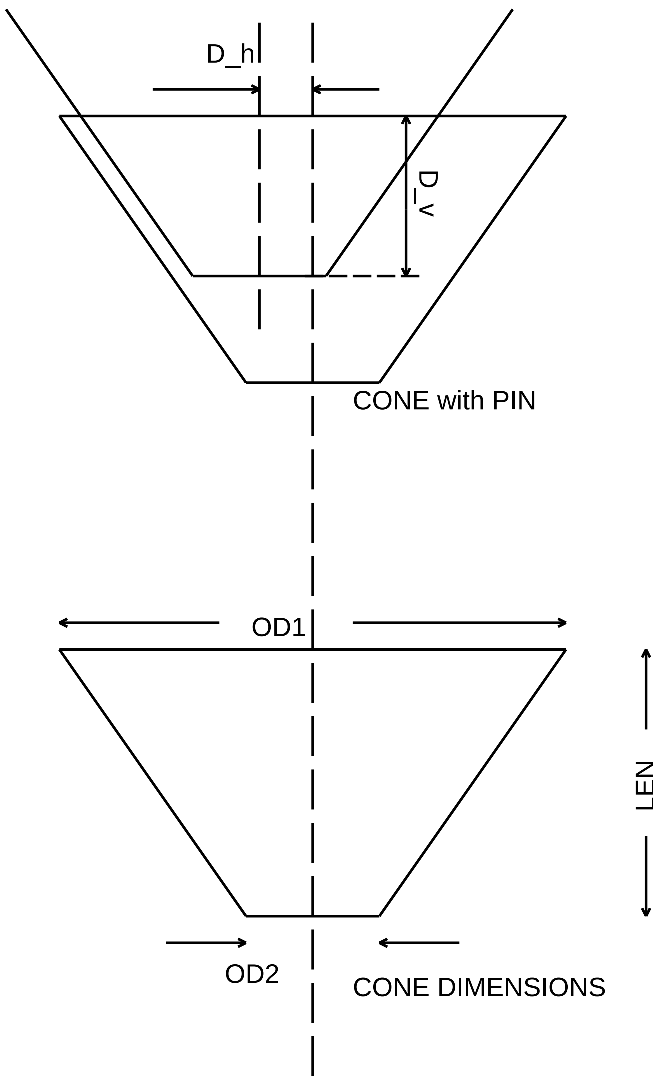
Delta_v = –Delta_h * T

Finally, if D_v is greater than LEN the two deformations are

Delta_v = D_v – LEN – (D_h * T)
Delta_h = D_h

Now, the force in the connector is computed from the two deformations and the two spring constants.

The **–IG_STIFF** option is applicable for flexible connector classes as well as



LMU CONNECTOR

FIGURE 7

restraint classes. With this option, the stiffness of connectors attached to other bodies will not influence the structural results of the desired body. Additional springs can be combined in series with the basic springs by including either (or both) of the options **–SEND** or **–X_PY**. The first of these defines a nonlinear spring where the stretch in the spring, DEL, resulting from a force, F, is given by:

DEL = (KE(1) + F * KE (2)) * F.

The units for KE(1) are ft./kips, ft./l–tons, ft./s–tons, m/mt or m/kn and for KE(2), ft./kips**2, ft./l–tons**2, ft./s–tons**2, m/mt**2 or m/kn**2. The –**CONVOLUTION** option allows one to add an additional spring, but the behavior here is defined by a convolution CVL_NAME. This allows one to define springs that are viscoelastic as well as elastic.

The **–X_PY**, **–Y_PY** and **–Z_PY** options define nonlinear springs in series with the basic x, y or z stiffness. Here, the letter following the – defines the "element" degree of freedom to which the extra stiffness will be applied. All of these options define points on the force/deflection curve of a spring. Here, P(i) is the force in bforce, and Y(i) is the resulting deflection in feet or meters. The curve defined must have the deflection being a function of the force. In other words, there can be no two points with the same P, and the points must be defined in increasing P. Also, MOSES assumes that the tension/compression behavior of the spring is the same so that the values should all be positive. Also, to avoid mathematical difficulties, Y should be a monotone increasing function of P.

The **–X_DAMPING**, **–Y_DAMPING**, and **–Z_DAMPING** options defines a nonlinear dashpot at the end of the element. As with the PY options discussed above, the letter following the – defines the degree of freedom in which these dashpots act. Here, the force is given by:

F = Co * (v **Ex)

for F <= Fo (bforce). Here v is the relative velocity and Co is in (bforce–sec/ft or bforce–sec/meter). This dashpot is only active when the spring will have a force in it.

The **–FRICTION** option can be used to limit the force in the element y and z directions based on the element x force, and the precise behavior depends on the stiffness data. If either of the Y or Z stiffnesses are non zero, then friction is used as follows. After the force components Fx and Fz have been computed, they are checked against the product of Fx and MU; i.e. we compute

```
Fh   = sqrt ( Fy*Fy + Fz*Fz )
Fm   = MU * abs ( Fx ) / Fh
```

FACT = min (Fm, 1)

and then actually apply FACT * Fy and FACT * Fz. If both lateral stiffnesses are zero then the lateral force applied will be Fm and it will be applied in the direction *opposite* the relative velocity.

Three classes are used to define the constituents of slings and pipe assemblies:

```
~CLASS, SLING,    OD,    –LEN, L, –OPTIONS
~CLASS, DAVIT,    OD,    –LEN, L, –OPTIONS
~CLASS, ROLLER, SENSE, DF(1), SPV(1), AF(1)  ... \
                DF(n), SPV(n), AF(n),     \
                               –OPTIONS
```

Slings and davits are similar in that they are really lines whose force depends only upon the distance between the ends. The cross–sectional area of these elements is computed from OD (inches or mm) assuming that the section is circular. The stiffness is then given by AE and L (feet or meters). The only other option available for this class is **–EMODULUS**.

A **ROLLER** is a limited GSPR element. It is limited in that all of the options discussed above are not available. Here the available options are:

```
–Y_PY  , P(1), Y(1), P(2), Y(2), ......., P(n), Y(n)
–Z_PY  , P(1), Y(1), P(2), Y(2), ......., P(n), Y(n)
–Y_ROLLER, Y–, Y+
–Z_ROLLER, Z–, Z+
```

If neither the **–Y_ROLLER** nor the **–Z_ROLLER** is specified, then the roller will automatically restrain lateral motion and prevent the pipe from going below the roller. The values Y–, Y+, Z– and Z+ (inches or mm) define "gaps" for the roller. Here a roller is assumed to consist of four physical rollers. A pair of these restrain motion in each of the pipe system Y and Z directions. The Y– and Y+ values are the dimension from the end of the connector to the physical roller. The same can be said for the Z dimensions. To get a one sided constraint, use Z– equal 0 and Z+ equal 10000. This says that the "bottom" roller is at the connector node and the "top" roller is 10000 above the node. Since the pipe is unlikely to move this much, the top roller will never be active. All of these elements must have a single node, and the second end will be taken care of in the assembly itself.

There are four classes which can be used to define a connector with finite length.

```
~CLASS, ROD,      OD, T,     –LEN, L, \
              –REFINE, N,     –OPTIONS
~CLASS, B_CAT,    OD,   FLAG, –LEN, L, \
              –DEPANCHOR, DEPTH –OPTIONS
```

~CLASS, **H_CAT**, OD, FLAG, –**LEN**, L, \
–OPTIONS
~CLASS, **SL_ELEM**, OD, FLAG, –**LEN**, L, \
–OPTIONS

Here, OD and T are the outside diameter and thickness (inches or mm). In reality, a ROD is the accurate solution to the problem (including local dynamics) and the others are different levels of approximation. Also, a ROD can have temperature, pressure, and contents as specified with **–T_PRESSURE** option on the **&ENV** command and an **T_PRESSURE** command. A SL_ELEM is an element whose force depends only upon the distance between the ends and acts in the direction of the vector from one end to the other. By default, an SL_ELEM can have both tension and compression. If, however, a value of **TENSION** is specified for FLAG, then is can have only tension. A B_CAT is a catenary to ground and a H_CAT is a hanging catenary. Both of the catenaries employ restrictions and assumptions and FLAG controls a second level of approximation:

- The only force which is assumed to act is the weight (in air weight for a H_CAT and in water weight for a B_CAT).
- A B_CAT can only connect a body to ground and DEPTH is the depth at the anchor.
- By default,a B_CAT tabulates the force vs horizontal distance at the initial depth. If one uses a value of **EXACT** for FLAG, then the table will not be used. This should be done when changes in depth are important.
- A H_CAT ignores both the bottom and the water line, so a spring buoy cannot be used here.
- By default, a H_CAT ignores the weight of the element so that it is really a tension only SL_ELEM. A value of **EXACT** for FLAG considers the weight.

The following are, in general, available:

–**REFINE**, N,
–**IG_STIFF**
–**SEND**, KE(1), KE(2)
–**X_PY**, P(1), Y(1), P(2), Y(2), ..., P(n), Y(n)
–**X_DAMPING**, Co, Ex, Fo
–**SPGRAVITY**, SPGR
–**DENSITY**, RHO
–**CONVOLUTION**, CVL_NAME
–**EMODULUS**, EMOD
–**POI_RAT**, POIRAT
–**ALPHA**, ALPHA
–**FYIELD**, FYIELD
–**WTPLEN**, WTPFT
–**DISPLEN**, DPFT
–**PISTON**, TYPE, LT, LD, VLONG, VSHORT, TMAX, TMIN

–**B_TENSION**, BTEN
–**C_SN**, CSN
–**TAB_LIM**, TABLE_LIMIT
–**DEPANCHOR**, DEPTH
–**CLUMP**, CW, CLEN
–**BUOYDIAMETER**, BOD
–**DRAGDIAMETER**, D_DIAMETER
–**WINDDIAMETER**, WOD
–**AMASDIAMETER**, AMOD
–**FRICTION**, BOTMU
–**SLOPE**, SLOP

A **ROD** is the *only* connector which can have forces, other than buoyancy and weight, which act along its length. It is also the *only* connector which is assumed to respond dynamically. MOSES treats rods as an assembly of large deflection tubular beam elements with inertia. The option **–LEN** is used to define the length of each segment of the rod, with L being the length (feet or meters). The option **–REFINE** is important for ROD elements. This option instructs MOSES to subdivide the segment into N elements, and thus create N+1 nodes. *If this option is omitted, a single element will be used for the entire segment.* The maximum number of elements (for all segments) of a rod is 100. Since a rod is nonlinear, the number of elements influences the accuracy of the solution. Any of the "material" options can be used to define the properties of the rod, and –**WTPLEN**, –**DISPLEN**, –**BUOYDIA**, –**WINDDIA**, –**DRAGDIA** and –**AMASDIA** can be used to change the force computations as for a beam.

The remainder of the flexible connectors, are in essence, a set of springs in series, a dashpot, and a hydraulic tension control device. All of these elements have linear stress–strain behavior, but they all can be combined with three additional springs (defined with –SEND, –CONVOLUTION, or –X_PY) which have non-linear force–deflection or viscoelastic behavior. Here, in contrast to the above "GSPR" connectors, damping and py curves can only be defined in the element X direction. A ROD cannot have a convolution. An **B_CAT** can *only* be used to connect a body to ground, while the other types of connectors can connect a body to ground or to another body.

The **–PISTON** option is used to define a hydraulic piston that is used to control the tension. Here TYPE must be either **MAX** or **MINMAX**, Lt and Ld (inches or mm) are the total stroke and the nominal position of the piston, VLONG and VSHORT (feet or meters/sec) are the lengthening and shortening velocities of the piston, and TMAX and TMIN (bforce) are the minimum and maximum tensions desired. The details of the behavior depend on TYPE. If type is MINMAX, then the piston starts at Ld and stays there until the tension exceeds TMAX or is less than TMIN. If the max is exceeded, the piston moves to increase the length with a velocity of VLONG and the tension is kept at TMAX. If the position wants to exceed LT, motion stops and the tension can exceed TMAX. If the tension is less than the minimum, then the piston moves to make the the decrease the length with a velocity of VSHORT and the tension is kept at TMIN. If the position wants to be less than zero, the motion stops and the tension constraint is removed. For values of tension between, there is no piston motion.

With a TYPE of MAX, the piston behaves as above when the tension exceeds Tmax. When the tension becomes smaller than Tmax, then the piston moves to shorten the device with a velocity Vshort until the position is at Ld. For both values of TYPE, there is no stiffness in the connector when the tension is being set at one its limits.

The **–B_TENSION** option is used to define a breaking tension, BTEN (bforce), for each segment of the connector. If this option is not specified, then one will be computed based on the area of the element and the ultimate tensile stress. This value is used to normalize the tension, to report a unity check, and in some cases to compute fatigue.

The option **–C_SN** defines the curve used to compute fatigue for an element. Here, CSN can be the name of any defined SN curve. The definition of SN curves is addressed with the &REP_SELECT command. Most of the time, CSN will be either **CHAIN** or **WIRE**. If this is true, then the API curves will be used for computing the fatigue. These curves are not really SN curves, but curves of tension ratios to cycles. For a multi segment connector of different materials, it is not obvious which segment will have the most fatigue. What is done is that the segment with the largest t/tb ratio is used when this segment has a tension type SN curve, and the segment with the largest stress is used when the critical t/tb segment has a normal SN curve. Nothing prohibits one from using a normal SN curve such as X for the fatigue.

If one has not used an EXACT FLAG for a B_CAT, MOSES will tabulate the force–distance properties of each line. The values required during execution are then interpolated from this table. This table will consist of thirty points for each line, and will start at zero horizontal force and increase up to some maximum value. This maximum is set to the maximum breaking tension over all segments, or it can be set by the user with the inclusion of **–TAB_LIM**, where TABLE_LIMIT (bforce) will be the maximum tension in the table. For values which exceed the range of the table, extrapolation will be used.

An **B_CAT** class describes a catenary line with possibly more than one segment. The depth at the anchor is defined with the **–DEPANCHOR** option where DEPTH (feet or meters) is the vertical distance (feet or meters) from the waterplane to the anchor. The length of the segment, L, (feet or meters) is defined with the **–LEN** option. The area of the cross–section is computed from the OD (inches or mm) assuming that the section is circular. The stretch in the line is computed based on the sectional area and Young's Modulus defined with the **–EMODULUS** option. The submerged weight per unit length of the catenary is computed from the area and the density, and the density of water. The weight per length can be changed with any combination of **–WTPLEN**, **–DISPLEN**, and **–BUOYDIA**. For example,

```
    –WTPLEN 10 –BUOYDIA 0
```

will set the submerged weight per length to 10, and

```
    –WTPLEN 15 –DISPLEN 5 –BUOYDIA 0
```

will accomplish the same purpose. The **–CLUMP** options adds a "clump weight" of weight CW (bforce) at the end of the current segment. If CW is less than zero, then it is the negative of the maximum buoy displacement and CLEN is the length of the pendent. A spring buoy cannot be part of the last segment where the segment attaches to ground. Thus, the connection will be constrained to lie on CLEN feet or meters below the water surface until the load that the lines exert on the buoy is equal to the maximum displacement. The buoy will then sink, so that the connection is in equilibrium. The solution for the catenary is exact *except* that the water depth is ignored for the first segment; i.e. MOSES does not consider the possibility of grounding between the spring buoy and the fairlead.

MOSES will assume that there is no friction between the seafloor and an **B_CAT** and that the sea bottom is flat, unless altered by one of the options: **–FRICTION** or **–SLOPE**. Here, BOTMU is the coefficient of friction between the line and the bottom, and SLOP is the slope of the bottom, from the vessel toward the anchor, positive if the depth increases from vessel to anchor, (vertical distance / horizontal distance).

Consider the following example for the description of a typical mooring line:

```
    ~WIRE B_CAT 4.625 –LEN 500  –BUOY 0 –WTPL 37.51/1000 \
                 –EMOD 29000*0.1 –CLUMP –100
    ~WIRE B_CAT 4.625 –LEN 4000 –BUOY 0 –WTPL 37.51/1000 \
                 –EMOD 29000*0.1
    ~WIRE B_CAT 4.375 –LEN 2000 –BUOY 0 –WTPL 158.2/1000 \
                 –EMOD 29000*0.15
    ~WIRE B_CAT 4.625 –LEN 2000 –BUOY 0 –WTPL 37.51/1000 \
                 –EMOD 29000*0.1 –DEPANCHOR 3000
```

Here, effort is saved by using minimum uniqueness and arithmetic in the input. Four line segments are described, with a spring buoy at the end of the first segment. The first, second and fourth segments share the same material properties, except for length. Notice the BUOYDIA is set to zero; therefore, the weight per length specified is the weight in water.

XII.M.5 Rigid Connector & Restraint Classes

Rigid connector and restraint classes are defined by using one of the following:

> ~CLASS, **FIX**, DF(1), ... DF(n),
> ~CLASS, **SPR**, DF(1), SPV(1), ... DF(n), SPV(n),
> ~CLASS, **GAP**, COEF

FIX and **SPR** classes define connections between specified degrees of freedom. For these connectors *it is not permitted to define offsets.* Instead, MOSES computes an offset for the second end of the connection so that the constraint is satisfied when the connection is defined. Thus, one should issue an **&INSTATE** command for each body *prior* to connection to establish the proper relative orientation. The values input when defining these classes are in the body system of the body to which the first node belongs. For both, DF(i) is the name of the degrees of freedom which will be restrained and must be chosen from the list: **X**, **Y**, **Z**, **RX**, **RY**, **RZ**. A **FIX** class will "fix" the specified degrees of freedom of the element nodes. If all degrees of freedom are to be restrained, then one can simply specify **FIX** with no other values. The **SPR** class is used to define linear springs. Here, SPV(i) is the value of the spring (bforce/blength for degrees of freedom X, Y, and Z, and bforce–blength/radian for degrees of freedom RX, RY, and RZ). These classes are used for defining both restraints and connectors. During a stress analysis, the behavior of a restraint and a connector with the same class is identical. During a simulation, however, a connector will behave differently than one may expect. During a simulation, rigid constraints are capable of restraining only translation. Thus, if one selects a rotational connector, it will *only* be applied during the stress analysis. Also, the connector applied during a simulation will be the same regardless of whether **FIX** or **SPR** was used to define the connection. The difference, however, will appear during the stress analysis where the specified flexibility will be applied.

A **GAP** connector is a rigid connector between *two* nodes. It will produce a force between the two nodes acting from the second node to the first to keep the distance between them greater than or equal to the distance between them when the gap was defined. Notice that for a **GAP** connector to be properly defined, the two nodes *cannot be* coincident. The vector from the second node to the first is called the gap direction and the length of this vector when the gap was defined is called the gap distance. The gap direction is considered to be a vector in the body to which the first node is attached. During a simulation, MOSES will compute the distance measured along the gap direction between the two nodes, and not allow this distance to be less than the gap distance. A gap cannot produce tension. If a gap has a specified friction coefficient, COEF, then it will behave differently during a simulation than during a stress analysis. For the stress analysis, the treatment is according to Coloumb's Law. For a simulation, a rigid connection is created perpendicular to the gap direction whenever the gap is active, i.e. when the gap is active, it prevents *all* relative motion between the two points.

XII.M.6 Propulsion Connector Classes

In MOSES, one can simulate the effect of a propulsion unit. These units have a thruster and optionally, a rudder. A propulsion connector class is described using:

~CLASS, **PROPULSION**, E_NAME, MAX_THRUST, R_ALPHA, R_GAMMA, R_DIST

where the available options are:

> –**R_ANGLE_LIMITS**, RA_MIN, RA_MAX
> –**T_ANGLE_LIMITS**, TA_MIN, TA_MAX

Here, E_NAME is the name of an "efficiency" curve and MAX_THRUST is the maximum thrust of the unit in bforce units. The thrust applied is the efficiency at the relative water particle velocity times the maximum thrust times the fraction of thrust applied. The last factor and the thrust and rudder angles are defined with an option on a **&CONNECTOR** command. The next three values define the optional rudder. R_DIST is the distance (feet or meters) of the rudder shaft aft of the connector point. R_ALPHA (ft**3 or m**3) and R_GAMMA (ft**2 or m**2) define the force that is exerted normal to the rudder as

```
Fn = p * R_GAMMA
p  = .5 * rho * s * vn
```

Here Fn is the force normal to the rudder, rho is the density of water, p is the pressure, s is speed the relative water velocity, and vn is the component of the relative velocity normal to the rudder. The relative water particle velocity, v, is given by

```
v  = vr + vt
vt = abs ( thrust / ( .5 * rho * R_ALPHA ) )
```

Here, vr is the relative water particle velocity in the absence of the thrust, and vt is the water particle velocity induced by the thrust.

The options –**R_ANGLE_LIMITS** and –**T_ANGLE_LIMITS** are used to define limits on the angles of the rudder and the thruster, and they should be between –90 and 90. If no limits are given, then –90 and 90 will be used. The thrust fraction can be between –1 and 1, so that by default the thruster can act in any direction. For a thruster which can act in a fixed direction only, one simply limits the angles with –**T_ANGLE_LIMITS**.

XII.M.7 Tug Connector Classes

In MOSES, one can simulate the effect of a tug boat attached to a body. Basically, this is a connector that applies a force in a constant global direction. When dynamic simulations are considered, the force the tug applies varies with wave amplitude at the tug position. A tug connector class is described using:

~CLASS, **TUG_BOAT**, FORCE

where the available options are:

–**T_DYNAM**, PERCENT_FORCE, PHASE
–**DAMPING**, C

Here, FORCE is the force of the tug in bforce units. Statically, this force is simply the amount specified. Dynamically, the force varies according to %FORCE, which is the significant percentage of force change. PHASE is the phase angle in degrees, relative to the wave crest. This data is specified using the –**T_DYNAM** option. The –**DAMPING** option defines a dashpot at the end of the tug with a constant C (bforce–sec/ft or bforce–sec/meter).

XII.N Structural Elements

In MOSES there are four types of structural elements: beams, generalized plates, part connectors, and structural post–processing elements. All of the properties of these elements are defined in terms of the "element local system". It is also the system in which loads and stresses will be reported. With respect to local system, the structural post–processing elements operate exactly the same as beams and generalized plates. The definition of the element system and the options that control it are considered in the next section. All of the structural elements have additional attributes that can be associated with them and which are defined with options. Most of these options can be used with any type of structural element and these are considered in a section below. Options which are specific to a given type of element are considered with the element definition.

While one can alter existing elements by issuing a new BEAM or PLATE command, this is often cumbersome since *all* of the data must be redefined. An alternative is offered by the command:

ED_ELEMENT, OBJECT, –OPTIONS

where OBJECT is the name of the object to which the options will apply. If OBJECT is two node names (they may include wild characters, but must begin with an *), the attributes will apply to all beams between those two nodes. If OBJECT is an attribute class name (begins with a ∼), then the attributes will apply to all elements which belong to classes which match OBJECT. If OBJECT begins with neither an * nor a ∼, then the attribute will be applied to all members whose names match OBJECT. Here, any option which is valid for the elements being edited can be specified. The **ED_ELEMENT** command works in the input channel through **INMODEL**, or under the **MEDIT** menu.

The string function which returns information about an element is:

&ELEMENT(ACTION, DATA)

Where ACTION must be either **EN_NODES**, **CLASS**, **CATEGORY**, **ELE_TYPE**, **STRW_USE**, **WEIGHT**, **BUOYANCY**, **DCOSINES**, **BLENGTH**, **LENGTH**, **SEG_LENG**, **RATIO**, **STRESS**, **CDR**, **NODES**, **RELEASES**, **E_COORDINATES**, **OFFSETS**, **CFB**, **CM**, **KFACT**, **LAMBDA**, **HAS_P–D** and **FLOODED**, **WIDTH**, **AREA**, **CENTROID**, **THICKNESS**, **SUBELEMENT**, and DATA is normally an element name. The first one, however, is different. Here, DATA is a set of node names and the data returned will be a list of element names all ends of which are in the set of node names. If one specifies only one node, then it returns all elements connected to the specified node. The remainder of ACTIONs take an element name and return element information based on action. Most of these are obvious, for example the next four types of ACTION take an element name and the last computed unit check value, the last computed axial load, and the last computed cumulative damage ratio.

The next set of options are applicable to all types of elements. The first of these returns the class name, the next the category name, and the next the element type. "BEAM" is returned for beams, "PLATE" for plates, and "OTHER" for anything else. **STRW_USE** returns YES if the stiffener weight is included, NO otherwise. **WEIGHT** and **BUOYANCY** returns respectively, the weight and maximum buoyancy of the element. **SN** and **SCF** returns the SN curves used and the SCFs. Here, the order is from the first vertex to the interface between segments and finally the second vertex for a beam. For a generalized plate, there is only one value. **DCOSINES** returns the 3X3 direction cosine matrix which transforms vectors in the element system into the part system. The next three option return element length and element buckling length, and the length of each segment in feet or meters respectively. No value is returned for **SEG_LENG** when the element is a generalized plate.

The next three set of options are valid for beams and plates. The **SN** option returns N values of SN where N is the number of segments plus one for a beam and one for a plate. The **SCF** option returns the SCFs used for beam fatigue. There are three time the number of segments plus one for a beam and one number for plates. The **J_SCF** option returns the eight SCFs used for joint fatigue at each end of the beam if both ends are parts of tubular joints. If an end is not part of a tubular joint, the NOT_A_TUBULAR JOINT will be returned for that end. If the end is part of a tubular joint, but has the default values, then TUBULAR_JOINT_DEFAULTS will be returned.

The next set of options are again valid for all types of elements. The next four options **NODES**, **RELEASES**, **E_COORDINATES**, and **OFFSETS**, return the nodes, the releases (six values of YES or NO), the coordinates (three values in feet or meters), and the offsets (three values in inches or mm) at each vertex of the element.

The next set of the actions return values only for beam elements. **CFB** returns the compression flange bracing in inches or mm and **CM** returns the CM factor used in the AISC and API code checks. **KFAC** results the K factor used in the code checks and **LAMBDA** returns the strong and weak axis values of the slenderness parameter. **HAS_P–D** returns YES of the element includes p–delta loads, NO otherwise. **FLOODED** returns YES if the element is flooded, NO otherwise.

The last set of options return non empty values only for generalized plate elements. **WIDTH**, **AREA**, **CENTROID**, and **THICKNESS** Here, width and length are the maximum distance (feet or meters) across the Y and X element axis respectively. The area is returned in ft**2 or m**2 and the centroid in feet or meters. The thickness is returned in inches or mm. **SUBELEMENT** returns the subelement names for the specified element excluding the base element itself. In other words, this action will return a null value for a triangular plate.

The string function which returns information about a subelement is:

&SUBELEMENT(ACTION, NAME)

Where ACTION must be either **RATIO**, **STRESS**, **CDR**, **NODES**, **RELEASES**, **E_COORDINATES**, or **OFFSETS**, and NAME is the name of the subelement. The options here return the same data as the same one for the &ELEMENT string function.

XII.N.1 Element System

Both beams and generalized plates have a special direction: along the length for a beam and normal for a generalized plate. Thus for a beam, the local X axis will be from the first end to the second one, and for a plate the local Z axis will be normal to the surface. One of the other two directions can be chosen arbitrarily and the third will be defined by a cross product of the two vectors.

By default, MOSES has two additional vectors which will be used to complete the local system definition: a primary direction and a secondary one. Normally, the primary direction defines the direction that an axis "points". For beams this is the Z axis, and for generalized plates it is the X. The secondary direction is used if the special direction and the primary direction are parallel. The primary and secondary vectors, SAV1 and SAV2, are defined with the –DIR_PLATE or –DIR_BEAM options of a &DEFAULT command.

The SAV1 and SAV2 vectors are "part system" vectors. This makes the business of connecting parts special since here one is dealing with two parts. The local system of a part connector will be discussed below.

For a beam the local system is constructed as follows:

- If the beam X axis *is not parallel* to the SAV1 vector, then the beam Y axis will be defined by the cross product of the SAV1 vector and the beam X axis.
- If the beam X axis *is parallel* to the SAV1 vector, then the beam Y axis is determined by the cross product of the SAV2 vector with the local X axis.

and for a generalized plate

- If the generalized plate Z axis *is not parallel* to the SAV1 vector, then the generalized plate Y axis will be defined by the cross product of the SAV1 vector and the generalized plate Z axis.
- If the generalized plate Z axis *is parallel* to the SAV1 vector, then the generalized plate Y axis is determined by the cross product of the SAV2 vector with the local Z axis.

For both beams and generalized plates, the default behavior can be changed with two options on the element definition command:

–**REFN**, *REFNOD
–**DIR_LOC**, SAV1(1), SAV1(2), SAV1(3), SAV2(1), SAV2(2), SAV2(3)

The –**REFN** option replaces the default SAV1 vector with a unit vector from the element origin (first point specified) to the point defined by *REFNOD. The –**DIR_LOC** option can be used to completely redefine both the SAV1 and SAV2 vectors. If less than 4 numbers are specified with this option, then only SAV1 will be redefined.

The local system definition described can be altered again for both beams and generalized plates. In particular, for generalized plates, the **–DIR_LOC** option can be used with only the string **NODES** following. This instructs MOSES to use the vector from the local origin to the second node as the SAV1 vector. For beams, one can specify the option:

–CA, CHANG

This option rotates the local system about the local X axis an angle of CHANG (degrees). The rotation of the system is about the beam x axis, positive towards the beam negative Y axis (right hand rule).

If for a beam one wants the strong axis of beams to be "horizontal", the SAV1 vector should be the vector defining "vertical" in the part system. In other words, if Z is the part "vertical" then SAV1 would be 0, 0, 1. Alternately if Y is the part "vertical" then SAV1 would be 0, 1, 0. In either case, SAV2 simply handles the special case and can be chosen to either suit one's fancy or to conform to some existing practice. The relationship of the local coordinate system is shown in the following figure.

XII.N.2 Element Options

In general, the available options are:

> –**DIR_LOC**, SAV1(1), SAV1(2), SAV1(3), SAV2(1), SAV2(2), SAV2(3)
> –**CA**, CHANG
> –**REFN**, *REFNOD
> –**SCF**i, VALUES(1), ... VALUES(n)
> –**SN**i, CURVE
> –**COLOR**, NAME_COL
> –**TEXTURE**, NAME_TEX, X_SCALE, Y_SCALE
> –**GO**i, X, Y, Z
> –**LO**i, X, Y, Z
> –**REL**i, REL(1), REL(2), ..., REL(5)
> –**CATEGORY**, CAT_NAME
> –**USE**, USE(1), USE(2), ..., USE(i)
> –**NUSE**, NOT_USE(1), NOT_USE(2), ..., NOT_USE(i)
> –**FLOOD**, YES/NO
> –**STW_USE**, YES/NO
> –**NUM_APPLIED**, NUMBER

The options –**DIR_LOC**, –**CA** and –**REFN** where discussed above, and for a discussion of the –**SCF** and –**SN** options, see the sections on associating SCFs Associating SCFs with Tubular Joints. and SN curves with fatigue points.

The options –**COLOR** and –**TEXTURE** define the color and texture of an element. These will be used when one asks for a picture with –COLOR MODELED. Here, NAME_COL is any color which has been previously defined. See the section on Colors for a discussion on defining colors. The NAME_TEX value for –**TEXTURE** is the name of a file in either /X/data/textures or /X/data/local/textures (here MOSES is store in /X). The X_SCALE and Y_SCALE are scale factors which will be applied to the texture. The NAME_TEX of **NONE** will yield a null default texture.

If one wishes to offset the vertices of an element from the nodes, he can employ one of the options: –**GO** or –**LO**. The –**GO** options are used to define offsets at the "ith" vertex of the element. Here, –**GO1** defines offsets at the first end, etc. The values X, Y, and Z define the coordinates of the offset (inches or mm). For the –**GO**i options, these coordinates are defined in the part system, while for the –**LO** options, they are defined in the member system. If only –**GO**, or –**LO** are specified, then all vertices will have the same offsets. *In all cases*, an offset is defined as the vector from the node to the vertex of the member.

Releases are governed by the options –**REL**. These options are used to define releases at the "ith" vertex of the element. Here, –**REL1** defines releases at the first end, etc. The values REL(i) define the particular releases to be applied. They must come from the list: **FX**, **FY**, **FZ**, **MX**, **MY**, **MZ**, *but only* as many as 5 may be specified. If –**REL** is specified, then all vertices of the element will have the same releases.

The loads due to the properties of the element are controlled with the options: –**CATEGORY**, –**USE**, –**NUSE**, and –**FLOOD**. The first three of these has been discussed with Categories and Load Types. If the –**FLOOD** option is used with a YES/NO value of **YES**, then a tubular member will be allowed to fill with water if it's below the water surface. Otherwise it will be assumed to be buoyant. The option –**STW_USE** allows one to "turn off" the weight of stiffeners. Here, YES/NO must be either **YES** in which case the weight will be used or **NO** where the weight will not be used. The option overrides the value set via **&DEFAULT**. The option –**NUM_APPLIED** allows one to have a multiplier, NUMBER, for a beam. The results for a beam are first computed based on the specified properties, and then all results are multiplied by NUMBER. In particular, the damping, stiffness, mass, matrices and the force are multiplied by this number.

XII.N.3 Beams

In MOSES, an element is viewed as being the union of three sets of attributes: its vertices, its class properties, and its "additional attributes". Defining the first two of these has already been discussed. To complete the definition of a beam, one uses:

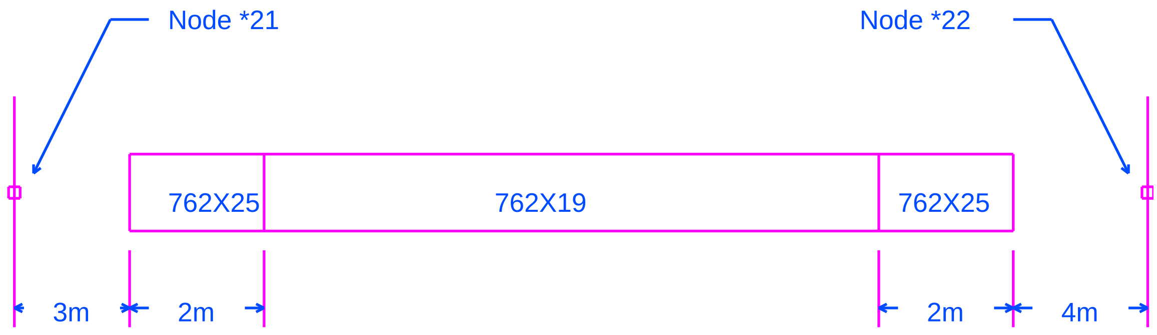
BEAM, ELE_NAME(1), ~CLASS(1), –OPT(1), *NODE(1), ... *NODE(m), \
ELE_NAME(2), ~CLASS(2), –OPT(2), NODE(n), ...

and in addition to the available options discussed above we have:

–**CMFAC**, CMY, CMZ
–**KFAC**, KY, KZ
–**BLY**, BUKLENY
–**BLZ**, BUKLENZ
–**BLENG**, BELE_NAME
–**CFB**, CFSPAC
–**HAS_P–DELTA**, YES/NO

BEAM defines a string of beams. Here, the ~CLASS(i) defines the section attributes, and *NODE(1), *NODE(2), ...*NODE(n) is the list of connected nodes where, *NODE(1) connects to *NODE(2), *NODE(2) connects to *NODE(3), and so on. The locations of the command marked by –OPT(i) are positions where one may place as many element options as desired, and may be omitted if defaults are suitable for the beam in question. The beams from *NODE(1) to *NODE(m) have the properties defined by ~CLASS(1) and –OPT1, and the beams from *NODE(m) to *NODE(n) have properties defined by ~CLASS(2) and –OPT(2). In other words, a beam in the string has the properties defined by the last ~CLASS and/or –OPT data issued before the beam was defined. Here, ELE_NAME(i) is a name which can be assigned to the element. If it is omitted, MOSES will assign a name. The ~CLASS is a name for a set of element attributes which define the "sectional properties". An example of a beam definition accompanied by a sketch is shown in Figure 8.

The options specified above are used when checking codes. To alter the CM values for an element, one can use the **–CMFAC** option. Here, CMY is the factor for bending about the Y axis and CMZ is the factor for bending about the Z axis. The K factors and the "buckling lengths" are multiplied to obtain the effective length of the element about each axis. By default, the buckling lengths are taken to be the element length for beams and the square root of the area for generalized plates. In particular, to define the effective length multipliers for beams, one uses the option **–KFAC**. Here, KZ is used for bending about the Z axis, while KY is used for bending about the Y axis. If this option is not used, both factors will be



```
&DEFAULT  -FY 290
~TUB762 TUBE 762 25 -LEN 2
~TUB762 TUBE 762 19
~TUB762 TUBE 762 25 -LEN 2
BEAM ~TUB762 -LOA  3000  -LOB  -4000  *21 *22
```

DESCRIPTION OF BEAM

FIGURE 8

set to 1.

Element lengths for beams include the effect of any offsets, invoked by either the **–GO** and **–LO** options on the BEAM card, or the **–OFFSET** option on the **INMODEL** command.

The **–BLENG** or **–BLY** and **–BLZ** options can be used to alter the buckling lengths about each axis. Here, the dimensions are feet or meters. The **–BLENG**, BELE_NAME construct offers a way to "bind" the buckling length of an element to the load state of another element. Here, BELE_NAME is the name of the "brace element". If this option is exercised, then the KL factors of the basic element will be those input if the brace element is in compression. If the brace element is in tension, then the factor for out of plane will be the same as for inplane. When this option is used, the "compression" lengths will be used in any report or computation outside of the Structural Post–Processing Menu. An alternative way of defining this type of dependence is provided in the **MEDIT** Menu with the **XBRACE** command.

The **–CFB** option defines compression flange brace spacing, CFSPAC, (inches or mm). If this option is omitted, then the value will be taken to be the element length. Again, this is the length after accounting for any offset. The **–HAS_P–DELTA** option tells MOSES that this beam has nonlinear effects taken into account. If YES/NO is **YES** then no interaction effects will be taken into account when the codes are checked. If YES/NO is **NO**, or no option is used, then standard interaction formulae will be used in the code checking. For a tube, they are generally computed based on the load path and the formulae specified. For other type sections, they are input.

As an example of defining beams, consider:

```
~STF  TUBE  30  .75  –FY  42 –LEN 3
~STF  TUBE  30  .675  –FY  42 –LEN 0.
~STF  TUBE  30  .75  –FY  42 –LEN 3
BEAM ~STF –RELA MY –GO1 10 12 *AAA1010 *BBB1010
```

Here we have defined a single beam connecting the nodes *AAA1010 and *BBB1010, with properties defined by the class name ~STF. The beam consists of 3 segments, hence the three STF commands. The first and third segments are tubular sections with 30 inch OD and 0.75 inch wall thickness, while the middle section has the same OD, but only a 0.675 inch thickness. The first and third segments have a length of 3 ft. and the length of the middle segment is to be computed by the program. All segments have a yield stress of 42 ksi.

As a second example, consider:

```
~LONG, W18X40
BEAM, ~LONG, *A, *B
```

This defines 1 beam as a wide flange beam (18 x 40). This is a prismatic beam, as only one ~LONG command is given.

The example below uses a string of nodes for a beam description, which can sometimes be quite useful:

```
~TUBE TUBE 42  1.625
BEAM ~TUBE –GO1 10 10 10 *1 *2 *3  \
          –GO2 12 12 12 *4 *5
```

In the above example, there are five nodes used, and four individual beam elements created. Each of the beam elements has a name assigned to it by MOSES. The options apply in the order the beam is defined. For instance, *1 is the A end of the beam joining nodes *1 and *2, and the –GO1 is a global offset at *1. *2 is the A end of the beam joining nodes *2 and *3, with *3 as the B end of the beam. This sequence continues for as many nodes as there are to define the entire series of beams. For the last beam of this series, *5 is the B end of the beam joining node *4 and *5, and the –GO2 option is a global offset at *5. The backslash (\) used above is for a continuation of the same command on the next line.

In many instances, one needs to define beam load attributes other than those defined via the element description. Three mechanisms for this are provided: two commands and the **–T_PRESSURE** option on the **&ENV** command and a **T_PRESSURE** command. This option allows one to specify the temperature, internal pressure and the density of the contents of a beam. The first of the command provided is:

#ELAT, OBJECT, WTPFT, DPFT, BOD, DOD, AMOD, WINOD – OPTIONS

and the available options are:

–**TOTAL**
–**CATEGORY**, CAT_NAME
–**A**, XA
–**B**, XB
–**LENGTH**, LEN

This command is used to define additional intrinsic load attributes for a logical beam, and OBJECT is the name of the object to which they apply.

If OBJECT is two node names (they may include wild characters, but must begin with an *), the attributes will apply to all beams between those two nodes. Remember, there may be more than one beam defined between the same two nodes. Also, these two selectors can be used to define load attributes on a "logical" beam. For example, suppose that there is a beam between *1 and *2 and a beam between *2 and *3. If these two beams are colinear, then #ELAT *1 *3 will apply the load to both beams. If OBJECT is an attribute class name (begins with a ∼), then the attributes will apply to all elements which belong to classes which match OBJECT. If OBJECT begins with neither an * nor a ∼, then the attribute will be applied to all members whose names match OBJECT.

The data defines the attributes which will be added to the element. Here, WTPFT is a weight per foot (bforce/blength), DPFT is a buoyancy per foot (bforce/blength), BOD is the diameter (inches or mm) which will be used for buoyancy, DOD is the diameter (inches or mm) which will be used for viscous drag, AMOD is the diameter (inches or mm) which will be used for added mass, and WINOD is the diameter (inches or mm) which will be used for wind. With this command, one defines a line buoyancy with DPFT as well as a diameter which will be used for computing buoyancy. Both can be used at the same time. The added mass and viscous drag computed will be based on Morison's Equation. The –**TOTAL** option denotes the fact that the values input for WTPFT and DPFT are total quantities and should be divided by the length to obtain the distributed properties.

By default, loads produced from the properties specified with #ELAT are assigned to the default Extra Category. This can be changed with the –**CATEGORY** option. Also, the load attribute defined here will, by default, act over the entire length of the elements selected. If one wishes, he may alter this distribution with the options: –**A**, –**B**, and –**LENGTH**. Here, the attributes are defined over a segment of the original beam beginning XA (feet or meters) from the "A" end of the beam, and extending to XB (feet or meters) from the "B" end, and the length over which it is applied is LEN. The defaults are that both XA and XB are zero and LEN is the length of the beam. Thus, for a load over the entire beam, none of the option list is needed. Notice that the "B" end of an attribute can be defined by either –XB or –LENGTH. *Both should not be used on the same command.*

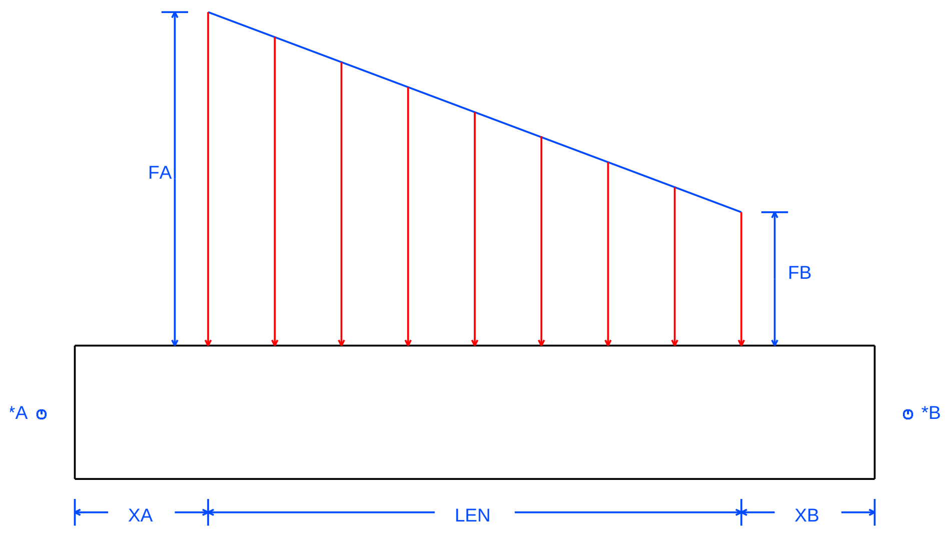
The second command for beam attributes defines an applied load which belongs to user defined load set,

> #LSET, OBJECT, VALA(1), ... VALA(6), –OPTIONS

and the available options are:

> –**VB**, VALB(1), ... VALB(6)
> –**LOCAL**
> –**TOTAL**
> –**A**, XA
> –**B**, XB
> –**LENGTH**, LEN

Here, VALA(i), shown as FA, are the values of the attribute XA (feet or meters) from the "A" end of the beam, and VALB(i), shown as FB, are the values at XB (feet or meters) from the "B" end. If –**VB** is not specified, then the values at XB will be taken to be the same as those at XA. The units for VAL(i) are in bforce and bforce–blength. To define an attribute in beam coordinates, one simply adds the –**LOCAL** option. To input a concentrated attribute, one should specify LEN to be zero. Figure 9 illustrates how the options are used. The –**TOTAL** command denotes the fact that the values input for VALA(i) and VALB(i) are total quantities and should be divided by the length to obtain the distributed properties. The options –**A**, –**B**, and –**LENGTH** operate in the same manner as the corresponding option for #ELAT and were discussed above.



USER DEFINED LOAD SET
FIGURE 9

XII.N.4 Generalized Plates

A generalized plate is a structural element defined with a set of nodes along the exterior, and perhaps, a set of points defining a free edge of the generalized plate. An edge of the generalized plate is the line segment between two adajent nodes on the plate, and a "face" is all edges which are on a line.

A generalized plate is defined with the command:

> **PLATE**, ELE_NAME, ~CLASS, –OPTIONS, *NODE(1), *NODE(2), \
> *NODE(3), *NODE(4), ... *NODE(n)
> –FSOPT *FP(1), ...

This command defines a generalized plate element of class ~CLASS, connected to nodes named *NODE(1) through *NODE(n). The nodes must be specified so as to go around the element in one direction. Here, ELE_NAME is a name which can be assigned to the element, omitted, MOSES will assign a name. The –OPTIONS have been discussed above. They may be omitted, or valid options may be inserted directly at this location within the command. The class name, ~CLASS, is used to specify the generalized plate's attributes in exactly the same manner as for the beam element.

A generalized plate is composed of subelements (triangular plates) contained within the defined perimeter, and can be any *convex* shape and some special concave ones. The number of subelements depends on the shape of the perimeter and the number of nodes on each face. For three nodes you get no subelements and with four nodes you get four subelements. If the generalized plate has four faces, the subelements are generated as if it were composed of strips of quadrilateral elements. If not, the subelements are generated along rays (line segments from the average of the coordinates of the specified nodes) to the nodes on the boundary. Subelements will be generated so that the maximum distance of a side is less than or equal to the maximum distance between any two specified nodes. The internal nodes will have names which begin with **IN. Figure 10 shows four typical generalized plates

and Figure 11 shows the corresponding subelements.

The –FSOPT options can be used to define concave generalized plates. Here *FP(i) are points which define a "free edge". The points define the geometry of the edge. The nodes on the other edges will determine where the nodes which will define the subelements will be placed.

If –FSOPT is **–HOLE** then the free edge defines a hole contained within the generalized plate and *FP(i) are points defining the geometry of the hole. Here *FP(1) should be closest to NODE(1) and the order of the *FP(i) should be the same order as NODE(i). The actual nodes on the free edge are at the intersection

Plate Exterior Nodes

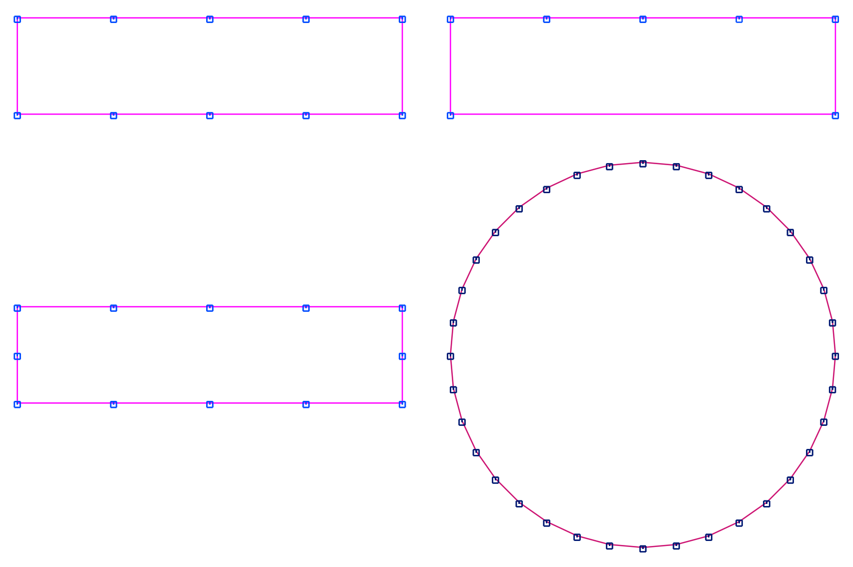


FIGURE 10

Plate Subelements

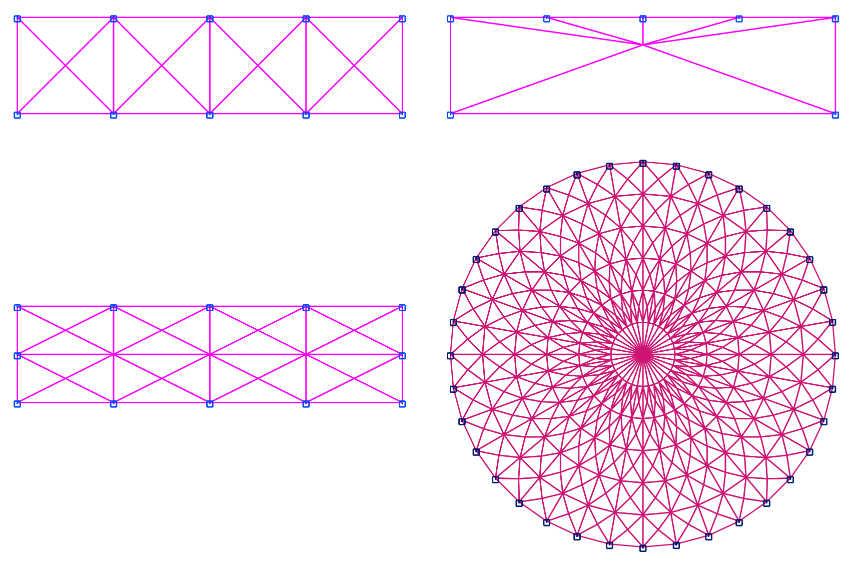


FIGURE 11

of the ray from the center of the generalized plate with the free edge. Each of these defining rays passing through an exterior node. Thus, the more nodes on the exterior, the better representation one gets of the hole. The nodes created around the free edge will be named beginning with **CN. A word of *caution* here. The weight and centroid of this generalized plate are computed properly, but any other load is computed ignoring the hole.

If –FSOPT is **–FREE_EDGE** then the free edge defines a concave portion of the exterior of the element exterior. Here, the points *FP(2) and *FP(2) are actually NODES defining a face "opposite" the free surface; i.e. here we actually have a shape which can be mapped into a lattice with the face between the nodes *FP(1) and *FP(2) opposite the free edge. Also, the NODES(i) and FP(3), ... FP(n) should be in order around the element. Another word of *caution* since the generalized plate may not be convex, it also may not be star shaped with respect to its centroid. If it is not any load other than weight will be in error. Figure 12 shows a generalized plate with a free edge defined with 17 points beginning with the characters *CP,

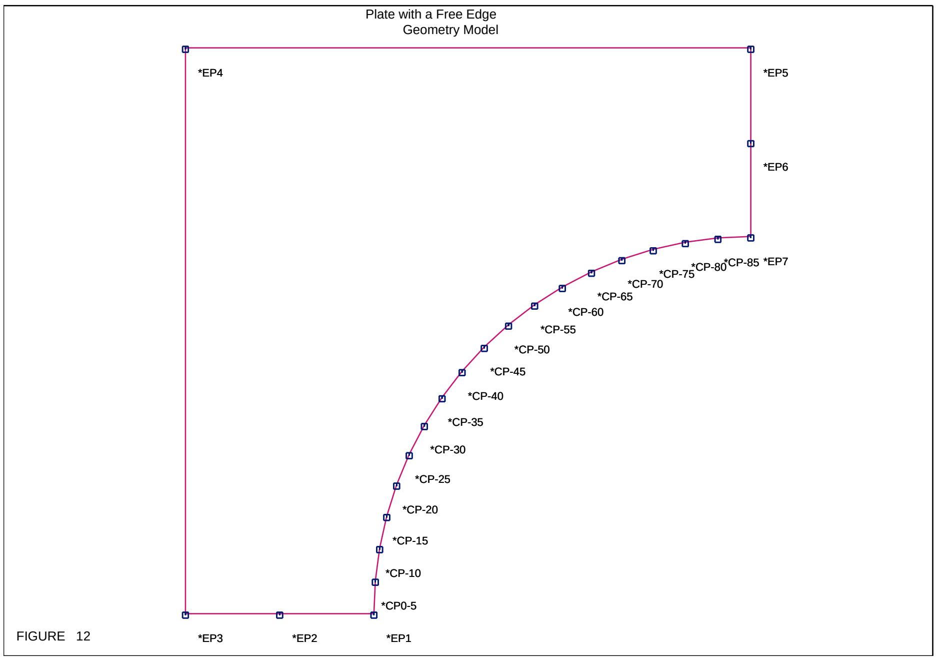


FIGURE 12

and Figure 13 shows the corresponding nodes and subelements.

Finally, and Figure 14 shows what happens if the number of nodes opposite the free edge is increased.

The only non–intrinsic load attribute one can specify for a generalized plate is a temperature specified with the **–T_PRESSURE** option on the **&ENV** command

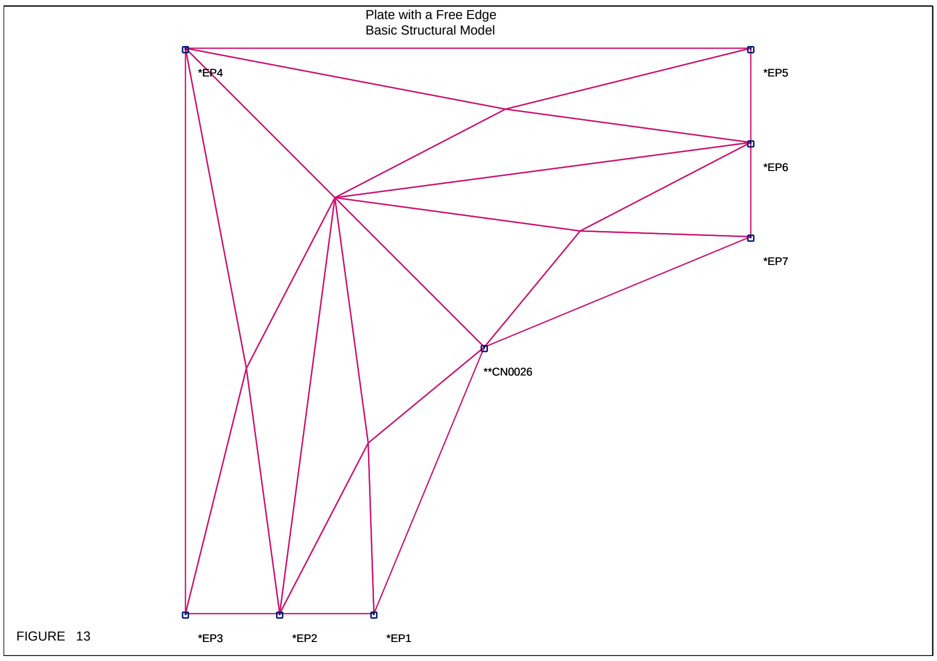


FIGURE 13

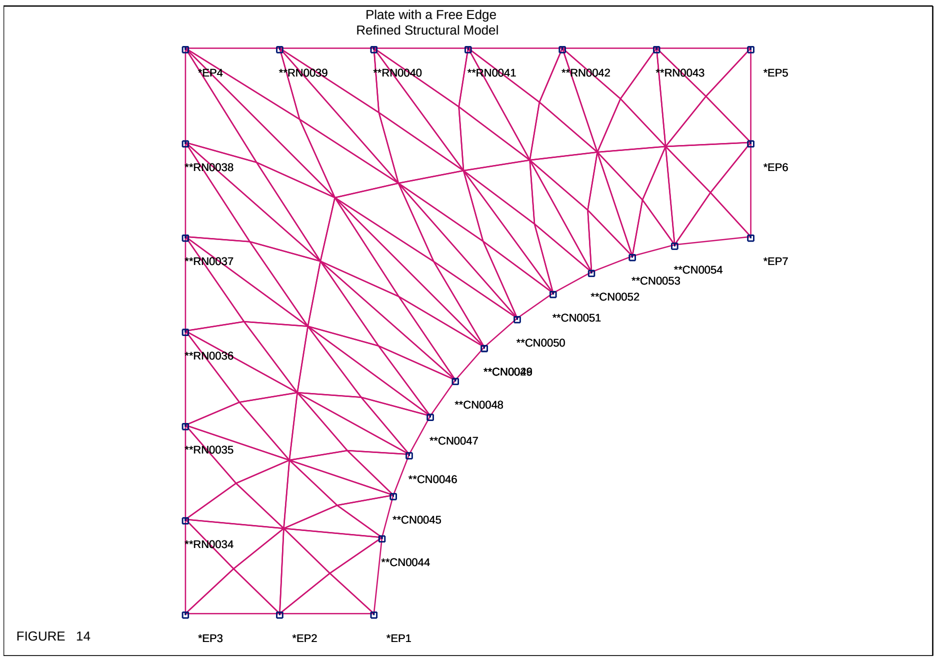


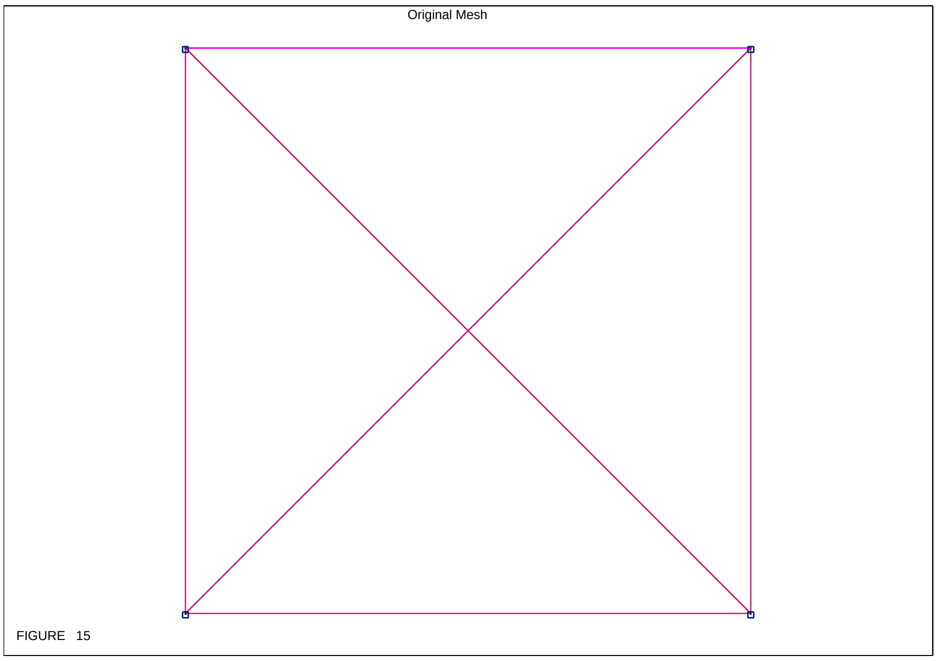
FIGURE 14

and a **T_PRESSURE** command.

One can refine the edges of generalized plate elements with the command:

REFINE, MAX_DIST, WHAT, SEL(1), SEL(2), ...

This command causes the edges of all generalized plates selected by the selectors specified to be refined so that the maximum length of an edge is MAX_DIST (feet or meters). Here WHAT can be either: **EDGE**, **ELEMENT**, or **BOX**. If one specifies EDGE, the following SEL(i) should be in pairs of selectors; e.g. a pair *Q@ *R@ will select any edge that has two nodes which match the two selectors. If one specified ELEMENT, the following selectors select elements based on element name; e.g. QP@ refines all edges of elements whose name matches QP@. Finally, a WHAT of BOX refines edges which are totally with a box. Here, the values of SEL(i) should be X_MIN, X_MAX, Y_MIN, Y_MAX, Z_MIN, and ZMAX. These are distances (feet or meters) specified in the part system. Notice, all of these things can be used together. For example, one can have a BOX inside of a BOX with the interior one having a smaller MAX_DIST. Then the interior one will have the interior MAX_DIST and the exterior one the larger distance. The nodes added as a result of a refinement have names which begin with **RN. The REFINE command can be issued both when performing an INMODEL and in the MEDIT Menu. Figures 15 through 18 show a square generalized plate with none, one, two, and four edges refined.



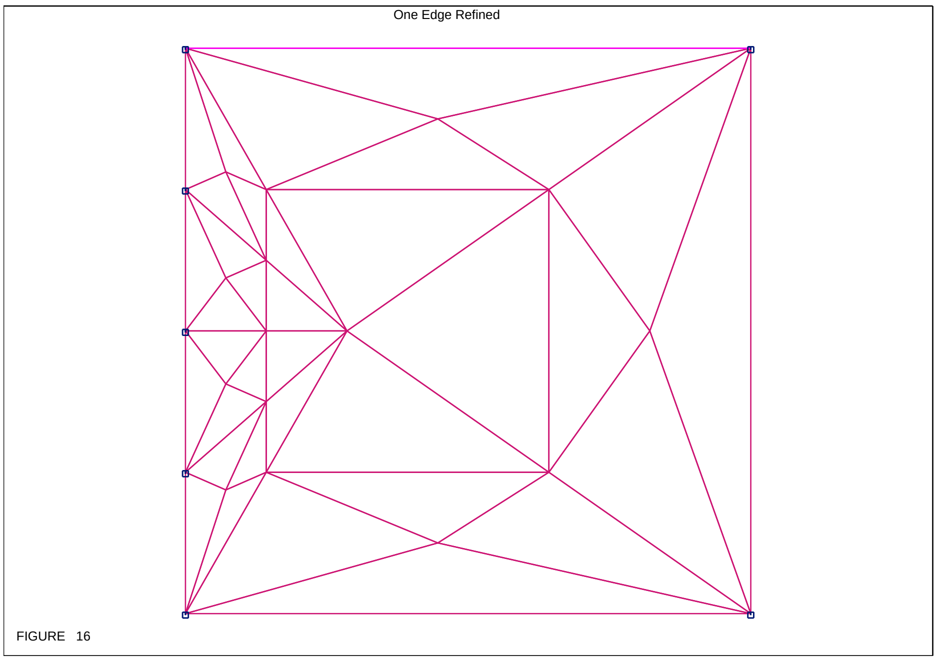


FIGURE 16

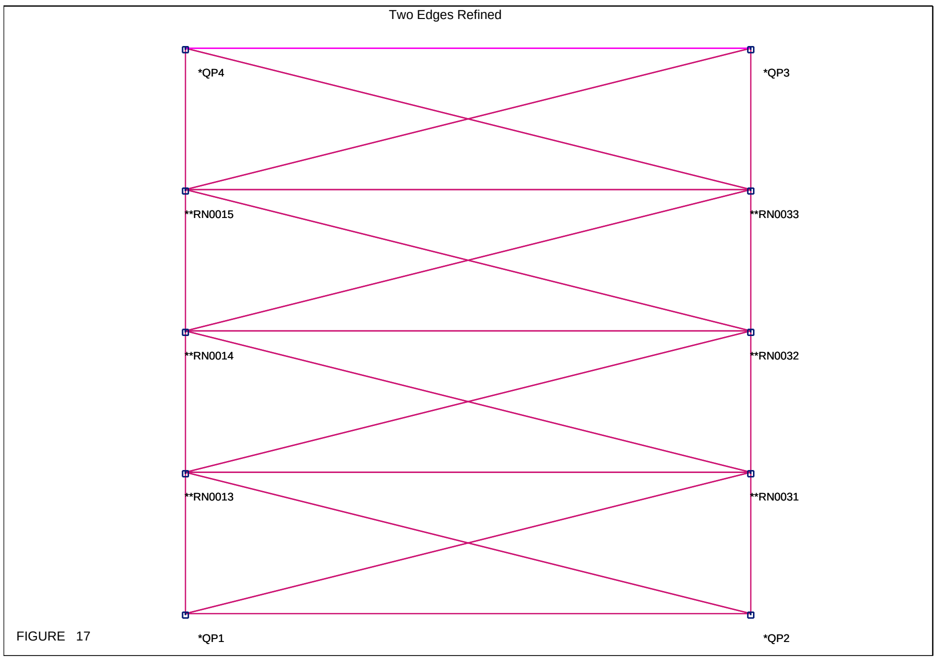


FIGURE 17

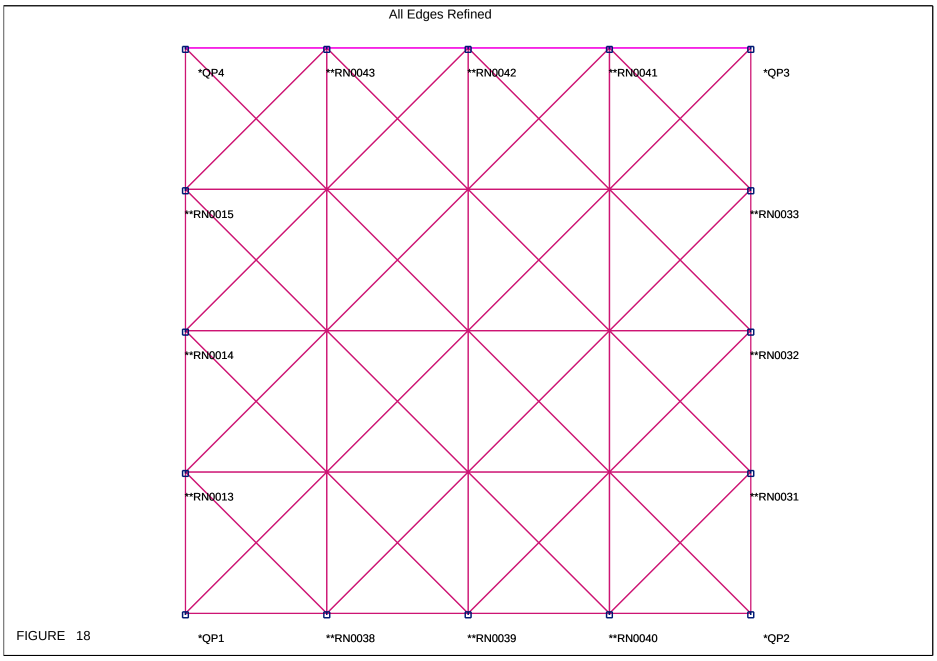

All Edges Refined
*QP4
**RN0043
**RN0042
**RN0041
*QP3
**RN0015
**RN0033
**RN0014
**RN0032
**RN0013
**RN0031
*QP1
**RN0038
**RN0039
**RN0040
*QP2

FIGURE 18

XII.N.5 Connecting Parts

Connecting parts is a rather delicate proposition. *Normally*, the only elements which do not belong to a part are connectors. Connectors, however, are not designed to carry moments and are not as general as beams. If, however, beams are allowed to span parts, their element system is not properly defined. Two alternatives are provided to circumvent these difficulties. The first is that nodes can have an alias. In other words, a node which has an alias will have the same deflection as the alternative name of the node. Defining a node alias is accomplished via:

ALIAS, *SLAVE(1), *MASTER(1), ... *SLAVE(n), *MASTER(n)

When this command is issued, an element will be generated between *SLAVE(i) and *MASTER(i). No checking will be done as to the congruence of the locations of the two nodes, and these elements will not be used during simulations. Instead, their only purpose is to insure that when a stress analysis is performed, the two nodes will have identical deflections. One will normally use this technique to connect two nodes in different parts which have the same location in space.

The second method of connecting parts is with special connectors called part connectors. These elements belong to special parts which have a part type of **PCONNECT**. These elements may be defined by standard **BEAM** and **PLATE** commands, or they may be defined in the **MEDIT** Menu by commands which are similar to the ones used in defining tiedowns. The additional commands are:

PCONNECT, ~CLASS, *JN, :SEL(2)
PCONNECT, DX, DY, DZ, ~CLASS, :SEL(1), :SEL(2)

The first format generates part connectors at a single node in one part to several nodes in the other part, using beams of section ~CLASS. Thus ~CLASS is the name of the class property defining the section properties of the part connector member, *JN is the name of the node to be tied down, and :SEL(2) is a selector for the nodes to which *JN is connected.

The second format generates part connectors at several nodes, where the orientation of each part connector is constant. In effect, DX, DY, and DZ define the far end of a beam element at each node which matches the selector :SEL(1). This far node is then connected by a rigid link to the nearest node matching :SEL(2). Here, ~CLASS is as before, and DX, DY, DZ is the distance measured from the node to the body attachment point, in the second part system (feet or meters).

A special method for connecting two parts for the transportation of a structure on a vessel is provided by MOSES. To utilize this method, one *must* have a model which consists of a single body with a body type of **VESSEL**, and there *must* be a part named **JACKET** with a part type of **JACKET**. This allows the part system of the jacket to be different from the vessel, and MOSES will automatically rotate the jacket so that the part systems are the same. Here, one *must* also establish connections between the parts so that an analysis can be carried out. The connections are of two basic types: launchways, which are continuous beams fastened to the vessel and upon which the jacket rests, and tiedowns, which fix the jacket to the vessel.

To use this feature, one should issue the command:

> **TRANS_CON**, –**LOCJ**, XO, YO, ZO, \
> –**JLLEGS**, *JS(1), ... *JS(n), \
> –**JLLEGP**, *JP(1), ... *JP(n), \
> –**LWAYP**, X1, ZNA, L, ~CLASS, :BPSEL \
> –**LWAYS**, X1, ZNA, L, ~CLASS, :BSSEL

The launchways are generated by the program as two continuous beams. Nodes on these beams are automatically generated by MOSES to match up with the nodes on the jacket launch legs and with suitable nodes on the vessel. The jacket launch leg nodes are defined with the –**JLLEGS** and –**JLLEGP** options where *JP(i) are nodes on the port launch leg, and *JS(i) are nodes on the starboard launch leg. In both cases, the nodes must be ordered so that a given node is further aft than all of the nodes which precede it.

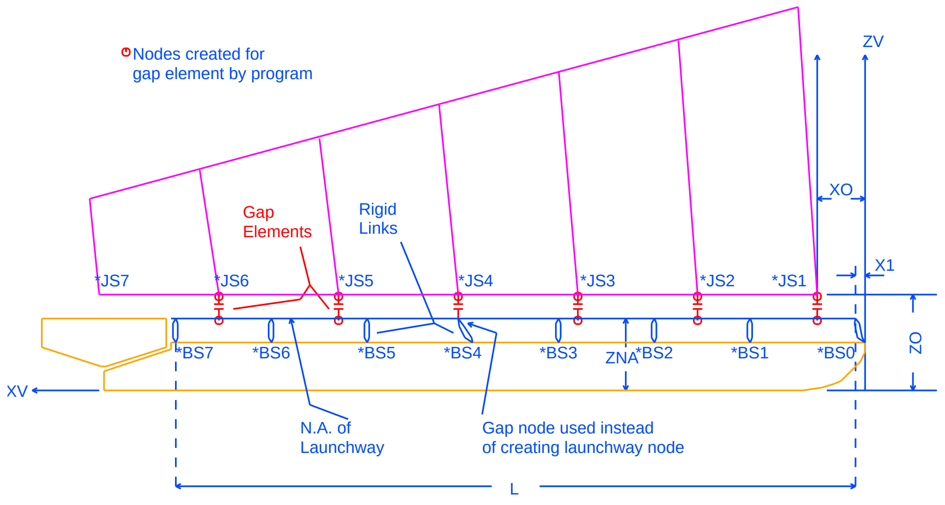
The jacket location on the vessel is given on the –**LOCJ** option where: XO, YO, ZO are distances (feet or meters), in the body system, from the vessel origin to the point midway between *JP(1) and *JS(1). With the jacket located on the vessel, the two launchways are defined using the –**LWAYP** and –**LWAYS** options. Here, X1 is the vessel coordinate of the beginning of the launchway (feet or meters), ZNA is the vessel coordinate of the launchway neutral axis (feet or meters), L is the length of the launchway (feet or meters), ~CLASS is the name of the class property defining the section properties of the launchway, :BPSEL is a selector for the nodes to be rigidly connected to the port side (LWAYP) launchway, and :BSSEL is a selector for the nodes to be rigidly connected to the starboard side (LWAYS) launchway. Figure 19 shows the effect of the above command.

To connect parts elastically, one employs tiedowns in one of two formats:

> **TDOWN**, ~CLASS, *JN, :SEL(2)
> **TDOWN**, DX, DY, DZ, ~CLASS, :SEL(1), :SEL(2)

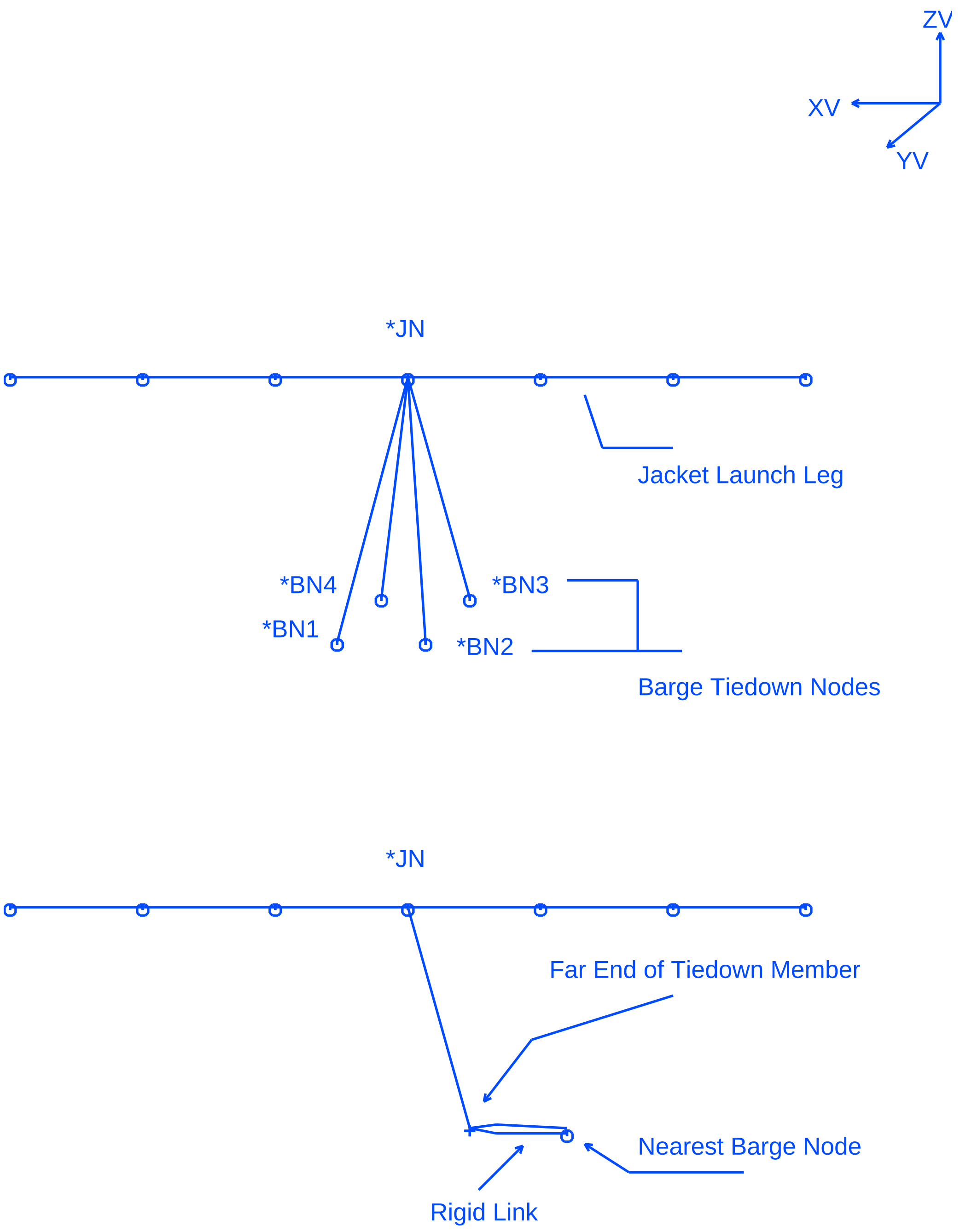
The first format generates tiedowns at a single jacket node which are connected using beams of section ~CLASS to several vessel nodes. Thus ~CLASS is the name of the class property defining the section properties of the tiedown member, *JN is the name of the jacket node to be tied down, and :SEL2 is a selector for the vessel nodes to which *JN is connected.

The second format generates tiedowns at several jacket nodes, where the orien-



JACKET/BARGE/LAUNCHWAY GEOMETRY
FIGURE 19

tation of each tiedown is constant. In effect, DX, DY, and DZ define the far end of a beam element at each jacket node which matches the selector :SEL(1). This far node is then connected by a rigid link to the nearest node matching :SEL(2). Here, ~CLASS is as before, and DX, DY, DZ is the distance measured from the jacket node to the body attachment point in the vessel system (feet or meters). The tiedown geometry is illustrated in Figure 20.



TIEDOWN GEOMETRY

FIGURE 20

XII.N.6 Structural Post–Processing Elements

MOSES has a concept of "pseudo elements" which can be used for structural post–processing. The pseudo elements are called sp_beams or sp_plates, and they are defined in the same manner as beams or plates except that the command BEAM is replaced with SP_BEAM or PLATE is replaced with SP_PLATE; i.e.

SP_BEAM, ELE_NAME(1), ~CLASS(1), –OPT(1), *NODE(1), ... *NODE(m), \
ELE_NAME(2), ~CLASS(2), –OPT(2), *NODE(n), ...
SP_PLATE, ELE_NAME, ~CLASS, –OPTIONS *NODE(1), ... *NODE(m)

and there is an additional option *which must be used*:

–ELEMENTS, :ELSEL(1), ... :ELSEL(n)

These commands create a pseudo elements which have all of the properties defined with their class and their geometry, but gets their load information from the elements specified via the **–ELEMENTS** option. Here :ELSEL(i) are selectors for the elements which will be considered to "be a part of" the SP_BEAM or SP_PLATE.

As mentioned above, these pseudo elements add *nothing* to the model – no stiffness, no weight, nothing. They are simply a useful way to look at parts of the model "in the large" for structural post–processing. In particular, suppose that one had a plate model of a semi–submersible. After a structural analysis, he has stresses in the plates, but that is it. Now, to really analyze this situation, one needs to check global buckling of the columns, etc. By creating SP_BEAMS of the various pieces, this can be easily accomplished. For purposes of structural post–processing pseudo elements are treated the same as structural elements. Thus, one can also find bending moments and shears in them, stresses in them, etc.

XII.O Load Groups

The load group is a generalization of a nodal load. Here, a collection of load attributes is associated together as a group. For simulations, the load due to each of the attributes is computed and applied to the proper body. A load group also has a list of points associated with it. This list is given a name (normally the name of the load group) and is called a load map. When a stress analysis is performed, the total load on the group will be distributed to the nodes associated with the points by a least squares technique. Thus, if there is only one node associated with the group, it acts as a nodal load.

There are two primary differences between a load group and a nodal load. First, load groups allow for the association of not only loads, but also load attributes, with nodes. Secondly, the load attributes of a load group can be distributed to more than one node. In addition to the obvious advantages of using attributes instead of loads, the load group allows one to define gross properties to entire bodies when he is not interested in the structural details.

The load map may be either defined with a set of selectors as described below, or MOSES will define it for you. In particular, if one has not defined a map explicitly, the loads will be mapped to the points where load attributes have been defined. Whenever a body or part is defined, MOSES will automatically define a load group with the same name as the body or part.

Load groups are defined in much the same manner as bodies and parts. In other words, one issues:

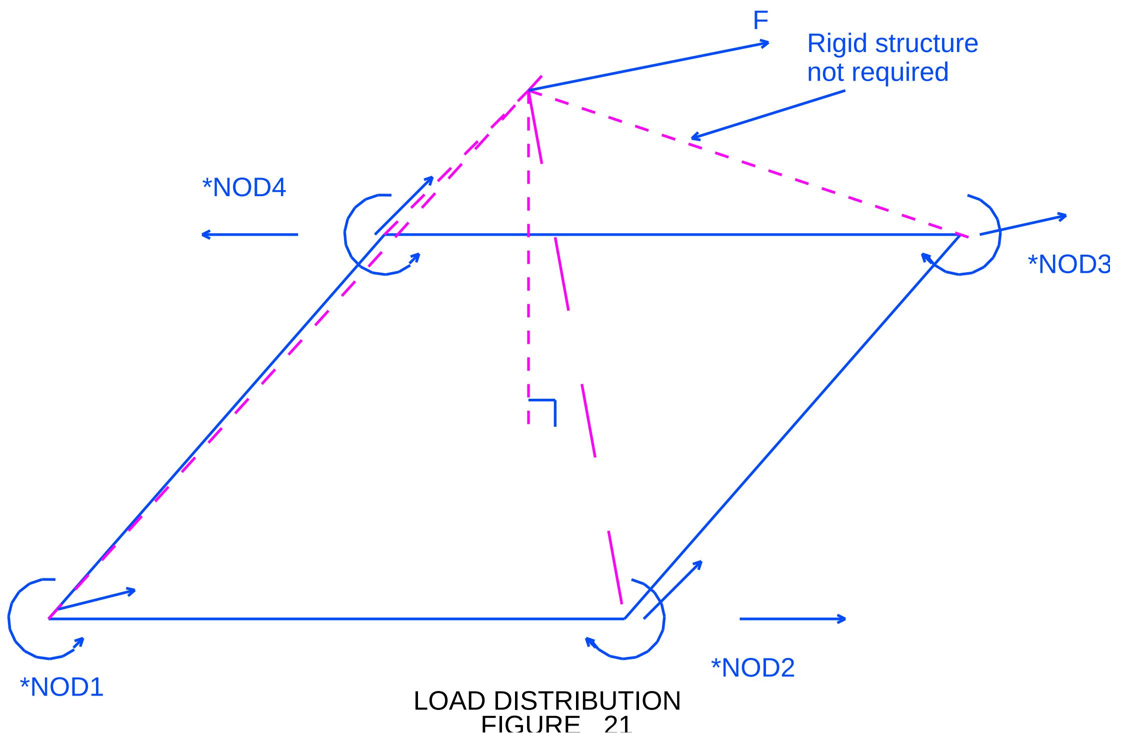
&DESCRIBE LOAD_GROUP, LG_NAME, :NODE_SEL(1), .. :NODE_SEL(n), –OPTIONS

where the options are:

–**DAMP_FUSE**, YES/NO
–**AMASS_FUSE**, YES/NO

and all load group attributes which follow belong to the specified load group. Here, LG_NAME is the name which one wishes to give to the following load group, and :NODE_SEL(i) are a set of selectors defining the points to which the loads will be mapped. The force from a load group is automatically distributed to the specified nodes as if the nodes were connected to the point of application by a rigid structure. Note, however, the rigid structure is *not* required, since the force is distributed by a least squares technique, as shown in Figure 21. The options –**DAMP_FUSE** and –**AMASS_FUSE** control the use of matrices defined with **#AMASS** and **#DRAG** commands in the frequency domain. By default, they are *not* used. If YES/NO is set to **YES**, the matrices will be included.

One can obtain the values of some of the present attributes of a load group with



the string function:

&LOADG(:LSEL, –OPTION)

the valid options are: **–PERCENT**, **–WEIGHT**, **–RADII**, and **–CG**. The string returned here is the name of the load group followed by the applicable values for each load group which matches the selector :LSEL. This value is multiplied by the load group multiplier and margin. The CG and radii of gyration are returned in the part system.

One can define mass, added mass, viscous damping, linear damping, wind, and buoyancy attributes for a load group. Here, the linear damping is not strictly associated with wave radiation, since it is a constant. Both the added mass and viscous drag are computed according to Morison's Equation. Most of the load attributes defined are applied at a point previously defined by the user. If the point reference is omitted, then the loads will be applied at the part origin.

Many option apply to all load attributes described, In particular, all load group attributes are assigned a Category of the default "extra" category by default. If one wishes to alter this, he may use the **–CATEGORY** option where indicated. Thus, one can have load attributes associated with different categories within the same load group. Also, MOSES associates a multiplier with the load group as a whole. This multiplier may be changed as can the multipliers for categories with the &APPLY command. This allows one to be able to "turn off" an entire load group as well as alter the force computations for some of the attributes.

Each of the attributes allows one to define a NUMBER, defined by an option, –**NUM_APPLIED**, which multiplies the results of an attribute before it is applied. In other words, the force, mass, drag, and added mass are first computed based on the properties defined. Then, each of these quantities are multiplied by the number specified before they are applied to the body.

The forces which will be applied are determined by a set of multipliers defined by the –**WIND**, –**DRAG**, and –**AMASS** options. These multipliers are similar to shape coefficients in that a force is computed and then the multiplier is applied. If any of these options are omitted, the corresponding multiplier is set to one. The forces which result from these commands are computed according to Morison's Equation. Wind only acts on the area defined by **#AREA** if its center is above the water surface. Similarly, water loads are only attracted when the center of area is below the water surface. –**WAVE_PM** is used to define WAVMUL which is a multiplier for wave particle velocity and acceleration. If WAVMUL is greater than zero, it is used to factor the wave particle velocity and acceleration before it is added to the other velocities and accelerations to compute a force. The default value of WAVMUL is zero, in which case wave velocity and acceleration will not be considered for the load attribute.

The buoyancy due to these commands is defined by the –**BUOY_THICK** option. Here, BTHICK is a thickness (inches or mm) which, when multiplied by the submerged area, will yield the buoyancy force.

Weight can also be defined with the other commands by using one or both of the options –**TOT_WEIGHT** or –**MULT_WEIGHT**. Here, WT is a weight, in bforce, which will be applied at the point of application. With the –**MULT_WEIGHT** option, WMULT is a weight/area (bforce/blength**2) which is multiplied by the area defined by either a **#PLATE** or a **#AREA** command to yield a weight applied at the centroid. Both options can be used on the same command.

For any of the load attributes that follow, the options:

–**TEXTURE**, NAME_TEX, X_SCALE, Y_SCALE
–**COLOR**, COLOR(1), FRAC(1), ... COLOR(n), FRAC(n)

can be used to define the color and texture of the attribute. These will be used when one asks for a picture with –COLOR MODELED. Here, NAME_COL is any color which has been previously defined. See the section on Colors for a discussion on defining colors. The NAME_TEX value for –**TEXTURE** is the name of a file in either /X/data/textures or /X/data/local/textures (here MOSES is store in /X). The X_SCALE and Y_SCALE are scale factors which will be applied to the texture. The NAME_TEX of **NONE** will yield a null default texture.

Perhaps the most popular of the load group class commands is the one which associates a weight with the group. The form of this command is:

#WEIGHT, *PT, WT, RX, RY, RZ, –OPTIONS

and the available options are:

–**LDIST**, X1, X2
–**NUM_APPLIED**, NUMBER
–**CATEGORY**, CAT_NAME

This command instructs MOSES that a weight of WT bforce is attached to the part at the location specified by the point *PT. This weight has radii of gyration RX, RY, and RZ (feet or meters) about the point *PT. The **–LDIST** option defines the longitudinal distance over which the weight will be applied when computing traditional longitudinal strength. Here, X1 and X2 are the beginning and ending longitudinal coordinates (feet or meters) of the interval over which the weight will be applied.

To define an applied force within a load group, one uses the command:

#LSET, *PT, FX, FY, FZ, MX, MY, MZ

This command defines an applied generalized force with the magnitude of the components given by FX, FY, etc. (bforce and bforce–blength). The force is applied at the point defined by *PT, and is a member of the load set #LSET. As with all user defined load sets, one must "activate" the load set with an **&APPLY** command before it will actually be applied.

To instruct MOSES to build an added mass matrix for the load group, one should issue:

#AMASS, *PT, DISP, CX, CY, CZ, RX, RY, RZ –OPTIONS

and the available options are:

–**CATEGORY**, CAT_NAME

Here, the added mass will be DISP/G, where G is the gravitational constant. CX, CY and CZ are the added mass coefficients, and RX, RY, and RZ are added radii of gyration, taken about the point specified by *PT.

Similarly, one can define a constant, linear drag matrix with the command:

#DRAG, *PT, DISP, D(X), D(Y), D(Z), R(X), R(Y), R(Z) –OPTIONS

and the available options are:

–**CATEGORY**, CAT_NAME

Here, the force that will be produced is:

F(X) = DISP/G * D(X) * V(X)
F(Y) = DISP/G * D(Y) * V(Y)
F(Z) = DISP/G * D(Z) * V(Z)
M(X) = DISP/G * R(X)**2 * OMEG(X)
M(Y) = DISP/G * R(Y)**2 * OMEG(Y)
M(Z) = DISP/G * R(Z)**2 * OMEG(Z)

Where G is the gravitational constant, DISP, D(i) and R(i) are the quantities specified, V(i) is a velocity at the point *PT, and OMEG(i) is an angular velocity in rad/sec.

Point buoyancies can be associated with the load group by:

#BUOY, *PT, DISP, –OPTIONS

and the available options are:

–**CATEGORY**, CAT_NAME
–**NUM_APPLIED**, NUMBER
–**TOT_WEIGHT**, WT

Here, DISP (bforce) is the magnitude of the buoyancy which will be applied. Again, this is applied at the location specified by the point *PT, and is applied only when this point is below the water surface. Weight can also be specified with this command by use of the **–TOT_WEIGHT** option. Here, WT is a weight (bforce) which will be applied at the same point as the buoyancy.

The following two commands, **#AREA** and **#PLATE**, are somewhat similar in that they both define an area which attracts water and wind forces. The format of the first of these is:

#AREA, *PT, AX, AY, AZ, –OPTIONS

and the available options are:

–**CATEGORY**, CAT_NAME
–**NUM_APPLIED**, NUMBER
–**WIND**, WINMUL
–**DRAG**, DRGMUL

–**AMASS**, AMSMUL
–**WAVE_PM**, WAVMUL
–**BUOY_THICK**, BTHICK
–**TOT_WEIGHT**, WT
–**MULT_WEIGHT**, WMULT

Here, AX, AY, and AZ (ft**2 or m**2) define the components of an area concentrated at the point specified by *PT.

In contrast to **#AREA**, a **#PLATE** command defines a distributed area, and load attraction does not depend on the location of the center of area. The form of this command is:

#PLATE, *PNT(1), *PNT(2), ..., –OPTIONS

and the available options are:

–**CATEGORY**, CAT_NAME
–**NUM_APPLIED**, NUMBER
–**WIND**, WINMUL
–**DRAG**, DRGMUL
–**AMASS**, AMSMUL
–**WAVE_PM**, WAVMUL
–**BUOY_THICK**, BTHICK
–**TOT_WEIGHT**, WT
–**MULT_WEIGHT**, WMULT

Here, one specifies up to four vertices of a polygon by points *PNT(i). These vertices *must* be input in the order of one tracing the outline of the plate. MOSES will compute the portions which are submerged and those which are above the water surface, applying the proper forces in each regime. The level of detail used in the force calculation is defined by the –**MAXAREA** and –**MAXREFINE** and the method of computing drag are defined with options of the **&PARAMETER** command. The options for **#PLATE** function in the same manner as they do for **#AREA**, with one exception. When the submerged portion of a plate is computed, an aspect ratio is also computed. The added mass for the plate is that computed for a rectangular plate according to DNV Classification Notes 30.5.

Occasionally, it is convenient to describe a load attribute as a tubular, but without actually adding a tubular element to the model. This can be performed with the command:

#TUBE, OD, T, *PT1, *PT2 –OPTIONS

where the available options are:

–**CATEGORY**, CAT_NAME

–**NUM_APPLIED**, NUMBER
–**WIND**, WINMUL
–**DRAG**, DRGMUL
–**AMASS**, AMSMUL
–**WAVE_PM**, WAVMUL
–**TOT_WEIGHT**, WT
–**MULT_WEIGHT**, WMULT
–**BUOY_DIA**, BOD

Here, OD (inches or mm) is the diameter of the tube used to calculate environmental forces and T (inches or mm) it the thickness. The tube is positioned between the points *PT1 and *PT2. The first eight options behave the same as for the #AREA command, and the **–BUOY_DIA** option defines the diameter used to calculate buoyant forces for the attribute. The weight computed for this element is as described about for #AREA *except* that another weight is computed using the thickness, the OD, the current default density, and the length of the tube.

Wind and current forces can be input to MOSES and then associated with a load group. This is achieved with the command:

#TABLE, T_NAME, *PNT, –OPTIONS

Here *PNT is the point of application of the force and the available options are:

–**WAVE_PM**, WAVMUL
–**CATEGORY**, CAT_NAME

When computing current forces, the wave particle velocity is computed a the minimum depth of the location of *PNT or the water surface. The wind speed is always computed at standard anemometer height so the wind is always allied regardless of the *PNT location.

For this command, T_NAME is the name of a table of user defined force coefficients for wind and current. This table is defined using the **&DATA** menu:

&DATA A_TABLE, T_NAME, FLAG

where T_NAME is the name of the force table. In this menu, the commands available are:

ANGLE, ANG1
WIND_ARE, WAX, WAY, WAZ, WAMX, WAMY, WAMZ
CURR_ARE, CAX, CAY, CAZ, CAMX, CAMY, CAMZ

The **ANGLE**, **WIND_ARE**, and **CURR_ARE** commands are repeated for each angle, for up to 36 angles. If FLAG is specified as **REFLECT**, the angles specified should range from 0 to 180 degrees. If this is not specified, the angles specified should be from 0 to 360 degrees. These angles are either wind angles relative to the body system, or relative body/current velocity angles. The body velocity used is that computed at *PNT, specified in body coordinates as X, Y and Z. The force acting at *PNT is calculated by multiplying the specified coefficients by the square of the relative velocity. These force coefficients also create damping in the frequency domain. For the **WIND_ARE** and **CURR_ARE** commands, X, Y and Z are force coefficients, while MX, MY and MZ are moment coefficients. The prefix WA is for wind area, and CA is for current area. Remember to exit this menu using **END_&DATA**.

If one wishes, he can associate a Tanaka Damping load attribute with a load group. This is accomplished with the command:

#TANAKA WETSUF –OPTIONS

and the available options are:

–**ROLL**, SECTION, FRACTION, BLOCK, DEPKEEL, KG, BEAM, BILRAD
–**PITCH**, SECTION, FRACTION, BLOCK, DEPKEEL, KG, LENGTH, BILRAD
–**PERIOD**, T(1), T(2), ..., T(n)
–**ANGLE**, AN(1), ..., AN(n)

The two options –**ROLL** and –**PITCH** are instructions to compute a set of Tanaka data and add it to what exists, and one can have up to 20 different occurrences of these options. The degree of freedom which will be effected by the damping is defined by the option name. Here, WETSURF is the total wetted surface of the body (ft**2 or m**2), and one can think of each pair of –**ROLL** and –**PITCH** options as defining the damping for a part of the body which has a fraction, FRACTION, of the wetted surface. All of the other variables for the option apply to the piece being defined. SECTION defines the type of section, and must be either **BOW**, **MIDBODY** or **STERN**. BLOCK is the block coefficient, DEPKEEL is the distance from the waterline to the keel (feet or meters), KG is the vertical center of gravity above the keel (feet or meters), BEAM is the breadth of the body (feet or meters), LENGTH is the length of the body (feet or meters), and BILRAD is the bilge radius (feet or meters). No pitch damping is produced by default. If you define *no* additional data on the #TANAKA command, you will be put into a new submenu menu.

Once in the submenu, the equivalent linear damping coefficients are defined with the commands

R_TANAKA PER, VDM(1), ..., VDM(N)

or

P_TANAKA PER, VDM(1), ..., VDM(N)

Here, PER is one of the periods specified with the **–PERIOD** option, and VDM(i) are the coefficients corresponding to the angles specified with **–ANGLE**. *Notice* that VDM(1) corresponds to ANGL(1), VDM(2) with ANGL(2), etc. The units for VDM are bforce–sec–feet or meters. **R_TANAKA** refers to roll damping coefficients, while **P_TANAKA** is for pitch. This menu is "exactly" the same as the one you have before with I_TANAKA.

The final command which can be used for defining load group attributes is different from the others. Here, instead of defining specific load attributes, one defines the wind and current load attributes of an entire vessel. The particular form of this command is:

#TANKER, SIZE, TLEN, TDEP, TBEAM, AEX, AEY, LCP, –OPTIONS

and the available options are:

–**CATEGORY**, CAT_NAME
–**CBOW**
–**YAW_FACTOR**, YF
–**WAVE_PM**, WAVMUL

This command causes wind and current forces to be computed for a tanker of the specified size according to the data presented in *Prediction of Wind and Current Loads on VLCCs* by the Oil Companies International Forum (OCIMF). *NOTICE, these curves assume that the vessel is defined so that the X axis goes from bow to stern and that the keel of the vessel is at the origin of the part.* These forces will be added to any other forces computed for the load group. The option **–CBOW** should be used if the tanker has a bulbous bow, and the option **–YAW_FACTOR** controls the yaw viscous damping during a time domain simulation. The OCIMF formulae were derived for computing the wind and current force on a stationary tanker. Instead of simply using the current velocity in the formulae, MOSES uses the relative current tanker velocity, so that viscous damping is obtained as well as applied force. This approach works quite nicely for the basic forces and yaw moment, but produces zero moment for a tanker which has zero velocity about the center of pressure regardless of the yaw angular velocity. To overcome this problem, an additional term has been added which depends on the lateral coefficient and the yaw angular velocity. The **–YAW_FACTOR** option specifies a multiplier for this extra term. A value of zero for YF means that the term will not be used while a value of one means that it will be used with no modification.

Here, SIZE is the size of the tanker in thousands of deadweight tons (e.g. a SIZE of 100 would denote a 100,000 DWT tanker). If all of the other dimensions are omitted, they will be interpolated from an internal database. The other data are: length, depth, beam, extra frontal area, extra lateral area, and distance from the origin to the longitudinal center of the tanker. Here the units are feet or meters for distances, and ft**2 or m**2 for areas. MOSES uses the major dimensions to compute the wind and current areas, and the extra areas are for wind only. There is an internal database of default extra areas for AEX and AEY. These values are replaced when a non–zero value is used for AEX or AEY. In particular, the lateral wind area is given by TDEP minus the draft times TLEN plus AEY. Likewise the frontal wind area is (TDEP – DRAFT) * TBEAM + AEX.

In some cases, using **#TANKER** may cause forces in a direction opposite from one expects. This is a function of the OCIMF data, and not a problem with the software. There are lift forces involved with the ship shaped hull forms this data represents, sometimes causing this behavior. The OCIMF results are based on extensive wind tunnel and tank tests on typical tankers. We have incorporated digitized versions of these curves into MOSES, therefore, what is derived is essentially what was measured. The question now becomes why are these forces in a direction opposite to what one may expect – the answer is probably lift (but this depends on your expectations!), and is what permits us to sail and fly. The hull will behave like an aerofoil where the flow does not separate immediately at the bow (and particularly if it is cylindrical as far as wind loads are concerned). As a consequence, the longitudinal force components may be "negative", i.e. up current or upwind, for some directions. Rest assured, however, the resultant force is always downweather so we can't sail or fly for nothing – there is a net drag!

XII.P Compartments

Compartments are entities used for three purposes:

- To define hydrostatic, wind, and current forces on complicated shapes,
- To define the sea/body interaction forces on things for which Morison's Equation is not applicable, and
- To define ballast in the interior of a body.

In essence, compartments are used to model the exterior of bodies and the interior compartmentation of bodies. Thus, we have two types of compartments: interior and exterior.

The distinction between interior and exterior compartments is made when the compartment is named. Whenever a body is defined, MOSES will automatically define a compartment with the same name as the body. This compartment is an exterior compartment. Any other compartment defined whose name is not also the name of some *part* is an interior compartment. Only exterior compartments can attract hydrodynamic loads, and only interior compartments can be flooded or ballasted.

Compartments are a collection of two basic entities: pieces and tube tanks. A tube tank is used to model the interior of a tube, and pieces are used to model everything else. In essence, a piece is a "ship like" part of the compartment. When a piece is part of an exterior compartment it creates buoyancy, drag force, wind force, and behaves according to either Strip Theory, Three Dimensional Diffraction Theory, or attracts no hydrodynamic load at all. When part of an interior compartment, it can be damaged (flooded) or ballasted and defines part of the inertia of the system. The way to define pieces and tube tanks will be discussed below.

Compartments are defined in a fashion similar to bodies and parts. In other words, the data for a compartment follows an **&DESCRIBE** command naming the compartment:

&DESCRIBE COMPARTMENT, CNAME –OPTIONS

Here, CNAME is the name of the compartment and the options define additional attributes for *interior* compartments and will be discussed later.

Tube tanks are a computationally efficient way to define interior compartments. The only restriction is that the physical shape defined by a tube tank must be a circular cylinder. A given compartment can have as many tube tanks as desired, and each one is defined by:

TUBTANK, DIA, PNT(1), PNT(2)

Here, DIA is the diameter (inches or mm) of the tank and PNT(1) and PNT(2) are the names of two points defining the ends of the tank. Alternately, the coordinates of the ends (feet or meters) can be input instead of point names. If one uses coordinates, points will be defined with these coordinates. These tanks can be mixed with other pieces for an internal compartment. *If one attempts to define a tube tank for an exterior compartment, an error will result.*

XII.P.1 Pieces

For more complicated geometries, compartments are defined with pieces. In essence, a piece is a *closed* surface in space. Each piece is, in turn, defined by a set of connected areas called "panels". Both interior and exterior compartments can be composed of as many pieces as required. There are two basic ways to define a piece: by defining each panel, or using a more powerful language to generate the panels. In the next section, an even more powerful method of generating panels by Boolean operations on existing pieces will be discussed.

In addition to geometry, pieces have several additional attributes, all of which are defined as options on the command which defines the piece. Here, we will first discuss defining pieces via panels and then turn to the easy way. A panel description of a piece begins with the command:

&DESCRIBE PIECE, PIECE_NAME, –OPTIONS

and the available options are:

–**PERMEABILITY**, PERM
–**OBSTACLE**,
–**DIFTYPE**, TYPE
–**CS_WIND**, CSW_X, CSW_Y, CSW_Z
–**CS_CURRENT**, CSC_X, CSC_Y, CSC_Z
–**CS_VELOCITY**, CS_VELOCITY_NAME
–**DD_MULT**, DDR(1), MULT(1), ..., DDR(n), MULT(n)
–**AMASS**, AMA_MULT, AM_CURVE
–**TANAKA**, TANAKA_FACTOR
–**ROLL_DAMPING**, ROLL_DAMP_FACTOR
–**COLOR**, COLOR(1), FRAC(1), ... COLOR(n), FRAC(n)
–**TEXTURE**, NAME_TEX, X_SCALE, Y_SCALE

Here, PIECE_NAME is the name of the piece. If this is omitted, a name will be automatically generated by MOSES. The **–PERMEABILITY** option can be used to alter the buoyancy of a piece. Here, PERM is the permeability of the piece, and it should be positive for a buoyant piece and negative for a hole or a flooded compartment. A value of one corresponds to 100 percent of the piece's volume. The default value for permeability is 1 for an exterior piece, and –1 for an interior compartment.

The **–OBSTACLE** option provides a method for defining an obstacle in the water, with which a floating body will have hydrodynamic interaction. The obstacle belongs to the body defined with the last **&DESCRIBE BODY** command, but does not move in a simulation. Multiple obstacles may be defined, so that several situations may be analyzed. For instance, three obstacles may be defined and arranged to simulate a floating body inside a dry dock. A single obstacle can be arranged underneath a body to represent a sloping sea floor. It is important to note that an obstacle will move only when the body it is attached to moves as a result of a positioning command, such as **&INSTATE –LOCATE**.

The remaining options of the **PIECE** command are applicable *only* to exterior compartments and are used to define how forces on the piece will be computed. The **–DIFTYPE** option defines which hydrodynamic theory will be used for the piece. Here, TYPE must be either **3DDIF** if one wishes to use Three Dimensional Diffraction Theory, or **NONE** if the piece is to be ignored in computing hydrodynamic properties. If one wants to use Strip Theory, he must define the piece with a **PGEN** command which is discussed later. The **–CS_WIND** and **–CS_CURRENT** options define how the panels of the piece attract wind and current loads.

In general, for each panel in a piece, MOSES will split it into a portion above the water and a portion below. The wind force on the out of water portion and the current force on the below water portion are computed as if the area was a **#PLATE**. Here, however, the components of the velocity are multiplied by the values of CSC_X, CSC_Y, and CSC_Z respectively. In addition, the current force is multiplied by a multiplier, MULT, that depends on the water depth to draft ratio, DDR. This value is obtained by interpolation from the table defined with the **–DD_MULT** option. The default values for the CSs and the multiplier table are set with the **&DEFAULT** command.

Several other things need to be said here. First, the "current" force actually depends on both the current velocity and the velocity of the panel, so using the word current is really incorrect. Second, the results are quite dependent upon the setting of **–AF_ENVIRONMENT** option of **&PARAMETER**. If the option is set to **NO** and the above is used for "round" pieces, the results may be surprising. The force produced for a rectangular box is exactly the same as one would get by using a pair of **#PLATE**s. For a circular cylinder, however, one gets a force of .785 (pi/4) times that of a square box. This is simply a result of the recipe used to compute the force. Thus, to get "the correct" force on a cylinder, one should specify:

–CS_WIND .5/.785 .5/.785 1
–CS_CURRENT .5/.785 .5/.785 1

If the setting is **YES** then this does not apply.

The option **–CS_VELOCITY** can be used to change the first component of the CS_CURRENT values. Here, CS_VELOCITY_NAME is the name of a "curv" what has been defined with the command CS_VELOCITY with the command &DATA CURVE CS_VELOCITY. If this option is used, then the relative speed at each panel will be used to interpolate a drag coefficient.

The **–AMASS** option defines a multiplier for an estimate of the added mass of a piece. If the name AM_CURVE is omitted, then MOSES will estimate the added mass in these three directions as a function of submergence. These estimate are based on the projection of the "box" containing the submerged piece onto the body coordinates for surge and sway and on the waterplane for heave. Here you will get three polygons, one for each coordinate direction. The added mass in each direction is obtained as if this polygon is an isolated plate. If AMA_MULT is zero, no added mass will be accumulated for this piece. The slopes of the heave curve are also used to compute slam loads in heave. If you wish, you can define a "curve" with a type of AM_PRESSURE with the command &DATA CURVE AM_PRESSURE. Notice that this is an excellent way to estimate added massed for isolated plates and for things which will change their submerged shape considerably, but it should be turned off for pieces which are used for diffraction or strip theory hydrodynamics.

The option **–TANAKA** defines a multiplier for the damping due to "eddy making". In MOSES, we refer to this as "Tanaka" damping after the person who first formulated the results for eddy making damping. In MOSES, we use the formulation outlined in a paper by Schmidke in *The Transactions of the Society of Navel Architects and Marine Engineers* (1978). The default value here is defined by an option of the same name for the **&DEFAULT** command. The option **–ROLL_DAMPING** defines a quadratic damping factor, ROLL_DAMP_FACTOR in (sec**2 – feet or meters – bforce)/rad**2. When defined, it applies a roll moment given by:

roll_moment = ROLL_DAMP_FACTOR * omega**2

and omega is the roll angular velocity. *CAUTION* if either of these option are used in conjunction with **–CS_CURRENT** then roll will be over damped.

The **–COLOR** and **–TEXTURE** options can be used to define the color and texture of the piece. These will be used when one asks for a picture with –COLOR MODELED. Here, NAME_COL is any color which has been previously defined. See the section on Colors for a discussion on defining colors. The NAME_TEX value for **–TEXTURE** is the name of a file in either /X/data/textures or /X/data/local/textures (here MOSES is store in /X). The X_SCALE and Y_SCALE are scale factors which will be applied to the texture. The NAME_TEX of **NONE** will yield a null default texture.

The string function:

&PIECE(ACTION, NAME)

can be used to return information about a piece. Here, NAME is the piece about which information is desired and ACTION must be either **CURRENT**, **PANELS**, **DIFTYPE**, **PERMEABILITY**, **CS_WIND**, **CS_CURR**, **AMASS**, **LOCATION**, or **OBSTACLE**, and the type CURRENT returns the current piece name, PANELS returns the names of the panels in the specified piece, and all of the others return the data defined with the options of the same name.

After the piece has been defined, its geometry is defined by a set of convex polygons called panels. Each panel is defined by points at its corners with the command:

PANEL, PAN_NAME, PTNAM(1), ..., PTNAM(n)

Here, PAN_NAME is the optional name assigned to the panel, and PTNAM(i) are names of its vertices. If PAN_NAME is omitted, a panel name will be assigned internally. A panel can contain from three to fifty vertices. The order of the definition of the vertices should be *clockwise* when the panel is viewed from *outside* of the body. In other words, the normal to the surface when defined by the right hand rule should point into the body. One can alter this convention by using the command,

REVERSE, –OPTIONS

and the available options are:

–YES
–NO

The counterclockwise order for the vertices should be used following a **REVERSE –YES** command. If one wishes to change back to the basic convention, he should then issue **REVERSE –NO**.

The string function:

&PANEL(ACTION, NAME)

can be used to return information about a panel. Here, NAME is the panel about which information is desired and ACTION must be **AREA**, **NORMAL**, **E_COORDINATES**, **PERIMETER**, or **G_CENTROID**. These return the obvious things with E_COORDINATES returning the part coordinates of the ends and G_CENTROID returning the global centroid.

While the above method is certainly general in that all possible surfaces can be defined as accurately as desired, actually defining a mesh with panels can become quite tiresome. As an alternative, MOSES provides a menu for "generating" a piece. This menu begins with the command:

PGEN, PIECE_NAME, –OPTIONS

and the available options are:

–PERMEABILITY, PERM

–**OBSTACLE**,
–**DIFTYPE**, TYPE
–**CS_WIND**, CSW_X, CSW_Y, CSW_Z
–**CS_CURRENT**, CSC_X, CSC_Y, CSC_Z
–**CS_VELOCITY**, CS_VELOCITY_NAME
–**DD_MULT**, DDR(1), MULT(1), ..., DDR(n), MULT(n)
–**AMASS**, AMA_MULT, AM_CURVE
–**TANAKA**, TANAKA_FACTOR
–**ROLL_DAMPING**, ROLL_DAMP_FACTOR
–**COLOR**, COLOR(1), FRAC(1), ... COLOR(n), FRAC(n)
–**TEXTURE**, NAME_TEX, X_SCALE, Y_SCALE
–**STBD**
–**PORT**
–**BOTH**
–**TOL_OFF**, TOL
–**LOCATION**, X, Y, Z, ROLL, PITCH, YAW

and ends with an **END_PGEN** command. Here, most of the options are exactly the same as those for the **PIECE** command, and the last five define how to interpret the plane data which will follow.

In this menu, the closed surface of the piece will be defined as a sequence of polygons called planes. In contrast to panels, these planes do not define the surface directly, but define "cuts" through it. To proceed, consider two coordinate systems, a local system used in this menu only, and the part system. The local X axis lies along the length of the piece, positive aft. The local Z axis is perpendicular to the local X axis, and is positive "up". The local y axis is defined by the local X and Z axes, and the right–hand rule (positive to starboard.) The surface of the piece is now defined by its intersection with a set of planes. These planes have constant local X values and the intersection is defined by a set of local points, (Yi,Zi). The options define how the points will be interpreted. The default is to define only the positive Y portion of the plane, allowing MOSES to automatically produce the negative half. If one specifies **–STBD**, no reflection will be performed. If **–PORT** is specified, then the plane is reflected, and the positive portion is deleted. Finally, if **–BOTH** is specified, one must define both halves of the plane. In this case, one first defines the positive portion and proceeds around the contour to define the negative portion. In all cases, one starts at the bottom of the station (minimum z and zero y) and proceeds around counter clockwise (point 2 is has non decreasing z from point 1). If using Strip Theory, the user should set TYPE to be **STRIP** and the **–STBD**, **–PORT** and **–BOTH** options cannot be used. The **–CS** options work exactly the same as described above.

The **–TOL_OFF** option allows one to omit offsets which lie on a line from the model. This is often desirable since MOSES gives the correct answers for planes of any size. Here, TOL is the cosine in the angles minus 1 which will be considered to be colinear.

The **–LOCATION** option allows one to place and orient the generated piece with respect to the part system. The position of the piece is defined by X, Y, and Z (feet or meters). These are the part coordinates of the origin of the local coordinates of the piece. The orientation of the piece is defined by the three angles: ROLL, PITCH, and YAW. These are defined as follows: Suppose that the piece is given successive rotations about the local Z axis, then about the new local Y axis, and finally, about the new local X axis, so that the piece, after the three rotations is in its proper orientation in space. These three angles can be thought of as a yaw, followed by a pitch, followed by a roll to move the piece from when it is aligned with the global system to its required position in space.

In this menu, the actual definition of the surface is accomplished by a set of commands:

PLANE, X(1), X(2), ..., X(n), –OPTIONS

and the available options are:

–RECTANGULAR, ZBOT, ZTOP, BEAM, NB, NS, NT
–CARTESIAN, Y(1), Z(1), Y(2), Z(2), ..., Y(n), Z(n)
–CIRCULAR, Y, Z, R, THETA, DTH, NP
–E_CIRCULAR, Y, Z, R, THETA, DTH, NP

where: X(i) is a local X coordinate of the plane (feet or meters), and the offsets of the section are defined by the options. The X values for a piece must be defined in non–decreasing order; i.e. one should not have

PLANE 10 –10 ...

If the section is rectangular, it can be completely defined by one **–RECTANGULAR** option. Here: ZBOT is the local Z coordinate of the bottom of the section (feet or meters), ZTOP is the local Z coordinate of the top of the section (feet or meters), and BEAM is the width of the section (full width, feet or meters). Normally, the section is defined with 4 points, but this can be changed with the values NB, NS, NT. NB defines the number of points along the bottom, NS the side, and NT the top. For other types of sections, the **–CARTESIAN**, **–CIRCULAR**, or **–E_CIRCULAR** options are used. With the **–CARTESIAN** option Y(i), Z(i) are local Y and Z coordinates of the ith offset point (feet or meters). The **–CIRCULAR** option allows one to input data in polar coordinates. Here: Y and Z are local coordinates of the center of the circle (feet or meters), R is the radius of the circle (feet or meters), THETA is the angle of the first point (degrees) measured from the negative local Z axis positive toward Y, DTH is the increment in angle (degrees), and NP is the number of points to be generated. It is important to notice that any number of **–CIRCULAR** and **–CARTESIAN** options can be used to define the section. The **–E_CIRCULAR** option is the same as the **–CIRCULAR** option except that the radius which is specified is *not* the one which will be used. Instead, a new radius will be computed so that the area of the polygon defined by the new radius is the same as that of the circular sector.

The number and size of the panels determine the accuracy of the results. Of course, the computational effort required is quite sensitive to the number of panels and so one is constantly seeking a good compromise between fidelity and efficiency. Complicating this, however, is the fact that there are several different things for which accuracy is important. The algorithm which computes hydrostatic results from the mesh is "exact" in that the results obtained from a given mesh are the same as those obtained from a refined mesh for the same body. Thus, from a pure hydrostatic point of view, the most desirable mesh is the coarsest one which properly defines the surface of the body.

In a structural analysis, however, a different problem arises. By default, the structural loads on a panel are mapped to the nodes closest to the vertices of the panel. If you have a large number of nodes in comparison to the number of panels, then the default will not yield a good load distribution. If the default scheme is to work properly, then there should be a reasonable relationship between structural nodes and corners of panels. In other words, if one has a single panel for the side shell of a barge, he should *at least* have points on the panel at each longitudinal location where he has structural nodes.

To eliminate many of these difficulties, MOSES has a command available in the **MEDIT** menu:

M_PAN_FIX, TOL_OP, TOL_B, –OPTIONS

where the available options are:

–BOUNDARY
–NODES, YES/NO
–COMPART, :CMP_SEL
–PIECE, :PIECE_SEL
–PANEL, :PAN_SEL
–POINTS, :PNT_SEL
–X, X_BOX1, X_BOX2
–Y, Y_BOX1, Y_BOX2
–Z, Z_BOX1, Z_BOX2
–DIRECTION, DIR

The purpose of this command is to change the nodes to which the loads will be mapped for selected panels. The details of the options will be addressed later. The two numbers TOL_OP and TOL_B control the remapping. To see how these work, consider the situation discussed above. In particular, suppose that you have large panels and small plates, so that there are several plates which are inside a panel. By default, there will be no load applied to the nodes at the vertices of the interior generalized plates. **M_PAN_FIX** will solve this problem by adding points on the boundary and interior of the panel to the list of points which will receive the load. Here, TOL_OP (feet or meters) defines an out of plane tolerance. Only points within TOL_OP distance along the normal of the plane will be considered. Likewise, TOL_B (feet or meters) defines a boundary tolerance, and only points with a distance of TOL_B of the boundary will be considered to belong to the boundary. These two tolerances define what is meant by ”on the boundary” and ”inside the panel”. If a point is within TOL_B of the boundary in the plane of the panel and within TOL_OP of the plane of the panel, then it is ”on the boundary” of the panel. If a point is ”inside” the panel (a distance greater than TOL_B away from the boundary and inside the panel) and within TOL_OP of the plane of the panel, then it is inside the panel. If the **–BOUNDARY** option is used, all points inside the panel and on the boundary of the panel will be used for the map. Alternatively, without the **–BOUNDARY** option, only points inside the panel will be used. To ”fix” a mesh that is being mapped to a generalized plate model, this is normally all that is necessary.

The panels are available for selection are defined by selectors by matching the selectors defined with the **–NODES**, **–COMPART**, **–PIECE**, **–PANEL**, and **–POINTS** options. In other words a panel is available for selection if its name matches :PAN_SEL, and it is in a piece selected by :PIECE_SEL which is in a compartment selected by :CMP_SEL. Finally, for a panel to be selected, it must be inside a box defined in the part system with the options **–X**, **–Y**, and **–Z**. If the average of the vertices of the panel lie inside this box, the panel will be remapped. The default for each selector is @, and the default box is all of space. Thus, if none of these options is used, all panels will be remapped. Normally, only *nodes* will be considered for the mapping. If, however, one wants to map to points which are not nodes, then they should use the **–NODES** option with a value of **NO** for YES/NO. With the options **–X**, **–Y**, or **–Z** the first dimension defines the beginning of the box (minimum dimension) and the second one defines the end of the box (maximum dimension). For example, if one issues:

```
M_PAN_FIX –X 0 50
```

then only panels which have the average of their vertices in the x direction between 0 and 50 will be remapped.

The **–POINTS** option defines the set of points which will be used for the mapping. By default, :PNT_SEL is set to @, so that all points will be used.

The last option **–DIRECTION** completely alters the behavior of the remapping. This option is useful for remapping meshes that are connected to beam models. In this case, the concepts of closeness used above are changed to measures along a single axis. Here, DIR which must be chosen from **X**, **Y**, or **Z** defines the "direction" of the mapping. Here, the remapping works as above, except that now "in the panel" means that a point has its "selected" coordinate between the extremes of the "selected" coordinates of the panel. By "selected" coordinate, we mean either the x, y, or z coordinate depending on the value of DIR. Thus, suppose that we specified X for DIR. A point will be used for the mapping if its x coordinate is between X_MIN − TOL_B and X_MAX + TOL_B where X_MIN is the minimum value of the x coordinates of the vertices of the panel and X_MAX is the maximum of the x coordinates of the vertices. Additionally, the y and z coordinates must be between the minimum values of the vertices minus TOL_OP and the maximum values plus TOL_OP.

Finally, the **MEDIT** command:

> **MAP**, MAPNAM :MAP_SEL(1), :MAP_SEL(2), ...

can be used to completely define the map for a panel. Here, MAPNAM is the name of the panel map, and :MAP_SEL(i) are a set of selectors which define the map. One can look at the maps with the commands **&STATUS MAP** and **&STATUS N_MAP**.

If Strip Theory is used to compute hydrodynamics, the planes defined in a **PGEN** piece are used for the Two Dimensional hydrodynamic computation. Thus, for these pieces, one needs a reasonable longitudinal distribution and several (over 10) planes, even if two would suffice hydrostatically.

In contrast to most programs, the mesh used for Three Dimensional Diffraction is the same as that used for hydrostatics. Whenever one asks for hydrodynamic pressures to be computed, MOSES will convert this mesh to one which describes the submerged portion of the body in the initial configuration. Thus, one can analyze different drafts and trims without changing the mesh definition. MOSES has an automatic refinement capability controlled by the two options **–M_DISTANCE** and **–M_WLFRACTION** of the **&PARAMETER** command. When the mesh is being used for hydrodynamic computations, MOSES will automatically refine the mesh so that no side of a panel is greater than the distances given by these two options. This allows one to build a quite crude mesh and yet obtain solutions as accurate as desired by simply changing one of the two parameters. The same caveat about structural loads also applies here. The basic panels should have corners in a reasonable relationship to the structural nodes where one will ultimately apply the loads.

To examine how a mesh is refined by the two mesh refinement options, the user can use **&PICTURE –TYPEL MESH –DETAIL**. This will produce the mesh refined according to the **&PARAMETER** settings in effect when the command was issued. As mentioned above, the accuracy depends on the panel size, particularly at higher frequencies. The execution time, however, increases as the square of the number of panels for small meshes and as the cube for large ones. Thus, one should strive to have the minimum number of panels which are still small enough to produce an accurate solution.

Even though the same set of panels are used for both hydrostatics and hydrodynamics, the computations are quite different. For hydrostatics, the computation is a simple integration over each panel. Here, it does not matter if two different pieces have common boundaries. For Three Dimensional Diffraction, however, it is essential that the panels represent the true surface and that no part of space belongs to two different panels. This makes the generation methods of the **&SURFACE** menu quite valuable, since dealing with the intersections between pieces is provided.

Once a diffraction mesh has been defined, it can be exported to a file for later use with the command:

&EXPORT MESH

XII.P.2 Defining Surfaces with Polygons

MOSES provides a method for combining closed surfaces in space using simple closed surfaces called blocks. These blocks can then be combined in a variety of ways, to produce more complex shapes. This block combining is accomplished in a menu which one enters with the command:

&SURFACE

and exits using an

END_&SURFACE

command.

At this menu level, one can proceed to build blocks with the command

BLOCK, BLOCK_NAME, –OPTIONS

and the available options are:

–**LOCATION**, X, Y, Z, ROLL, PITCH, YAW
–**STBD**
–**PORT**
–**BOTH**

Here, BLOCK_NAME is the block name, and is optional. If this name is not supplied, MOSES will provide one. Once at the **BLOCK** menu level, building blocks can be generated using any of the commands for describing planes, found in the section describing pieces. The BLOCK menu can be exited with the **END_BLOCK** command.

If one has an existing mesh description, it can be entered into the program through the **MESH** menu, by typing the command

MESH

In this menu, the valid commands are the same ones used to define panels, also found in the section describing pieces. The sole purpose of this menu is to accept previously defined meshes. The mesh generation commands from prior versions of the program are accepted here. For these earlier meshes, the vertices *must* be defined before the panels. Commands are also available for defining a plate mesh. In fact, the term mesh now refers to a structural plate mesh, a hydrostatic mesh, and a hydrodynamic mesh. The differences between these types of mesh are discussed later. Remember to exit from the **MESH** menu by typing **END_MESH**.

The **BLOCK** and **MESH** menus basically serve to describe the polygons that make up a surface. When these have been described, one returns to the **&SURFACE** menu to complete the building process.

Blocks can be manipulated using the following commands:

> **LIST_BLOCK**
> **MOVE_BLOCK**, BLOCK_NAME, ANSNAM, X, Y, Z, RX, RY, RZ
> **DELETE_BLOCK**, BLOCK_NAME(1), ... BLOCK_NAME(n)
> **REFLECT_BLOCK**, BLOCK_NAME, ANSNAM, AXIS

The **LIST_BLOCK** command will provide a list of all blocks currently defined. **MOVE_BLOCK** will create a new block, ANSNAM, based on BLOCK_NAME, moved from its original location to that specified by X, Y, Z, RX, RY, RZ. To delete an existing block, use the **DELETE_BLOCK** command and specify the block name or names to delete. Remember that wild characters are valid here. Finally, the **REFLECT_BLOCK** command will take an existing block, BLOCK_NAME, and create a reflection of the specified AXIS, X, Y or Z. The new block created by the reflection will be named ANSNAM.

Blocks can be combined in a variety of ways, using the commands:

> **UNION**, BLOCK_NAME(1), BLOCK_NAME(2), ANSNAM
> **INTERSECT**, BLOCK_NAME(1), BLOCK_NAME(2), ANSNAM
> **DIFFERENCE**, BLOCK_NAME(1), BLOCK_NAME(2), ANSNAM

Here, BLOCK_NAME(1) and BLOCK_NAME(2) are names of blocks to be combined, and ANSNAM is the name of the resulting combination. The command names describe what happens when two blocks are joined in the specified manner. For instance, a **UNION** simply joins two blocks together. Since, however, we want a closed surface as a result, the volume *inside* will be removed. Both the **INTERSECTION** and **DIFFERENCE** commands can be thought of a "throwing away" part of a block. A **DIFFERENCE** of two blocks "subtracts" the part of BLOCK_NAME(2) inside BLOCK_NAME(1) from the resulting block. An **INTERSECTION** keeps only the part of BLOCK_NAME(1) inside BLOCK_NAME(2). In either case, the result is a closed surface. With either of these commands, strange things may happen if BLOCK_NAME(2) does not contain any of BLOCK_NAME(1). As a general rule, **UNION** is used to combine two blocks for creating the exterior of a body, **DIFFERENCE** is used to make holes in a body, and **INTERSECTION** is used for making compartments from the exterior.

The basic design here is that one defines a set of simple blocks and combines them to create the exterior. Now, one uses the exterior, more special blocks and **DIFFERENCES** to define all of the compartments for the body. After everything has been defined, one deletes the temporary blocks, renames those he wishes to keep and emits the model. Any time during the process, one can make pictures of the blocks with

PICTURE, BLOCK_NAME(1), ..., BLOCK_NAME(n)

One renames blocks with the command:

RENAME_BLOCK, :BLOCK_SEL, –OPTIONS

where the options are:

–**POINT**, *PNAM,
–**PANEL**, PNL,
–**SORT**, ORDER, JUMP_NUM, JUMP_TOL
–**EQUIVALENT**, DIST

The **RENAME_BLOCK** command removes all blocks except those selected with :BLOCK_SEL, and renames the point and panel names according to the options used. The **–POINT** option specifies a point name prefix, and should begin with an *****. The **–PANEL** option describes a panel name prefix, while the **–SORT** option defines a criteria for sorting the resulting point and panel names. Here, ORDER can be any combination of the letters XYZ. Using this option, points output to the file will be sorted according to their coordinates, in the order specified. JUPNAM is the integer amount added to a point name when there is a jump in the coordinates of a point of JUMP_TOL. The **–EQUIVALENT** option defines a distance DIST (feet or meters) which is used for point equivalence. Two points within this distance are declared to be the same and references to the deleted point are removed from all panels. This option is quite useful in removing small pieces of trash which results from combining blocks. Renaming is not necessary, but it provides a set of results which are much easier to read and which have a smaller number of panels and points.

The final step in this process is to **EMIT** the results created. This is accomplished with the command:

EMIT, :BLOCK_SEL(1), ... :BLOCK_SEL(n), –OPTIONS

where the options are:

–**PART**, PART_NAME, PAROPT
–**BODY**, BODY_NAME, BODOPT
–**USE_NAME**, YES/NO
–**PERM**, PERM
–**NAME**, NAME
–**COMPARTMENT**, CMPOPT
–**PIECE**, PIEOPT

–POINTS

Here, the **–PART** or **–BODY** option instructs MOSES to emit the points for the blocks matching the :BLOCK_SEL(i) selectors and to specify that they all belong to either body BODY_NAME or part PART_NAME. This should normally be the first thing emitted. After the points have been emitted, one should then emit the panels for the exterior and any interior compartments. Each one of these should have a single emit command. Panel names will be used if YES/NO is set to **YES** on the **–USE_NAME** option. If YES/NO is **NO**, no panel names will be provided. The **–PERM** option is used to specify the permeability of the piece being emitted. When emitting panels, both compartments and pieces will be generated: one compartment is defined for each emit command and a piece for each block emitted. Thus, if there is on only a single block emitted one gets a compartment and a piece. The **–NAME** option specifies the name of the compartment. If it is omitted, then the compartment name will be the same as the first block emitted.

The **–BODY**, **–PART**, **–COMPARTMENT** and **–PIECE** options allow one to specify options which are emitted on the &DESCRIBE command. When using this capability, one needs to remember to enclose the various data within " or ' marks.

For complicated models with multiple pieces, it can be useful to emit only the points for a piece without the associated &DESCRIBE BODY or &DESCRIBE PART commands. This flexibility allows the emitted points to be assigned to a previously defined body or part, which facilitates automatic generation of a complete model into one post processing file. To emit only points, use the **–POINTS** option.

The tools described here are quite powerful, but as with all powerful things, one must use them with care. The way this operation works is that MOSES finds the part of one block "inside" the other and then eliminates it. For this to work properly, however, MOSES must be able to unambiguously distinguish what is "inside". This looks quite simple, but it is not numerically. Consider, for example two blocks:

```
BLOCK BOX
   PLANE –50 50 –RECT 0 10 100
END_BLOCK
BLOCK HOLE
   PLANE –10 10 –RECT 0 10 20
```

```
END_BLOCK
```

and suppose one removes the hole from the box with

```
DIFFERENCE, BOX, HOLE, NEW
```

In all likelihood, this command will fail. The reason is simple, the top and bottom surfaces of the hole lie in the same plane as those of the box. Numerics being what they are it is possible that MOSES thinks that the hole is entirely inside the box, so nothing is subtracted. To make things work properly, one should define the hole as:

```
BLOCK HOLE
   PLANE –10 10 –RECT –1 11 20
END_BLOCK
```

Now, there is no confusion as to whether or not the box intersects the hole. The same rule applies to the other two operations: **Always make sure that the two blocks penetrate one another and two lines do not intersect.**

Continuing with this example, we can now create tanks in the box.

```
BLOCK AT
   PLANE 40 51 –RECT –1 11 120
END_BLOCK
BLOCK PTW –LOC 0 –40 –port
   PLANE –60 60 –RECT –1 11 11
END_BLOCK
INTERSECTION BOX AT ATT
INTERSECTION ATT PTW PAW
```

The blocks AT and PTW are temporary blocks, the first being a slice across the stern and the latter a slice along the port side. Notice that in both cases, care was taken so that these slices "extended beyond" the basic box. The first intersection uses the temporary block to clip a true slice from the stern. Here, due to the simplicity, this is not necessary, but this produces a true piece of the ship from 40 feet aft. The last intersection produces a tank at the port aft of the box. If one defines more longitudinal and transverse blocks, all of the tanks can be easily created.

To complete this example, we would emit the exterior of the box and the tank with:

```
RENAME_BLOCK  BOX PAW –EQUIV .1 –POINT *P
EMIT @ –PART BOX
EMIT BOX –PIECE ' –DIFTYPE 3DDIF –PERM 1.00'
EMIT PAW –COMPART "–DESCRIPT 'Port Aft Fuel Tank' " \
–PIECE   '–PERM –0.98'
```

Notice here the use of **"** to delimit the options which are used when describing the compartment. This is necessary because a **'** is already in use to delimit the text for the **–DESCRIPT** option.

XII.P.3 Interior Compartments

As mentioned above, interior compartments have attributes that exterior compartments do not need. These are specified with the options on the **&DESCRIBE COMPARTMENT** command that defines the compartment. The first or these options:

–**DESCRIPTION**, DESC

allows one to specify a description for this compartment. If quotes are used this description can include blanks and be up to 40 characters in length. This description is simply printed with some **&STATUS**es and at the top of Tank Capacity reports.

Sounding tubes are used to define the level of contents in a compartment, and are defined with the option:

–**S_TUBE**, *PT, *PB

Here, *PT is a point which defines the top (at the deck) of the tube and *PB defines the other end. If you do not define a sounding tube, then MOSES estimates a location for you.

The option:

–**MINIMUM**, MPERC, SPGC

is used to define the "residual water" in compartments. When this value is defined, one cannot ballast below this level, and there is no free surface correction due to the minimum amount of water. The specific gravity of this "residual water" is stated using SPGC.

The last option defines the "holes" which pierce the surface of the compartment:

–**HOLES**, HOLE(1), ... HOLE(n)

Holes are defined with the command

&DESCRIBE HOLE, HOL_NAME, HOL_TYPE –OPTIONS

Here HOL_NAME is the name of the hole and HOL_TYPE is its type which must be either **F_VALVE**, **WT_VENT**, **M_VENT**, **VENT**, or **V_VALVE**. The type is used to control when water may flow into the compartment, so the distinction between vent and flood valves is strictly for operational purposes; i.e. both allow water into the compartment the same way when open. Of the vents, the **WT_VENT** is peculiar. It never allows water to flow into the compartment and is used strictly for reporting. A **M_VENT** is a "magic" vent in that it never goes under the water. The difference between a **VENT** and a **V_VALVE** is that a vent cannot be closed and hence a compartment with a vent can not have an internal pressure. VENTs and WT_VENTs are used to define traditional down–flooding points. The difference between the two is that the "non weather tight" points can get water inside due to splash, etc. These points are used primarily when assessing stability, but one can obtain &STATUSes of them. Down–flooding points (vents) are ignored when a compartment is flooded.

Holes have: a location, an area, a normal, and a friction factor and these are defined with the options:

–**POINT**, POINT_NAME
–**AREA**, AREA
–**NORMAL**, NX, NY, NZ
–**FRICTION**, F_FACTOR

Here, AREA is the cross sectional area of the valve (inches**2 or mm**2), POINT_NAME is the named of a point defining the centroid of the hold, NX, NY, and NZ are the components of the normal of the hole, and FRICT is the friction coefficient for the hole and piping system. The normal to the hole is actually the normal to the area of the hole and points out of the compartment. If some of this data is omitted, defaults will be used.

The flow rate is calculated by MOSES using the following equation:

Q = U * AREA * SQRT(2GH) / SQRT(F_FACTOR)

where Q is the flow rate in cubic feet per minute or cubic meters per minute, G is the gravitational constant, H is the differential head (feet or meters), and U is a constant which makes the units work out correctly. The average flow rate between two subsequent events of a static process is used to calculate the time to flood a compartment.

For many cases one does not need all of this complexity. In particular if one only wants to check the intact stability of a vessel all he needs is a list of down–flooding points. To save work in this case, MOSES allows these to be defined directly on the &DESCRIBE COMPARTMENT command with the options:

–**WT_DOWN**, *WD(1), *WD(2), ...
–**NWT_DOWN**, *ND(1), *ND(2), ...

Here, the *WDs and *NDs are the names of points. Often one does not wish to build a complete model, so these options can be used on an exterior compartment to define down–flooding into non modeled volumes.

The string function

&COMPARTMENT(:CMP_SELE –OPTION)

is useful when writing macros. It returns a string containing the information defined by –OPTION for each compartment selected by :CMP_SELE. For all values of –OPTION, there are more than one token returned for each compartment and the first token is the compartment name. The available options are:

–PART
–PIECES
–HOLES
–F_TYPE
–MIN_WT_DOWN
–MIN_NWT_DOWN
–PERCENT
–MAX_PERCENT
–AMOUNT
–MAX_AMOUNT
–SOUNDING
–ULLAGE
–CG
–FULL_CG
–CG_DERIVATIVE
–FS_MOMENT
–MAX_DHEAD
–PRESSURE
–FLOW_RATE

–PIECES returns the name of the compartment and the name of each piece used to define the compartment. In a similar fashion, **–PART** provides the name of the compartment, and the part name to which the compartment belongs. **–F_TYPE** returns the flooding type, and **–HOLES** returns the names of the holes piercing the compartment. The next two options return a single token in addition to the compartment name: with **–MIN_WT_DOWN** it is the minimum height of all weather tight

down–flooding points and with **–MIN_NWT_DOWN**, the minimum height of all non weather tight down–flooding points. For the next six options, two tokens are returned in addition to the compartment name: a measure of the amount of ballast in the compartment and the specific gravity of the contents. The measure returned (the second token) is defined by the option selected: with **–PERCENT** it is the percentage full, with **–MAX PERCENT** it is the maximum percentage full, with **–AMOUNT** it is the amount in the compartment, in bforce units, with **–MAX_AMOUNT** it is the maximum amount in the compartment, in bforce units, with **–SOUNDING** it is the sounding in feet or meters, and with – **ULLAGE** it is the ullage in feet or meters. With the next five options, one token is returned in addition to the compartment name. **–CG** and **–FULL_CG** return four tokens, the name of the compartment and the X, Y, and Z part coordinates of the current location and full location of the center of gravity. The option −**CG_DERIVATIVE** also returns three tokens: the name of the compartment, and the longitudinal and transverse derivatives of the CG with respect to angle changes (feet or meters/deg). The option −**FS_MOMENT** returns three tokens: the name of the compartment, and the longitudinal and transverse free surface moments in bforce–blength. the compartment. The last three options return a single token in addition to the compartment name:

- the maximum differential head with −**MAX_DHEAD**,
- the internal pressure (ksi or mpa) for −**PRESSURE**,
- the flow rate −**FLOW_RATE**,

XII.P.4 Filling Interior Compartments

Interior compartments can get "contents" in several ways:

- An accident can occur which breaks a hole in the compartment,
- One can open a flood valve and let water run into the compartment,
- One can open the vent valves and let water run into the compartment, or
- Fluid can be pumped directly into the compartment.

Any of these processes is controlled by one of two commands. One of these "automatically" pumps water into compartments and will be discussed below. The other one is:

&COMPARTMENT, –FLAG(1), –OPTION(1), ... –FLAG(n), –OPTION(n)

Now, this syntax is different from the normal in that here there are two things which begin with a –. The basic distinction between a FLAG and an OPTION is that FLAGS specify a class of action and OPTIONS are more specific. A single FLAG can apply to more than one OPTION.

To control accidental or intentional flooding one uses the four flags:

–FLOOD, :CMP_SEL(1), ... :CMP_SEL(n)
–NO_FLOOD, :CMP_SEL(1), ... :CMP_SEL(n)
–OPEN_VALVE, :CMP_SEL(1), ... :CMP_SEL(n)
–DOWN_FLOOD, :CMP_SEL(1), ... :CMP_SEL(n)
–DYNAMIC, :CMP_SEL(1), ... :CMP_SEL(n)

The **–FLOOD** flag tells MOSES that the compartments which match :CMP_SEL(i) will be open to the sea and that they will normally be full of water up to the waterline. The **–NO_FLOOD** flag is used to reverse the process. When this is used, compartments revert to being filled as before it was flooded, but with the *current* filling type (filling type is discussed below).

The **–OPEN_VALVE** specifies that all flood valves attached to the compartment are open. Here, ambient water will flow into the compartment, or contents will flow out depending on the location of the valves, internal pressure, and amount of ballast in the compartment. This change of fluid in the compartment occurs *statically*, and can be observed using **&STATUS COMPARTMENT**. The maximum volume of fluid in the compartment is artificially limited by that specified using the add ballast options.

The **–DOWN_FLOOD** option instructs MOSES to fill the compartment up to the waterline whenever the lowest vent point goes below the water. Here no maximum need to be specified. This option is particularly useful for hydrostatic stability and capsizing studies. Down–flooding points on tanks that have been marked with **–DOWN_FLOOD** are not used when computing down–flooding during a RARM command. Thus, this option can be used to get a correct picture of the stability. Notice that when this command is issued, water may be drained from the compartment.

The **–DYNAMIC** option defines tanks which will be flooded dynamically when a time domain simulation is performed. In essence, this option opens the valves of all holes with a type of F_VALVE at the beginning of a time domain simulation. Using this option is reasonably delicate. It should be the last item specified on the command. Any reference to "dynamic" compartments after they are selected will *turn off* the dynamic behavior.

There are three options which influence this flooding:

–INT_PRE, :CMP_SEL(1), ... :CMP_SEL(n), INTPRE, EMP_FRACT
–COMPRESSOR, :HOL_NAME(1), ... :HOL_NAME(n), PCOMP, FLCOMP
–PUMP, :HOL_NAME(1), ... :HOL_NAME(n), PCOMP, FLCOMP

The **–INT_PRE** option is used to define the initial internal gage pressure (air pressure in the compartment minus atmospheric pressure) in the compartment (ksi or mpa). If INTPRE is zero, then all vents in the compartment are open, if INTPRE is greater than zero, then all vent valves on the compartment will be closed. If there are holes with a type of VENT, then there will be no internal pressure, otherwise the internal pressure will limit the capability of the compartment to flood. EMP_FRACT specifies the percentage full of air in the compartment at the stated internal pressure. If this value is left blank, MOSES assumes INTPRE is the pressure acting when the compartment is 100 percent full of air. The **–COMPRESSOR** and **–PUMP** compressor or a pump attached to a compartment which will act during a time domain simulation. Here, PCOMP is the rated gage pressure (ksi or mpa), while FLCOMP is the rated flow rate of the pump or compressor in cubic feet per minute or cubic meters per minute. The pump or compressor are connected to holes in the compartment and here :HOL_NAME(i) are selectors for the holes which will receive the pump of compressor.

Now, to pump contents into a compartment one has the flags:

–CORRECT, :CMP_SEL(1), :CMP_SEL(2), ...
–APPROXIMATE, :CMP_SEL(1), :CMP_SEL(2), ...
–APP_NONE, :CMP_SEL(1), :CMP_SEL(2), ...
–APP_WORST, :CMP_SEL(1), :CMP_SEL(2), ...
–FULL_CG, :CMP_SEL(1), :CMP_SEL(2), ...
–FCG_NONE, :CMP_SEL(1), :CMP_SEL(2), ...
–FCG_WORST, :CMP_SEL(1), :CMP_SEL(2), ...
–INPUT, AL, AT, Gx, Gy, Gz, :CMP_SEL(1), :CMP_SEL(2), ...
–INITIAL,

–**ADDITIONAL**,

and the options:

–**PERCENT**, :CMP_SEL(1), PERC(1), SPGC(1), ...
–**FRACTION**, :CMP_SEL(1), FRAC(1), SPGC(1), ...
–**AMOUNT**, :CMP_SEL(1), BAL(1), SPGC(1), ...
–**SOUNDING**, :CMP_SEL(1), S(1), SPGC(1), ...

are available. In essence, the flags tell MOSES how to treat the contents and the options tell the amount of contents.

Before this can be described intelligibly, a bit of history is in order. When a compartment is not filled completely, its center of gravity moves about relative to the body when the body changes angle. Now, we simply compute the correct location of the center of gravity. In the past, however, this was a laborious computation and several methods were developed to approximate its location; i.e.

$$Xg = Xo + Ac$$

where Xg is the current CG location, Xo is the CG location in a reference position, A is a matrix (with only two non zero terms) of the derivative of the CG with respect to angle changes, and c is the change in angle from the reference position. In perhaps more familiar terms, the A matrix contains all zeros except for the first and second diagonal elements and these are the "free surface moments" divided by the current amount of ballast in the tank. We are being a bit vague here for a reason. Different stability rules **require** that certain values be used for Xo and A in certain conditions. Thus the different filling types specified by the flags above – a way to treat the problem approximately and satisfy the rules!

The filling FLAGS correspond to different choices of Xo and A as follows:

- –**CORRECT** compute the CG and its derivative at each point in a simulation.
- –**APPROXIMATE** use the correct CG and its derivative when the compartment is filled.
- –**APP_NONE** use the correct CG when it is filled and use zero for the derivative (no free surface correction).
- –**APP_WORST** use the correct CG when it is filled and the derivative which yields the largest free surface moment. Here maximum is the maximum over the depth of the tank.
- –**FULL_CG** use the CG of the tank when it is full for the reference CG and use the correct derivative when the tank was filled.
- –**FCG_NONE** use the CG of the tank when it is full for the reference CG and zero for the derivative.
- –**FCG_WORST** use the CG of the tank when it is full for the reference CG

and use the derivative which produces the maximum free surface moment.

- **–INPUT** use the input values for the CG and its derivative. Here AL and AT are the longitudinal and transverse derivatives in feet or meters/deg and Gx, Gy, and Gz are the X, Y, Z coordinates (feet or meters) of the CG in the part system.

The two flags **–INITIAL** and **–ADDITIONAL** define whether or not the compartments will be emptied before adding, or whether the following fluid will be added to that which exists. For all of these flags, :CMP_SEL(i) is a selector which defines the compartments which will have their properties altered.

The –FLAGS define how the amounts defined with **–PERCENT**, **–FRACTION**, **–AMOUNT** or **–SOUNDING** will be treated. These options instruct MOSES to add ballast to the specified tanks; the only difference is the manner in which the amount of ballast is specified. With **–PERCENT**, one specifies the percentage full, with **–FRACTION**, he specifies the fraction full, with **–AMOUNT**, he specifies the amount of ballast (bforce), and with **–SOUNDING** he specifies a sounding (feet or meters). SPGC is the specific gravity of the contents of a tank. If it is omitted, then the last specific gravity provided for a tank will be retained. For new tanks, the default is the specific gravity of ambient water. If none of these flags is specified, then all tanks mentioned will have a fill type defined by the **–FILL_TYPE** option of the **&DEFAULT** command. The values :CMP_SEL(i) which may follow these options allows one to change the type of filling without altering the amount of ballast in the tanks.

This command is quite complicated and a few examples are in order:

```
&COMPARTMENT –CORRECT ONE TWO S@
```

will change the type of filling for tanks ONE, TWO, and all tanks which match S@ to CORRECT. A tank half full can be specified as:

```
&COMPARTMENT –PERCENT ONE 50
```

or

```
&COMPARTMENT –FRACTION ONE .5
```

A tank full of two different fluids can be defined by:

```
&COMPARTMENT –PERCENT     ONE 50 1.025
&COMPARTMENT –ADDITIONAL ONE 50  .800
```

Here, the heavier fluid will be at the bottom of the tank, and the ballast will correctly reflect the combination of the fluids. The vertical CG, however, will be approximated by the center of volume of the two fluids.

For an example of moving water in a compartment statically, consider:

```
&COMPART –OPEN_VALVE –PERCENT ONE 50 –INT_PRE ONE 3 100
```

In this case, fluid will flow out of the compartment if the specified internal pressure is greater than the ambient pressure at the valve location.

To initially ballast a tank half full and have it flood dynamically, one issues:

```
&COMPARTMENT –PERCENT ONE 50 –DYNAMIC ONE
```

and to turn off dynamic flooding,

```
&COMPARTMENT  –PERCENT ONE 50
```

To simulate a compressor attached to a compartment, consider the following:

```
&COMPART –DYNAMIC ONE –PERC ONE 50 –INT_PRE ONE 1E–5 .5  \
        –COMPRE ONE .05 10000
```

Here, the vent valve is closed by providing a non–zero value for INTPRE. To analyze what happens as the compartment empties, one should issue a TDOM command.

To automatically ballast a body to achieve a desired condition, one should use the command:

&CMP_BAL, BODY_NAME, :CMP_SEL(1), –OPTIONS, ... :CMP_SEL(n) –OPTIONS

where the available options are:

–LIMITS, MIN, MAX
–HARD

Here, BODY_NAME specifies the name of the body to be ballasted and :CMP_SEL(i) specifies the compartments in which ballast will be altered. MOSES will then move water into and/or out of specified tanks until the body is in equilibrium. The minimum and maximum amount compartments will be ballasted can be defined with the **–LIMITS** option. All compartments mentioned *after* this option will not be ballasted below MIN percent full, nor will they be ballasted greater than MAX percent full. If a **–LIMITS** was not used, all active compartments will have a minimum of the minimum allowed in the tank and a maximum of 100.

XII.Q Editing a Model

In the **MEDIT** Menu, one can define and delete elements, nodes, load groups, and element load attributes in much the same way as when defining a model. One can also redefine mapping of panel loads to nodes. In fact, all commands used during **INMODEL** can be used in **MEDIT**.

To delete objects already in the database, the following commands are provided:

MEDIT
    ED_ELEMENT, OBJECT, –OPTIONS
    ED_CLASS, ~CLASS_NAME, SEGNO, ...
    CL_DELETE, :CLS_SEL(1), :CLS_SEL(2), ...
    ELA_DELETE, :ELE_LOAD_SEL(1), :ELE_LOAD_SEL(2), ...
    LG_DELETE, :LG_SEL(1), :LG_SEL(2), ...
    EL_DELETE, :ELE_SEL(1), :ELE_SEL(2), ...

These commands edit elements and classed and delete classes, element load attributes, load groups, and elements respectively, and the selectors are selectors defining the quantities to be deleted. In using the commands which delete quantities, care must be taken, since if a class is deleted, then all elements which use that class will also be deleted. Also, when using the **ELA_DELETE** command, notice that the selectors operate only on the element name. In other words, while one can define element load attributes by either class, nodes defining the element, or name, one can only delete them by element name. A word of *caution* is in order here. Once something has been deleted from the database, all data associated with this quantity is lost. Thus, if one has already performed a structural analysis and he deletes an element or class, then the structural results for the elements deleted will be lost. It is better to ”deactivate” elements, as described later, than delete them.

Notice that there is no command for deleting a node. Since nodes are simply points that are connected with structural elements, there is no need for such a command. In order to relocate a node, simply redefine its associated point. This will move the end of a beam or the corner of a plate that references the relocated node.

Often, elements exist for some stages of an analysis and are absent for other stages. This notion can be modeled by deactivating elements when they are not present with the command:

EL_ACTIVE, –OPTIONS, OBJECT(1), OBJECT(2), ... –OPTIONS, OBJECT(3), ...

Here, OBJECT(i) is as described above and the available options are –**ACTIVE** and –**INACTIVE**. All elements selected with OBJECT(i) will have the activity defined by the option immediately preceding the object.

The command:

GEN_OFFS

It serves the same purpose as the **–OFFSET** option on the **INMODEL** command. Thus, if a model is defined entirely in the **MEDIT** menu, member end offsets can still be generated by issuing this command after the model has been defined. As with all things in **MEDIT**, **GEN_OFFS** only generates offsets on elements which have been defined before the command is issued.

The command:

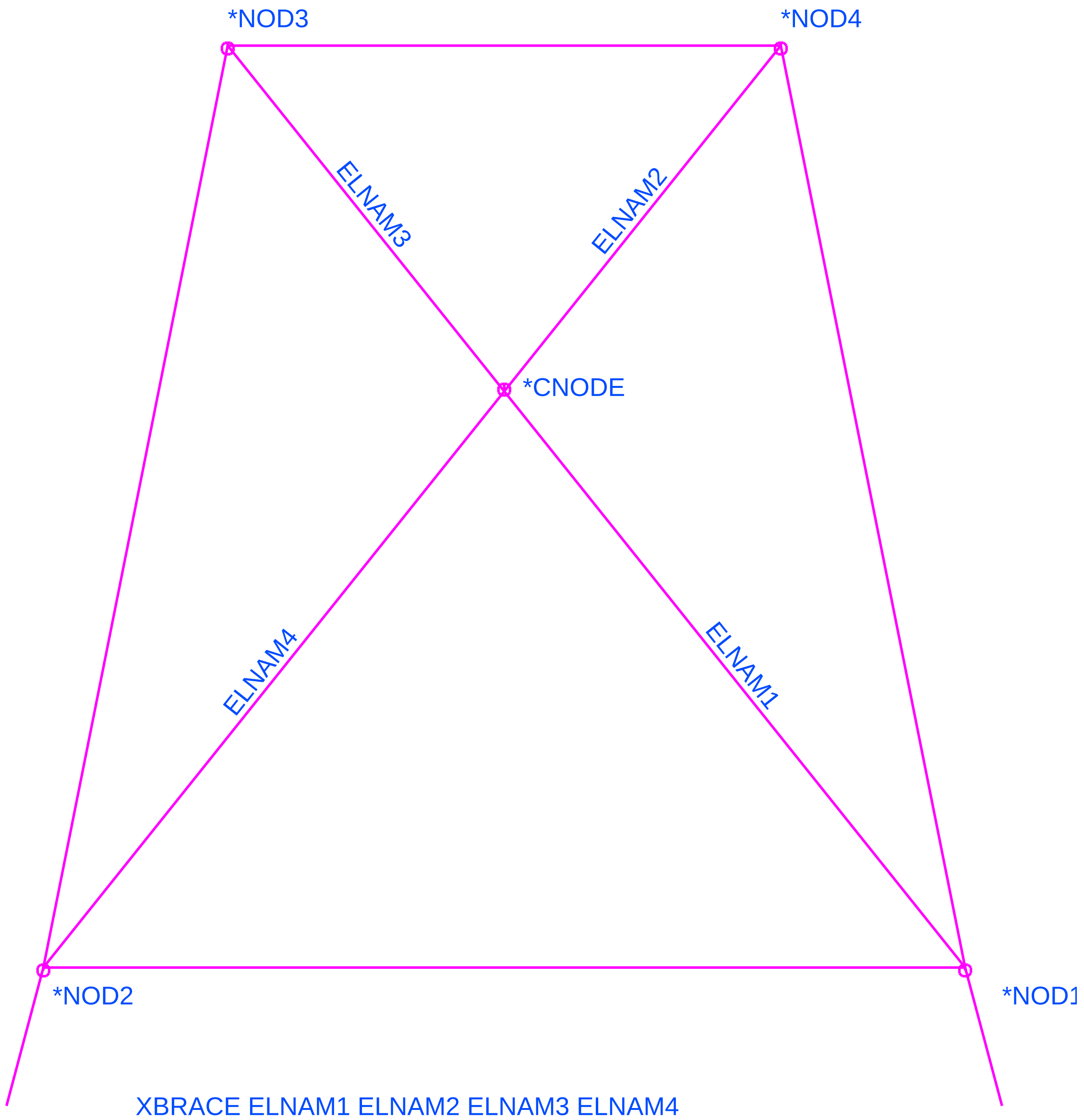
CL_D/T_RESIZE :C_SELECT, D/T_MIN, D/T_MAX, D_INCREMENT

will change the diameter and thickness of tubular members with classes which match the selector :C_SELECT. It will take the original area and compute a new diameter and thickness which produces d D/T ratio between the two limits and which has the same area as the original sizes. The new dimension will be changed to the nearest D_INCREMENT. By default, all classes will be checked, D/T_MIN is 30, D/T_MAX is a huge number, and D_INCREMENT is 1/8 inch or 2 mm depending on the units being used. With the defaults, the new dimensions will be multiples of either 1/8 inch or 2 mm.

A simple way to define element dependent buckling lengths is provided with either of the two commands:

XBRACE, ELE_NAME(1), ELE_NAME(2), ..., ELE_NAME(n)
XBRACE, *CEN_NODE, *NODE(1), ..., *NODE(n)

With either of these commands, one is defining the buckling length of an element to be based on the tension–compression behavior of another element. With the first form of this command, the buckling length of ELE_NAME(1) is based on ELE_NAME(2), ELE_NAME(2) on ELE_NAME(3), ..., and ELE_NAME(n) on ELE_NAME(1). The second form accomplishes the same thing, but here *CEN_NODE is the common node between a set of elements, and *NODE(i) are the other ends. In other words, the two forms are identical provided ELE_NAME(1) is an element between *NODE(1) and *CEN_NODE, ELE_NAME(2) is between *NODE(2) and *CEN_NODE, etc. An illustration of the use of the **XBRACE** command is shown in Figure 22. For a complete discussion on how the buckling length depends on the brace element, see the section on defining elements. Notice that with either of these forms, one should define either the nodes *NODE(i) or the elements ELE_NAME(i) *in order* around the center node.



XBRACE ELNAM1 ELNAM2 ELNAM3 ELNAM4
or
XBRACE *CNODE *NOD1 *NOD2 *NOD3 *NOD4
equivalent:
BEAM ELNAM1 -BLENG ELNAM2 ~CLASS *NOD1 *CNOD
BEAM ELNAM2 -BLENG ELNAM1 ~CLASS *NOD4 *CNOD
BEAM ELNAM3 -BLENG ELNAM4 ~CLASS *NOD3 *CNOD
BEAM ELNAM4 -BLENG ELNAM3 ~CLASS *NOD2 *CNOD

BUCKLING LENGTH RELATIONSHIP FOR X BRACES
FIGURE 22

XIII. CONNECTIONS AND RESTRAINTS

In MOSES, bodies are connected together or to ground with special elements called connectors or restraints. A restraint is an element which acts like a connector during a stress analysis, but does not act during a simulation. Basically, restraints are of limited use. Connectors, however, are quite important. In general, there are three categories of connector: flexible, rigid, and connector assemblies. Flexible and rigid connectors connect a body to either another body or to ground. Connector assemblies are sets of simple connectors which act in unison and may connect more than two bodies. Rigid and flexible connectors actually serve the same purpose – they create a force between two points. The real difference is how the force is computed. With flexible connectors, the force is computed based on a force–deflection rule and the geometry of the system. Rigid connectors yield constraints on the motion of the bodies. Excepting numerics, a very stiff flexible connection and a rigid connection should yield the same result. In reality, when the stiffness gets substantial, numerical precision is lost and the solutions are either difficult or impossible to obtain. In these cases, rigid connections can ameliorate the difficulties.

Here, we will call connectors "simple" if they are not assemblies. All simple connectors are defined with the following command in the **MEDIT** menu:

CONNECTOR, CNAME, –OPTIONS, ~CLASS, *NODE(1), *NODE(2)

and the available options are:

–**GO**i, X, Y, Z
–**ANCHOR**, THET, DTA
–**EULER**, E_DATA
–**NUM_APPLIED**, NUMBER
–**TUG**, ANG, DIST

and restraints are defined with

REST, ~CLASS, *NODE(1), *NODE(2)

Here, ~CLASS is a previously defined class and *NODE(1), and optionally *NODE(2), are the nodes to which the connector will be attached. The details of the options and the number of nodes required depends on the category of connector class. The –**GO**i options are used to define offsets at the "ith" vertex of the element. Here, –GO1 defines offsets at the first end, etc. The values X, Y, and Z define the coordinates of the offset (inches or mm) and are defined in the part system. If only –GO is specified, then all vertices will have the same offsets. In all cases, an offset is defined as the vector from the node to the vertex of the member. To connect a body to ground, either the second node should be omitted, or it should be a *fixed* node. Fixed nodes are nodes which belong to a special part **GROUND**.

The **–ANCHOR** option is applicable only to connectors which have a class type of **ROD** or **B_CAT**, and *must* be used to define the location of the ground attachment for these classes. Here, the location is defined relative to *NODE(1) by a heading and a distance. THET is the heading (deg.) from the vessel X axis, (in the current configuration, positive toward Y) to the anchor line, and DTA is the horizontal distance (feet or meters) from the attachment point to the anchor. The vertical coordinate is not defined since it is given by the depth of the anchor on the class definition.

The **–EULER** option is applicable only to connectors which have a class type of **GSPR**, **LMU**, or **FOUNDATION** and it is used to change the element system of the connector. By default, the element system is aligned with the body system of the body to which the first node belongs. E_DATA can be a set of Euler angles which changes this orientation. There are three angles: a roll, a pitch, and a yaw. These are rotations about the element system. In particular, suppose that the element system is in its default position. Now, give it a yaw (a rotation about the element Z axis), and then a pitch (a rotation about the new element Y axis), and finally a roll (a rotation about the new element X axis) to put it into the desired position. As a quick way of defining these angles, MOSES will also accept values of **+X**, **–X**, **+Y**, **–Y**, **+Z**, or **–Z**. These align the element X axis with the body axis specified. In other words, a value of –Y is the same thing as an value of 0, 0, –90.

The option **–NUM_APPLIED** allows one to have a multiplier, NUMBER, for a connector. The results for a connector are first computed based on the specified properties, and then all results are multiplied by NUMBER. In particular, the damping, stiffness, mass, matrices and the force are multiplied. This option works only with flexible connectors.

The final option is applicable only to **–TUG** connectors. Here, ANGLE is the global direction of the tug, positive when measured from the global X axis towards the global Y axis. DIST is the distance (feet or meters) from the attachment point on a body, *PT, to the tug.

The string function:

&CONNECTOR(:CON_SELE, –OPTION)

returns the current data for each connector selected by the selector :CON_SELE. The first token is the name of the connector and the remainder depends upon the options. For a type of

- **–TENSION** the remainder is by the magnitude of the force in the connector,

- –**HORIZONTAL** the remainder is the magnitude of the X and Y components of the force,
- –**FORCE** the remainder is the X, Y, and Z components of the force in the connector,
- –**RATIO** the remainder is the ratio of the acting force in the connector to the breaking strength,
- –**E_NAMES** the remainder is the names of the points at the "ends" of the connector,
- –**E_COORDINATES** the remainder is the coordinates of the ends of the connector in the body system of the body,
- –**ANCHOR** the remainder is the three coordinates of the anchor,
- –**LENGTH** the remainder is the current length of the first segment of the connector,
- –**HEADING** the remainder is the local body heading of the connector,
- –**G_HEADING** the remainder is global heading of the connector,
- –**FRICTION** the remainder is the friction in the connector,
- –**T_MULT** the remainder is the current trust multiplier for the connector,
- –**T_ANGLE** the remainder is the current body angle of the thrust of the connector,
- –**R_ANGLE** the remainder is the current angle of the rudder for the connector,
- –**HP_ANCHOR** the remainder is the horizontal pull on the anchor for the connector,
- –**VP_ANCHOR** the remainder is the vertical pull on the anchor for the connector, and
- –**L_ON_BOTTOM** the remainder is the line on bottom for the connector.

Since there are many different situations that MOSES can analyze there is a command that allows one to change many of the properties of connectors as different situations arise. This command is discussed in the section on altering connectors.

XIII.A Defining a Pulley Assembly

A pulley assembly is simply a set of connectors connected with pulleys. Such an assembly is defined by the command

ASSEMBLY PULLEY, PUL_NAME, EL(1), ..., EL(i)

Here, PUL_NAME is the name you wish to give to the pulley assembly, and EL(i) are previously defined connector element names. The connector elements *must* be either **H_CAT** (without the exact option) or **B_CAT** elements. If one has a **B_CAT** element, then it must be the *last* element of the assembly. In essence, a **ASSEMBLY PULLEY** assembly is a single line which passes through n–1 pulleys. If all of the elements are H_CATs, then the computation is exact, and all of the properties of each segment of each connector are properly accounted for. If, however, one has an assembly with a B_CAT element, then the idealization is exact *only* when the size of the wire for the H_CAT element is the same as the top segment of the B_CAT. Here, also, any additional springs on the H_CAT element *will be ignored.* These elements can be used in any way that H_CAT or B_CAT connector can be used. In particular, the **&CONNECTOR** command can be used to change the length of the first segment or the change the activity of the element.

XIII.B Defining a Launchway Assembly

A launchway is a complicated set of connections which are collectively referenced within MOSES by the name **&LWAY**. The launchway is composed of at least two launch legs each one of which is defined via the command:

ASSEMBLY LLEG, *J(1),*J(2), .. *J(n), BODY_NAME(1), XB, YB, ZB, :B1(1), .. \
:B1(n) BODY_NAME(2) :B2(1), ... :B2(n),
–OPTIONS

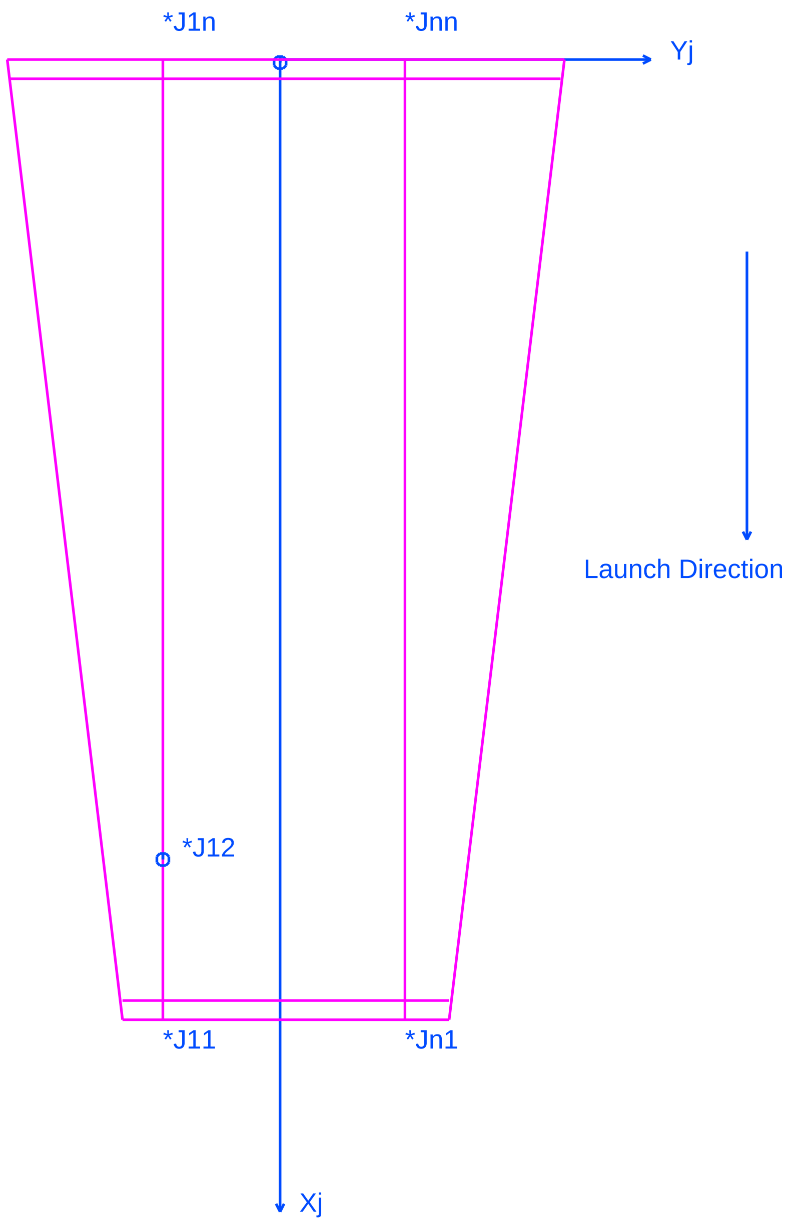
and the available options are:

–**FRIC**, DYNFRC
–**TPIN**, TPRIDEP, XP, YP, ZP, MAX_ANGLE, TSECDEP, DIST
–**BEAM**, LENP, EIP, LENS, EIS

Here, *J(1), ... *J(n) are the node names of the nodes along the launch cradle of the jacket, *in order*, where *J(1) is the first node which will enter the water and *J(n) is the last node which will enter the water. The launch cradle is considered the part of the jacket that rests on the barge skidway. An illustration of the node arrangement is shown in Figure 23. BODY_NAME(1) is the body name assigned to the barge where the tiltpins are attached, and XB, YB, and ZB are the coordinates, in the BODY_NAME(1) body system, of the beginning of the skidway on body BODY_NAME(1). Here, the skidway should be considered to be at the height of the jacket launch leg centerline above the barge origin. Also, :B1(i) are selectors for the nodes in BODY_NAME(1) which will be used for connecting the jacket to the barge. BODY_NAME(2) is the name of an optional second body over which the skidway may pass, and :B2(i) are again node selectors.

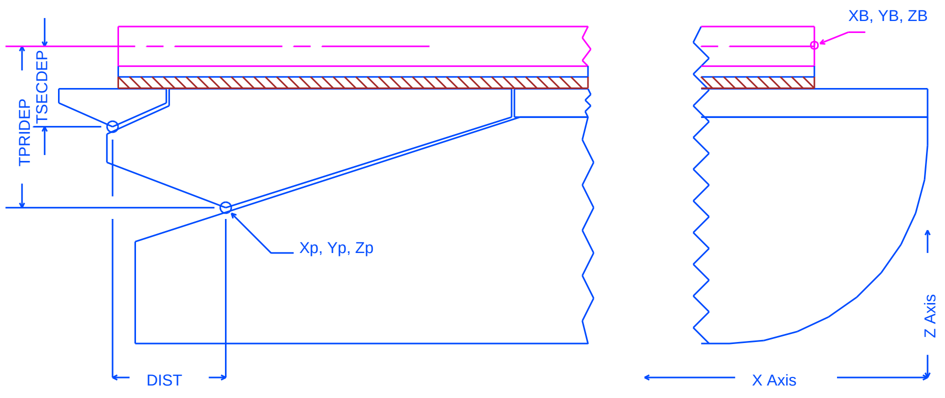
The dynamic coefficient of friction for the launchway is specified with the –**FRIC** option as DYNFRC. The tiltbeam geometry for the launchway is specified via the –**TPIN** option. Here, XP, YP, and ZP are the body coordinates of the primary tiltpin (feet or meters), TPRIDEP is the height of the primary tiltbeam (feet or meters), MAX_ANGLE, the maximum angle the tiltbeam is allowed to rotate until the secondary tiltbeam becomes active (deg.), TSECDEP, is the height of the secondary tiltbeam (feet or meters), and DIST is the distance along the skidway from the primary tiltbeam to the secondary one (feet or meters). Here again, the depth of the beams should be considered to be the vertical distance from the tiltpin to the centerline of the jacket leg. If there is no secondary tiltbeam, one should omit the values for MAX_ANGLE, TSECDEP, and DIST. An illustration of the tiltbeam is shown in Figure 24.

The stiffness of a tiltbeam is input using the –**BEAM** option. Here, LENP is the length of the primary tiltbeam (feet or meters), and EIP is the stiffness (bforce–ft**2 or bforce–meters**2). LENS and EIS are the length and stiffness values for the secondary tiltbeam, respectively. If there is no secondary tiltbeam, these



PLANE VIEW SHOWING JOINTS USED ON LLEG COMMAND

FIGURE 23



TILTBEAM GEOMETRY

FIGURE 24

values should be omitted.

The *order* of input of the **ASSEMBLY LLEG** commands is important as it is used to establish the launch coordinate system of the jacket. The axes of this coordinate system are set as follows: The X axis is parallel to a line connecting *J(1) and *J(n), and is directed towards *J(1). The jacket is launched in the positive X direction. The origin of this system is midway between the trailing joints given on the first and last **ASSEMBLY LLEG** commands, and the Y axis is along the line connecting the *J(n) on the last **ASSEMBLY LLEG** input with *J(n) on the first one input. The Z axis is determined from the right hand rule.

At the conclusion of the **MEDIT** Menu, the orientation of the body systems will, in general, change. MOSES will change the body system of the jacket as described above. It will also change the orientation of the barges so that their body X axis will be aligned with that of the jacket. In contrast to the jacket, there is no translation of the barges' body system.

XIII.C Defining a Sling Assembly

A sling assembly is an abstraction of the lines which a lift vessel can apply to another body. Here, the point at the boom where the connection begins is called the boom point. This point is connected to the hook by a length of line, and in turn the hook is connected to several points on the body by other lines. In MOSES, the hook can be connected to the body with up to four elements. This set of lines is called a tip–hook set, and is referred to by a name given to the hook. Several different tip–hook sets can be defined connecting different bodies.

Each tip–hook set is defined by a command:

ASSEMBLY T–H_DEFINITION, NAME, BHE, EL(1), ..., EL(4), –OPTIONS

and the available options are:

–INITIAL,
–ORIENT,
–VERTICAL
–DEACTIVATE

Here, NAME is the name you wish to give the tip–hook set, BHE is the name of the element connecting the boom point to the hook, and EL(i) are the names of the elements connecting the hook to a body. All of these elements *must* have been previously defined with **CONNECTOR** commands, *must* have a class category of **SLING**, and *must* have a single node. Additionally, all of the nodes of the EL(i) must belong to the same body.

The **–INITIAL** option instructs MOSES to "initialize" the set. If this option is specified and this is the first tip–hook set defined, the body will be moved so that the hook point is directly below the boom point. If the option is selected and this is not the first tip–hook set, then the length of the tip–hook elements will be changed so that there is no sag in the lines.

The **–DEACTIVATE** option instructs MOSES to to deactivate all previously defined tip–hook sets.

The **–ORIENT** option can be used to alter the definition of the body system of the body to which the EL(i) are connected. With this option, the order of the sling nodes is used to define the local body system. The origin of this system is defined as the midpoint of the vector connecting the first two sling nodes. The local Y axis is in the direction of the first node toward the second, the local Z axis is from the second node to the third, and the local X axis is given by the right hand rule.

During static processing, the configuration of the body is defined by the height of the hook attachment point above the waterplane and two angles. These two angles are called pitch and roll. The two angles are defined by vectors which can be defined via a **&DEFAULT** command. The **−VERTICAL** option can be used to automatically define these three quantities. Here, pitch is the angle which the normal to the plane formed by the sling nodes makes with the waterplane, roll is the rotation of the line defined by the first two sling nodes, and height is measured to the midpoint of the line connecting the first two nodes.

Once the sling is defined, the lengths of the tip–hook elements can be changed with **&INSTATE –SL_SET**. MOSES will place the system in the specified configuration and compute the lengths of each tip–hook element so that there is no slack in the assembly. Also, the **&CONNECTOR** command can be used to change the length of the boom element and deactivate the tip–hook set. The name to use here is the, NAME, the name given the set.

XIII.D Defining a Pipe or Riser Assembly

A pipe assembly is a **ROD** element along with a set of elements which connect the **ROD** to a set of bodies. The connecting elements can be either **DAVIT** or **ROLLER** elements. Here, **DAVIT** elements are lines which connect the pipe, while **ROLLER**s are one–sided constraints which keep the **ROD** on a pipe–lay route. Such an assembly is defined by the command

PIPE, ~PIPE_CLASS, EL(1), ..., EL(i), –OPTIONS

and the available options are:

–PIPE_TENSION, TLOWER, TUPPER

Here, ~PIPE_CLASS is the name of a **ROD** class which will be used to define the properties of the pipe, and EL(i) are the connector element names of elements which have been previously defined. The connector elements *must* be either **DAVIT** or **ROLLER** elements, and these types of elements *cannot* be mixed. MOSES assumes that pipe is laid in the negative X global direction, so that all of the connectors must lie along the global X axis when the assembly is defined. The pipe itself is given a name **&PIPE** and the assembly has a name of **&PIPE/ASSEMBLY**.

The use of the **–PIPE_TENSION** option depends on the type of connection. For an assembly of **ROLLER** elements, TLOWER defines the smallest allowable tensioner value (bforce) and TUPPER the largest allowable tension (bforce). With a **ROLLER** assembly, the last roller defined is the tensioner. With a **DAVIT** assembly, the value of TLOWER (bforce) is the nominal initial tension in the pipe where it connects to the first davit, and TUPPER is not used.

A special situation occurs when one has only a single roller. In this case, MOSES will treat the pipe assembly as a *vertical* pipe with a tensioner, so that one can simulate a riser. Here, the bottom of the rod is placed directly below the roller connection so that the pipe will have a tension of TUPPER.

When initializing a pipe assembly other than a vertical one, it is assumed that the pipe behaves as a catenary. (Thus, the use of the word nominal as regards the initial tension.) MOSES uses this assumption as initial estimates for the configuration of the assembly, and then iterates a solution to the nonlinear problem. A stinger is initially tensioned so that either the slope of the pipe matches the slope of the last two rollers, or all of the pipe is off of the bottom. The assembly problem is particularly difficult with davits. Here, the assembly is initialized by using only the length of the first davit and the nominal pipe tension. The lengths of the other davit lines are then computed to conform to this estimate.

XIII.E Defining a Control Assembly

A control assembly is simply a set of connectors connected to a control. Such an assembly is defined by the command

ASSEMBLY CONTROL, CONTROL_NAME, PE(1), ..., PE(i) –OPTIONS

Here, CONTROL_NAME is the name you wish to give to the control assembly, and PE(i) are previously defined PROPULSION connector element names.

Here the options are either

–**SENSORS** SG(1), SEN(1), ... SG(n), SEN(n)

Simply, the control system will compute an x and y force and a yaw moment given by:

F(i) = sum [SG(k) SR(k,i)]

Here SG(k) are the multipliers defined with the –**SENSORS** option and SR(k,i) is the "sensor reading" for the kth sensor. The i is the degree of freedom for the sensor. The thrust is allocated amount the thrusters based on a least squares fit.

Normally, the sensors are "VECTOR" sensors where the first point of the vector is a point on the body and the second point is a point on ground. The signal is the vector between the two points which we want to be zero.

Propulsion units with rudders are treated differently. They can control only yaw and the thrust is not altered by the control system. In other words, one sets the thrust with a **&CONNECTOR** xxx –SET_PROPULSION command and the control system will control the rudder to achieve a given heading.

XIII.F Defining a Winch Assembly

A winch assembly is simply a set of connectors connected to a winch. Such an assembly is defined by the command

ASSEMBLY WINCH, WINCH_NAME, EL(1), ..., EL(i) –OPTIONS

Here, WINCH_NAME is the name you wish to give to the winch assembly, and EL(i) are previously defined connector element names. The connector elements *must* be either **B_CAT**, **H_CAT**, **SL_ELEM**, **ROD**, **GSPR**, **LMU** elements, or the **SLING** element connecting the boom to the hook on a tip–hook assembly.

Here the available option is

–**WINCH** FULL_WEIGHT, MAX_TORQUE, S_MOMENT, D_MOMENT, \
TOT_LENGTH, FULL_GYRADIUS, FULL_RADIUS

This option is used to define the mechanical properties of the winch and alternatively the second defines the velocity of the winch as a function of time. Here, FULL_WEIGHT is the weight (feet or meters) of the drum plus the wire.The next three values define either the maximum applied torque, MAX_TORQUE, maximum applied torque or two moments applied by a brake. S_MOMENT is the static moment due to the brake and D_MOMENT is a factor which when multiplied by the square of the angular velocity is the dynamic moment due to the brake All three of these are in bforce–blength. The last three values are: the total length of wire on the drum, the radius of gyration of the drum plus wire when full, and the radius of gyration of the drum plus the wire when the winch is full in feet or meters.

XIII.G Altering Connectors

So that many different situations can be analyzed, MOSES provides the ability to partially alter the definition of the connector system. This is performed with one command, **&CONNECTOR**, which has several options, most being applicable to a particular class of connector. The form of this command is:

&CONNECTOR, :CONN_SEL(1,1), ..., :CONN_SEL(n,1), –OPTION(1),
\
:CONN_SEL(1,2), ..., :CONN_SEL(n,2), –OPTION(2),
\
:CONN_SEL(1,m), ..., :CONN_SEL(n,m), –OPTION(m)

where the options applicable to all connectors are:

–INACTIVE
–ACTIVE

which makes a set of connectors inactive or active. Each option operates on the list of connectors immediately preceding it. The first option defines the connectors whose names are selected by the selectors :CONN_SEL(i) to be inactive. Inactive connectors can be reactivated by simply issuing the second option.

For connectors with types of **ROD**, **SL_ELEM**, **B_CAT**, and **H_CAT** more things can be accomplished with **&CONNECTOR**. To alter the length of the first segment of a line, use the options:

–LENGTH, LEN
–L_DELTA, DLEN
–L_HORIZONTAL, FORCE
–L_TENSION, FORCE

Here, :CONN_SEL(i,j) are the selectors for lines whose length will be altered, and the method of specifying the new length is controlled by the options. If one uses **–LENGTH**, he is simply defining each line which matches the selectors to have a length of the first segment of LEN (feet or meters). Using **–L_DELTA** is similar to the above except that DLEN (feet or meters) is added to the existing length. Alternately, one could specify either **–L_HORIZONTAL** or **–L_TENSION**. With these options, a new length of the first segment will again be defined, but here, the new length is calculated so that either the tension or horizontal force has the value FORCE in the initial configuration.

To alter the location of the anchor, use the options:

–ANCHOR, XA, YA, ZA
–A_HORIZONTAL, FORCE

−**A_TENSION**, FORCE

Here, :CONN_SEL(i,j) are the selectors for lines whose anchor location will be altered, and the method of specifying the new location of the anchor is controlled by the options. If one uses −**ANCHOR**, he is simply defining the global x, y and z coordinates (feet or meters) of the anchor of each line which matches the selectors. The z coordinate specified here is honored for all flexible connectors except a type of H_CAT, where it is ignored. Alternately, one could specify either −**A_TENSION** or −**A_HORIZONTAL** which instructs MOSES to compute the location so that either the tension or horizontal force has the value FORCE in the initial configuration.

The force of **H_CAT** connectors is computed with the aid of a "lookup table". The force properties of the line are computed as a function of distance from the anchor and stored in a table when the connector is defined or its properties changed. Now, this table produced quite accurate results for changes in horizontal directions, but the changes due to vertical motion are approximated. If one is interested in moving a body vertically, he may need to recompute the table as the body moves. The The option

−**G_TABLE**}

will force MOSES to recompute the table at the current position.

For **ROD** connectors, three additional things can be altered with the **&CONNECTOR** command with the options:

−**A_STIFF**, STADGX, STADGY, STADGZ
−**TOP_MOMENT**, YES/NO
−**ST_ADDITION**, INONUM, STADGX, STADGY, STADGZ
−**ZERO_BSTIF**, YES/NO

Normally, the top of the rod is "pin" connected to the body, and a default stiffness is assigned for the connection at the first node (the "anchor"). The default depends on the type of rod: if it is a "straight rod" a large stiffness is used, while for a "mooring line" type of rod a much smaller stiffness is defined. The option −**A_STIFF** allows one to redefine the anchor stiffness. Here, STADGX, STADGY, and STADGZ are the *global* X, Y, and Z values of the stiffness in bforce/blength. If one uses −A_STIFF 0 0 0, then the rod will have no restraint at the anchor. Similarly, the −**TOP_MOMENT** option changes the connection behavior at the top. If it is used with a YES/NO of **YES**, then the top connection will apply a moment; otherwise, it will apply only forces. The −**ZERO_BSTIF** option changes the connection behavior at the bottom. If it is used with a YES/NO of **YES**, then the stiffness at the bottom will be applied, otherwise the bottom will be free to move.

The option **–ST_ADDITION** allows one to add a diagonal stiffness matrix at intermediate points along the rod. Here, INONUM is the intermediate node number where the stiffness will be added. These nodes are numbered with the bottom being 1, so that to add a stiffness one node above the bottom, it should be added with INONUM equal 2. STADGX, STADGY, and STADGZ are the same as for **–A_STIFF**.

One may alter the settings of a propulsion connector with: with the option:

–SET_PROPULSION, T_MULT, T_ANGLE, R_ANGLE

of the **&CONNECTOR** command. Here, T_MULT is the fraction of the maximum thrust (–1 <= T_MULT <= 1) which will be applied, T_ANGLE is the angle that the thrust will be applied, and R_ANGLE is the angle of the rudder. Both of these angles should be –90 <= ANGLE <= 90.

One may alter a tug connector with: with the options:

–T_FORCE, FORCE
–T_LOCATION, ANG, DIST
–T_DYNAMIC, PERCENT_FORCE, PHASE

of the **&CONNECTOR** command. The values for FORCE, ANG, DIST, PERCENT_FORCE and PHASE have the same meaning as on the commands defining the connector.

To define the special states of a foundation element for checking, one uses the option

–SET_STATE, TYPE, MULT

of the **&CONNECTOR** command. Here, TYPE must be either **PRELOAD** or **NOMINAL**, and if TYPE is NOMINAL, then MULT is a multiplier which will be use in computing the unity ratios otherwise it should be omitted. See the discussion in the section on Process Post–Processing of Connectors for details on the unity ratio computation.

A launchway assembly can be activated or deactivated by using the name **&LWAY** for :CONN_SEL(i,j). Also, one can alter some of the settings for the assembly of launchways with the options:

–LWA_FRICT, DYNFRC
–LWA_ANGLE, MAX_ANGLE

Here, one selects the launchways which will have their properties altered with

:CONN_SEL(.). The **–LWA_FRICT** option sets the dynamic coefficient of friction of the selected runners to DYNFRC, and the **–LWA_ANGLE** option sets the maximum angle for the first tiltbeam on the selected launchways to MAX_ANGLE (degrees). The names of the launchways are **&LLEG**i where i is a number assigned as the launchways were defined.

Some things with pipe assemblies can also be changed with the **&CONNECTOR** command. The options are:

–PIPE_TENSION, TLOWER, TUPPER
–DAV_LENGTH, NEWLEN,
–MOVE_ROLLER, DX, DY, DZ,
–LOC_ROLLER, X, Y, Z,
–A_STIFF, STADGX, STADGY, STADGZ
–TOP_MOMENT, YES/NO
–ST_ADDITION, INONUM, STADGX, STADGY, STADGZ

Unless otherwise specified, :CONN_SEL should be **&PIPE**. The **–PIPE_TENSION** option sets the lower and upper bounds for the tensioner tension, TLOWER and TUPPER (bforce). The **–DAV_LENGTH** option changes the length of the davit elements selected by :CONN_SEL to be NEWLEN (feet or meters). If :DELEM selects the first davit element, then the entire assembly will be re–initialized and all of the lengths will be changed as when the assembly was defined. If the first element is not altered, then the selected lengths will be changed and a new equilibrium configuration of the pipe will be computed. The two options **–MOVE_ROLLER** and **–LOC_ROLLER** both result in the assembly being completely re–initialized. The first of these options moves rollers which match :CONN_SEL from their current location by an amount DX, DY, and DZ (feet or meters). The second simply defines the new location of the roller to be X, Y, and Z, (feet or meters), in the stinger body system. The last three options: **–A_STIFF**, **–ST_ADDITION**, and **–TOP_MOMENT** operate on the pipe exactly as they do for a simple rod connector, and were discussed above.

Winch assembly properties can also be altered with the **&CONNECTOR** command by using the option

–L_DYNAMIC, ACTION, MULT, BOUND}

where :CONN_SEL selects the winches to be altered. Here ACTION must be either **MOTOR**, **BRAKE** or the name of a **CT_LENGTH** "curve" that defines the rate of change of length (feet or meters/sec) of the line. For the first of these, the MAX_TORQUE will be multiplied by MULT and the winch motor will be turned on and the brake will be released. For the second, S_MOMENT and D_MOMENT will but multiplied by MULT and winch motor will be turned off and the brake will be engaged. For the last action, the velocity curve will be multiplied by MULT. Here, BOUND is a bound on the line length. If MULT is greater than zero (letting out line) the BOUND is an upper bound on the line length. If MULT is less than zero (letting in line) the BOUND is an lower bound on the line length.

Normally control assemblies are changed only in the time domain. One can use the **–SET_PROPULSION** option to set the values for statics, or use the option

–CONTROL

With this option, the control system will attempt to counteract the static mean wind, current, and wave forces.

XIV. PROCESSES

One thing which is fundamental to MOSES is the concept of a process. Whenever MOSES performs a simulation, the results are stored in the database by the process name. The name it uses is the ”current one”, or the last one which was referenced. To alter the ”current one”, one should issue the following command:

&DESCRIBE PROCESS, PRCNAM, –OPTIONS

and the only option is:

–EVENTS

When an **&DESCRIBE PROCESS** command is encountered, and the name for the specified process is not already defined in the database, it will be added to the database with the name specified. MOSES will then set the ”current” process to be the one specified.

Here, by simulation results, we mean the initial and current configurations, the events for the process, the activity status of the connections, the lengths of each mooring line, the load group multipliers, and the load set multipliers. In other words, all events of a simulation, the initial conditions, and any of the data defined by **&COMPARTMENT**, **&CMP_BAL**, **&APPLY** and **&CONNECTOR** commands is stored by process name. By altering the process name, one can actually consider several different connection situations using the same model. This is important, since one can also perform a structural analysis for several processes with the same factored stiffness matrix.

Two types of data are stored by process name: constants and data which depends upon event. Most of the data mentioned above are constants, and only one value will be associated with a process. The data which is stored by event is the configuration, the forces acting on the bodies, and any results of the **&COMPARTMENT**, **&CMP_BAL**, **&APPLY** and **&CONNECTOR** commands. The constants are the results of the **&ENV** command. Thus, a given process can have only a single environment.

When the process name is altered, the new process will have all of the same settings and initial condition as the situation immediately before the name was changed, but the results of the previous simulations will appear to be ”lost”. They are, however, not really lost. To recover them, one should issue another **&DESCRIBE PROCESS** command to change the current process back to the name under which the results are stored. Thus, the process concept allows one to have many different sets of results available for further processing.

If one wishes to have the events of the previous process available in the new one, he can specify the **–EVENTS** option when he created the process. Now, the new process is a true copy of the old process, and it can be altered as desired.

Most processes will have events defined by either a time domain simulation (the results of a **TDOM** or **LAUNCH** command) or an upending sequence. At the conclusion of a time domain or launch simulation, MOSES will revert to the situation at the beginning. Thus, issuing **&STATUS** after a time domain simulation will produce the same result as issuing it before the simulation. In some cases, however, one may wish to construct a series of events which correspond to a set of equilibrium configurations with different ballast conditions. This can be accomplished by computing an equilibrium configuration, and storing the results with a given event number via the command:

> **&EVENT_STORE**, EVE_NUMBER

The results of this "simulation" can be post–processed as any other simulation in the Process Post–Processing Menu. *Notice*, if this command is used when a process already exists (a launch, time domain, upend, etc.) then it will corrupt the existing data.

The string function which returns data for processes is:

> **&PROCESS**(ACTION)

and ACTION must be either: **PROCESS**, **C_EVENT**, **MAX_EVENT**, or **MIN_EVENT**. Which return the process name, the current event, the maximum event and the minimum event for the current process.

XV. AUTOMATIC OFFSHORE INSTALLATION

Both the strength and weakness of MOSES is its flexibility; there is virtually nothing that one cannot accomplish provided they are willing to spend the time. While the flexibility allows one to analyze everything, the price one pays is the cost of learning. If, however, one is willing to establish some rules and to decide on a fixed objective for a given analysis, then that analysis becomes routine.

A system of macros has been developed for the analyses of the installation of a jacket, deck, or other structure. In general, they can be used for loadout, transportation, lift, upend and launch. Considerable flexibility has been incorporated into this system so that most cases can be easily considered. This system has been designed so that several simulations that represent different phases of a structure can be performed, load cases generated, and one code check over all load cases can be produced.

With this system, one defines all of the data necessary for a transportation, launch, upend, lift or loadout analysis of a structure in a single data file with each analysis using the same data. There are two general rules here to which one must adhere:

The dimension cannot be changed during the analysis
One must use a barge which conforms to conventions

By a barge conforming to conventions, we simply mean that all of the basic variables have been set. A library of vessels is provided which conform. If one of these is not suitable, you can prepare a model of your own. In preparing this model, you should use the data file SAMPLE.DATA in the vessels library and which is discussed here.

To use this system, you should first read the documentation and then copy files install.dat and install.cif from /ultra/hdesk/tools/install to your working directory or by clicking here, and then modify them to suite your problem. Most of the work is involved with changing install.dat and we will discuss it first.

Most of the data required is defined in this file. The exceptions are the basic model data of the structures to be analyzed and the barge data. The structures data is assumed to be in other files which are inserted. The barge data is also inserted, but it is assumed to be in a special directory which contains all of the barge models.

Basic Data

The data required here is broken down into sections. The first section contains "basic" data required for any analysis and is basically self explanatory. The line

&DIMEN –SAVE –DIMEN METERS K–NTS

defines the units which will be used in the analysis. Two definitions are available to control the pictures:

I_SET DO_MOVIE = .TRUE.

and

I_SET RENDER = –RENDER GL

If the first of these is set **.TRUE.** then "movies" of a launch and or upending will be created *provided* you are rendering the pictures with GL. The second one defines the rendering mode of the pictures. If **–RENDER GL** is set, then pictures will be rendered with GL. To save time, you could use **–RENDER WF**. Finally, if you are using a "TERMINAL" interface, all pictures will be rendered as WF pictures and no movies will be produced.

The following:

I_SET WDEPTH = 390

defines the water depth. The "margins", or weight contingencies are defined by the two lines:

I_SET MARGIN = 5
I_SET PER_APPLY = 105

The first of these is for the basic steel in the model and the next is for any "joint loads".

The line

I_SET T_CODE = API WS

defines the type of code checks which will be computed. Here one can specify any code that is accepted by the BEAM_POST or JOINT_POST commands. For transportation, the macros will automatically build the proper load cases, but more information is required:

I_SET T_CODE = API LRFD SO_FACTOR SM_FACTOR DM_FACTOR

Here, the static internal forces and restraints are multiplied by SM_FACTOR and combined with DM_FACTOR time the dynamic component before reporting or checking a code. Also, the static case times SO_FACTOR is also checked. One should not use an LRFD code for any other type of analysis.

The line

I_SET CODE_LIM = 1.33,1e6 1.,1.33 0.9,1.0 0.,0.9

defines the limits for the code check and joint check reports. If you do not change this line, then the checks will be broken down ratios between 1. and 1.33, the next for ratios between .9 and 1, and the last for ratios between 0 and .9.

The elements for which code checks and/or fatigue will be performed are defined by the following:

I_SET C_CODE = –SELECT @ –EXCEPT ∼dum@
I_SET N_CODE = –SELECT @ –EXCEPT
I_SET N_FAT = –SELECT @ –EXCEPT

Here, CLASS defines a variable that is used to determine the classes which will be considered for all three categories. If a class is not defined here, no element in this class will be considered for Beam Check, Joint Check, or Joint Fatigue. N_CODE is a variable which defines the joints which will be considered. If a joint is not selected, no results for Joint Check or Joint Crushing will be produced for this joint. Finally, N_FAT defines the joint to be considered for Joint fatigue.

Fatigue Data

If one wants to consider fatigue, then he must define several things. One thing, which is essential, is that the duration data which will be used must be specified. This is with an option of the transportation macro and is discussed below. Also, one may wish to alter one or more of the following:

I_SET FAT_LIM = 1.,1.e6 0.25,1 0,0.25
I_SET SCF = Efthymiou
I_SET SN = XP
I_SET B_SN = AWSE

The first of these defines the limits for which the fatigue reports will be broken down. If this is not changed, one will receive 3 reports for joint and beam fatigue: The first will have CDRs above 1, the second will have CDRs between .25 and 1, and the last will have CDRs between 0 and .25. The next two lines deal with tubular joint fatigue. Here, SCF defines the type of SCFs which will be computed during fatigue. The SN variable defines the SN curve for Joint Fatigue. In either of the variables, multiple curves can be used. For example,

I_SET SN = XP X

will produce fatigue results for both the XP and the X curves. The B_SN variable defines the SN curve which will be used for Beam Fatigue, and the &REP_SELECT command defines a new type of SN curve, AWSE. Here, the default SCFs for beam which are not tubes are set to 1, and the AWSE curve is used for beam fatigue.

Report Data

The following command defines the cover page for the report. Be sure to enclose the variables in single quotes(') if they are more than one word.

I_BEGIN,–OPTIONS

And the available options are:

–**TLINE1**, 'XYZ EXPLORATION and PRODUCTION'
–**TLINE2**, '8 Pile Jacket for the COWABUNGA Field'
–**TLINE3**, 'Installed Offshore Timbuktu'
–**CLIENT**, 'QRS Engineering, Inc'
–**FOOTER**, '8 Leg Jacket'

Barge Data

This data is necessary only for a launch or transportation.

USE_VES BARGE

The data for the barge being used is more restrictive than that for the structure. Again, guidance on how to make your own barge model can be found here. The reason for this is that the barge tilt beam data and many other things are necessary. Thus, the barge used must be one of the barges supplied, or a new barge defined in the same format. Here, BARGE, should be the name of the barge one wishes to use.

Structure Data

This section of data needs to be completed for any analysis. Here, one is calling a macro which sets up the data required for each structure to be analyzed. The syntax of the macro is:

MODEL_IN NAME FILE X Y Z –OPTIONS

And the available options are:

–**PORT_NODES**, *P1, *P2, *P3 ...
–**STBD_NODES**, *S1, *S2, *S3 ...
–**ORIENT**, *O1, *O2, *O3
–**TOP_NODE**, *TOP_NODE

–EXTREMES, P_NAM(1), *P_NODE(1), P_NAM(2), *P_NODE(2) ...

The variable "FILE" defines the file which contains the model for the structure.

Even thou the –PORT_NODES and –STBD_NODES are listed as options they are necessary. If you have a transportation analysis with multiple structures, you should have a "MODEL_IN" for each structure. Sometimes you may have a jacket and deck on the same barge, or two deck sections on the same barge. The MODEL_IN command can also be used to input files that describe miscellaneous cargo, such as piles or boat landings. In this case, FILE for each cargo would contain the appropriate commands to adequately describe the cargo, such as PGEN and #WEIGHT.

The variable "NAME" defines the type of structure. Any name up to eight characters may be used, but the names JACKET and TRIPOD are special. Also, for multiple structures on one barge, the first three characters of the name must be unique. This definition controls the type of connections which are established for transportation and the initial setup for upending. For upending with a type of TRIPOD, the axes system will not be moved when slings are added.

The variables X, Y and Z define the location of the "origin" of the structure on the barge. Here, X is the location aft of the bow, Y the distance off of the centerline and Z is the height above the barge deck. The meaning of origin changes with how the structure is oriented. Normally, the two options **–PORT_NODES** and **–STBD_NODES** define the orientation. If they orient the body, the origin is the midpoint of the trailing port and starboard nodes. If the **–ORIENT** option is used, the origin is the first node specified.

The two options **–PORT_NODES** and **–STBD_NODES** are used to define the names of the nodes on the "launch legs". The first node for each variable is the node at the leading edge of the jacket and the last is the node at the trailing edge of the jacket. The leading edge is defined as the end of the jacket that enters the water first. The PORT_NODES are on the port side of the barge while the STBD_NODES are on the starboard side. When specified, these nodes define the orientation of the structure on the barge. They are also used to define barge/structure connections if a V_LWAY connection is specified or if one performs a loadout analysis.

For a structure which is not symmetrical about the barge centerline the orientation scheme is different. Here, the **–ORIENT** option is used. This option defines three nodes. The first node is where distances for positioning will be measured and is normally at the bottom of the leg that is parallel with the deck edge, assuming the top of jacket faces aft. The second node is along the leg from the first node, and the third node is on the other side of the barge, usually along the horizontal level in line with the first node. Y is the distance from the centerline of the barge to the first node, positive towards starboard. Note that if one specifies starboard nodes and a negative Y then the jacket will be placed under the barge.

–TOP_NODE option is used for an initial guess during upending and stability springs for other types of analysis. This node should be on the face of the structure which is the highest above the water in the initial floating position and should not be attached to any slings.

Points used for reporting purposes can be specified with the **–EXTREMES** option. Here, a point name and node name is required for each point of interest. For a jacket, these are normally the top and bottom nodes of each corner leg.

For deep water fixed leg structures, the inside diameter of leg compartments can vary substantially. For these situations, there is a useful command for defining very accurate tank definitions, which has the following syntax:

I_TANK TANK_NAM, *BOT_NOD, *TOP_NOD, E_BOT, E_TOP, –OPTIONS

And the available options are:

–F_VALVE, VF_DIA, VF_DIST
–V_VALVE, VV_DIA, VV_DIST
–ELEVATION
–PERMEABILITY, PERM
–B_NODES, BN(1), BN(2), ...

Here, TANK_NAM is the name given to the tank, and *BOT_NOD and *TOP_NOD are the names of the bottom and top nodes on the jacket leg where the tank resides. The variables E_BOT and E_TOP provide the locations of the bottom and top of the tank, respectively. These are the bulkhead locations inside the leg. If this information is not supplied, the bulkhead locations will be assumed to be at the bottom and top nodes. If the **–ELEVATION** option is used, these bulkhead locations will be assumed to be jacket elevations, where the jacket origin is at the inplace waterline, and Z is vertical up. Without this option, the bulkhead locations are assumed to be the length along the leg from the bottom node. Bulkhead locations are specified in feet or meters, depending on the current units. The diameter and location of the flood and vent valves are specified with the **–F_VALVE** and **–V_VALVE** options. For this information, valve diameter is specified in inches or millimeters, and the valve location is specified in feet or meters. The valve locations here are according to the use of **–ELEVATION**. If no valve information is provided, a 4 inch flood valve will be located at the bottom node, while a 4 inch vent valve will be placed at the top node. The **–PERMEABILITY** option allows one to specify the permeability for the tank. Normally legs which contain tanks are straight. The **–B_NODES** option allows one to specify joints at which the leg has a break in slope. The **I_TANK** command will use the jacket model to prepare the proper TUBTANK definitions, capturing all the changes to inside diameter along the elements defining a jacket leg, including changes to segments in the elements.

Connector Data

There are three different categories of connectors that can be defined for use in a simulation: SLINGS, TIEDOWN CONNECTORS and VERTICAL SUPPORTS. Each of these categories uses the **I_CONNECTOR** command, followed by a type description and then the required data.

To define an upending sling assembly for use in a jacket upending analysis, a type of UP_SLING is used, and has the following syntax:

I_CONNECTOR UP_SLING *U1 L1 *U2 L2 ...

The data that follows the connector type is a set of node names and harness lengths in feet or meters. The number of pairs defined gives the number of sling elements which will be attached. For a body name of JACKET, the order of the nodes is used to define the local body system. The origin of this system is the midpoint of the vector connecting the first two sling nodes. The local Y axis is in the direction of the first node toward the second, the local Z axis is from the second node to the third and the local X axis is given by the right hand rule.

To perform a lift analysis, you will need a connector type of LIFT_SLING:

I_CONNECTOR LIFT_SLING *L1 LEN1 *L2 LEN2 *L3 LEN3 *L4 LEN4

A sling will be constructed from each of the nodes specified to the common hook point.

Tiedown connectors for a transportation analysis can be defined using the following **I_CONNECTOR** types:

I_CONNECTOR 4_TIE ~TD_CLASS *TIE1 *TIE2 ...
I_CONNECTOR V_BRACE ~TD_CLASS *TIE1 *TIE2 ...
I_CONNECTOR P_BRACE ~TD_CLASS *TIE1 *TIE2 ...
I_CONNECTOR H_BRACE ~TD_CLASS *TIE1 *TIE2 ...
I_CONNECTOR PCONNECT TIEDOWN DATA
I_CONNECTOR XY_DELTA ~TD_CLASS DELTA_X DELTA_Y *TIE1 *TIE2 ...

When the **4_TIE** connector type is specified, 4 tiedowns with the properties of ~TD_CLASS will be generated at each node specified. The ~TD_CLASS must be defined before it is referenced on the **I_CONNECTOR 4_TIE** command. The tiedowns will be arranged in star pattern, with each tiedown 45 degrees from a longitudinal axis that passes through the tiedown node and is parallel to the barge centerline. The longitudinal and transverse distance from the referenced structure node to the deck end of the tiedown is the same as the vertical distance of the referenced node above the barge deck.

Connectors types of **V_BRACE**, **P_BRACE** and **H_BRACE** are all very similar to one another. The names here refer to Vertical Brace, Pitch Brace, and Horizontal Brace, respectively. The **V_BRACE** takes only dynamic vertical load, no gravity load, and creates an element from the referenced node to the barge deck. The **P_BRACE** takes only longitudinal dynamic load, and creates a horizontal element that is 5 feet or meters long. The **H_BRACE** takes only transverse dynamic load, and creates a horizontal element from the referenced node to the side shell. As with the **4_TIE** type, the referenced ~TD_CLASS must have been previously defined. For all these tiedown types, the connection at the barge end of the tiedown takes no moments, meaning a pinned connection.

If none of the above tiedown connector types are suitable, one can still define connectors explicitly, and place this definition in this file. For tiedowns, this is done with the **ICONNECTOR PCONNECT** command. While this format allows for any valid **PCONNECT** data, the following information is normally provided:

> **I_CONNECTOR PCONNECT** DX DY DZ ~TD_CLASS *NOD *B@

For structure descriptions that include tiedowns, the tiedowns should be removed from the structure file and placed in this file using the above **I_CONNECTOR PCONNECT** method.

The **I_CONNECTOR XY_DELTA** command provides an easy way to define tiedowns where the barge end remains at the same height as the referenced node on the structure. DELTA_X and DELTA_Y refer to the distance from the referenced node to the barge end of the tiedown.

Vertical supports that take gravity load can be defined with these I_CONNECTOR types:

> **I_CONNECTOR V_LWAY**
> **I_CONNECTOR V_CAN** ~CAN_CLASS *C1 *C2 ... –OPTION
> **I_CONNECTOR V_REST** ~REST_CLASS *R1 *R2 ...

The **V_LWAY** type will create a vertical connector using the node names provided on the **–PORT_NODES** and **–STBD_NODES** options of **MODEL_IN**. This will actually create a structural element that simulates the launchway on a barge, using the launchway information provided in the barge model. If this launchway information is not available, a WBOX beam is generated that is 48 inches deep, 48 inches wide, with 1 inch plates for the flanges and sides, and 2 inch plate for the center plate. The connections created here are gap elements. For spectral load cases, only a linear structural solution can be performed, so the gap elements have no effect on these cases. For time domain load cases, a nonlinear structural solution is performed, which involves iteration over the support nodes to release those supports that show tension.

The connector type **V_CAN** provides a vertical support can with the properties provided by the can class ~CAN_CLASS. A beam element is created from the referenced node to the barge deck, with moments about the local Y and Z axes released at barge end. If one specifies the option –DO_HORIZONTAL, then restraints will be added to prevent lateral motion of the cargo on the barge during stages when the tiedowns are not connected.

A connector type of **V_REST** works in a similar fashion. Here, simply specify the restraint class and support nodes, and restraints will be provided at each node.

Command File

After fixing up install.dat, one should turn to install.cif. Here one has the option to perform different installation simulations, perform the structural analyses, and do one comprehensive structural code check for all load cases. Below, we will discuss the commands to make this happen. The first of these is for a structural loadout analysis:

INST_LOADOUT –OPTIONS

And the available options are:

–VERT_RST
–GAPDIS, GAPDIS
–LENSKD, LENSKD
–FXLOC, FXLOC
–TOPLOAD
–PSUPNOD, PNODE(1), PNODE(2), ...PNODE(n)
–SSUPNOD, SNODE(1), SNODE(2), ...SNODE(n)
–NO_STRUCT

It is easier to discuss these commands assuming a jacket is the structure being loaded onto a barge, but this can just as easily be used for a deck loadout. This command will move the structure from the land onto the barge and create a load case for structural analysis whenever a structure hard point leaves the land support. Gap elements are used for the supports and a nonlinear structural solution is performed, unless **–VERT_RST** is used. In this case, rigid restraints would be used instead of gap elements.

The **–GAPDIS**, **–LENSKD**, and **–FXLOC** options define the geometry of the loadout. Here, GAPDIS is the distance between the land skidway and the beginning of the barge skidway and LENSKD is the length of skidway on the barge that actually provides support. FXLOC is the final location of the jacket on the barge. This is a distance from the end of the barge skidway (nominally the bow) to the trailing edge of the jacket. It is positive if the trailing edge of the jacket is aft from the bow, negative otherwise. The trailing edge is defined as the end of the jacket that would come off last if the jacket were being launched from the barge.

It is assumed that the jacket is loaded out with the base of the jacket moving onto the barge first, unless the option **–TOPLOAD** is exercised. It is further assumed that the stern of the barge is towards the fabrication bulkhead. An over the bow loadout can be analyzed by specifying the proper values for GAPDIS and FXLOC.

The options **–PSUPNOD** and **–SSUPNOD** can be used to specify support nodes, if the supports are different from PORT_NODES and STBD_NODES. The variables PORT_NODES and STBD_NODES are defined in install.dat.

This macro is designed to perform a simulation and a corresponding structural analysis by default.

Next, we will discuss the transportation analysis, which has the following syntax:

INST_TRANSP, –OPTIONS

And the available options are:

–NO_SEAKEEPING
–NO_VORTEX
–NO_STAB
–NO_STRUCT
–DRAFT, T_DRAFT
–TRIM, T_TRIM
–BALLAST, BAL_SEL
–AMOUNT, BAL_AMT
–CMP_BAL, CMP_SEL
–EQUI
–DAMAGE, DAM_CMP
–S_COND, S_C
–PERIOD, PERIOD(1), ...
–HEADING, HEADING(1), ...
–WIND, W_INTACT, W_DAMAGED W_VORTEX, W_STRUCTURAL
–MO_POINTS, P_NAMES
–SPEED, SPEED
–TYPE_SPECT, SPECT_TYPE
–STEEP, STEEPNESS
–DO_FREQ
–DO_TIME, TOB TINC

–FLEXIBLE
–DURATION DUR_FILE DUR_TIME DUR_VELOCITY
–TIETEN

If one does not want some of the default results, they can turn them off with the **–NO_SEAKEEPING**, **–NO_VORTEX**, **–NO_STAB**, or **–NO_STRUCTURAL** options. The two options **–DRAFT** and **–TRIM** define the draft at the bow and the trim for the tow. If these two options are not used, then a trim of .57 degrees will be used and the draft will be set so that the draft amidships is half the depth. A weight is then computed so that the specified condition is achieved. If, however, the **–BALLAST** option is used, the situation is different. Here, the variable BAL_SEL is a string containing a set of pairs of tank names and percentages full. If this is specified, then this ballast condition will be used and equilibrium found as the transportation condition. In a similar fashion, the **–AMOUNT** option allows one to specify a ballast amount, in the current big force units. This must also be in the form of tank name and amount pairs. If the **–CMP_BAL** option is invoked, MOSES will compute the ballast amount required in each tank listed in CMP_SEL to achieve the specified draft and trim. If the **–EQUI** option is used, MOSES will consider all the information in the barge and cargo input files, and find an equilibrium condition. The **–DAMAGE** option defines DAM_CMP which is a list of compartment names which are damaged. If this is omitted, only intact stability will be computed.

The option **–S_COND** defines the sea states to be considered. Here, specify several sea triples. These three tokens are first a character sea–state identifier, next a wave height and finally a period.

The options **–PERIOD** and **–HEADING** define the periods and headings at which the response operators will be computed. If they are omitted, then headings of 0, 45, 90, 135, 180, 225, 270 and 315 and periods of 4, 5, 5.5, 6, 6.5, 7, 7.5, 8, 8.5, 9, 9.5, 10, 10.5, 11, 13, 15 and 20 seconds are used.

The option **–WIND** defines the wind speeds used in the analyzes. Here, W_INTACT is the wind speed for intact stability, W_DAMAGED for damaged stability, W_VORTEX for vortex shedding, and W_STRUCTURAL for structural analysis. The defaults are 100, 50, 100, and 100 knots respectively. If W_STRUCTURAL is zero then wind load is not included in structural load cases.

The option **–MO_POINTS** will provide statistics of motions for the point names specified with P_NAMES. This is an easy way to determine the motion accelerations at specified locations on the cargo.

Control of the spectral motions computation is provided with the **–SPEED**, **–TYPE_SPECT** and **–STEEP** options. Here, SPEED is the forward speed of the vessel in knots, and the default is 0 forward speed. SPECT_TYPE can be either ISSC, JONSWAP or a previously defined user spectrum, where ISSC is the default. The **–STEEP** option specifies the reciprocal of wave steepness, and uses a value of 20 as the default. A value of SPECTRAL can also be used, in which case the first environment specified on **–S_COND** will be used to linearize the equations of motion.

The default for producing structural load cases for transportation is to create frequency domain spectral load cases, and no particular option is required to make this happen. However, one can also prepare time domain load cases by using the **–DO_TIME** option. Here, TOB is the total time of observation in seconds, while TINC is the time step increment. What happens next is quite involved for such a deceptively simply option. A time domain synthesis will be performed for the motions of the center of gravity for each piece of cargo, for each environment specified on **–S_COND**. Then, for each of these environments, the time for the extreme force or moment for each of the six degrees of freedom will be determined. By extreme here, we mean a maximum or minimum, such as positive and negative roll. These times are then used in the creation of deterministic structural load cases. Regardless of the time of observation specified with TOB, the results will be adjusted to reflect a 3 hour simulation. Note that the time domain results used here come from a time domain synthesis, where the waterplane is assumed to remain constant. If only the **–DO_TIME** option is used, only time domain cases will be created. To provide the frequency domain and time cases in the same run, use both the **–DO_FREQ** and the **–DO_TIME** options.

It is prudent to make a quick preliminary run to check the position of the structures on the barge before investing in the longer duration run that performs the entire analysis. For these types of runs, use of the various **–NO_** options will turn off the specified computation.

If the **–FLEXIBLE** option is exercised, the flexibility of the barge will be considered. Otherwise, the barge will be considered as rigid. If tiedowns are included in the model, a sequential structural analysis will be performed. The first pass through the structural solver will create a dead load case without tiedowns, while the second pass will create dynamic load cases including tiedowns. The default action regarding tiedowns is to assume that a tension connection does not exist at the barge end of the tiedown. Another way to say this is the footprint of the tiedown brace lands on a doubler plate, and the welding of the barge deck plate to the web frames underneath is not sufficient to develop tension. For these situations, the load cases for the tiedowns are conservatively multiplied by two. The assumption here is that tiedowns are arranged as inboard/outboard pairs, and the tension that would have otherwise developed on one side goes into compression on the opposite side. If the tiedowns can really develop tension at the barge deck, use the **–TIETEN** option. In this case, the multiplier for the tiedown load cases will be one.

The **–DURATION** option is used to define the duration data for fatigue during this process. DUR_FILE is a file containing the duration data for the tow.

Also, DUR_TIME is the total time for which the data in DUR_FILE will act and DUR_VELOCITY is the average velocity of the tow.

One can issue several INST_TRANSP commands. For each command issued a process will be created and the results will be post–processed. This is an automated way in which to consider situations with different drafts, trims, etc and still have a single fatigue results for all of them.

The automated lift analysis needs almost no user involvement, and is invoked with the following command:

INST_LIFT" –OPTIONS

And the available option is :

–NO_STRUCT

This command will use the information provided in install.dat to setup the analysis, and prepare lift load cases with appropriate load factors according to API–RP2A. If you only want to determine the equilibrium position using the specified sling lengths and not perform the structural analysis, use the **–NO_STRUCT** option.

The syntax for the automated launch analysis is shown below:

INST_LAUNCH, –OPTIONS

And the available options are:

–**DRAFT**, L_DRAFT1, L_DRAFT2 ...
–**TRIM**, L_TRIM1, L_TRIM2 ...
–**BALLAST**, BAL_SEL
–**AMOUNT**, BAL_AMT
–**CMP_BAL**, CMP_SEL
–**EQUI**
–**FRICTION**, FRICT
–**MAXANGLE**, MAX_ANGLE
–**MAXTIME**, MAX_TIME
–**STOP_SEP**
–**MAXOSC**, MAXOSC
–**WINCH**, WINCH
–**NO_REPORT**
–**NO_STRUCT**
–**FLEXIBLE**
–**NONLINEAR**
–**ALL_POINT**
–**FLX_RIG**

–**AMOD**, L_AMOD

This command assumes that an equal number of drafts have been specified with the –**DRAFT** option and trims have been specified with the –**TRIM** option. It will perform a launch for each draft and trim pair. If no draft and trim are specified, a single launch will be performed with a trim of 3 degrees and a draft so that the tilt pin is at the water surface. The draft specified on this command is measured at midships.

The data expected after the –**BALLAST**, –**AMOUNT**, –**CMP_BAL** and –**EQUI** options are the same as for the **INST_TRANSP** command defined above.

The skidway friction is specified via the –**FRICTION** option. Likewise, the maximum angle of tilt for the primary tilt beam is specified with the –**MAXANGLE** option. Normally a launch will proceed until the maximum time (specified with –**MAXTIME**) is reached or until 5 oscillations of the jacket have been made. However, if the –**STOP_SEP** option is used, the simulation will stop when the jacket separates from the barge. The –**MAXOSC** option is used to specify the number of jacket oscillations allowed after separation before the simulation stops. The initial winch speed of the jacket is specified with –**WINCH**, and the default is 1 foot/second.

If one uses the option, –**NO_REPORT**, then detailed post–processing will not be performed. This macro is designed to perform a simulation and a corresponding structural analysis by default. If the structural analysis is not required, simply use the –**NO_STRUCT** option.

The options –**FLEXIBLE**, –**NONLINEAR**, –**ALL_POINT**, –**FLX_RIG** and –**AMOD** are used to control various aspects of the structural analysis of a jacket launch. The –**AMOD** option specifies the allowable stress modifier for the structural code check, and has a default of 1. The other options control the way the solution is constructed:

- –**FLEXIBLE** Flexibility of the barge is included, pre–tipping and post–tipping load cases use gap elements.
- –**NONLINEAR** Rigid barge assumption, pre–tipping and post–tipping load cases use gap elements.
- –**ALL_POINT** Provides gap elements for post–tipping load cases. Without this option, post–tipping cases use the rocker load as applied loads based on a trapezoidal load distribution.
- –**FLX_RIG** Makes two passes through the structural solver. Before tipping, barge flexibility is included, after tipping, a rigid barge is assumed.

Of course, with any option that provides gap elements, a non–linear structural solution is produced. If none of the above options are used, a rigid barge is assumed, and the reactions between the jacket and barge are applied to the jacket as distributed loads.

To perform an Automated Upend analysis, one uses the **INST_UP** command. Which assumes a typical upending sequence, which includes lifting the jacket to provide the specified minimum bottom clearance, flooding the bottom side legs, and then flooding the top side legs. The flooding is performed with a constant hook height. Two upending simulations are actually performed to determine the proper lifting height needed to obtain the minimum clearance. The syntax of the command is:

INST_UP –OPTIONS

And the available options are:

–**LIFT_INCREMENT**, L_INCREMENT
–**FILL_INCREMENT**, F_INCREMENT
–**VENTS_CLOSED**, C_LEGS
–**MIN_BOTTOM_CLEAR**, MIN_BOT
–**TOP_OF_LEG**, TOP_OF_LEG
–**FIRST_FLOOD**, FF_TANKS, FF_DESC
–**SECOND_FLOOD**, SF_TANKS, SF_DESC
–**DAMAGED_LEG**, DAMAGED_LEG
–**NO_STRUCT**

The options of this command are used to convey the information needed to perform the upend analysis, and the variable names used here are fairly obvious. The variable L_INCREMENT is the lift increment for the lifting stage of the upend, in the present big length units. F_INCREMENT is the flood increment for the flooding stages, in percent. C_LEGS refers to the names of tanks that have their vent valves closed during flooding, and can be a list of tank names, a selection criterion, or a wild character. MIN_BOT specifies the minimum bottom clearance, and TOP_OF_LEG specifies the distance from the waterline to the top of leg in the final installed position. If this option is used, the jacket will be lowered to this location. In this position, the reported hookload would also be the on bottom weight. The **–FIRST_FLOOD** option provides input for the names of the tanks to be flooded first, along with a description of these tanks. Tank names used here would normally use the wild character, as shown:

–**FIRST_FLOOD** B@ Row B Legs

In this example, all tanks beginning with "B" would be flooded, and tank names would normally be defined as B1Leg and B2Leg, for instance. In a similar fashion, the second stage flooding is described using the **–SECOND_FLOOD** option. The **–DAMAGED_LEG** option is used to define the tank assumed to be damaged. With this option, MOSES will return to the original undamaged floating position, and compute a new floating position assuming the specified tank to be open to sea.

This macro is designed to perform a simulation and a corresponding structural analysis by default. If the structural analysis is not required, simply use the **–NO_STRUCT** option.

The final command in this sequence of simulations and structural solutions provides structural post–processing for all the load cases previously created, and has the following syntax:

> **INST_SPOST**, –OPTIONS

And the available options are:

> **–RESIZE**
> **–UP_CLASS**
> **–MEMLOD**
> **–DEFL**

The **–RESIZE** option instructs MOSES to automatically resize any over stressed members in the model. If the **–UP_CLASS** option is used, these changes are saved to the database. The **–MEMLOD** and **–DEFL** options will provide detailed member loads and joint deflections, respectively.

XVI. THE CONNECTOR DESIGN MENU

There are two distinct reasons to analyze a system containing connectors: one needs to assess their effect on the behavior of the system, and one needs to design a connector system to perform a given task. The first is the result of simulating a process. To aid in accomplishing the second, MOSES has several commands which may be invoked from the Connector Design Menu. This menu is entered by issuing the command:

CONN_DESIGN,

from the Main Menu. When one has completed investigating his design, he can return to the main menu by issuing **END_CONN**.

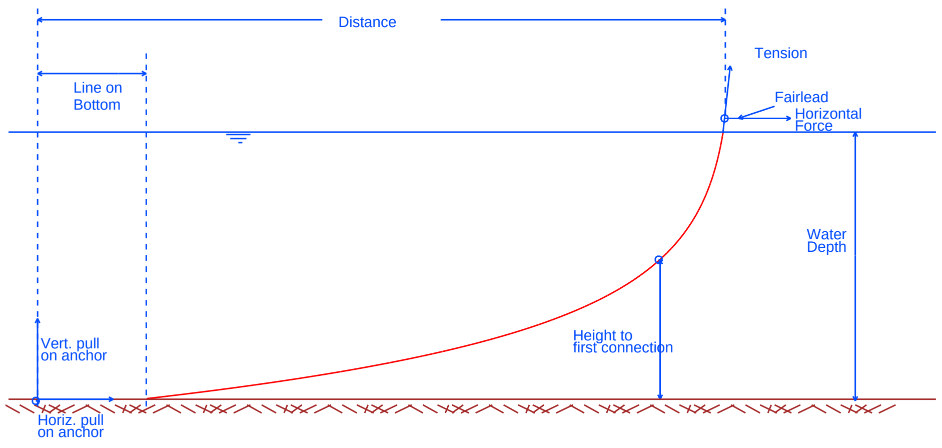
Most of the commands in this menu will place the user in the Disposition Menu at the conclusion of each command. Here, he can dispose of the results as he sees fit. After leaving the Disposition Menu, he is again in Connector Design Menu.

XVI.A Obtaining Connector Tables

The **TABLE** command is used to obtain a list of the force–distance properties of a connector. The form of this command is:

TABLE, LNAME

Here, LNAME is the name of the connector for which force–distance properties are desired, for each connector specified. The results of this command will be a table of distance from anchor, horizontal force and tension at the attachment, the derivative of horizontal force with respect to distance from anchor, the vertical and horizontal pull on the anchor, the active length of the connector, the height of the first connection, and the applied force on this connection. If there is a spring buoy at the first connection, the last two entries will, of course, be the height of the buoy and its displacement. An illustration of some of these quantities is shown in Figure 25.



PROPERTIES OF MOORING LINES

FIGURE 26

XVI.B Obtaining Connector Geometry

The **GEOMETRY** command is used to obtain the current geometry of a connector. The form of this command is:

GEOMETRY, LNAME

Here, LNAME is the name of the connector for which the geometry is desired. The results of this command will be a table of the distance along the connector from "end 2" to the point in question, the horizontal distance from end 2 to the point, and the X, Y, Z, coordinates of the point. After the properties are computed, the user is placed in the Disposition Menu.

XVI.C Finding the Restoring Force

The **MOVE** command is used to obtain the restoring force on a body as a function of either excursion or angle. The form of this command is:

MOVE, BODY_NAME, –OPTIONS

and the available options are:

–**LINE**, TH, DIST, NUMBER
–**ROTATE**, EXCUR, TH_INC, NUMBER

where BODY_NAME is the name of the body to be moved. If this is omitted, the current body will be used. If no options are specified, then the default is the same as –**LINE** with no data.

The option –**LINE** is used to move the body in a line. Here, TH is the direction (deg.) in which the body will be moved, measured from the global X axis positive toward global Y, DIST is the total distance of the move, and NUMBER is the number of positions calculated. Notice that the position increment is the quotient of the distance and the number of positions. If only the body name is specified, then this command will move the body in thirty (30) equal increments in the direction TH from the initial configuration until a termination criteria is met. This criteria is when the most heavily loaded line has a horizontal force equal to the maximum in the table.

The –**ROTATE** option is used to move the body around in a circle. Here, EXCUR is the distance from the initial position (feet or meters) and this will be a constant. TH_INC and NUMBER are the angle increments (deg.) and the number of angles respectively. If these are omitted, 10 degrees and 36 will be used.

At each position, the X and Y components of the restoring force and the magnitude of the restoring force are reported. Also, the tension, the horizontal force, and ratio will be reported for the lines with maximum and minimum tensions. During this process, all other bodies are held fixed in their position in the initial configuration.

XVI.D Obtaining the Results for a Pile

To aid in designing or checking a pile, MOSES has the command:

> **PILE_DESIGN**, PILE_NAME, –OPTIONS

where the options are

> –**FORCE**, FX, FY, FZ, MX, MY, MZ
> –**DISPLACEMENT**, DX, DY, DZ, RX, RY, RZ

When this command is issued, MOSES will take the pile defined by PILE_NAME and apply either a force or a displacement at the top of the pile. A force is specified by the **–FORCE** option where forces and moments are specified in the global coordinates in bforce and bforce–blength. An imposed displacement is specified via the **–DISPLACEMENT** option and is defined in the global system (inches or mm and deg.).

When this command is issued, MOSES will iterate a solution for the pile with the conditions specified. At the conclusion of the computation, the user is placed in the Disposition Menu where he may dispose of the results as he sees fit. When reporting these results, three different reports are available: **LOCATION**, **FORCE**, and **STRESS**.

XVI.E Designing a Lifting Sling

To aid in the design of a lifting sling, MOSES has a special command:

SLING_DESIGN, *NOD(1), ..., *NOD(4), BEGHEI, MAXHEI, NUM

When this command is issued, MOSES will take the body to which the nodes are connected in the initial configuration and compute the lengths of each element of the sling so that the hook point is above the center of gravity. This computation is made for a hook height beginning at BEGHEI for NUM heights (feet or meters), measured from the highest sling attachment point until the height exceeds MAX-HEI. At each height step, the tensions in each sling element are estimated so that the hook load equals the body weight. The height step increment is calculated as (BEGHEI–MAXHEI)/(NUM–1).

At the conclusion of the computation, the user is placed in the Disposition Menu where he may dispose of the results as he sees fit.

XVI.F Obtaining Propulsion/Weather Envelopes

The **PROPULSION** command is used to obtain an envelope of the maximum wind, current, and wave which yields one propulsion unit having maximum thrust. The form of this command is:

PROPULSION, CNAME

Here, CNAME is the name of the control assembly which will be checked. When issued, this command will use the current environment and iterate a the maximum wind, wave and current. These will be done in order so that the results will be the maximum maximum environmental component which can act with the others at their nominal values. For example, suppose that the current environment is defined with

```
&ENV –WIND 100 90 –CURRENT 3.0 45 –SEA ISSC 135 10 7
```

The results of this comand will be a table with four columns labeled Heading, Max Wind, Max Current, and Max Wave. The column labeled Max Wind is the maximum wind speed at 90 degrees acting in concert with a current of 3 at 45 degrees and a sea of significant height of 10 at 135. These are reported as a function of vessel heading.

XVII. THE REPOSITION MENU

To investigate different ways to reposition bodies using connectors, MOSES provides the Reposition Menu. This menu is entered using the command:

REPO

and must be exited using an

END_REPO

command. At this menu level, several commands are available to specify preferences for performing the repositioning, display the selected values, and perform the repositioning. These commands are:

DO_REPO
WEIGHT_CONN, WT, :SC(1), :SC(2), ..., :SC(n)
BOUNDS_CONN, UB, LB, :SC(1), :SC(2), ..., :SC(n)
SELECT_CONN, :SC(1), :SC(2), ..., :SC(n)
DESIRE_VALUE, DES, :SC(1), :SC(2), ..., :SC(n)
TUG_DCHANGE, WFMUL, WDMUL, :TS(1), :TS(2), ..., :TS(n)
SHOW_SYS

The **DO_REPO** command instructs MOSES to find new line lengths and tug forces so that the system will be in equilibrium at the current position, subject to the controls specified with the other commands. This problem will not always have a solution as specified and it is up to the user to check that he finds the results suitable for his purpose.

This command changes only those connectors selected by the latest **SELECT_CONN** command, and will keep the tensions (tug forces) between the upper and lower bounds as specified for each connector using **BOUNDS_CONN**. Here, UB is the upper bound, while LB is the lower bound, both in bforce units. The equilibrium solution keeps the force in each connector as close as possible to the desired value, DES (bforce), defined with **DESIRE_VALUE**. The command **WEIGHT_CONN** is used to specify the relative desirability of changing a given connector. With this command, WT is the relative weight factor to be used for the specified connectors. For all these commands, :SC(n) and :TS(n) refers to a selection criteria containing connector names, or can refer to the connector name itself. The **SHOW_SYS** command will produce a report showing the selection status, current force, desired force, upper and lower bounds, and the weight value for each connector.

Normally, MOSES will change force in tugs instead of changing their direction. The command **TUG_DCHANGE** allows for a tug to change direction as well. Here, WFMUL and WDMUL are the relative desirability factors for changing force and direction, respectively.

XVIII. THE HYDROSTATIC MENU

To perform hydrostatic computation with MOSES, one should issue the command

HSTATICS, BODY_NAME

from the main menu. This command will place the user in the Hydrostatics Menu, where he can compute hydrostatic results for a single body. The body which will be considered will be either the current body, or the one specified on the **HSTATICS** command. In this menu, MOSES can perform four classes of hydrostatic analysis:

- Curves of Form,
- Righting Arm Curves,
- Damage Stability
- Longitudinal Strength, and
- Tank Capacities.

Unless specified, the results obtained will be computed assuming that the water surface is flat. To consider the effect of a wave, one can use the option:

–**WAVE**, WLENGTH, STEEP, CREST

on any of the commands in this menu. Here, WLENGTH is the length of the static wave (feet or meters), STEEP is the reciprocal of the wave steepness, and CREST is the distance of the crest from the vessel origin (feet or meters). Notice that the wave height here is obtained by MOSES as the quotient WLENGTH/STEEP. Once specified, the wave will remain in effect until it is nullified by specifying a –**WAVE** option with zero height.

As far as MOSES is concerned, there is no difference between hydrostatic properties for an intact vessel and a damaged one. There are simply a different set of active compartments for one analysis than for another. As an example, suppose that one had just performed a set of righting arm computations for an intact vessel. To perform a similar analysis for a damaged condition, he would simply issue a **&COMPARTMENT –FLOOD** :TNK command to tell MOSES which tanks or compartments he wished to flood. Next, he would issue the **EQUI** command to find the floating damaged position, and then issue a **RARM** command to examine the stability of the damaged system.

XVIII.A Tank Capacities

To compute tank capacities, the user should issue a command of the form:

> **TANK_CAPACITY**, TNAME, INC, –OPTIONS

Where the options available are:

> –**ROLL**, ROLL_ANGLE
> –**PITCH**, PITCH_ANGLE

When this command is issued, MOSES will compute the weight, volume, and center of the compartment name TNAME in increments of INC (feet or meters). Normally, it is assumed that the vessel has zero roll and pitch. This can be changed, however, with either of the two options. The angles one specifies here are in degrees. At the conclusion of the command, the user is again placed in the Disposition Menu so that he can dispose of the results.

XVIII.B Curves of Form

Perhaps the most primitive hydrostatic results are those which normally comprise the curves of form. To generate this type of result with MOSES, one should issue the **CFORM** command. After the properties have been computed, MOSES will place the user in the Disposition Menu so that he can dispose of the results as he desires. This command produces two hard copy reports in the Disposition Menu: **BASIC** and **COEFFICIENT**. The first of these contains the condition, displacement, center of buoyancy, waterplane area, center of flotation, and the metacentric heights. The second contains the condition, the wetted surface, the load to change draft, and the moment to trim. If no data is specified on the **REPORT** command in the Disposition Menu, then both reports will be printed.

The form of the command for curves of form is:

CFORM, DRAFT, ROLL, TRIM, –OPTIONS

and the available options are:

–**DRAFT**, INC, NUM
–**ROLL**, INC, NUM
–**PITCH**, INC, NUM
–**WAVE**, WLENGTH, STEEP, CREST

Here, the *initial* draft, roll, and trim are specified by DRAFT, ROLL, and TRIM, and the number of conditions and the increment are defined by one of the first three options. If the –**DRAFT** option was specified, then draft will be incremented by INC (feet or meters) NUM times. Similar results are obtained with either –**ROLL** or –**PITCH**, except that here INC is in degrees and roll and/or pitch will be incremented. Finally, the –**WAVE** option controls the static wave{!!li} as discussed previously.

XVIII.C Finding Floating Equilibrium

The method for finding an equilibrium position is fundamental to the manner in which MOSES considers hydrostatics. Instead of using a specified draft and trim to determine the weight, ballast, and cargo for a vessel system, MOSES allows the user to input these weights and ballasts directly via internal commands discussed earlier. After the system has been defined, the question remains as to where the vessel will float. This question can be answered by issuing the following command:

EQUI_H, –OPTIONS

and the available options are:

–**ECHO**, YES/NO
–**FIX**, DOF(1), ..., DOF(N)
–**NUMITER**, ITER_MAX
–**TOLERANCE**, HE, RO, PI
–**WAVE**, WLENGTH, STEEP, CREST

When this command is issued, MOSES will iterate until equilibrium is found. The program simply iterates until the center of buoyancy is above or below the center of gravity, and the buoyancy equals the weight. If one wishes to see the details of the resulting position, he should issue the **&STATUS** command afterward.

The −**ECHO** option controls the trace of iterations which is printed at the terminal. If YES/NO is **YES**, then the trace will be printed, otherwise it will not. The option −**FIX** fixes the specified degrees of freedom during the iteration. Here, DOF(i) can be HEAVE, ROLL, and/or PITCH. The option −**NUMITER** is used to override the default (20) number of iterations, and −**TOLERANCE** is used to override the default closure tolerances for heave, roll, and pitch respectively. The values are a percentage of weight for heave, and arms for the angular motions. The defaults are 0.0001, 0.01, 0.01. Finally, the −**WAVE** option controls the static wave as discussed previously.

XVIII.D Longitudinal Strength

To produce longitudinal strength results, the user should issue the command,

MOMENT, –OPTIONS

where the available options are:

–**WAVE**, WLENGTH, STEEP, CREST
–**ALLOW**, ALLOW_STRESS, ALLOW_DEFLECT

When it is issued, MOSES will include all applied loads, the ballast in all tanks, the defined weight, and the buoyancy in the strength calculations. The values used for the buoyancy are those of the current state, i.e., the buoyancy is computed considering all parts which are active, and the current draft, roll, and trim. The current state can be set with either an **&INSTATE**, **EQUI_H**, or **&EQUI** command.

It is possible that the shear curve will not close, particularly if the vessel is not in static equilibrium. To close the shear curve, one should either issue an **EQUI_H**, **&EQUI**, **&WEIGHT –COMPUTE**, or **&CMP_BAL** command, to establish equilibrium *before* issuing the MOMENT command. If one is interested in longitudinal strength in waves, he should issue **EQUI_H** with the **–WAVE** data before the **MOMENT** command. If not, the vessel will normally not be in equilibrium and the shear curve will not close.

In applying the loads, the default procedure is to include a load even if all of its distribution is not applied within the geometric limits of the vessel. When this occurs, the overhanging part of the load is transferred as an applied load and moment at the vessel end. If the end of the load is forward of the beginning, the end is set to be very close to the beginning. When a load is completely off of the vessel, it is applied as a concentrated load and moment at the end, and again the shear and moment curves will not close. After the results have been computed, the user is placed in the Disposition Menu so that he can dispose of the results generated. The **–ALLOW** option is used to define the allowable deflection and stress in the vessel. Here ALLOW_STRESS is the allowable stress (ksi or mpa), and ALLOW_DEFLECT is the allowable deflection (inches or mm).

XVIII.E Righting and Heeling Arm Curves

To compute righting and heeling arm results, the user should issue a command of the form:

RARM, INC, NUM, –OPTIONS

The options available are:

–**ECHO**, YES/NO
–**FIX**
–**NUMITER**, ITER_MAX
–**TOLERANCE**, HE, RO, PI
–**WAVE**, WLENGTH, STEEP, CREST
–**YAW**, YAW_ANGLE
–**WIND**, WIND_SPEED
–**CEN_LATERAL**, X, Y, Z
–**U_CURRENT**, FLAG
–**W_COEFF**, WC0, WC1, WC2, WC3
–**R_COEFF**, RC0, RC1, RC2, RC3
–**STOP**, HOW
–**WEIGHT**, SF_WEIGHT

The –**ECHO** option controls the trace of iterations which is printed at the terminal. If YES/NO is **YES**, then the trace will be printed, otherwise it will not. The option, –**FIX**, fixes the trim of the vessel during the iterations. The option –**NUMITER** is used to override the default (20) number of iterations, and –**TOLERANCE** is used to override the default closure tolerances for heave, roll, and pitch respectively. The values are a percentage of weight for heave, and arms for the angular motions, and the defaults are 0.0001, 0.01, 0.5. Finally, the –**WAVE** option controls the static wave as discussed earlier.

When the command is invoked, it will rotate the vessel NUM times adding INC to the roll angle. For each increment, the program will iterate an equilibrium position for the other degrees of freedom and then compute the righting and wind heeling arms. Since the righting arm is based on the equilibrium of the buoyancy and weight of the vessel, the vessel weight must have been previously defined either in the model itself, or via an **&WEIGHT** command. *For this command only*, roll is defined as a rotation about an axis which can be changed. The default is, of course, the vessel X axis.

The –**YAW** option is used to compute righting arms about a skewed axis. YAW_ANGLE is the angle of the axis for computing the arms from the vessel X axis. If this option is used, then the axis for the "roll" is yawed to the angle specified. Here, the angle is measured positive from the X axis positive toward Y. If one uses a angle of 90 degrees, the "roll" axis will be moved 90 degrees toward Y and the righting arms will be about the Y axis. In other words, here a "roll" of 2 degrees will make the vessel stern go down.

In addition to computing the righting arms of the vessel, MOSES will compute the wind heeling arms when the **−WIND** option is used. Here, any load attributes which attract wind (#AREA, #PLATE, #TABLE, #TANKER, structural elements, or pieces) will be used with WIND_SPEED (knots) to compute a wind force. A heeling moment is computed from this force and an assumed point of application. There are three alternatives here:

- By default, MOSES assumes that the force is equilibrated by a pressure distribution which has a center of pressure at the vessel center of buoyancy. For certain types of vessels, this assumption may not be applicable.
- This center can be specified using the **−CEN_LATERAL** option. Here, X, Y and Z are the coordinates of the center of lateral resistance in the local body system.
- Finally, the Current Model can be used to compute the assumed application point. To utilize this method, one simply specifies the option **−U_CURRENT**. MOSES will then compute a current force on the vessel which equilibrates the wind force and use this to compute the application point. If FLAG is **INITIAL**, then the application point will be computed at the first condition and the same point used for all other conditions. If FLAG is any other value, MOSES will compute the application point for each heel angle. This probably should be the default, but it requires a proper drag model.

The **−W_COEFF** and **−R_COEFF** options allows one to define a "heeling/restoring moment" which will be added to that computed from the load attributes. Here, WC0, WC1, WC2, and WC3 or RC0, RC1, RC2, and RC3 define that additional moments as:

MW = (WCO + WC1 * H + WC2 * H*H + WC3 * H*H*H) * WIND_SPEED **2
MR = (RCO + RC1 * H + RC2 * H*H + RC3 * H*H*H)

where H is the roll angle in degrees, and MW is in bforce–blength, and MR is in feet or meters.

The **−STOP** option is used to stop the computation after a specified event has occurred. If HOW is **RARM** then it will stop when the righting arm crosses zero. For a value of **NET**, termination will occur at the second intercept point. Finally, **DOWN** will terminate when the minimum NWT down–flooding height becomes negative.

One of the results computed during this process is the minimum height of all down–flooding points on the vessel. The user can define and alter this set of points with **&DESCRIBE** COMPARTMENT commands. If he has not defined a point, then the height of the vessel origin will be reported. The **−WEIGHT** option is used to redefine the scale factor used to convert righting moments into righting arms. By default, the apparent weight of the vessel is used. If the option is exercised, SF_WEIGHT (bforce) will be used.

At the conclusion of the command, the user is again placed in the Disposition Menu so that he can dispose of the results.

XVIII.F Stability Check & Allowable KG

Stability is a simple concept but is extremely complicated in practice. To simplify assessing stability, MOSES has two commands **STAB_OK** and **KG_ALLOW**. What they do is compute several things from the basic results reported by the **RARM** command; and if you ask it to, it will check to see if the value computed is greater than a specified value. These checks are given by the stability criteria of the different regulatory bodies.

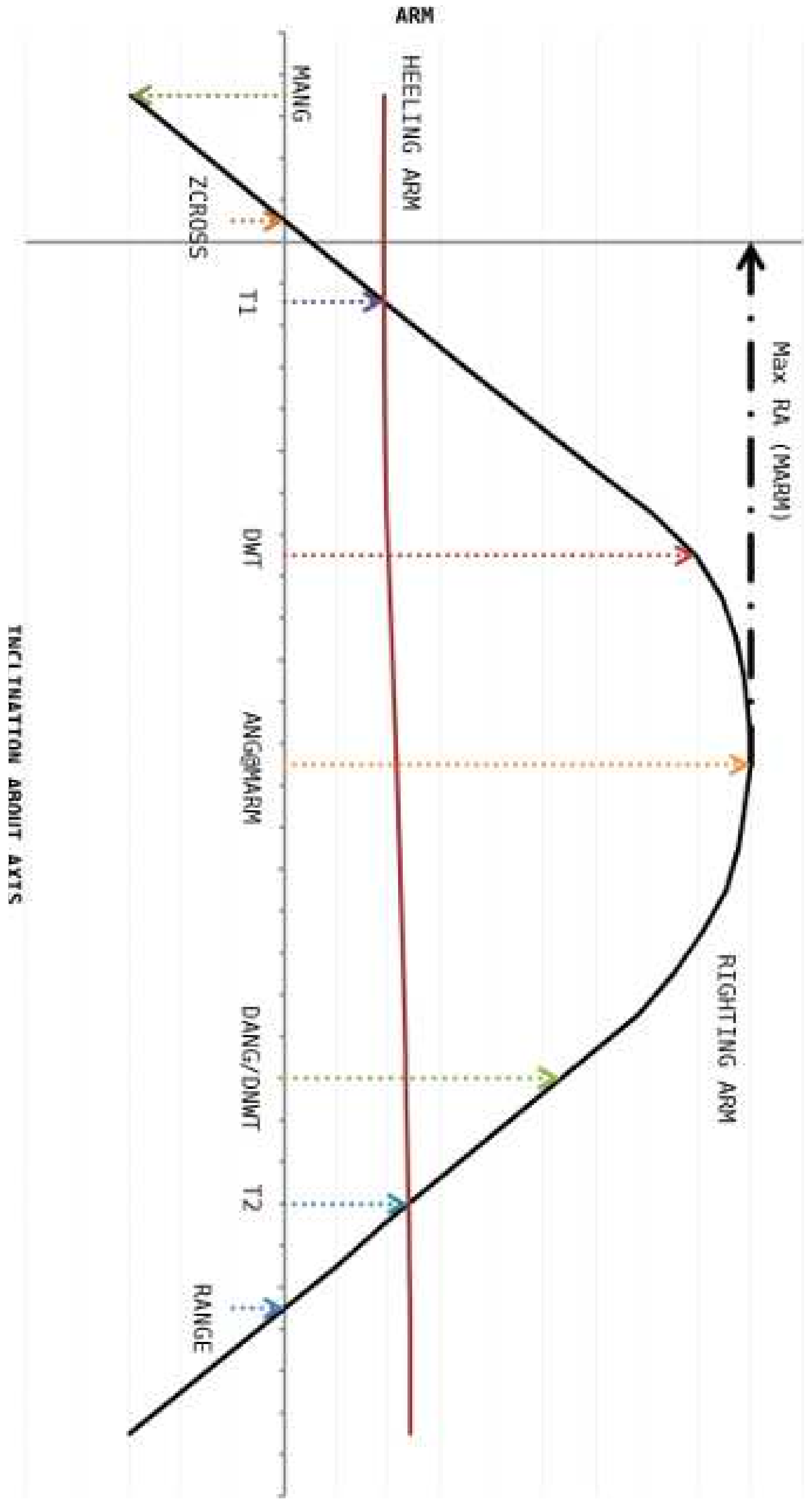
All angles (except the equilibrium angle) defined below are relative to the equilibrium position. First, let us define the symbols:

- R(a) = The righting arm at the angle a.
- H(a) = The wind heeling arm at the angle a.
- RA(a) = The area under the righting arm at the angle a.
- HA(a) = The area under the wind heeling arm at the angle a.
- DWT = The smallest angle at which a WT point goes below the water.
- DNWT = The smallest angle at which a NWT point goes below the water.

The angles are shown in the figure below.

Now, the macros compute:

- GM = The distance from the metacenter to the center of gravity.
- DOWN_H = The down–flooding height at equilibrium.
- ZCROSS = The equilibrium angle without wind.
- THETA1 = The first angle, T, at which R(T) = H(T).
- THETA2 = The second angle, T, at which R(T) = H(T).
- RANGE = (of stability) Is the second angle, T, at which R(T) = 0.
- R_M_EQUI = The RANGE – FACTOR * ZCROSS. Here, FACTOR is specified on the option.
- DANG = DNWT.
- DANG_T1 = DNWT – THETA1.
- DWT_T1 = DWT – THETA1.
- ANG_DIFF = THETA2 – THETA1.
- ANG@MARM = The angle at which the RA peaks.
- AR_RATIO = The ratio of the RA/HA with both measured at the minimum of DANG and THETA2.
- ARR@30 = The ratio of the RA/HA with both measured at the minimum of THETA2, DANG, or 30 degrees.
- AR_RESID = RA(T) – RA(THETA1) – [HA(T) –HA(THETA1)] where T is the smallest of the down flooding angle, or the second intercept.
- RARM@30 = R(30).
- ARM_AR = RA(THETA2).
- ARE@DFLD = min (RA(DANG), RA(THETA2)
- ARE@MARM = RA(M) where M is the angle where R is a maximum.


ARM
HEELING ARM
MANG
ZCROSS
T1
DWT
ANG@MARM
DANG/DNWT
T2
RANGE
Max RA (MARM)
RIGHTING ARM
INCLINATION ABOUT AXIS

- ARE@30 = RA(T) where T is minimum of 30 degrees, DANG, or THETA2.
- ARE@40 = RA(40d), where 40d is the minimum of 40 degrees, DANG, or THETA2. degrees.
- AREBTW = RA(40d) – RA(30), where 40d is the minimum of 40 degrees, DANG, or THETA2.
- ARM_RATIO = R(M)/H(M) where M is the minimum of DANG, THETA2, or the angle of maximum righting arm.
- AR_WRATIO = [RA(TW) – RA(MANG)] / AD, Here the heeling arm is assumed to be a constant HAW = 1.5 * HA(0). Now, TW is the second intercept of the constant wind arm and the righting arm, AD = HAW * [TW + TT], and TT = MANG – T1. Here we use the RA at a positive MANG even though the angle is drawn as negative on the figure. This is equivalent to assuming that RAA is skew symmetric about the origin.

For checking intact stability, the two commands discussed below will accept the following options.

–**I_GM**, IGM
–**I_AR_RATIO**, IARATIO
–**I_RARM@M30**, IRARM@M30
–**I_AR_WRATIO**, IAWRATIO, MANG
–**I_ARM_RATIO**, IARMRAT
–**I_DOWN_H**, I_DOWNH
–**I_ARE@MARM**, IARE@MARM
–**I_ARE@DFLD**, IARE@DFLD
–**I_ARE@30**, IARE@30
–**I_ARE@40**, IARE@40
–**I_AREBTW**, IAREBTW
–**I_ARM_AR**, IARMARE
–**I_AR_RESID**, IARRESID
–**I_ZCROSS**, IZCROSS
–**I_THETA1**, ITHETA1
–**I_RANGE**, IRANGE
–**I_R_M_EQUI**, IRMEQUI, FACTOR
–**I_ANG_DIFF**, IANGDIF
–**I_DANG_T1**, IDANGT1
–**I_DANG**, IDANG
–**I_ANG@MARM**, IANG@MARM

Here, what follows the –I_ is the quantity defined above and the second thing is which will be compared to the computed quantity. Two of the options require a second value. AR_WRATIO needs MANG, the nominal roll due to waves, and R_M_EQUI needs the FACTOR to multiply THETA1. These options will be used to check intact stability. There is a second set of option which begin with –D_ which are used to check damaged stability. For example:

–I_GM, 5, **–D_GM** 1

will demand that the GM exceed 5 for an intact case and 1 for a damaged case.

STAB_OK produces the righting arm, heeling arm, and area ratio curves for the draft specified. Since the righting arm is based on the equilibrium of the buoyancy and weight of the vessel, the vessel weight must have been previously defined. This can be accomplished with the stanza:

```
  $
  $********************************      set transit condition
  $
  &INSTATE –CONDITION 7
  $
 $********************************      compute weight for
condition
  $
  &WEIGHT –COMPUTE  5 32 85 85
```

Now, stability checks with the commands of the form:

```
  $
  $************************************      check one intact
  $
    hystat
    stab_ok 5 2.5 10 –wind 100 –yaw 0
  $
  $************************************      check one damaged
  $
    stab_ok 5 2.5 10 –wind 100 –yaw 0 –damage 5p
```

These two checks are identical, except that the first one checks intact stability for a draft of 5 feet, while the second one checks stability with compartment 5p damaged.

When using STAB_OK (or KG_ALLOW discussed below), the NWT_DOWN points are used to check intact stability, and both the NWT_DOWN and WT_DOWN points are used to check damaged stability. The general form of the command is:

STAB_OK DRAFT RANG_INC NR_ANGLES –OPTIONS

where the options are:

–R_TOLERANCE HE, RO, PI

–**YAW**, Y_ANGLE
–**DAMAGE**, DAM_CMP
–**WIND**, WIND
–**THWAV**, ANGLE_WAVE
–**CEN_LATERAL** XC, YC, ZC
–**U_CURRENT**
–**COEF_WIND**, W_COEF
–**COEF_RARM**, R_COEF
–**WIND_MAC**
–**RARM_MAC**

+ any of the options discussed above.

The variable DRAFT sets the draft for which stability will be checked. When the command is invoked, it will rotate the vessel NR_ANGLES times adding RANG_INC to the roll angle. For each increment, the program will iterate an equilibrium position for the other degrees of freedom and then compute the righting and wind heeling arms.

The –**R_TOLERANCE** option is the same as the one on the **RARM** command. The values specified here will be passed to **RARM** whenever it is called. Likewise the –**YAW** is analogous to the same option on the **RARM** command and is used to compute righting arms about a skewed axis. The –**DAMAGE** option is used to select tanks that will be damaged for damaged stability. If one does not wish to check damaged stability then this option should not be used.

The next set of options control the computation of wind heeling. The –**WIND** option is used to define the wind which will be considered. The –**CEN_LATERAL**, –**U_CURRENT**, –**COEF_WIND** and –**COEF_RARM** options simply pass their data directly to the RARM command, so that their data is the same as that for the option to the RARM command of the same name.

To include the roll owing to wave action, a wave angle can be included with –**THWAV**. The righting and wind arm calculations will begin at the angle ANGLE_WAVE to windward.

Alternatively, the –**WIND_MAC** option will call the macro

COEF_SET yaw draft

and the –**RARM_MAC** option will call the macro

COEF_RSET yaw draft

immediately before each invocation of RARM. If you use this option, then you must write the macro, COEF_SET or COEF_RSET. It takes the two arguments and set a variable. An example is:

```
&MACRO COEF_SET YAW DRAFT
  &SET COE_WIN = –W_COEF A B C D
&ENDMACRO
&M_ACT COEF_SET RARM

&MACRO COEF_RSET YAW DRAFT
  &SET COE_WIN = –R_COEF A B C D
&ENDMACRO
&M_ACT COEF_RSET RARM
```

One can add any logical he wishes here to change the coefficients based on draft and yaw. The values A, B, and C are numbers depending upon the situation.

By default, righting arms are computed about the equilibrium position which is computed before starting. If the **–NO_EQUI** option is used, the basic position is used. Notice that the GM may not be defined when using this option.

The results of this command are plots of the righting arm, heeling arm, area ratio curves, and two reports. The first report is the standard stability report. The second presents: the condition, the allowables, and the results for each stability criterion along with a statement of "PASS" or "FAIL".

We apologize for the complexity here, but we tried to make these commands applicable to as many rules as possible. Only the checks which are specified will be checked and reported.

We compute the maximum allowable KG for a set of drafts, intact wind speed, and a damage wind speed with the command:

KG_ALLOW

where the options are:

–WIND, I_WIND, D_WIND
–YAW, Y_ANGLE(1), ...
–DAMAGE, DAM_CMP(1), ...
–DRAFTS, D1, D2, ...
–KG_TOL, KG_TOL
–KG_MIN, KG_MIN
–KG_MAX, KG_MAX
–CEN_LATERAL XC, YC, ZC
–U_CURRENT
–COEF_WIND, W_COEF
–WIND_MAC

+ any of the options discussed above.

The options here are "the same as" those for the **STAB_OK** command, *except* that one should specify wind speeds for both intact and damaged cases with –**WIND** and one can specify more that one thing with –**YAW** and –**DAMAGE**. The only new options are –**KG_TOL** which define the tolerance (feet or meters) for the computation of the allowable KG, –**DRAFTS** which defines the drafts to be considered, –**KG_MIN**, and –**KG_MAX**. The last two of these are used in setting the limits which will be searched and normally should not be needed. **KG_MIN** has a default of 0 and thus it is assumed the vessel has been coded up according to the documentation or that if the KG is at the keel, the vessel will pass the stability requirements. If you get a message that "LOWER BOUND FAILS", then you need to use this option with some negative KG so that the message goes away. The –**KG_MAX** value defaults a value which yields zero GM. If you have a partial run which establishes an upper bound on the allowable KG, you can use it here to minimize the computational effort.

For each draft specified, the command will find an "allowable KG" for the set of damages and yaw angles specified. By allowable, we mean that any KG greater than (to within KG_TOL) that found will fail one of the stability requirements for some damage and yaw angle. Basically, this command simply incorporates an iterative algorithm and repeatedly calls **STAB_OK** to find the allowable. The command uses the following search technique:

- It first sets a lower bound, KL, to MIN_KG and checks to make sure that this passes all cases (intact and all damages for all yaws).
- It then sets the upper bound, KU, to about where the GM is MIN_GM or that specified with –**MAX_KG**.
- An estimate of the KG, KC, is obtained as a*KL+b*KU, and each cases is considered; i.e. for each intact and damaged condition, each yaw angle is considered until one fails or all pass. Here, a and b are chosen based on the number of cases to be considered (number of damage compartments + 1 times the number of yaw angles). The reason for doing this is that it costs much more to check a condition that passes than it does to check one that fails! Thus, the coefficients are chosen to minimize the total cost of the search.
- If all pass, the KL is replaced by KC and the above process is repeated.
- If one fails, KU is replaced by KC and the process is repeated.
- This continues until KU – KL is less than TOL and the allowable is taken to be KL.
- All of the above was done with no reporting. After an allowable KG has been found, all of the cases are again considered and a report of the stability are printed.

If more than one draft was specified, a plot of the allowable KG vs draft will be made. Since the algorithm favors failure, it is much more efficient if you order the data "properly". In particular, you should input the damage cases in order of most likely failure. The same can be said of the yaw angles.

XIX. THE HYDRODYNAMIC MENU

The Hydrodynamics Menu contains commands which allow the user to build, alter, and maintain databases of hydrodynamic properties which act on body panels. The Hydrodynamic Database provides MOSES with the ingredients used in almost all types of analysis. This database can be input directly in this menu, or it can be computed from the model. The menu is entered with

HYDRODYNAMICS

and when one has completed his task here, he exits with

END_HYDRODYNAMICS

In order to fully understand the implications of the pressure database, it helps to have some information about what the program computes and how it uses the intermediate results. Basically, the primitive quantity is a set of velocity potentials on each panel which results from the interaction of the panel with the sea. The first six of these velocity potentials arise due to unit motion of the body, at a given frequency, in each of the degrees of freedom. These are called radiation velocity potentials. The remaining potentials result from a wave being stopped by the body. These are called diffraction potentials. The diffraction potentials depend not only on wave frequency but also on wave heading. All of the potentials are complex numbers (a real and an imaginary part for each potential). Collectively, we will refer to these potentials as "diffraction results" since they are normally computed via some type of diffraction analysis. The forces on Morison's Equation elements are not a part of the pressure database, are computed whenever they are needed, and are not considered in this Menu. In addition to the diffraction potentials, the "incident wave potentials" are also required, but since they are easy to compute, they are not stored in the pressure database. Also, for some computations, MOSES needs to have not only the potentials, but their gradient, so they are included in the database.

From these basic ingredients, MOSES then computes a new, or "*Total*" database which includes:

- The forces which unit amplitude regular waves exert on a stationary body,
- The bodies' added mass and damping matrices.
- The mean drift force which a regular wave exerts on a stationary body,
- The change in mean drift force with a motion of the body, and
- The damping on a body due to time variation of the mean drift force.

These results are a function of wave heading, forward speed, and encounter frequency.

The Hydrodynamic database actually consists of two different types of data for each body: Pressure data and Mean Drift Force Data. Also, the data is stored by

"Packet Name". Thus it is possible to have several different sets of hydrodynamic data available for each body. For example, one can have different sets for different draft and trim conditions, or different sets computed with different methods. A packet of data is associated with body whenever the packet is: *Generated*, or *Imported*. In general, the user is free to define the packet name for the data when it is created, *but* if the name already exists, then a new name will be created. The same name can exist for each of the different types of data, but all names for a given type of data must be unique. A special reserved packet name is **NONE**. If this name is associated with a body for a given type of data, then it has the same effect as null data.

The data in the Hydrodynamic database is used in both frequency and time domain computations. Since the database consists of frequency domain quantities, its use in the frequency domain is easily inferred. For a time domain simulation, use of this data is not obvious. Here, three things happen. First, an excitation force is created as a cosine series of the frequency domain forces. The periods for the series are those specified with the **–S_PERIOD** option on the &ENV command. The amplitudes are chosen to conform to the specified sea spectrum, and a set of phases are chosen. Next, a mean drift force is computed from the drift force "RAOs" and the sea spectrum. A time varying drift force is created as a cosine series at periods specified with the **–MD_PERIOD** options of the &ENV command and amplitudes which conform to the drift spectrum and a set of phases. Finally, the frequency domain added mass and damping matrices are transformed by an inverse Fourier transform into a convolution kernel, or "retardation function", and the equations of motion are integro–differential equations. Thus, any time domain simulation in a seaway first requires the basic data discussed above.

The basic method of computing mean drift force from the hydrodynamic pressures is to integrate the results over each submerged panel. If, however, one has used strip theory, then this results in zero surge drift force (the surge diffraction potential is ignored). In this case, a representation by Salvesen which employs an assumption of the body being a "weak scatterer" is used to estimate a surge component.

There are basically two types of data considered in this menu: pressure and mean drift. Each of these will be considered in detail later, but as a general rule commands which generate data begin with **G_**, those which post–process with **V_**, those which import with **I_**, and those which export with **E_**. The commands for importing hydrodynamic data are designed to allow the user to completely describe a hydrodynamic data base. Although the commands for doing this are documented in the following sections, the user is encouraged to examine the samples of data provided with this software release. To input your own data, it is helpful to first export a hydrodynamic data base to get an understanding of the MOSES file format. Then, modify this file as desired, and import it to the program.

XIX.A Pressure Data

To compute the sea pressures on the vessel, the program must know the form of the vessel below the water. This is communicated to MOSES by a set of vessel description data, defined earlier, and the current condition of the body. To initiate the pressure computations discussed above, one must issue:

G_PRESSURE, BODY_NAME, PKT_NAME –OPTIONS

and the available options are:

–**HEADING**, H(1), H(2), ..., H(n)
–**PERIOD**, T(1), T(2), ..., T(n)
–**MAX_DIST**, DIST

When this command is issued, MOSES will take the system in its current configuration and compute frequency domain pressures and a total hydrodynamic database for the body BODY_NAME. This data will be stored by the name PKT_NAME in the database.

The –**HEADING** option is used to change the default values of vessel heading for which results will be computed. H(i) is the ith value of heading (deg.) to be considered. Here, heading is measured as an angle from the X axis, positive toward Y. Hence, for a vessel described with the origin at the bow, head seas are 180 degrees. The –**PERIOD** option is used to alter the default values of encounter period, and T(i) is the ith value of encounter period in seconds. As many of these options as needed can be input, so long as the number of periods does not exceed 200. The defaults for heading and period are set with options on **&DEFAULT**.

When the **G_PRESSURE** command is issued, MOSES takes the vessel description and the condition defined, and converts the vessel description into one describing the vessel below the waterplane. The program will then compute the added inertia and damping matrices, and the applied forces on the vessel. These computations will be performed for the periods and headings defined by the command. In converting the model, the two options –**M_DISTANCE** and/or –**M_WLFRACTION** of the **&PARAMETER** command are used to refine the computation. Use of these options allows one to define a quite crude mesh and have MOSES automatically refine it to achieve any desired degree of precision. Also, the option –**MAX_DIST** provides a way to get approximate solutions to large diffraction problems with reduced computational effort. This option defines a maximum distance (feet or meters) for panel interaction. Any two panels which have a distance between them greater than DIST will have a zero for their coupling terms in the diffraction matrix. For very long slender bodies, this option can be used quite effectively to save computer effort. For a body 4000 ft long with 6000 diffraction panels, answers within 10% of the exact ones can be obtained in

66% of the time.

Once one has a pressure database, it can be examined in the Disposition Menu. In particular, the command

V_MATRICES, BODY_NAME

will gather the added mass and damping matrices for the pressure packet currently associated with body BODY_NAME and place the user in the Disposition Menu. Alternately, the command

V_EXFORCES, BODY_NAME

will do the same thing with the wave excitation forces. For both of these commands, the results are about the origin.

As mentioned earlier, it is possible to define the hydrodynamic pressure distribution and/or the total hydrodynamic database via commands. The same device is used if one has computed hydrodynamic pressures and wished to "save" them for use later. This capability can be quite useful when analyzing a large diffraction model. For instance, one could alter the mooring lines of system, and use the importation feature to save time in the reanalysis. Caution should be used here, so that only changes to the model that do not effect hydrodynamic properties are performed. To emit a hydrodynamic database, one issues:

E_PRESSURE, BODY_NAME –OPTION

where BODY_NAME is the name of the body for which hydrodynamic databases will be emitted. This command will export a pressure database for the packets currently associated with the body BODY_NAME. Here, the only option is –**NOTE** which is just like the same option on model definition commands. In particular, you need to specify all five characters of the option, and the note that you attach will be included as a comment in the emitted file. Also, the title and subtitle will be included if they are not blank. No drift or total data will be emitted since it is recomputed when the pressure data is imported. It is possible to export (and subsequently import) a total hydrodynamic database. This is accomplished with:

E_TOTAL, BODY_NAME –OPTION

The resulting output is substantially smaller than the pressure database. Here the options and titles are the same as for the pressure database. This file, however, consists data for both Total force and drift force. For simulation purposes, this is all that is necessary, but it cannot be used for computing structural loads. When such a database is input, a *single* warning to this effect will be given.

The format of an exported pressure database is the same as the one used to define the data directly to MOSES. It begins with the command:

I_PRESSURE, BODY_NAME, PKT_NAME, DISPL, –OPTIONS

which places the user in a submenu. Here, BODY_NAME is the name of the body for which the database is being generated, PKT_NAME is a desired packet name, DISPL is the displacement at the condition being defined, and the options are:

–**PERIOD**, T(1), T(2), ...
–**HEADING**, H(1), H(2), ...
–**CONDITION**, DRAFT, ROLL, PITCH

Even though these items are called options, the first three of them are necessary to properly define the database. The **–PERIOD** option defines the periods (sec) for which the database will be defined, and **–HEADING** defines the headings (deg) for which the exciting forces will be defined. The **–CONDITION** option defines the vessel condition for which the database is defined and DRAFT, ROLL, and PITCH are the draft (feet or meters), roll (deg), and pitch (deg) defining the condition. The remaining options were described previously.

Once the menu has been entered, several commands are available. First, the command:

FP_MAP PANEL_NAME, :PNT_SEL(1), :PNT_SEL(2), ...

defines a how the structural loads on the panel PANEL_NAME will be mapped to all points matching :PNT_SEL(i). The command:

FPANEL PANEL_NAME, AREA, XC, YC, ZC, NX, NY, NZ, WLLEN

defines a panel. Here, PANEL_NAME is the name of the panel, XC, YC, and ZC are the coordinates (feet or meters) of its centroid, NX, NY, and NZ are the components of its normal, and WLLEN is the length of the intersection of the panel with the waterline (feet or meters).

After a panel has been defined, the pressures acting on it are defined through a set of velocity potentials with commands:

FPPHI PER, RPRX, IPRX, RPRY, IPRY, ... \
        RPRRZ, IPRRZ, RPDH(1), IPDH(1), ...

Here, PER is one of the periods T(i), and the remainder of the data are velocity potentials per unit wave amplitude (feet or meters). These commands *must* be in decreasing order of period. In other words, the value of PER for a given command must be less than the value of PER for the previous one and greater than PER for the next one. The velocity potentials are pairs of real and imaginary numbers. The first six pair (twelve numbers) are the radiation potentials, and the remainder are diffraction potentials. The diffraction potentials correspond to the headings H(i), and are in the same order.

The final type of data for a panel is defined with:

> **FDELP** PER, RPRXX, IPRXX, RPRXY, IPRXY, ... \
> RPRRZZ, IPRRZZ, RPDHX(1), IPDHX(1), ...

These quantities are the gradients of the potentials defined with the **FPPHI** command, and thus there will be 3 times as many values as on the **FPPHI** command. If these are viewed as complex numbers, then the first three numbers are the derivatives with respect to X, Y, and Z of the first potential on **FPPHI**. The first 36 values (18 complex numbers) are gradients of the radiation potentials. The gradients of the diffraction potentials follow for each heading.

After all of the panels have been defined, the menu is exited with an **END_I_PRESSURE** command.

To define a "total hydrodynamic database" one first issues:

> **I_TOTAL**, BODY_NAME, PKT_NAME, DISPL, –OPTIONS

One can then describe the hydrodynamic data, and issue **END_I_TOTAL** to exit the menu. Here, BODY_NAME is the name of the body for which the database is being generated, DISPL is the displacement at the condition being defined, and the options are:

> –**PERIOD**, T(1), T(2), ...
> –**HEADING**, H(1), H(2), ...
> –**CONDITION**, DRAFT, ROLL, PITCH
> –**SCFACT**, SCLEN, SCMASS, SCDRAG, SCFOR

Even though these items are called options, the first three of them are necessary to properly define the database. The –**PERIOD** option defines the periods (sec), for which the database will be defined, and –**HEADING** defines the headings (deg), for which the exciting forces will be defined. The –**CONDITION** option defines the vessel condition for which the database is defined and DRAFT, ROLL, and PITCH are the draft (feet or meters), roll (deg), and pitch (deg) defining the condition.

The option –**SCFACT** defines a set of scale factors which can be used to convert to the units required. In other words, all of the quantities input via the commands discussed below will be multiplied by the scale factors prior to being stored in the database. The manner in which these factors will be combined with the input numbers will be discussed with each command.

After the **I_TOTAL** command, the database is defined by a sequence of the following commands:

> **H_ORIGIN**, OX, OY, OZ
> **H_EULERA**, EROLL, EPITCH, EYAW
> **H_PERIOD**, T
> **H_AMASS**, AM(1,1), AM(2,1), ..., AM(6,6)
> **H_DAMP**, DAMP(1,1), DAMP(2,1), ..., DAMP(6,6)
> **H_FORCE**, H, RFKX, RFKY, ... RFKYAW, IFKX, ..., IFKYAW \
> RDIX, RDIY, ... RDIYAW, IDIX, ..., IDIYAW \

The **H_ORIGIN** and **H_EULERA** commands define a change of coordinate system from the one being input to the one employed by MOSES. Here, OX, OY, and OZ are the components of a vector, in the MOSES body system from the origin in the local body system to the origin of the system in which the input quantities are computed. Likewise, the quantities EROLL, EPITCH, and EYAW are three Euler angles (deg). These angles, when applied as a yaw followed by a pitch, followed by a roll, define the direction cosine matrix which transforms the system in which the quantities are computed to the local body system. If either of these two commands is omitted, the corresponding transformation will be assumed to be the identity. Once a transformation has been defined, it will be used until it is redefined by another similar command.

The **H_PERIOD** command defines the period for all quantities which follow until a new **H_PERIOD** command is encountered. Here, T is the period (sec) and it must have been defined by the –**PERIOD** option on the **I_TOTAL** command. If the period defined by an **H_PERIOD** command is the same as one previously used, the data following will be added to the previous data for the same period. Thus, one can define the properties of a complicated body by inputting the properties for each piece of the body and letting MOSES combine them to form the properties of the body.

The remaining commands are used to actually define the hydrodynamic properties for the "current" period, and they can be repeated as many times as desired until an **END** is encountered. This marks the end of the hydrodynamic database definition for a given body.

The **H_AMASS** and **H_DAMP** commands define the added mass and linear damping matrices respectively. The data is input on the command by columns of the matrix. The values which MOSES needs for the added mass matrix are added mass divided by displaced mass, and the length units should be feet or meters. When an **H_AMASS** command is input, the top 3x3 is multiplied by SCMASS, the two coupling 3x3 matrices are multiplied by SCMASS*SCMASS, and the bottom 3x3 is multiplied by SCMASS*SCLEN*SCLEN, as defined with the –**SCFACT** option. It is the product of the input values times the scale factors which should have the dimensions defined above. The scaling for the damping matrix is similar to that for the added mass except that SCDRAG is used instead of SCMASS. The desired units for the product of the input values and the scale factors is damping coefficient divided by displaced mass.

The **H_FORCE** command defines the wave exciting force for the heading, H, which has been defined by the **–HEADING** option on the **I_TOTAL** command. The first twelve values define the Froude–Krylov force, the next twelve define the diffraction force. The first three values for each force will be scaled by SCFOR, and the last three will be scaled by SCFOR*SCLEN, as defined on the **–SCFACT** option. The product of the input values times the scale factors should be either bforce, and feet or meters, depending upon the last **&DIMEN** command.

XIX.B Mean Drift Data

In most cases, the mean drift database is created as a consequence of creating the pressure database. This is simply the average of the nonlinear forces over time. In general, there are three contributions to this force: diffraction/incident potentials, radiation potentials, and the coriolis acceleration. The mean force due to diffraction/incident potentials is independent of the motion while the other two depend on the motion. Here, the last RAOs computed for the current process will be used to compute theses contributions. The diffraction/incident and the radiation mean forces are also used to apply a slowly varying force in the time domain as described earlier. The coriolis acceleration contribution to the mean is not used in the time domain. Instead, it is computed exactly.

One can examine the drift data with the command

V_MDRIFT, BODY_NAME

which will place one in the Disposition Menu to do whatever one wishes. Simple estimates of the mean drift force can be created with the command

G_MDRIFT, BODY_NAME, PKT_NAME –OPTIONS

and the available options are:

–**MD_TYPE**, DTYPE
–**DIMENSIONS**, LENGTH, BEAM, DRAFT
–**HEADING**, H(1), H(2), ..., H(n)
–**PERIOD**, T(1), T(2), ..., T(n)

Here, DTYPE must be either **FORMULAE**, or **SEMI**. The –**DIMENSIONS** option defines the size (feet or meters) of the body used in estimating the drift. If it is omitted, the actual body dimensions will be used. The wave drift data computed here is not sophisticated, and only mean drift is considered.

One can save a set of drift data for later use by issuing the command:

E_MDRIFT, BODY_NAME

which writes the drift data currently associated with body BODY_NAME to a file for later use.

The user can input his own drift database. To define mean wave drift response operators, one first enters a submenu with the command:

```
I_MDRIFT, BODY_NAME,  PKT_NAME –OPTIONS
```

and the available options are:

```
–HEADING, H(1), H(2), ...., H(n)
–PERIOD, T(1), T(2), ...., T(n)
```

Here, PKT_NAME is the name of the set of response operators, the **–PERIOD** option defines the periods (sec) and the **–HEADING** option defines the headings (deg) for which drift values will be defined. Once in the menu, the components of the mean wave drift force for the specified headings are defined with the command:

```
M_DRIFT, PER,  FXR(1), FXI(1), ...., FYAWI(1), ...  \
           FXR(n), FXI(n), ..., FYAWI(n)
```

Here PER is the period for which this set of forces is applicable and must be one of the T(i) specified with the **–PERIOD** option. FXR(1), FXI(1), .. FYAWI(1) are the real and imaginary parts of the mean drift forces and moments per unit of wave amplitude squared (bforce/blength**2 for force, bforce/blength for moments) for the heading HED(1). Likewise, the values FXR(2) ... are for heading HED(2), etc. The corrections to the mean force due to motions are a complex 6x6 matrix per wave amplitude**2 defined with the command:

```
MD_MOTION, PER,  HED, MDR(1,1), MDI(1,1), ... MDI(6,6)
```

the units here are feet or meters, bforce, seconds, and radians. When all of the data has been defined, the menu should be exited with a **END_M_DRIFT** command.

XX. THE FREQUENCY RESPONSE MENU

Frequency response is a linear approximation to the equations of motion with assumed harmonic input. In MOSES, a menu is devoted to computing frequency response and its post–processing. To enter the menu, one inputs:

FREQ_RESPONSE

When completed, the menu is exited with **END_FREQ_RESPONSE**.

The traditional way of obtaining frequency response is to consider a set of unit amplitude waves and to linearize the equations of motion for each wave. This is exactly what is done with the **RAO** command. Results obtained in this way can later be easily combined with different spectra to obtain approximations to the extremes in many different situations, and are thus quite popular. The major disadvantage of this approach is that one can only look at the response to wave frequency excitation. In other words, with the response operator approach, one cannot investigate the effect of wind or slow drift wave excitation. To cope with these other effects, one can no longer look at unit amplitude waves, but must consider the simultaneous effect of all environmental forces. This is what is done with the **SRESPONSE** command. The major disadvantage with spectral response is that this response is applicable to a *single* environment and thus the post processing options are limited.

Before getting into the details of these commands, a few general words are in order. The frequency response calculations depend not only on the sea pressures, but also on the stiffness of the connectors and the mass properties of the system. Instead of requiring the user to directly specify the weight and radii of gyration of the system, MOSES uses those computed from the element and weight data supplied in the structural model, and from currently active tanks, weights, etc. This is necessary to insure that when structural loads are computed, there will not be seriously out of balance in the loads. It is therefore *important* to accurately model the weights not only from a local sense, but also in a global one. Another important thing to remember is that the "response operators" produced with the **RAO** command are not really independent of wave height. For most cases, nonlinear damping (and perhaps forcing) is important. Thus, some *real* wave amplitude must be used to linearize the system. If post processing results are obtained for an environment which is radically different from that used in the linearization, then the applicability of the results is open to question.

MOSES provides two ways to linearize the equations for RAO computations: specified wave amplitudes, and a spectral linearization. With the specified wave height method, one specifies a steepness, a period, and a wave height. This is the default method. MOSES will for each period and heading, use a constant wave steepness to obtain a "real" wave amplitude for linearization for periods less than the specified period. For larger periods, the specified height will be used. The energy from the linear system and the real system over a period are set equal to obtain the linearized results. For damping, this appears to be a rational approach. When, however, one looks at the forcing from a Morison's equation element, it does not appear as attractive. Here one is saying that the linearized coefficient depends only on the velocity for a single Fourier coefficient. The alternative is a "spectral" linearization. Here, it is assumed that the seas are unidirectional so that the RAOs for a set of periods and given direction are linearized at one time. The spectrum is used to compute the RMS of the relative velocity at a point, and this is used to compute the "drag" coefficient. The two approaches yield somewhat different results depending on the difference in the peaks of the response and the spectrum. For **SRESPONSE**, the spectral method is always used. However, the linearization here is not only over periods, but also over headings.

In this light, it can be seen that frequency domain forces are no longer for zero speed, and the interpretation of wave heading is different than before. That is, the frequency domain forces are those used to compute the response. Sometimes these are force/wave amplitude results (RAOs), and sometimes these are Fourier coefficients, depending on the use of **SRESPONSE**.

Response operators are computed by simply issuing the command:

RAO, –OPTIONS

where the available options are:

–**HEADING**, H(1), H(2), ..., H(N)
–**PERIOD**, T(1), T(2), ..., T(N)
–**SPEED**, VR
–**ITER**, MAXIT
–**SPECTRUM**, ENV_NAME
–**STEEP**, ST, PBCHEI, CHEI
–**ROD_STEEP**, ST, PBCHEI, CHEI

When this command is issued, MOSES will compute a new set of response operators, for the same frequencies and headings as those of the basic pressure data (those specified on the **G_PRESSURE** command), unless the options –**HEADING** or –**PERIOD** are exercised. One can obtain response operators for any *one* speed by including the option –**SPEED**, where VR is the speed desired in knots.

If either –**PERIOD** or –**HEADING** are used, MOSES will interpolate values of the hydrodynamic forces, added mass, and damping from the values contained in the pressure database. If no data for a quadrant exists in the pressure database, then MOSES will assume symmetry about the vessel centerline or about amidships when interpolating hydrodynamic results.

The remaining options describe how MOSES deals with viscous damping. In general, there are three types of viscous damping which are considered: empirical roll damping, Morison's drag for bodies, and Morison's drag for rod elements. Normally, MOSES will iterate up to thirty times to achieve a proper solution considering the viscous damping. In some cases where there are many bodies connected by rigid connectors, there is a substantial computational effort involved with this iterative solution. The **–ITER** option may be used to limit the number of iterations. Here, MAXIT is the maximum number of iterative steps which will be taken, where 30 is the default value.

The **–SPECTRUM** option specifies the wave spectrum of the environment, ENV_NAME, which will be used to linearize the equations spectrally. Here, any nonlinear dependence will be replaced by the RMS value times SMULT, where SMULT can be specified with the **–SPE_MULTIPLIER** option of an **&DESCRIBE** BODY command. If **–SPECTRUM** is omitted, then an equivalent linearization will be performed, where the drag is linearized by either constant wave steepness or constant wave height. The default is to use constant wave steepness for roll damping and Morison's drag on bodies and constant wave height for drag on rod elements. These assumptions can be altered by using the **–STEEP** option for roll damping and body drag, and **–ROD_STEEP** for rod drag. When either of these options is used, the wave steepness will be held constant at 1/ST for periods less than PBCHEI seconds and a constant wave height of CHEI (feet or meters) will be used for larger periods. If neither PBCHEI nor CHEI are specified, then constant steepness will be used for all periods.

During the RAO computation, the dependence of drag coefficient with Reynolds Number is *not* considered. Instead, the drag coefficient corresponding to the value specified with the **–F_CD_TUBE** option of **&DEFAULT** command is used. During the computation of nonlinear damping, the computed value is multiplied by a factor to obtain that which is used in the computation. For a rod, a drag multiplier can be defined with **&DEFAULT –FM_ROD**. For bodies, the **–FM_MORISON** option of **&DESCRIBE BODY** is used to define the multiplier.

Spectral frequency response is computed by issuing the command:

SRESPONSE, ENV_NAME, –OPTIONS

where the available options are:

–PERIOD, T(1), T(2), ..., T(N)
–HEADING, H(1), H(2), ..., H(N)
–FIX_TEN, YES/NO

–ITER, MAXIT

Here, ENV_NAME is the name of the environment which will be used to compute the response, and the options operate in the same manner as with the **RAO** command. If ENV_NAME is omitted, then the current environment will be used. Here, one will definitely want to specify the **–PERIOD** option since the objective is to investigate the effect of non wave excitation frequencies. When this command is issued, MOSES will take the environment and expand the direct wave excitation, the wave drift force, and the wind force in a Fourier series of the periods specified with **–PERIOD**. The direct wave force and drift force series will also be expanded with headings specified with **–HEADING**. The wind series will be applied to the heading closest to the specified wind heading. These series of forces will be used to obtain the Fourier coefficients of the response series. This is precisely what is called spectral response. The option **–FIX_TEN** can be used to "fix the tensions". Since the connector tensions are normally a nonlinear function of the motions, the linearization necessary in the frequency domain will not be nearly as effective for predicting the connector forces as it will be for predicting the motions. "Fixing the tensions" is a partial remedy for this problem. If the option is used with a value of YES/NO of **YES**, then MOSES will compute the value of the tension at the maximum position and compute a ratio of this maximum to that predicted by the force response. The force response is then scaled by the ratio so that the predicted maximum will be that computed for the extreme position.

After obtaining frequency response results, one can examine them with other commands in this Menu. Some of the commands produce response operators, others produce statistics for irregular seas, and others produce equation force data. At the conclusion of most of these commands, the user is placed in the Disposition Menu where he is given the option of reporting, viewing, or graphing the results of the command.

The behavior of the commands in this menu differ with the type of frequency response data to be examined. With response operators, (obtained with an **RAO** command) one *must* specify an environment to obtain statistical or time synthesis results, while for nonlinear spectral results (obtained with a **SRESPONSE** command), an environment *cannot* be specified. With RAOs, the commands in this menu which deal with sea–states have a final syntax which is identical to that of the **&ENV** command. With nonlinear spectral results, the environment data must be omitted. In other words, with spectral results, no environmental data will be allowed. With RAOs, these commands not only initiate the computation of quantities in an irregular sea, but are also **&ENV** commands. Thus, when one issues one of these commands with a non–blank ENV_NAME, he is altering the definition of this environment within the database. If ENV_NAME is omitted, then the environment used will be totally defined by the options specified.

To produce a time domain process from the frequency response and and an environment, one should issue:

FR_2TIME, ENV_NAME, –OPTIONS

where the available options are:

–**SEA**, SEA_NAME, THET, HS, PERIOD, GAMMA
–**SP_TYPE**, TYPE
–**SPREAD**, EXP
–**TIME**, TOBSERV, DELTA_TIME, TTRA_SET, NCYCLES
–**T_REINFORCE**, TB

This command generates a set of configurations of the system and the connector forces by summing the frequency response with the sea. One is not put into the Disposition Menu. Instead, one can enter the Process Post–Processing menu where on can look at the position of points, the relative motion of points, connector forces, etc. You cannot, however, use the **TRAJECTORY**, **POSITION**, **STABILITY TANK_FLD**, **TANK_BAL**, **HOLE_FLOODING**, **R_VIEW**, **R_ENVELOPE**, or **R_DETAIL** commands because data for them is not generated. Also, notice that this command creates events for a process and it will overwrite any existing events you may have. Finally, this command is not only useful for looking at a true sample but also for making movies.

While commands discussed later give the user complete control over the results he obtains, a single command has been provided to produce a set of "standard results" which suffice in many circumstances. The form of this command is:

FP_STD, X, Y, Z, –OPTIONS

where the options are

–**WEIGHT**, WEI, RX, RY, RZ
–**HEIGHT**, WAVE_HEIGHT

This command is not applicable to nonlinear spectral results. If it is issued with no options, then the response operators will be computed at the point X, Y, and Z (feet or meters) and these results will be reported and graphed. If the –**HEIGHT** option is used, then statistics of the motions will be computed in an ISSC sea of height WAVE_HEIGHT (feet or meters) for periods from 4 to 18 seconds. Again, these results will be reported and plotted. Finally, if the –**WEIGHT** option is used, the response operators of the forces acting on a body of weight WEI (bforce) with radii of gyration RX, RY, and RZ (feet or meters) located at this point will be computed. The response operators will be reported and graphed. If both the –**WEIGHT** and –**HEIGHT** options are used, then the statistics of the forces will be computed, reported, and graphed for the same set of conditions as the motions.

XX.A Equation Post–Processing

The commands discussed in this section allow one to post–process the data used to compute the response. Basically, two types of data are used: added mass and damping matrices, and exciting forces. MOSES always uses these properties about the vessel origin, but *all* commands discussed here will produce data applicable to the point specified on the *last* **FR_POINT** command. In other words, if one wants the results for the vessel origin, then he should issue a **FR_POINT** command with a point of 0, 0, 0 before issuing any command discussed here.

While commands are discussed below which give one complete control over the results he wishes to obtain, a single command has been provided which produces a set of "standard results" which suffice in many circumstances. The form of this command is:

EQU_SUM

When this command is issued, MOSES will simply generate reports of all of the equation results. The Disposition Menu will not be entered.

Alternately, one can have the option of disposing of the data for the matrices by issuing:

MATRICES, –OPTION

When this command is issued, the program will compute the added mass and damping matrices about the point specified on the last **FR_POINT** command. If the option, **–FILE** option is used, then the matrices will also be written to the PPO file. In the file, the full 6x6 matrices will be written. *Care is needed here*. The results reported for damping are not the ones really used, but are the maximum over all headings. This is correct for a spectral linearization, but not for a steepness one. Also, a report to paper gives the radii of gyration while the report to file gives the actual terms of the matrix.

To obtain the exciting forces about the point specified on the last **FR_POINT** command, one uses:

EXFORCE, –OPTION

When this command is issued, the program will compute the forces operators at the specified location for every period and heading where response was computed. Again, if the option **–FILE** option was used, the forces will also be written to the PPO file. In the file, the forces will be real and imaginary parts instead of the magnitude and phase elsewhere.

When placed in the Disposition Menu, the results for all headings are available. The names of the variables are followed by **HEDXXX** where XXX is the heading angle in degrees. When using the **REPORT** command in the Disposition Menu, one can selectively report the response operators. If there is no data on the **REPORT** command, all headings will be reported. To report data for only some headings, one should specify the angles of the heading to be reported on the **REPORT** command.

To compute statistics of the exciting forces in irregular seas, one issues the command:

> **ST_EXFORCE**, ENV_NAME, –OPTIONS

where the available options are:

> –**SEA**, SEA_NAME, THET, HS, PERIOD, GAMMA
> –**SP_TYPE**, TYPE
> –**SPREAD**, EXP
> –**E_PERIOD**, EP(1), EP(2), ...

Again, the point where moments are referred is the last point specified on a **FR_POINT** command. The statistical result is the statistic specified with the last **–PROBABILITY** option on a **&DEFAULT** command.

The remainder of the commands available for connector forces have a similar syntax in that the final portion of the command is identical to that of the **&ENV** command. In fact, these commands not only initiate the computation of quantities in an irregular sea, but are also **&ENV** commands. Thus, when one issues one of these commands with a non–blank ENV_NAME, he is altering the definition of this environment within the database. If ENV_NAME is omitted, then the environment used will be totally defined by the options specified. The options –**SEA**, –**SPREAD** and –**SP_TYPE** are used to define the sea state to which the vessel will be subjected. The –**E_PERIOD** option can be used to generate results for seas of several different periods. If this option is omitted, then a single period of PERIOD will be considered. With the option, periods of PERIOD, EP(1), EP(2), ... will be produced.

XX.B Motion Post–Processing

The commands discussed in this section all deal with the frequency response of a point or relative motion of two points, and they allow one to obtain: the frequency response of the point, statistics of this response, and a time realization of the motion. Basically, one first finds the frequency response at a given point, and then the other commands discussed here deal with this response until the response at another point is obtained. As mentioned previously, the data for these commands depends on the manner in which the original frequency response data was computed. If it was computed with an **RAO** command, then all options and data discussed here are available. If it was computed with an **SRESPONSE** command, then only geometrical data can be input. In other words, no environmental data can be specified.

Many of the commands here compute statistics of quantities and as a result have many common options. In particular:

> –**SEA**, SEA_NAME, THET, HS, PERIOD, GAMMA
> –**SPREAD**, EXP
> –**SP_TYPE**, TYPE
> –**E_PERIOD**, EP(1), EP(2), ...
> –**CSTEEP**, YES/NO

The statistical result is the statistic specified with the last –**PROBABILITY** option on a **&DEFAULT** command, and If the original response data was produced with the **SRESPONSE** command, then *no* additional sea data can be specified.

The remainder of the commands available for motions have a similar syntax in that the final portion of the command is identical to that of the **&ENV** command. In fact, these commands not only initiate the computation of quantities in an irregular sea, but are also **&ENV** commands. Thus, when one issues one of these commands with a non–blank ENV_NAME, he is altering the definition of this environment within the database. If ENV_NAME is omitted, then the environment used will be totally defined by the options specified. The options –**SEA**, –**SPREAD** and –**SP_TYPE** are used to define the sea state to which the vessel will be subjected. The –**E_PERIOD** option can be used to generate results for seas of several different periods. If this option is omitted, then a single period of PERIOD will be considered. With the option, periods of PERIOD, EP(1), EP(2), ... will be produced. If –**CSTEEP** is specified with a YES/NO of **YES**, then the height of the wave will be altered so that all seastates have the same steepness as the initial one. Otherwise, the wave height will remain constant.

To obtain the frequency response at a point, one issues the command:

FR_POINT, WHERE, –OPTIONS

Here, WHERE can be either the body coordinates (feet or meters) of the point in question, the name of a point, or the names of two points. If a single point is specified, then the results are the motion of the point. If two points are specified, then the results are the relative motion of the two points. The options are: **–VELOCITY** and **–ACCELERATION**. If no option is specified, then the motion response is produced, with **–VELOCITY**, velocity response is produced, and with **–ACCELERATION**, acceleration response results. If one specifies coordinates, MOSES assumes them to be coordinates of the *current body*. To remove confusion in multi–body situations, it may be a good idea to issue an &DESCRIBE BODY command to establish the current body before using coordinates.

When this command is issued, the program will compute the response at the specified location for every period and heading at which the original response was computed. When placed in the Disposition Menu, the results for all headings are available. The names of the variables are prefixed by **HEDXXX** where XXX is the heading angle in degrees. When using the **REPORT** command in the Disposition Menu, one can selectively report the response. If there is no data on the **REPORT** command, all headings will be reported. To report data for only some headings, one should specify the angles of the heading to be reported on the **REPORT** command.

To compute statistics of responses in irregular seas, one should issue:

ST_POINT, ENV_NAME, –OPTIONS

where the available options are:

–**SEA**, SEA_NAME, THET, HS, PERIOD, GAMMA
–**SPREAD**, EXP
–**SP_TYPE**, TYPE
–**E_PERIOD**, EP(1), EP(2), ...
–**CSTEEP**, YES/NO

The options were discussed above. When computing motions and using more than one period, there are three reports available with the **REPORT** command in the Disposition Menu. If the **REPORT** command is issued without data, then all three reports will be written. To select a subset of these three, the **REPORT** command should be given followed by: **MOTION**, **VELOCITY**, and/or **ACCELERATION**, in which case only the reports specified will be written.

When dealing with irregular seas, it is often of interest to know the variation of the sea and response spectra with frequency and period. To obtain results of this nature, one should issue:

SP_POINT, ENV_NAME, –OPTIONS

where the available options are:

–**SEA**, SEA_NAME, THET, HS, PERIOD, GAMMA
–**SP_TYPE**, TYPE
–**SPREAD**, EXP

The results produced here are based on the results of the last **FR_POINT** command and the options were discussed above.

The results obtained with the **ST_POINT** command consist of motions measured from a point in vessel coordinates. Often, one desires a global motion measured from a specified reference. Results of this type can be obtained via the command:

PMOTION, :PNT_SEL, ENV_NAME, –OPTIONS

where the available options are:

–**SEA**, SEA_NAME, THET, HS, PERIOD, GAMMA
–**SP_TYPE**, TYPE
–**SPREAD**, EXP
–**E_PERIOD**, EP(1), EP(2), ...
–**CSTEEP**, YES/NO

Here, :PNT_SEL is a selector for the points whose motion will be computed, and the other options are the same as for the **ST_POINT** command. When this command is issued, statistics for the global "dynamic motion" of the Interest Points selected by the selector :PNT_SEL will be computed. In addition, MOSES will compute the mean global position of the point, and the mean "motion" of the point. The mean motion is the vector from the global position of the point the last time the **&DESCRIBE INTEREST** command was issued. The mean motion is added to the dynamic motion with the sign of the mean to produce a "total" motion away from the marked global position of the point. The options were discussed above.

A command similar to the above is:

ST_CLEARANCE, :PNT_SEL, ENV_NAME, –OPTIONS

where the available options are:

–**SEA**, SEA_NAME, THET, HS, PERIOD, GAMMA
–**SP_TYPE**, TYPE

–**SPREAD**, EXP
–**E_PERIOD**, EP(1), EP(2), ...
–**CSTEEP**, YES/NO

The purpose of this command is to investigate the statistics of the relative motion between the points selected by :PNT_SEL and the water surface. Here, one will receive for each point selected:

- The mean distance above the water,
- The statistical "clearance change",
- The difference between the mean and the statistical change,
- The probable minimum significant wave height that will create a slam (an event when the water is above the point),
- The number of slams per hour, and
- The velocity the point minus the wave particle velocity in the global system.

XX.C Cargo Force Post–Processing

The commands discussed here apply to "dynamic forces" acting on pieces of cargo located at specified points. These forces, however, are derived completely from the motion of the body, and are quite useful in estimating the load which will act on any cargo. Here, the frequency response of the forces consist of two parts: 1.) The forces required to produce the given accelerations, and 2.) The component of weight which arises due to change in angle. These forces do not contain any static contributions such as the vertical component of weight, or any force on the body due to immersion in the water. Since all computations involving these forces have the correct phase relationship between the acceleration and the angular motion, they will be less conservative than adding the two components after an irregular sea computation.

The commands here are essentially the same as those discussed previously for the motions of a point, except that the results will be forces instead of motions. Again, if the original data was obtained with an **RAO** command, then all of the data discussed here can be specified. If instead, the original results were obtained with a **SRESPONSE** command, *then no environment, nor options can be specified.*

The **FR_FCARGO** command is used to produce the frequency response of the dynamic forces acting on a rigid body whose CG is located at the last position specified on a **FR_POINT** command. The form of this command is:

FR_FCARGO, WEIGHT, RX, RY, RZ

where WEIGHT is the weight (bforce) of the body, and RX, RY, RZ are the X, Y, Z radii of gyration (feet or meters) of the body. When placed in the Disposition Menu, the results for all headings are available. The names of the variables are prefixed by **HEDXXX** where XXX is the heading angle in degrees. When using the **REPORT** command in the Disposition Menu, one can selectively report the response. If there is no data on the **REPORT** command, all headings will be reported. To report data for only some headings, one should specify the angles of the heading to be reported on the **REPORT** command. If *no data* is specified on the **FR_FCARGO** command, then the data will be generated so that one get the "G" forces on the cargo and the angular accelerations, in the same output report table.

Many of the commands here compute statistics of quantities and as a result have many common options. In particular:

–**SEA**, SEA_NAME, THET, HS, PERIOD, GAMMA
–**SPREAD**, EXP
–**SP_TYPE**, TYPE
–**E_PERIOD**, EP(1), EP(2), ...

–**CSTEEP**, YES/NO

The statistical result is the statistic specified with the last –**PROBABILITY** option on a **&DEFAULT** command, and If the original response data was produced with the **SRESPONSE** command, then *no* additional sea data can be specified.

The remainder of the commands available for cargo forces have a similar syntax in that the final portion of the command is identical to that of the **&ENV** command. In fact, these commands not only initiate the computation of quantities in an irregular sea, but are also **&ENV** commands. Thus, when one issues one of these commands with a non–blank ENV_NAME, he is altering the definition of this environment within the database. If ENV_NAME is omitted, then the environment used will be totally defined by the options specified. The options –**SEA**, –**SPREAD** and –**SP_TYPE** are used to define the sea state to which the vessel will be subjected. The –**E_PERIOD** option can be used to generate results for seas of several different periods. If this option is omitted, then a single period of PERIOD will be considered. With the option, periods of PERIOD, EP(1), EP(2), ... will be produced. If –**CSTEEP** is specified with a YES/NO of **YES**, then the height of the wave will be altered so that all seastates have the same steepness as the initial one. Otherwise, the wave height will remain constant.

To compute statistics of responses in irregular seas, one should issue:

ST_FCARGO, ENV_NAME, –OPTIONS

where the available options are:

–**SEA**, SEA_NAME, THET, HS, PERIOD, GAMMA
–**SP_TYPE**, TYPE
–**SPREAD**, EXP
–**E_PERIOD**, EP(1), EP(2), ...
–**CSTEEP**, YES/NO

and the options are defined above. The statistics here are of the forces which resulted from the *last* **FR_FCARGO** command. If one is computing statistics of "G" forces, the angular accelerations should be the same as the angular accelerations produced from statistics of the motions. *They may, however, differ* due to numerics. The ones produced from the motions are "computed better". If the difference is too large to suit, you need more periods.

When dealing with irregular seas, it is often of interest to know the variation of the sea and response spectra with frequency or period. To obtain results of this nature, one should issue:

SP_FCARGO, ENV_NAME, –OPTIONS

where the available options are:

–**SEA**, SEA_NAME, THET, HS, PERIOD, GAMMA
–**SP_TYPE**, TYPE
–**SPREAD**, EXP

The results produced here are based on the results of the last **FR_FCARGO** command and the options are defined above.

XX.D Connector Force Post–Processing

If connectors were attached to the system when the frequency response was computed, the frequency response of the constraint forces was also computed. To obtain the frequency response of the forces which the connectors exert on the first body to which they are connected, one can issue:

> **FR_CFORCE**, CONN_NAME

Here, the user is placed in the Disposition Menu with the frequency response of the connector which matches the selector CONN_NAME. He can then proceed to dispose of these results.

To obtain statistics of the forces which the connectors

Many of the commands here compute statistics of quantities and as a result have many common options. In particular:

> –**SEA**, SEA_NAME, THET, HS, PERIOD, GAMMA
> –**SPREAD**, EXP
> –**SP_TYPE**, TYPE
> –**E_PERIOD**, EP(1), EP(2), ...
> –**CSTEEP**, YES/NO

The statistical result is the statistic specified with the last –**PROBABILITY** option on a **&DEFAULT** command, and If the original response data was produced with the **SRESPONSE** command, then *no* additional sea data can be specified.

The remainder of the commands available for connector forces have a similar syntax in that the final portion of the command is identical to that of the **&ENV** command. In fact, these commands not only initiate the computation of quantities in an irregular sea, but are also **&ENV** commands. Thus, when one issues one of these commands with a non–blank ENV_NAME, he is altering the definition of this environment within the database. If ENV_NAME is omitted, then the environment used will be totally defined by the options specified. The options –**SEA**, –**SPREAD** and –**SP_TYPE** are used to define the sea state to which the vessel will be subjected. The –**E_PERIOD** option can be used to generate results for seas of several different periods. If this option is omitted, then a single period of PERIOD will be considered. With the option, periods of PERIOD, EP(1), EP(2), ... will be produced. If –**CSTEEP** is specified with a YES/NO of **YES**, then the height of the wave will be altered so that all seastates have the same steepness as the initial one. Otherwise, the wave height will remain constant. exert on the first body to which they are connected, one can issue:

ST_CFORCE, :CONN_SEL, ENV_NAME, –OPTIONS

where the available options are:

–**SEA**, SEA_NAME, THET, HS, PERIOD, GAMMA
–**SP_TYPE**, TYPE
–**SPREAD**, EXP
–**E_PERIOD**, EP(1), EP(2), ...
–**USE_MEAN**, YES/NO

The command produces irregular sea results for the first connectors which are selected by the selector :CONN_SEL. This command works exactly as the **ST_POINT** command, except here the results are for constraint forces instead of motions. Additionally, one can use the **–USE_MEAN** option to instruct MOSES to add the mean value of the force to the computed deviation with the sign of the mean so that the reported force will be a measure of the total force, and the remainder of the options are discussed above.

When dealing with irregular seas, it is often of interest to know the variation of the sea and frequency response of the constraint forces connector with frequency and period. To obtain results of this nature, one should issue:

SP_CFORCE, :CONN_SEL , –OPTIONS

where the available options are:

–**SEA**, SEA_NAME, THET, HS, PERIOD, GAMMA
–**SP_TYPE**, TYPE
–**SPREAD**, EXP
–**E_PERIOD**, EP(1), EP(2), ...

and they are discussed above.

Fatigue can be computed on the connectors if one has computed frequency response with an **SRESPONSE** command. This is accomplished with the command

FAT_CFORCE, –OPTIONS

where the available options are:

–**INITIAL**
–**ACCUMULATE**, :CONN_SEL
–**REPORT**, TIME

This command was designed to accumulate fatigue for several different environments (**SRESPONSE**s). When the command is issued with the **–INITIAL** option *all* fatigue accumulators are zeroed. When it is used with the **–ACCUMULATE** option, cumulative damage is computed for all connectors which match :CONN_SEL, and this damage is added to that which exists. The duration used for the damage is that specified on the &ENV command. Finally, when the command is used with the **–REPORT** command, a report of the cumulative damage is written. The TIME (days) variable can be used to scale the damage from the time accumulated to TIME. In other words, if the sum of the durations was T1 and TIME was specified to be T2, then the damage will be multiplied by T2/T1 before reporting.

A command closely associated with fatigue is:

> **COUNT_CF**, –OPTIONS

where the available options are:

> –**F_BINS**, T(1), T(2), ... T(n)
> –**ACCUMULATE**, :CONN_SEL
> –**REPORT**, TIME

Instead of accumulating fatigue, however, this command accumulates cycles of tension in specified ranges. The ranges are specified with the **–F_BINS** option where T(i) is in bforce, and the other options function exactly as for the **FAT_CFORCE** command.

If some of the connectors are rods, then one has two additional commands available. These commands compute statistics of the internal forces and the stresses in the rod. To obtain the statistics of the internal forces, one should issue:

> **ST_RFORCE**, ROD_NAME, ENV_NAME, –OPTIONS

where the available options are:

> –**SEA**, SEA_NAME, THET, HS, PERIOD, GAMMA
> –**SP_TYPE**, TYPE
> –**SPREAD**, EXP
> –**USE_MEAN**, YES/NO

Here, ROD_NAME is the name of the rod one wishes to investigate. If one is interested in a pipe assembly, then he should use **&PIPE** for ROD_NAME. When issued, the statistics for the rod internal forces will be computed. One can use the **–USE_MEAN** option to instruct MOSES to add the mean value of the force to the computed deviation with the sign of the mean so that the reported force will be a measure of the total force. The remainder of the options are discussed above.

Stresses in rods are computed via the command:

ST_RSTRESS, ROD_NAME, ENV_NAME, –OPTIONS

where the available options are:

–**SEA**, SEA_NAME, THET, HS, PERIOD, GAMMA
–**SP_TYPE**, TYPE
–**SPREAD**, EXP
–**USE_MEAN**, YES/NO

which operates exactly the same as the **ST_RFORCE** command.

XX.E Pressure Post–Processing

The commands discussed in this section deal with the frequency response of the pressure on a panel. As mentioned previously, the data for these commands depends on the manner in which the original frequency response data was computed. If it was computed with an **RAO** command, then all options and data discussed here are available. If it was computed with an **SRESPONSE** command, then no environmental data can be specified.

To obtain the frequency response of the average pressure over a panel, issue the command:

> **FR_PANPRESS**, :PAN_SEL –OPTIONS

Here, :PAN_SEL is a selector for the panels you want pressure on, and the only available option is **–FILE**. If one uses this option, the pressures on panels selected by :PAN_SEL will be written to the ppo file, and the Disposition Menu will not be entered. Alternatively, if the option is not used, :PAN_SEL should select *only* a single panel, and the user is placed in the Disposition Menu to dispose of the results as desired.

To compute statistics of pressures in irregular seas, one should issue:

> **ST_PANPRESS**, :PAN_SEL, ENV_NAME, –OPTIONS

where the available options are:

> –**SEA**, SEA_NAME, THET, HS, PERIOD, GAMMA
> –**SP_TYPE**, TYPE
> –**SPREAD**, EXP
> –**E_PERIOD**, EP(1), EP(2), ...
> –**CSTEEP**, YES/NO

Here, the statistic of the pressure on panels selected by :PAN_SEL will be returned and the user will be placed in the Disposition Menu. Here, there is no limit on the number of panels selected, and the statistical result is the statistic specified with the last **–PROBABILITY** option on a **&DEFAULT** command, and If the original response data was produced with the **SRESPONSE** command, then *no* additional sea data can be specified.

The remainder of the options have a similar syntax to those of the **&ENV** command. In fact, commands using these options not only initiate the computation of quantities in an irregular sea, but is also an **&ENV** command. Thus, when one issues one of this command with a non–blank ENV_NAME, he is altering the definition of this environment within the database. If ENV_NAME is omitted, then the environment used will be totally defined by the options specified. The options **–SEA**, **–SPREAD** and **–SP_TYPE** are used to define the sea state to which the vessel will be subjected. The **−E_PERIOD** option can be used to generate results for seas of several different periods. If this option is omitted, then a single period of PERIOD will be considered. With the option, periods of PERIOD, EP(1), EP(2), ... will be produced. If **−CSTEEP** is specified with a YES/NO of **YES**, then the height of the wave will be altered so that all seastates have the same steepness as the initial one. Otherwise, the wave height will remain constant.

XXI. FINDING EQUILIBRIUM

To find the equilibrium configuration of the system due to the current loading, ballast, damage, and constraints, etc., one should issue:

&EQUI, –OPTIONS

and the available options are:

–DEFAULTS
–ITER_MAX, MAX_ITER
–TOLERANCE, TOL
–IGNORE, B_NAME, DOF(1), DOF(2), ...
–OMEGA, FRACT
–MOVE_MAX, MAX_TRANSLATE, MAX_ANGLE

When this command is issued, the program will iterate to find an equilibrium position until either the residual is less than the default convergence tolerances or until the default maximum iterations are taken. The options define parameters which are used to control the algorithm. The parameters are remembered from one invocation of the command to the next. To change a value, use the option which alters it. To get back to the default settings, use the **–DEFAULTS** option. Upon completion of this command, the initial condition is reset to be the new equilibrium condition. If this is not desirable, use the **&INSTATE –PREVIOUS** command to return to the initial condition. To alter the default tolerances or the maximum number of iterations, one should use the options **–ITER_MAX** and **–TOLERANCE**. Here, MAX_ITER is the maximum number of iterations to be taken (default is 50) and TOL is the acceleration convergence tolerance in G's for translational motion (default is 0.001).

The **–IGNORE** option can be used to ignore specified degrees of freedom only for the current equilibrium search. In other words,

```
&EQUI –IGNORE BARGE X Y RZ
```

is the logical equivalent of

```
&DESCRIBE BODY BARGE –IGNORE X Y RZ
&EQUI –IGNORE BARGE X Y RZ
&DESCRIBE BODY BARGE –IGNORE
```

Here B_NAME is the name of the body and DOF(i) must be either **X**, **Y**, **Z**, **RX**,

RY, or **RZ**.

To find equilibrium, MOSES simply hunts for a configuration where

F = 0

This is a deceptively simple equation; the exercise of finding an equilibrium configuration is one of the most difficult things that MOSES does. One of the complicating factors is that, since the equations may be nonlinear, there may be more than one equilibrium configuration. When one finds an equilibrium configuration, care should be used, it may not be the one that occurs in real life.

To solve for the configurations which satisfy the equilibrium condition, a modified Newton method is used. The modifications deal with two problems, the main one being that in many cases the stiffness matrix is singular. For example, consider a freely floating ship. Here, one has stiffness in heave, roll and pitch, but none in surge sway and yaw. Blindly applying a Newton method here is not effective. To avoid a possible singularity in the stiffness, we augment the stiffness with a fraction of the inertia, i.e. in place of the stiffness, we use

Kb = K + FRACT**2 I

where FRACT is a small parameter which we can specify. This should fix the singularity problems because for degrees of freedom where K is not singular, the inertia term will be negligible and for singular degrees of freedom of K it adds a term on the diagonal. The **–OMEGA** option is used to set FRACT, and the default values of it are .2236 if there are no flexible connectors and .02236 if there are flexible connectors.

When using this modified Newton method, there are two reasons that one may not find a configuration with tolerance: the step size MOSES takes may be too small or it may be too large. When one is far from equilibrium, large steps are needed if one is to get close within the maximum number of iterations. Here, limits on step size may need to be increased so that larger steps can be taken. This may not help if FRACT is what is limiting the step size. You see, the term we added to minimize the chance of a singular stiffness also reduces the step size. For systems with large inertia and small stiffness, the "small extra" can actually dwarf the stiffness. The fix here is to decrease FRACT. *Caution* is, however, in order. The defaults are set for reliability. Following the above advice can create the other problem – too large a step.

For very stiff systems or systems with tensions only element, equilibrium may fail because steps which are too large are being taken. Here, MOSES thinks it needs to move "a good bit", but once it gets there it finds it has gone too far. Here, the way to help is to use **–MOVE_MAX** to make the maximum step size change smaller. This option sets two additional parameters, MAX_TRANSLATE and MAX_ANGLE which limits step size. The defaults are 1 foot (.3 meters) for MAX_TRANSLATE and 2 degrees for MAX_ANGLE.

The advice given here may appear contradictory, but there are two distinct cases. If you have problems, you first need to find out the cause. The best way to eliminate difficulties is to begin with a good guess. This eliminates several problems: you no longer need "big steps" to find equilibrium, and you are much more certain that the configuration you have found is, in fact, the one you want.

XXII. TIME DOMAIN SIMULATION

To initiate a time domain simulation, use the command:

TDOM, –OPTIONS

where the available options are:

–**NO_CAPSIZE**, YES/NO
–**EQUI**
–**NEWMARK**, YES/NO, BETA, ALPHA
–**CONVERGE**, NUMB, TOL
–**RESTART**, RESTART_TIME
–**RESET**, RESTART_TIME
–**STORE**, STORE_INCREMENT

When a time domain simulation is requested, MOSES will convert the frequency domain hydrodynamic pressure data for use in the time domain. This will results in a constant added mass, a convolution kernel, and a set of forces. The total time domain force is obtained from the frequency domain force by using a Fourier series for each heading and by interpolating a total force from the heading ones and the current vessel heading. If there is no pressure data available, null data will be used for the conversion. This will result in MOSES using the input added mass and damping matrices for the hydrodynamic interaction, and there will be no wave exciting force due to the vessel. The only wave excitation will be due to wave drift and any Morison's elements.

Normally, if a body has a roll or pitch angle greater than 90 degrees, the program will stop the simulation anticipating that something occurred that was not intended. If the **–NO_CAPSIZE**, **YES** option is used, no checking for capsizing will be performed.

The **–EQUI** option is useful for finding equilibrium solutions for particularly difficult or complex situations. With this option, MOSES will enter the time domain with only the mean forces applied. This is conceptually the same as using the **&EQUI** command, except that here, the user can examine the results for each time step.

Two methods are available for integrating the equations of motions: a Predictor/Corrector method, and a Newmark method. The default is to use the Newmark Method. If, however, **–NEWMARK**, **NO** is specified, then the Predictor/Corrector method will be used. This method has been around for years, and works well in many cases. Its primary shortcomings are that for "stiff" systems, very short time steps must be used, and for larger time steps, considerable numerical damping is induced. The Newmark method ameliorate these difficulties. The two parameters BETA and ALPHA are the two Newmark parameters. If they are omitted, the defaults of BETA = .25 and ALPHA = .5 will be used. The

–CONVERGE option defines the convergence parameters when a Newmark method is used. Here, convergence is defined as the change in location between two iterations at the same time step. MOSES will take at most NUMB iterations until the norm of the change in location is less than TOL. The default for NUMB is 5 and TOL is 5e–2. The Predictor/Corrector method does not iterate.

In general, a time domain simulation will begin at time equal zero, with current locations and velocities. The environment to which the system will be subjected is that specified on the last **&ENV** command. Events will be computed every DELTA_TIME seconds until time TOBSERV is reached, where DELTA_TIME and TOBSERV are also specified on the **&ENV** command.

At the conclusion of the time domain simulation, no results are automatically reported. Instead, they are stored in the database for further use. To obtain reports, graphs, or other types of information about the simulation, one should issue the **PRCPOST** command to enter the Process Post–Processing Menu.

When one issues a **TDOM** command, the results of any previous simulation for this process name will be lost, unless one wishes to "restart" the simulation. This is accomplished by adding either the **–RESTART** or **–RESET** option to the **TDOM** command. In either case, RESTART_TIME is the time at which the previous simulation will be restarted. When one restarts a process, the events up to the restart event of the previous process will be saved, and a new process will be computed for events afterwards. Normally one restarts a time domain so that a different time step can be used or to extend the observation time.

In some cases, however, one wished to change the model at some time. Here one uses **–RESET**. Suppose, for example, that at time 100, one wishes to activate a set of connectors. This can be accomplished with:

```
&INSTATE –EVENT 100
&CONNECTOR C100@ –ACTIVE
TDOM –RESET 100
```

This first sets the initial state to that at event 100, activates the connectors and then "resets" the time domain at 100. With **–RESET** the connector and compartment settings at the initial state will be used at the reset time.

During the computation of a time domain, the data is stored in the database. The user has some control over this data storage with the option **–STORE**. The state of a configuration will be written to the database at STORE_INCREMENT increments of computed steps. If the option **–STORE** is omitted, the data will be written every computed time step. In other words, a default of STORE_INCREMENT equal 1 will be used.

The string function

&SLAM(:PAR_SELE, E(1), E(2), DE **)**

is useful for determining ”slam events.” This function looks at event between E(1) and E(2) in increments of DE in the current process and finds the events where any part of the parts selected by the part selector :PAR_SELE are submerged. If E(1), E(2), and DE are omitted, then all the events in the database will be used. The result returned is a set of pairs (e1, e2) where some of the selected parts is submerged between e1 and e2.

XXIII. LAUNCH SIMULATION

To initiate a launch analysis, the following command must be issued:

LAUNCH, –OPTIONS

where the available options are:

–**RESTART** RESTART_TIME
–**MAXTIME** MAX_TIME
–**MAXOSCILLATIONS** NUM_OSCILLATIONS
–**TSTEP** DT0, DT1, DT2
–**WINCH** V0
–**QTIP**
–**QSEP**
–**NOYAW**
–**NO_CAPSIZE**, YES/NO
–**SAVE**, SAVE_INCREMENT
–**STORE**, STORE_INCREMENT
–**OLD**, YES/NO

If –**RESTART** is specified then this is a continuation of a previous launch analysis and integration will be restarted at RESTART_TIME. Four of the options control the termination of launch simulation. For the options –**MAXTIME** and –**MAXOSCILLATIONS**, MAX_TIME is the final simulation time, and NUM_OSCILLATIONS is the number of oscillations allowed after separation. The time step increment during the simulation is controlled with the –**TSTEP** option, where DT0, DT1, and DT2 is the increment during sliding, tipping and after separation, respectively (defaults are 0.75 seconds for all 3). The –**WINCH** option is used to specify V0, the initial winch velocity, in feet or meters/second (default=1 feet/second). –**QTIP** causes the simulation to quit after the jacket begins to tip, and –**QSEP** causes the simulation to quit after the jacket separates from the barge(s). The simulation will terminate if any of the above conditions are met, whichever one occurs first. Using –**NOYAW** provides an additional force applied at the trailing end of the jacket to eliminate any relative yaw between the barges and the jacket until the jacket tips.

Normally, if a body has a roll or pitch angle greater than 90 degrees, the program will stop the simulation anticipating that something occurred that was not intended. If the –**NO_CAPSIZE**, **YES** option is used, no checking for capsizing will be performed.

During the computation of a time domain simulation, the data is stored in the database. The user has some control over this data storage with the options –**SAVE** and –**STORE**. The state of a configuration will be written to the database at STORE_INCREMENT increments of computed steps. If the option –**STORE** is omitted, the data will be written every computed time step. In other words, a default of STORE_INCREMENT equal to 1 will be used. Even though the data is stored in the database, it will be inaccessible unless the database itself has been saved. Control over how often the database is saved is provided by the **–SAVE** option. If this option is omitted, the database will be saved fifteen times during the time domain simulation. If the option is used, the database will be saved at SAVE_INCREMENT increments during the time domain. In other words, if –SAVE 30 is used, the database will be saved once every 30 computed time steps.

Historically, launch has used a predictor/corrector integration scheme. With Rev 7.07, this scheme is used until separation and then the integration is changed to a Newmark method. To keep the old method, one can use the **–OLD YES** option.

XXIV. CREATING A STATIC PROCESS

The Static Process Menu of MOSES is used to simulate the process of altering the position of a body by either lifting or ballasting. It is particularly useful for simulating the process of moving a jacket from the floating position achieved after launch to an upright position, or lifting a body from a barge and lowering it into the water. Since the results of most of these processes are sensitive to the order in which the operation is performed, MOSES's interactive structure is ideally suited to such a situation. To enter this menu, one should issue the command:

STATIC_PROCESS, BODY_NAME

where BODY_NAME is the name of the body which will be considered.

The method of analysis here consists of issuing a sequence of commands to MOSES which are in the same order in which the corresponding operation would be performed in the field. Collectively, the set of commands which one issues is called a static process. Two main commands are available: **LIFT** and **FLOOD**, defined later in this section. When one of these commands is issued, MOSES will then use the equilibrium position at the last event, change the situation as designated by the command, and iterate a new equilibrium position. In this menu, three degrees of freedom are considered: the movement of the body vertically, and two angular motions.

Before entering the Static Process Menu, one normally defines a lifting sling. This is accomplished by first entering the **MEDIT** Menu and issuing the proper sling assembly commands. In most cases, these commands will take care of all of the preliminaries so that one can perform a static process. If one does not have a sling assembly, or the situation is unusual, he may need to alter the orientation of the body via a **&DESCRIBE PART –MOVE** command before entering the menu. In general, it is advisable that the orientation of the body be such that the body X and Y axes will be close to parallel the waterplane when the process begins.

During a static process, MOSES iterates equilibrium positions, and as a result, there are certain instances in which a position satisfying the tolerance within the specified number of iterations is not found. MOSES employs two closure tolerances. It will first attempt to get the norm of the residual less than TOL(1). If this cannot be achieved within twenty iterations, it checks a second tolerance, TOL(2). If the residual is less than this value, MOSES will terminate iteration. If not, the above step will be repeated three times. The user can alter the default values of these tolerances with the option:

–CLOSURE, TOL(1), TOL(2)

on either a **FLOOD**, **LIFT**, or **BEGIN** command. The default values for these two tolerances are 1E–5 and 1E–4, and they are compared to the sum of the squares of the normalized residuals. In other words, the tolerance is compared against

RES = (RV/WTJ) ** 2 + (RR/WTJ/JL) ** 2 + (RP/WTJ/JL) ** 2,

where RV is the residual vertical force, RR is the residual roll moment, RP is the residual pitch moment, WTJ is the body weight, and JL is the body length. Notice that with the default tolerances, the maximum error in the vertical force is .32%, or 1% of the body weight, depending upon which tolerance is used. These will also result in a maximum difference in location between the centers of buoyancy and gravity of .32%, or 1% of the body length. Once the closure values have been set, they remain in effect until they are reset by using the option on another command.

As each event is computed, the results at that event are written to the screen. Since, however, the screen is only eighty characters wide, only six numbers will be displayed. The user can specify the ones displayed by the option

–**DISPLAY**, OLD(1), NEW(1), ... OLD(6), NEW(6)

n either a **BEGIN**, **FLOOD**, or **LIFT** command, where OLD defines the original type of data for a column and NEW defines the type of data which one wants to have in that column. The values for OLD must be either: **PITCH**, **ROLL**, **HOOKH**, **HOOKL**, **TBALLAST**, or **CBALLAST** while the values of NEW must be either one of the valid values for OLD, or **MTENSION**, **WPA**, **GMT**, or **GML**. The default values displayed are: PITCH, ROLL HOOKH, HOOKL, MTENSION, and TBALLAST. A –**DISPLAY** option with no parameters will reset the display to the default values.

Most of the commands here have two common options:

When the user is placed in the Static Process Menu, his task is to construct a static process which satisfies the installation criteria. The idea behind MOSES is that the user will alter an existing process until he achieves one he likes. Before this can be accomplished, however, he must have something to modify. The definition of an initial process to MOSES is accomplished by issuing:

BEGIN, –OPTIONS

and the available options are:

–**PERFUL**, TANK_NAME(1), PER(1), TANK_NAME(2), PER(2), ...
–**CHEIGHT**
–**CLOSURE**, TOL(1), TOL(2)
–**DISPLAY**, OLD(1), NEW(1), ... OLD(6), NEW(6)

which instructs MOSES to find an initial equilibrium position. If, however, one has generated a static process previously, the results are stored in the database so that when the **STATIC_PROCESS** command is re–issued, the last event of the previous process will be displayed. In either case, a process is now available for modification. When in the Static Process Menu, the quantities reported for roll and pitch have a different meaning than elsewhere. Here, pitch and roll are defined as angles which two vectors make with the waterplane. These two vectors are defined using the **–SP_ORIENT** option of the **&DESCRIBE BODY** command.

The initial position generated by the **BEGIN** command normally corresponds to the floating position with no applied hookload and no water in any of the compartments. One can alter the situation by using either of the options **–PERFUL** or **–CHEIGHT**. For the **–PERFUL** option, TANK_NAME(i) is the name of the ith tank to be initially flooded and PER(i) is the initial percentage full of tank i. If the **–CHEIGHT** option is specified, MOSES will set the location of the hook at the position it has in the initial configuration and keep the hook height constant. Otherwise, the hook height will be allowed to change and the hook will have zero load.

When the **BEGIN** command is issued, MOSES takes the "initial configuration" of the system as the starting point for finding equilibrium. Since the process of finding equilibrium is expedited if the starting point is reasonably close to an equilibrium position, it is a good idea to issue an **&INSTATE** command to set a reasonable guess for an equilibrium position prior to issuing **BEGIN**.

Now, suppose that the process is good up to some point, but it is desirable to change it after that event. Here, one instructs MOSES to move the last good event and delete the remainder of the process from the current "work process". This is accomplished by:

GOTO, LGEVENT, –OPTIONS

and the available option is:

–POST

If LGEVENT is positive, it is the event number which will become the last event in the process. Alternately, if LGEVENT is negative, it is the number of events back from the current event. Thus, if one currently has a process with 20 events, and he wishes to reconsider action at event 18, he can accomplish this by either:

GOTO 18

or

GOTO –2

When a **GOTO** command is issued, some information will be lost. To save this information, one can use an option **–POST** on the command so that he is placed in the "Process Post–Processing Menu" before the old work process is truncated. Here, the user can either report the results, graph them, or save them on a post–processing file.

MOSES simulates two types of field operations: altering the amount of water in compartments, or lifting with a crane vessel. Each action is invoked by one of two commands, the first one being:

LIFT, DZ, –OPTIONS

where the available options are:

–**NUMBER**, NUM
–**SHEIGHT**, HSTOP
–**SHOOK**, HOSTOP
–**STENSION**, TSTOP
–**CLOSURE**, TOL(1), TOL(2)
–**DISPLAY**, OLD(1), NEW(1), ... OLD(6), NEW(6)

This command instructs MOSES to change the elevation of the hook in equal vertical increments, DZ (feet or meters), until one of the termination criteria defined by the **–NUMBER**, **–SHEIGHT**, **–SHOOK**, or **–STENSION** options is satisfied. MOSES will increment the hook height until either NUM changes have been made, a specified point reaches a height of HSTOP (feet or meters) above the waterplane, the hook reaches a height of HOSTOP (feet or meters) above the waterplane, or a specified hookload, TSTOP, is reached. The point for which HSTOP is checked is defined by using the **–SP_HEIGHT** option on the **&DESCRIBE BODY** command. When the lift increment is negative, the specified heights and tension are lower bounds and when the increment is positive, they are upper bounds. In other words, when lifting, HSTOP, HOSTOP, and TSTOP are maximum values, while when lowering, they are minimum values. At each increment in height, MOSES will move the hook, iterating pitch and roll angles and a hookload which result in equilibrium.

During the lifting process, the lifting sling is treated as an assembly of elements which are rigid in tension, but can carry no compression. Thus, the hook attachment point is first found assuming all sling elements to be rigid. The load in each sling element is then computed assuming a distribution which will minimize the sum of the squares of the deviations if the problem is indeterminate. If any of the sling elements carry compression, they are eliminated, and a new hook attachment point is computed. This process is continued until the sling elements all carry tension or zero load. The **–CLOSURE** and –**DISPLAY** options were discussed above.

With the second command, MOSES will alter the ballast in selected compartments until a termination criteria is satisfied. Its form is:

FLOOD, :TNK(1), PERC(1), TRP(1), :TNK(2), PERC(2), TRP(2), ..
–OPTIONS

and the available options are:

–**CHEIGHT**
–**NDRAIN**
–**CLOSURE**, TOL(1), TOL(2)
–**DISPLAY**, OLD(1), NEW(1), ... OLD(6), NEW(6)

In general, all compartments which match the tank selector :TNK(i) will have their ballast altered by a percentage PERC(i) at each step. The process terminates when all selected compartments have reached their termination percentage, TPR(i). During this process, the water added to a tank is free to move around in the tank as the body changes orientation. In view of this, problems sometime arise if long compartments which are almost horizontal are flooded. Here, the sloshing of water from one end to the other may inhibit closure of equilibrium residuals. The flood increments PERC(i) may be either positive or negative. If one is positive, then TPR(i) is a maximum amount of ballast which will be placed in the compartments selected by :TNK(i). If PERC(i) is negative, then TPR(i) is a minimum amount which will be in each selected tank.

The **–CHEIGHT** option alters the action which will be taken with the hook during the flooding. Without the option, compartments are ballasted holding the hookload constant. Alternately, if **–CHEIGHT** is specified, the compartments will be ballasted holding the height constant unless negative hookload is required. If this occurs, the height is allowed to change with zero load.

Without the –**NDRAIN** option, MOSES attempts to simulate a tank with an open valve. In this case, at each iteration, a check of where the waterplane intersects the tank is made. If the volume of the tank below the waterplane is less than the current percentage full, then the flooded volume is taken to be the submerged volume instead of the current percentage full. Thus, compartments which are filled without the –**NDRAIN** option can later lose ballast if they come out of the water. Use of –**NDRAIN** simulates pumping water into compartments. Here, no check on the submerged volume is made, and the amount of water in a tank remains fixed during subsequent calculations. As with all static process commands, the conditions used at the beginning of a step are those which existed at the end of the last step. The **–CLOSURE** and **–DISPLAY** options were discussed above.

At the conclusion of a command, the user will be prompted for the next command. To see data which was not displayed during the execution of a command, the user can input the command:

> **REVIEW**, RTYPE, E1, E2

where RTYPE is the type of data reviewed. RTYPE must be either **POSITION**, **WATERP**, **CBCG**, **HEIGHT**, **ACTION**, or **PLOT**. The values E1 and E2 are the event numbers over which the data will be printed. If omitted, all events will be reviewed. If all types of data are reviewed, essentially all data in the output reports will be displayed. If one specifies **PLOT**, then an animation of the process will be plotted. To check the current amount of water in any tank, simply issue an **&STATUS COMPARTMENT**.

During a static process, the stability of the body can become a critical question. While the metacentric heights provide some measure of the initial stability of the system at each event, they do not address the question of the range of stability. This question is properly addressed via a righting arm computation. MOSES provides an easy method of computing the righting arms at any event during a process simulation by simply issuing the command:

> **RARM_STATIC**, EVE_NUMBER, –OPTIONS

where the options are:

> **–TRANS**, ANGLE_INC, NUM_ANGLE
> **–LONG**, ANGLE_INC, NUM_ANGLE

Here, EVE_NUMBER is the event number about which the righting arms are to be computed, ANGLE_INC is the angle increment (deg.) for the computation, and NUM_ANGLE is the number of angles for which the righting arms will be computed. The option key words specify which angle is to be incremented.

After the righting arms are computed, the user is placed in the Disposition Menu so that he can dispose of the results. MOSES then returns to the simulation as if the **RARM_STATIC** had not been issued.

XXV. POST–PROCESSING OF A PROCESS

By entering the command

PRCPOST

the user can enter the Process Post–Processing Menu to obtain results of a process. In this menu, one can select certain aspects of the process for further investigation. MOSES then places him in the Disposition Menu to either view results, find the extremes or statistics of the results, graph the results, store the results, or report the results.

The precise commands available in this menu depend on the type of process being considered and the detail of the model. MOSES checks the database to see if information for a given command is available. If it is not, then the user will not see the command in the command list. After obtaining all desired results, the user should issue the **END_PRCPOST** command to return to the Main Menu.

This menu offers extreme flexibility. However, at times, the user may wish to receive a fixed body of information. Two commands:

LAUP_STD

and

STP_STD

are available. The **LAUP_STD** command is used for fixed reports and graphs of a launch. It will produce a report of the trajectory, velocity, acceleration, and constraint forces, as well as pictures of the launch, and graphs of the tiltbeam reactions versus position. The **STP_STD** command is for post–processing a static process. Here, the reports of position, stability, height, draft marks, and sling tensions will be produced. Pictures of the process and graphs of the metacentric heights are also generated.

XXV.A Post–Processing Body Information

Two commands deal with post–processing information on bodies. They are the **TRAJECTORY** and **BODY_FORCE** command.

The options common to both commands are:

> –**EVENTS**, EVE_BEGIN, EVE_END, EVE_INC
> –**MAG_DEFINE**, A(1), .. A(n)
> –**BODY**, :B_SEL

The –**EVENTS** option selects the events which will be considered. Here EVE_BEGIN and EVE_END are the beginning and ending event numbers for which the results will be computed, and EVE_INC is the increment for computing results. After the results have been computed, MOSES places the user in the Disposition Menu so that he can dispose of the data. The corresponding form of the **REPORT** command is:

> **REPORT**, REP_NAMES(1), REP_NAME(2), ... –OPTIONS

Here, REP_NAMES(i) is a set of report names which may be selected and will depend on the command issued. The only option available for reporting is again –**EVENTS**. The –**MAG_DEFINE** option defines how the "Magnitude" is computed. You can have one, two or three A(i) and each on must be either **X**, **Y**, or **Z**. If you specify all three (the default) then the magnitudes will be the length of the vectors. Alternatively, the magnitude will be the length of the vector projected on to either a line (if one is specified) or a plane. For example

> –MAG_DEFINE X Y

will give you the length of the vector projected onto the X–Y plane. The –**BODY** option is used to specify the bodies for which results will be reported, only bodies which match :B_SEL are considered,

The **TRAJECTORY** command instructs MOSES to compute the location, velocity, acceleration, bottom clearance, and displacement for each selected body and to place the user in the Disposition Menu. The form of this command is:

> **TRAJECTORY**, –OPTIONS

The available options are:

> –**EVENTS**, EVE_BEGIN, EVE_END, EVE_INC
> –**MAG_DEFINE**, A(1), .. A(n)
> –**BODY**, :B_SEL
> –**CG**

–LOCAL, YES/NO

Normally, the location, velocity, and acceleration are computed for the body origin. If one uses the **–CG** option, then the results will be computed for the body CG instead. The final option is **–LOCAL**. If YES/NO is **NO**, MOSES will compute the velocity and acceleration in global coordinates instead of local body ones. The opposite is true if YES/NO is **YES**. The other options are defined above.

The corresponding form of the **REPORT** command is:

REPORT, NAME, –OPTIONS

Here, NAME is a selector which selects the available REPORTs: **LOCATION**, **VELOCITY**, and **ACCELERATION**.

The only option available for reporting is:

–EVENTS, EVE_BEGIN, EVE_END, EVE_INC

The **BODY_FORCE**, command reports stored values of the force breakdown by type much as the other two commands, but the report time increment has no effect on the results.

BODY_FORCE, –OPTIONS

The available options are:

–EVENTS, E_BEG, E_END, E_INC
–MAG_DEFINE, A(1), .. A(n)
–BODY, :B_SEL
–FORCE, FORCE_NAME(1), ..., FORCE_NAME(n)

The first three options are discussed above and the **–FORCE** option defines the types of forces which will be reported. Here, FORCE_NAME(i) is a selector which selects forces from the list: **WEIGHT**, **CONTENTS**, **BUOYANCY**, **WIND**, **V_DRAG**, **R_DRAG**, **WAVE**, **SLAM**, **W_DRIFT**, **CORIOLIS**, **DEFORMATION**, **EXTRA**, **APPLIED**, **INERTIA**, **A_INERTIA**, **C_INERTIA**, **FLEX_CONNECTORS**, **RIGID_CONNECTORS**, and **TOTAL**. If this option is omitted, only the total force will be computed. The meaning of these forces can be found in the section of FORCES.

XXV.B Post–Processing Drafts, Points, and Sensor Readings

One class of command is always available within the Process Post–Processing Menu – the ones which deal with Interest points, Draft Marks, and Sensors. There are four commands available: **SENSOR**, **DRAFT**, **POINTS**, **REL_MOTION**, and **P_MIN_DISTANCE**.

The options common to most commands here are:

> –**EVENTS**, EVE_BEGIN, EVE_END, EVE_INC
> –**MAG_DEFINE**, A(1), .. A(n)

The –**EVENTS** option selects the events which will be considered. Here EVE_BEGIN and EVE_END are the beginning and ending event numbers for which the results will be computed, and EVE_INC is the increment for computing results. After the results have been computed, MOSES places the user in the Disposition Menu so that he can dispose of the data. The corresponding form of the **REPORT** command is:

> **REPORT**, REP_NAMES(1), REP_NAME(2), ... –OPTIONS

Here, REP_NAMES(i) is a set of report names which may be selected and will depend on the command issued. The only option available for reporting is again –**EVENTS**. The –**MAG_DEFINE** option defines how the "Magnitude" is computed. You can have one, two or three A(i) and each on must be either **X**, **Y**, or **Z**. If you specify all three (the default) then the magnitudes will be the length of the vectors. Alternatively, the magnitude will be the length of the vector projected on to either a line (if one is specified) or a plane. For example

> –MAG_DEFINE X Y

will give you the length of the vector projected onto the X–Y plane.

The command:

> **SENSOR**, :DNAME, –OPTIONS

instructs MOSES to compute the sensor signals of all sensors who's names match :SNAME and there is no DATA for the report command.

MOSES is instructed to compute the draft readings along the draft marks selected by :DNAME with the command:

DRAFT, :DNAME, –OPTIONS

and there is no DATA for the report command.

The **POINTS** command instructs MOSES to compute the location, velocity, motion, and acceleration of the points selected by :PNT_NAME. Here, motion is the vector from the global location of a point the last time the command **&DESCRIBE INTEREST** was issued to the current position of the point. The form of this command is:

POINTS, :PNT_NAME, –OPTIONS

An additional option here is:

–MAG_DEFINE, A(1), .. A(n)

This defines how the "Magnitude" is computed. You can have one, two or three A(i) and each on must be either **X**, **Y**, or **Z**. If you specify all three (the default) then the magnitudes will be the length of the vectors. Alternatively, the magnitude will be the length of the vector projected on two either a line (if one is specified) or a plane. For example

–MAG_DEFINE X Y

will give you the length of the vector projected onto the X–Y plane. The available REP_NAMEs available here are: **LOCATION**, **MOTION**, **HEIGHT**, **GS**, or **REL_VELOCITY**. The first of these reports the location, velocity, and acceleration of the points. The second reports the location of the points, their motion, the wave elevation at the points, and the clearance between the point and the sea, while the third reports only the height of the points above the waterplane. With a report name of **GS**, the dynamic "G" loads for the selected points are reported. Finally, the last report gives the wave elevation, the wave clearance, and the wave particle velocity minus the point velocity in global coordinates. If no report name is specified, all reports are produced.

For certain situations, it is desirable to know the location, velocity and acceleration of one point *relative* to another point. The **REL_MOTION** command is provided for this, and instructs MOSES to compute the relative location, velocity and acceleration for a pair of points. The results are expressed in the body system of the first point. The form of this command is:

REL_MOTION PNT_NAME(1,1), PNT_NAME(2,1), ... –OPTIONS

The additional option here is:

–**MAG_DEFINE**, A(1), .. A(n)

and it has the same meaning as it they did for the **POINTS** command. There is no DATA for the report command.

The final command here:

P_MIN_DISTANCE PIECE :PNT_SELE –OPTIONS

reports the minimum distance of from the points selected by :PNT_SELE to the piece PIECE. Here the minimum distance is the smallest of the distance from the vertices of the piece to the points or the distance from the points to the panels perpendicular to the normal of the panel.

Normally one interested in finding whether of not a body "hits something". The reason for using a piece here instead of a body is that this computation can be quite lengthy (goes like the square of the number of panels in the piece). By using an arbitrary piece, you can define a piece that does nothing (zero permeability) and bounds the body for a quick check.

XXV.C Post–Processing Compartment Ballast

MOSES provides three commands for the post–processing of the ballast in compartments. For all of these, :CMP_SEL defines the compartments or holes for which results will be reported, and the only option is:

–EVENTS, EVE_BEGIN, EVE_END, EVE_INC

Here, EVE_BEGIN and EVE_END are the beginning and ending event numbers for which the results will be computed, and EVE_INC is an event increment. The corresponding form of the **REPORT** command is:

REPORT, –OPTIONS

where the only option is **–EVENTS**.

The **TANK_BAL** command simply reports the: percentage full, sounding, ullage, amount of ballast, pressure head, maximum differential head across the compartment wall, and flow rate in the selected compartments.

TANK_BAL, :CMP_SEL, –OPTIONS

The **HOLE_FLOODING** reports the pressure, external head, internal head, differential head, and flow rate for each hole.

HOLE_FLOODING, :CMP_SEL, –OPTIONS

The **TANK_FLD** command is something of a combination of

TANK_FLD, :CMP_SEL, –OPTIONS

the other two commands. It reports both capacity information, flow information, and if there is only a single active hole, head information. It also estimates a time required to perform the flooding and thus is useful when one has a static process but wishes to estimate the time required.

XXV.D Post–Processing Applied Forces

There are two commands which allow one to examine the forces applied to the system. These forces are computed in the post–processor, so the time step of the computation is that at which the forces are computed.

The options common to both commands are:

> –**EVENTS**, EVE_BEGIN, EVE_END, EVE_INC
> –**MAG_DEFINE**, A(1), .. A(n)
> –**FORCE**, FORCE_NAME(1), ..., FORCE_NAME(n)

The **–EVENTS** option selects the events which will be considered. Here EVE_BEGIN and EVE_END are the beginning and ending event numbers for which the results will be computed, and EVE_INC is the increment for computing results. After the results have been computed, MOSES places the user in the Disposition Menu so that he can dispose of the data. The corresponding form of the **REPORT** command is:

> **REPORT**, REP_NAMES(1), REP_NAME(2), ... –OPTIONS

Here, REP_NAMES(i) is a set of report names which may be selected and will depend on the command issued. The only option available for reporting is again **–EVENTS**. The **–MAG_DEFINE** option defines how the "Magnitude" is computed. You can have one, two or three A(i) and each on must be either **X**, **Y**, or **Z**. If you specify all three (the default) then the magnitudes will be the length of the vectors. Alternatively, the magnitude will be the length of the vector projected on to either a line (if one is specified) or a plane. For example

> –MAG_DEFINE X Y

will give you the length of the vector projected onto the X–Y plane. For the –**FORCE** option, FORCE_NAME(i) is a selector which selects forces from the list: **WEIGHT**, **CONTENTS**, **BUOYANCY**, **WIND**, **V_DRAG**, **R_DRAG**, **WAVE**, **SLAM**, **W_DRIFT**, **CORIOLIS**, **DEFORMATION**, **EXTRA**, **APPLIED**, **INERTIA**, **A_INERTIA**, **C_INERTIA**, **FLEX_CONNECTORS**, **RIGID_CONNECTORS**, and **TOTAL**. If this option is omitted, only the total force will be computed. The meaning of these forces can be found in the section of FORCES.

The command

> **ELMFORCE**, :ELE_NAME –OPTIONS

is used for elements with two vertices, and the

> **LDGFORCE**, LG_NAME, NODE_NAME(1), ... NODE_NAME(n), –

OPTIONS

LDGFORCE command is used for generalized plate elements or load groups. For an **ELMFORCE** command, forces will be computed at the two end nodes of all elements which are selected by the selector :ELE_NAME.

The **LDGFORCE** command operates somewhat differently. Here, one can select only a single group, LG_NAME. The other data on this command is a set of node names, NODE_NAME(i). If this data is specified, then the forces will be computed at the specified nodes. The computation of force at the specified nodes is accomplished by first computing a force applied to the element. A least square fit is then performed to reduce this total force and moment to forces at the specified nodes.

XXV.E Post–Processing Connector Forces

MOSES has several commands available for post–processing connector forces. All of these commands make the results available in the Disposition Menu and have a common option:

> **–EVENTS**, E_BEG, E_END, E_INC

Here, E_BEG and E_END are the beginning and ending event numbers for which the results will be computed, and E_INC is an event increment.

The corresponding form of the **REPORT** command is:

> **REPORT**, REP_NAMES(1), REP_NAME(2), ... –OPTIONS

Here, REP_NAMES(i) is a set of report names which may be selected and will depend on the command issued. The only option available for reporting is −**EVENTS**

Many of these commands accept an addition option:

> **–MAG_DEFINE**, A(1), .. A(n)

This option defines how the magnitude of the force is computed. You can have one, two or three A(i) and each on must be either **X**, **Y**, or **Z**. If you specify all three (the default) then the magnitudes will be the length of the vectors. Alternatively, the magnitude will be the length of the vector projected on two either a line (if one is specified) or a plane. For example

> –MAG_DEFINE X Y

will give you the length of the vector projected onto the X–Y plane. Often, the details of the forces are not required, only their magnitude. When this is the case, a more concise report can be obtained via the second command.

The first of the connector commands is the **C_LENGTH** command.

> **C_LENGTH**, :CONN_SEL, –OPTIONS

This command reports the length of the connectors selected by :CONN_SEL and the available option is:

> **–EVENTS**, E_BEG, E_END, E_INC

A the next three commands produce results on force in connectors selected by :CONN_SEL. The command

CONFORCE, :CONN_SEL, –OPTIONS

produces the force, the magnitude of the force, the magnitude of the force divided by the breaking strength, the length of line on bottom, the vertical pull on the anchor, and the horizontal pull on the anchor for the selected connectors. The available options are:

–**EVENTS**, E_BEG, E_END, E_INC
–**MAG_DEFINE**, A(1), .. A(n)

The command

CF_MAGNITUDE, :CONN_SEL, –OPTIONS

reports only the force magnitude and the magnitude divided by the breaking strength and the available options are:

–**EVENTS**, E_BEG, E_END, E_INC
–**MAG_DEFINE**, A(1), .. A(n)

The command

CF_TOTAL, BODY, :CONN_SEL, –OPTIONS

reports the total force on the body BODY due to the connectors selected by :CONN_SEL. In addition, the power of these forces on the body is reported. Here, the body for which the results are reported is the body connected to ”end one” of the first connector selected. The force is reported in the body system. The available options are:

–**EVENTS**, E_BEG, E_END, E_INC
–**MAG_DEFINE**, A(1), .. A(n)

The command

FOUNDATION, :CONN_SEL –OPTIONS

”checks” a foundation. It computes four unity ratios, one each for overturning, sliding, and capacity, or pre–load,

Uo = f (Vc – Vn) / Vn
Up = f Vc / Vp
Uc = f Vc / Vn
Us = f mu Hc / Vc

Here V stands for vertical load, H for horizontal load, the subscript c stands for the current state, the subscript n stands for the ”nominal” state, and the subscript p for the "pre–load" state. The factor f and the values for the pre–load and nominal states are defined with the &CONNECTOR command. The coefficient of friction, mu, is defined with the class. The available option is

–EVENTS, E_BEG, E_END, E_INC

Launchway connectors are not available for reporting with the above commands. Instead, if one is interested in the forces which a set of launchways exert on the jacket, then he should issue the command

LWFORCE, –OPTIONS

Here the only option is

–EVENTS, E_BEG, E_END, E_INC

The command

TIP–HOOK, –OPTIONS

provides the length of the boom line and the forces in the boom and sling elements as a function of time. Again, the only option is

–EVENTS, E_BEG, E_END, E_INC

XXV.F Post–Processing Rods and Pipes

When a system contains either rod elements or a pipe assembly, commands are available to obtain additional information about the behavior of the rod elements involved. Here, one can look at the configuration of the rod, the forces in the rod, and the stresses and utilization in the rod. If one is interested in a pipe, the value of ROD_NAME discussed below should be **&PIPE**, otherwise, it should be the name of the rod for which information is desired. At the conclusion of the command, the user will be placed in the Disposition Menu to dispose of the data as he wishes. All of these commands have a common option:

–EVENTS, E_BEG, E_END, E_INC

Here, E_BEG and E_END are the beginning and ending event numbers for which the results will be computed, and E_INC is an event increment.

The corresponding form of the **REPORT** command is:

REPORT, REP_NAMES(1), REP_NAME(2), ... –OPTIONS

Here, REP_NAMES(i) is a set of report names which may be selected and will depend on the command issued. If no REP_NAMEs are supplied, all reports will be printed. The only option available for reporting is **–EVENTS**

The first of these commands,

R_DETAIL, ROD_NAME, –OPTIONS

allows one to look at the situation at all points in the rod at a single event, EVENT_NUMBER defined by the option

–EVENTS, EVENT_NUMBER

and the valid REP_NAMES(i) must be either **FORCE** or **STRESS**. The command

R_ENVELOPE, ROD_NAME, –OPTIONS

yields the minimum and maximum values at all points in the rod over all events selected by the option

–EVENTS, E_BEG, E_END, E_INC

and the valid REP_NAMES(i) must be either **FORCE**, **STRESS**, or **LOCA-**

TION.

Finally, the command

R_VIEW, ROD_NAME, –OPTIONS

gives the maximum absolute values of the values over the points in the rod as a function of time. The available option is

–**EVENTS**, E_BEG, E_END, E_INC

and the valid REP_NAMES(i) must be either **FORCE**, **STRESS**, or **LOCATION**.

Most of the quantities are self explanatory, except with stresses. Here, you get the normal axial, bending, torsion, and shear stresses. Also, you get a column for the maximum axial normal force. This is just the combination of the axial and bending at the extreme fibers. Finally there are three utilization ratios: the maximum normal stress divided by the yield stress, the Von Mises stress divided by the yield stress, and a code check of the interaction of hoop stress with normal stress according to RP2A working stress edition.

XXV.G Post–Processing Static Processes

When one is post–processing a static process, two commands are available. Both of these commands have a single option:

–EVENTS, E_BEG, E_END, E_INC

Here, E_BEG and E_END are the beginning and ending event numbers for which the results will be computed, and E_INC is an event increment.

The corresponding form of the **REPORT** command is:

REPORT, –OPTIONS

The only option available for reporting is **–EVENTS**

The first considers the event number, the pitch angle, the roll angle, the hook height, (i.e., the elevation of the hook above the waterplane (feet or meters)), the hookload (bforce), the total ballast (bforce), the ballast in flooding compartments (bforce), the bottom clearance (feet or meters), and the maximum tension in one element of the harness (bforce). The form of this command is:

POSITION, –OPTIONS

and the only available option is:

–EVENTS, EVE_BEGIN, EVE_END, EVE_INC

Alternately, the results available with the **STABILITY** command are: the event number, the waterplane area (ft**2 or m**2), the transverse GM (feet or meters), the longitudinal GM (feet or meters), the error in the vertical force on the jacket (bforce), the displacement (bforce), the virtual CG of the jacket, (i.e., the location of the center of the jacket weight and hookload), and the jacket center of buoyancy, which should have the same X and Y coordinates as the virtual CG. The form of this command is:

STABILITY, –OPTIONS

The only option is:

–EVENTS, EVE_BEGIN, EVE_END, EVE_INC

XXVI. STRUCTURAL ANALYSIS & APPLIED LOADS

To perform a structural analysis or emit a set of applied structural loads, one must enter the Structural Menu. This is accomplished by issuing the command:

STRUCTURAL, –OPTIONS

where the available option is:

–INITIALIZE

This command places the user in a sub–menu where he can define a situation and perform an analysis. If the option is selected, then this will *delete all previous* structural results; otherwise, the results will be added to previous results so that all of them are available later for Structural Post–Processing. In general, there are three types of commands available in this menu: commands which define the load cases to be used, commands which define the portion of the system which will be used, and commands which produce the results. When the solution has been completed, one should issue **END_STRUCT** to return to the main menu. There are no reports produced directly in this menu. The results that are produced are the system deflections and element loads, but to obtain a report of them, one must enter the Structural Post–Processing Menu.

Three general types of results can be obtained in this menu: structural analysis results (system deflections and element internal loads), loads applied to the structure, or vibration modes. In the first two cases, both the load cases and the portion of the system to be considered must be defined before the commands to compute the results are issued. With vibration modes, a single command suffices. The remainder of this discussion is applicable only to structural analysis and applied loads.

Before proceeding, however, it is best to make a distinction between load cases and load sets. A load set is either one of the intrinsic load sets that the program generates, or a user defined load set. Load sets are combined to form load cases. It is these cases that are used to obtain the results. When emitting applied loads, or when performing a structural analysis, it is the load cases which will be used.

It is beneficial to think of the solution as being performed in one of four types:

- Frequency Domain,
- Time snap–shots of a frequency domain,
- Events during a MOSES generated process, or
- At events in a user defined process.

Here, the types refer to the type of loadings which will be applied to the structure. MOSES is different from most programs in that the structural dynamics is included directly in the analysis via generalized degrees of freedom. Thus, if generalized degrees of freedom are included during an analysis, the deformation inertia is automatically included when the load case is generated. The result is a true load case which accurately describes the static as well as the dynamic behavior. The type of solution is controlled by the type of loads generated.

When a structural analysis has been performed, the results can again be combined in the post–processor. If the problem is linear, combining the deflections in the post–processor or combining the loads prior to solution produce the same results. For a nonlinear problem, however, this is not the case.

For the linear problem in the frequency domain, either options 1 or 2 defined above are available. Of these two methods, the frequency domain allows the user more flexibility, while the time domain can offer some savings in computational effort. The frequency domain method *must* be used if one wishes to compute the cumulative fatigue damage for the system.

If one wishes to use nonlinear elements, the structural system of equations will be nonlinear. In this case, the frequency domain solution method will not really yield the proper results since the solution will be for unit wave. Here, it is better to combine loads to form the total load on the system prior to solution. Thus, one cannot correctly assess cumulative fatigue damage using nonlinear elements.

When the structural analysis is complete, one should issue:

END_STRUCT

XXVI.A Extracting Modes Of Vibration

If one wishes to investigate the vibration modes of a body, he should issue the command:

> **MODES**, BODY_NAME, –OPTIONS

and the options are:

> –**NUM_EVAL**, NEV
> –**NO_FIX**

Here, BODY_NAME defines the name of the body for which the modes will be extracted, and the option −**NUM_EVAL** defines the number of modes, NEV, which will be extracted. If −**NUM_EVAL** is omitted, 20 modes will be computed. The −**NO_FIX** option controls the manner in which connectors and restraints are treated. MOSES uses the subspace iteration method outlined by Bathe in the book *Finite Element Procedures in Engineering Analysis*.

The way in which MOSES can use the modes as generalized degrees of freedom make the extraction of the modes themselves an interesting question. Normally, one takes the body in the specified configuration with the defined connections and computes the modes. For use as generalized degrees of freedom, however, we really want the unconstrained modes. Also, the body will move and as a result, its mass matrix will also change. What is the correct mass to use, and what should be done about the connections? These questions are left to the user. The mass due to ballast, weight, and any Morison's Equation added mass will be used when the modes are extracted. No added mass due to diffraction will be used since it is frequency dependent. If the −**NO_FIX** option is used, then all active connectors and restraints will be used. Without the option, the last node in the stiffness matrix will be fixed and no restraints or connectors will be applied.

Neither the concern about mass nor about connectors is very important if you intend to use the modes as generalized degrees of freedom. Here, one is simply looking for a reasonable subspace of the N degrees of freedom which "adequately" describe the deflection of the system. The correct connections and the correct mass will be added when any generalized degree of freedom analysis is performed.

As with the other commands in the STRUCTURAL menu, **MODES** produces no reports directly. One can look at the modes and the eigenvalues in the Structural Post–Processing Menu.

XXVI.B Frequency Domain Transportation Solution

If one wishes to investigate the frequency domain behavior of a structure being towed on a vessel, MOSES provides an easy method to obtain the solution. To fully utilize this command, however, one should have created a model consisting of a single body with a part named jacket and connected this part to the vessel with transportation connectors. In this case, one does not need to worry about defining load cases, selecting parts, etc., but he simply issues a single command:

TOWSOLVE, –OPTIONS

and the options will define the type of solution desired. The available options are:

–RIGID
–GAP
–TIME, SEANAME, CASE(1), T(1), CASE(2), T(2), ..., \
CASE(i), T(i)

The **–RIGID** option is used to select only the jacket for structural analysis, and to support it on a rigid vessel. If omitted, all of the structural model will be used. The **–GAP** option is used to select a nonlinear structural connection between the vessel and jacket instead of a linear one. If this option is selected, the **–TIME** option should also be selected. This option is used to select a time domain solution of the structural system. Here, SEANAME, is the name of a seastate which was previously defined by a **&ENV** command, CASE(i) is the name which the user wishes to give to the case, and T(i) is the time at which the loads will be combined to produce the "snapshot". If this option is omitted, the solution will be in the frequency domain.

XXVI.C Defining Load Cases

The loads which will be generated by MOSES are determined by the **LCASE** command. This command has different forms to generate different loads. One can input as many of these commands as desired, and each one will define at least one load case. If two of these commands are used which specify the same load case name, then the results of both of them will be combined. Also, one can change the current process name when defining load cases in order to perform a single analysis for several different physical situations.

In general the form of this command is:

> **LCASE**, –OPTION, DATA

The next option is used for analyzing fixed structures subjected to an environment. It uses the options:

> –**STATIC**, ENV_NAME, LCNAME, CHECK, PERIODIC

This will produce a set of loads which result from the environment, ENV_NAME, and the resulting load case name will be LCNAME. Here, the system will be assumed to be *fixed in space*, and the loads which yield some maximum will be generated as the load case. The CHECK value defines what maximum will be checked. If it is **OVERTURN** the maximum overturning moment about the mudline will be used as a criteria; if it is **SHEAR**, it will be maximum shear. The actual operation of the program depends upon the type of wave specified. If a periodic wave was specified, then a search algorithm will be used which is substantially more efficient than simply "passing the wave" through the structure. If one wants to override this algorithm for some reason, he should set PERIODIC to **PERIODIC**. If PERIODIC is specified as **NO**. MOSES will then act as if a non–periodic wave had been specified. For non–periodic waves, the program will simply compute the forces on the structure for the times defined with the **–TIME** option of the **&ENV** command and pick the time which creates the maximum.

Two options of LCASE are available for use with frequency domain situations.

> –**RAO**
> –**TIME**, ENV_NAME, CASE(1), T(1), ... CASE(i), T(i)

The –**RAO** form is used to perform an analysis in the frequency domain. It will generate a load case corresponding to the mean position of the structure and two "RAO" load cases for each heading and each period at which response operators were computed. The names of these generated cases depend upon how many **LCASE** commands have been issued. Basically, the names are FRQMEAN(i) for the mean, and RAO(i)NUMBX for the ones corresponding to the response operators. Here, (**i**) is equal **A** for the first, equal to **B** for the second set, etc. The values NUMB are numbers corresponding to the headings and frequencies.

Both the frequencies and headings are in sorted numerical order, and the values for NUMB are 1, 2, ..., N for frequency 1 and headings 1 through N. For frequency 2, they are N+1, N+2, etc. The final part of the name, X, is used to denote the real and imaginary part. Here, **I** denotes the imaginary part, and **R** denotes the real part. The results from the structural solution for the RAO load cases are deflections and element loads resulting from a regular sea of given period and heading and a height of 1 database unit. In MOSES, the database unit is an inch regardless of the current values set with **&DIMEN**. Thus, if one wishes to interpret the results directly, he should use the **CASES –COMBINE** command to scale them to more meaningful units.

The **–TIME** option is used to perform a snap–shot analysis of the frequency domain results. Here, ENV_NAME, is the name of a seastate which was previously defined by a **&ENV** command, CASE(i) is the name which the user wishes to give to the case, and T(i) is the time at which the loads will be combined to produce the "snap–shot".

The next option for **LCASE** is used to perform a solution for selected times during a process. This is similar to the situation discussed above with two major exceptions: Here, a process must have been defined previously with a **TDOM** or **LAUNCH** command, within the Static_Process Menu, or by issuing some **&EVENT_STORE** commands. Also, here, the forces in the connections are explicitly taken into account, since MOSES will generate a set of loads which will sum to the resultant of the rigid constraint loads (i.e. MOSES will inertia relieve these load cases). The form of the command is:

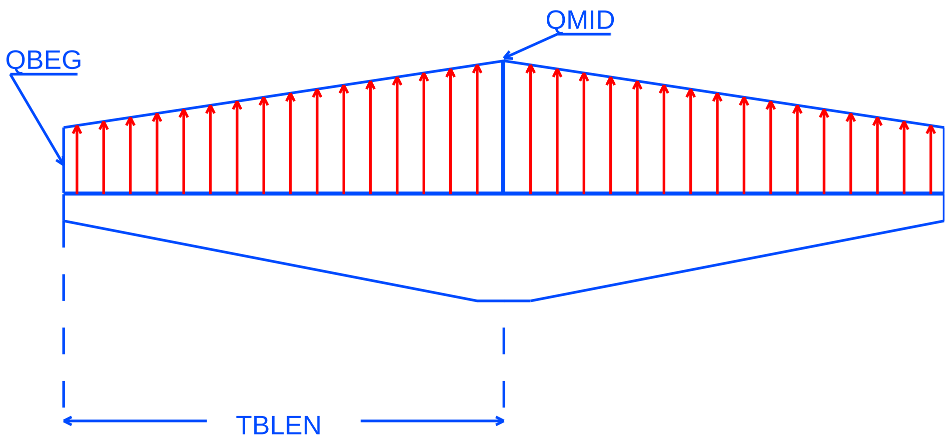
–PROCESS, CASE(1), T(1), ... CASE(i), T(i)

When **LCASE** is issued with the **–PROCESS** option, MOSES will generate a load case for each event in the process which was specified. Here again, CASE(i) is the name which the user wishes to give to the case, and T(i) is the event of the process for which the load case will be computed.

The final option for **LCASE** is used for generating a solution for a launch simulation as above but also approximating the loads that the launch barge applies to the jacket. The form is:

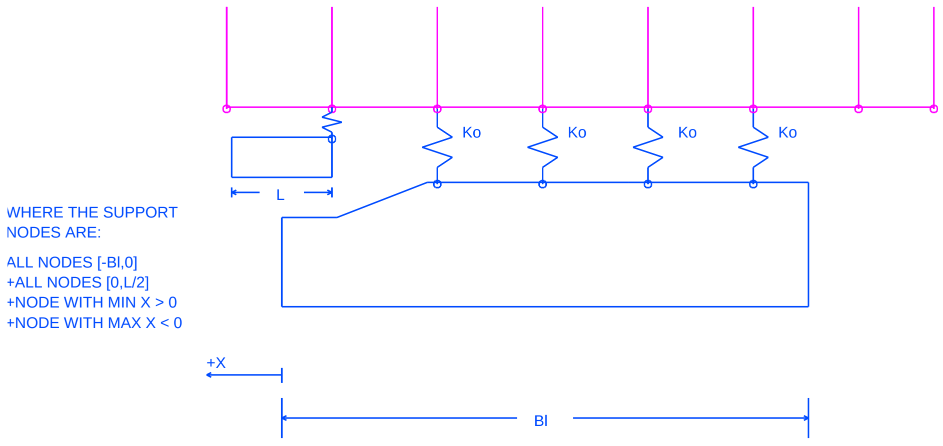
–LAUNCH, QMID, QBEG, TBLEN CASE(1), T(1), ... CASE(i), T(i)

The **–LAUNCH** option is used only for generating load cases during the launch of a jacket where it provides an approximate way of subjecting the launch legs of the jacket to a distributed load. *Caution: one should never use* **–LAUNCH** *in conjunction with* **BODSOLVE**. With this option, a **S_PART** command specifying only the jacket should be issued, and some restraints should be selected to completely define the system. The precise distribution generated depends on whether or not the jacket has tipped off the barge. Before tipping, a single distribution will be generated. After tipping, the distribution will be composed of two trapezoidal distributions, each TBLEN (feet or meters) long, which are symmetric about the tiltpin. The relative intensities at the pin and at the ends of each distribution are governed by two parameters, QBEG and QMID. Here, QMID is the relative load intensity under the tiltpin (percent), QBEG is the relative load intensity at the ends of the tiltbeam (percent), and TBLEN is one–half the tiltbeam length (feet or meters). Notice that QMID + QBEG should equal 100, and that they only describe one half of the loading on the tiltbeam. This load condition is illustrated in Figure 27. For a uniform loading, values of 50 and 50 should be used for QMID and QBEG, respectively. For a trapezoidal loading with twice the load at the pin compared to the ends, values of 66 and 34 should be used.



ROCKER ARM LOAD DISTRIBUTION
FIGURE 27

If either **SSOLVE –NONLINEAR** or **BODSOLVE** is used for solving a launch, MOSES will create connections modeling the launchway at each load case. The type of restraint supplied in the area of the tiltbeam is dependent on the use of the **–BEAM** option of the **LLEG** command. This option defines a bending stiffness and length for the beam. Before tipping, all jacket nodes between the aft end of the tiltbeam and the bow of the barge will be restrained. After tipping, only jacket nodes between the two ends of the tiltbeam will be restrained. In cases where this criteria does not yield at least two nodes, the node furtherest forward yet aft of the pin and the node furtherest aft yet forward of the pin will be added to the others. The selected nodes are then connected with compression only springs to the closest barge nodes. For nodes in contact with the barge proper, a nominal "stiff" spring is used. For those in contact with the tiltbeam, this stiff spring is put in series with the bending stiffness of the tiltbeam at the proper location. *In some cases, the compression only springs will be allowed to carry tension!* This happens when less than two nodes remain during the iterative process. The stiffness of the restraints and the number of restraints in the tiltbeam area are further explained in Figure 28

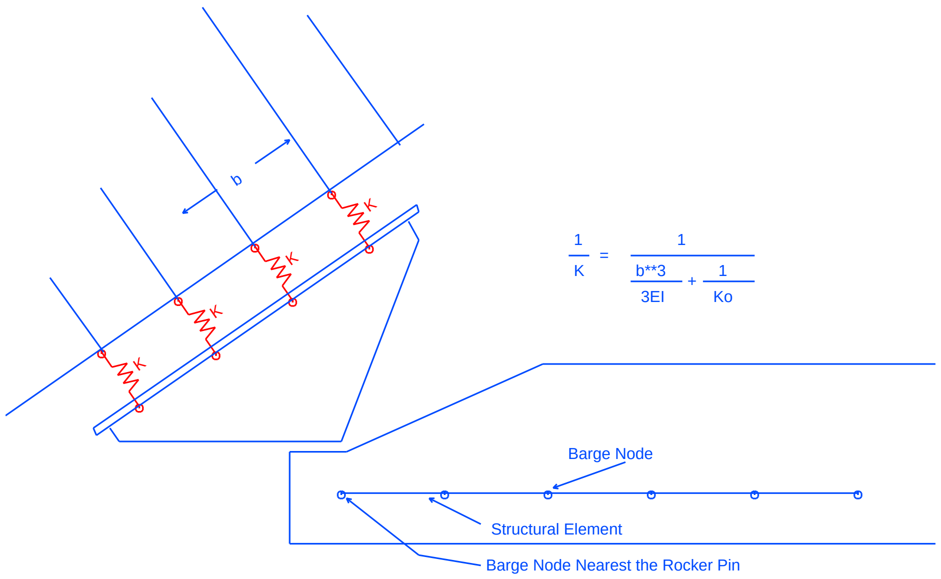


JACKET SUPPORT CONDITION FOR LAUNCH

BEFORE TIPPING

FIGURE 28

and Figure 29.


b
K
K
K
K
1/K = 1 / (b**3/3EI + 1/Ko)
Barge Node
Structural Element
Barge Node Nearest the Rocker Pin

JACKET SUPPORT CONDITION FOR LAUNCH
AFTER TIPPING

FIGURE 29

The final option of **LCASE** is :

–DEAD, CNAME, INT(1), .. INT(6)

which instructs MOSES to create the load case named CNAME by multiplying the inertia of the system by various intensities: i.e. the values are the six components of the accelerations which act on the structure (G's and rad./sec**2). Normally, however, one will use one of the other forms of the **LCASE** command.

XXVI.D Obtaining Applied Loads

If the goal of entering the Structural Menu was to generate a set of applied loads, one should first issue the appropriate **LCASE** commands, and then select the portion of the model for which he wishes to generate the loads. If the loads for the entire model are desired, then no further action is necessary. If, however, loads on only a portion are desired, the loads generated can be limited by the commands:

S_PART, PART_NAME(1), PART_TYPE(1), .. PART_NAME(n), PART_TYPE(n)
  S_BODY, BODY_NAME(1), BODY_NAME(n), ...

Here, PART_NAME(i) and BODY_NAME(i) are the part and/or body names to be selected, and PART_TYPE is the type of the part, and all can contain wild characters. When using **S_PART**, a part will be selected only when both part name and type match that of a PART_NAME(i) and PART_TYPE(i).

After the load cases and the model have been defined, one then exports the loads by simply issuing the command:

  EXP_ALOAD

and MOSES will write the loads applied to the selected system to a file for use elsewhere. By default, this file will be named root.ppo. The user may override this name by using the proper options of the **&DEVICE** command. There are three basic components to this file: the load case name, nodal loads (if present) and element loads. A sample of this file is:

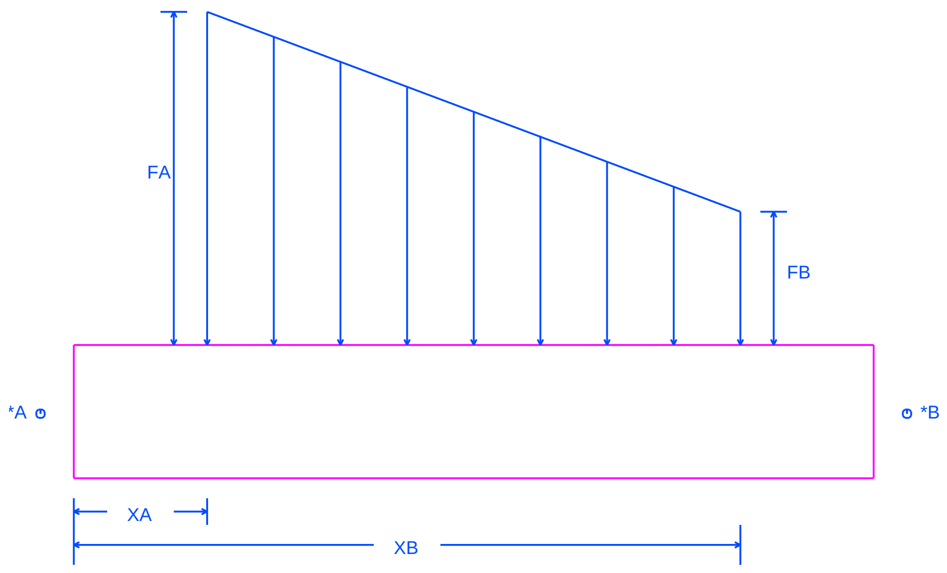
```
$&DIMEN –DIMEN  Feet     Kips
$APLOAD
LOAD CASE TWO
*2        6.4185E–16  6.4185E–16 –1.0000E+01  0.0000E+00  0.0000E+00
 6.4185E–14
*3        6.4185E–16  6.4185E–16 –1.0000E+01  0.0000E+00  0.0000E+00
 6.4185E–14
ELMAPL     T        ~T       *1       *2            .00     10.00
  0.0000E+00  0.0000E+00 –3.8378E–02  0.0000E+00  0.0000E+00
0.0000E+00
  0.0000E+00  0.0000E+00 –3.8378E–02  0.0000E+00  0.0000E+00
0.0000E+00
ELMAPL     T        ~T       *1       *2          10.00    131.42
  0.0000E+00  0.0000E+00 –4.2654E–02  0.0000E+00  0.0000E+00
0.0000E+00
  0.0000E+00  0.0000E+00 –4.2654E–02  0.0000E+00  0.0000E+00
0.0000E+00
ELMAPL     T        ~T       *1       *2         131.42    141.42
```

```
  0.0000E+00  0.0000E+00 –3.8378E–02  0.0000E+00  0.0000E+00
0.0000E+00
  0.0000E+00  0.0000E+00 –3.8378E–02  0.0000E+00  0.0000E+00
0.0000E+00
```

The first line in the file indicates the units used, and the second line is a comment line indicating the file content. The next line indicates the load case name for the applied loads that follow. All loads are reported in the *part coordinate system*.

If there are nodal loads in the structure, they are next. The format of the nodal loads starts with the node name, followed by the forces and moments acting at the node: FX, FY, FZ, MX, MY, MZ. The nodal loads are continued for this load case until an **ELMAPL** command is encountered.

Element loads have three lines per element. The first line begins with **ELMAPL** followed by the element name, the class name, the begin and end nodes, and the begin and end distance from the end of the beam over which the load acts. If the begin and end distances are equal then the load is concentrated. The next two lines define the load distribution. For concentrated loads, the two lines are the same and *each one* define the total load; i.e. one of these lines should be ignored. For distributed loads, the first line defines the load intensity at the beginning distance from the start of the beam and the second line at the end distance. There are six numbers on each line corresponding to force intensities in the X, Y, and Z directions and moment intensities in X, Y, and Z directions. These loads are trapezoidally distributed loads defined by the intensity at each end and the portion of the beam over which it acts, as shown in Figure 30.



TRAPEZOIDAL ELEMENT LOADS
FIGURE 30

Normally, there will be more than one **ELMAPL** command for each element. These come from a variety of sources, and there is no way to ascertain their origin. The **ELMAPL** commands are repeated for all the elements in the model that have load for this load case. The sequence of load case name, nodal loads and **ELMAPL** commands is then repeated for each load case specified.

XXVII. STRUCTURAL ANALYSIS & APPLIED LOADS

To perform a structural analysis or emit a set of applied structural loads, one must enter the Structural Menu. This is accomplished by issuing the command:

STRUCTURAL, –OPTIONS

where the available option is:

–INITIALIZE

This command places the user in a sub–menu where he can define a situation and perform an analysis. If the option is selected, then this will *delete all previous* structural results; otherwise, the results will be added to previous results so that all of them are available later for Structural Post–Processing. In general, there are three types of commands available in this menu: commands which define the load cases to be used, commands which define the portion of the system which will be used, and commands which produce the results. When the solution has been completed, one should issue **END_STRUCT** to return to the main menu. There are no reports produced directly in this menu. The results that are produced are the system deflections and element loads, but to obtain a report of them, one must enter the Structural Post–Processing Menu.

Three general types of results can be obtained in this menu: structural analysis results (system deflections and element internal loads), loads applied to the structure, or vibration modes. In the first two cases, both the load cases and the portion of the system to be considered must be defined before the commands to compute the results are issued. With vibration modes, a single command suffices. The remainder of this discussion is applicable only to structural analysis and applied loads.

Before proceeding, however, it is best to make a distinction between load cases and load sets. A load set is either one of the intrinsic load sets that the program generates, or a user defined load set. Load sets are combined to form load cases. It is these cases that are used to obtain the results. When emitting applied loads, or when performing a structural analysis, it is the load cases which will be used.

It is beneficial to think of the solution as being performed in one of four types:

- Frequency Domain,
- Time snap–shots of a frequency domain,
- Events during a MOSES generated process, or
- At events in a user defined process.

Here, the types refer to the type of loadings which will be applied to the structure. MOSES is different from most programs in that the structural dynamics is included directly in the analysis via generalized degrees of freedom. Thus, if generalized degrees of freedom are included during an analysis, the deformation inertia is automatically included when the load case is generated. The result is a true load case which accurately describes the static as well as the dynamic behavior. The type of solution is controlled by the type of loads generated.

When a structural analysis has been performed, the results can again be combined in the post–processor. If the problem is linear, combining the deflections in the post–processor or combining the loads prior to solution produce the same results. For a nonlinear problem, however, this is not the case.

For the linear problem in the frequency domain, either options 1 or 2 defined above are available. Of these two methods, the frequency domain allows the user more flexibility, while the time domain can offer some savings in computational effort. The frequency domain method *must* be used if one wishes to compute the cumulative fatigue damage for the system.

If one wishes to use nonlinear elements, the structural system of equations will be nonlinear. In this case, the frequency domain solution method will not really yield the proper results since the solution will be for unit wave. Here, it is better to combine loads to form the total load on the system prior to solution. Thus, one cannot correctly assess cumulative fatigue damage using nonlinear elements.

When the structural analysis is complete, one should issue:

END_STRUCT

XXVIII. STRUCTURAL POST–PROCESSING

At the conclusion of the Structural Menu, the deflections and element loads are stored in the database. To obtain reports of these results, or any results which can be derived from them, one must enter the Structural Post–Processing Menu. This is accomplished by issuing the command,

STRPOST.

At this point, the commands discussed in this section become available. When one is finished with Structural Post–Processing, he should issue:

END_STRPOST

to return to the main menu.

For any of the commands in this menu which produce reports, the selection criteria discussed for the **&REP_SELECT** command is operative. In fact, each of these report commands has as a subset of options, the options of the **&REP_SELECT** command. Thus, it is a simple matter to obtain reports for a single element, or for any subset of elements. In particular, for commands which deal with elements (beams, generalized plates, connectors, or restraints) only the following will be selected:

- The class name of the element must match the class selector defined by the last –**CLASS** option,
- The nodes which form the vertices of the element all must match a node selector defined by the last –**NODE** option, and
- The element name must match the element selector defined by the last –**ELEMENT** option.

A similar scheme is used for joints, and will be discussed later. In all cases, only cases which match the last –**LOAD** selector defined will be considered. A common mistake is to define a selector to limit one report and forget that it will also limit all subsequent reports until the selector is redefined. In most cases, the answer to the question of *why did I not get ...* is answered by a forgotten selector.

There is a string function

&str_pst(ACTION, –OPTION**)**

which gives information about the results obtained in the **STRPOST** Menu. Here ACTION should be chosen from: **E_CHECK**, **E_FATIGUE**, **E_LOADS**, **E_STRESS**, **J_CHECK**, **J_FATIGUE**, **C_CHECK**, **C_LOADS**, **C_FOUNDATION**, or **DURATION**, and the only option is

  –INITIALIZE.

MOSES saves a set of data about the "worst" situation encountered for each type of report in the STRPOST Menu. This information is initialized for all sets of data each time the menu is entered, or for a particular set when the –INITIALIZE option is used. The prefix stands for the type of data to be returned: E_ for elements, J_ for joints, and C_ for connectors. The word following the _ is the type of command for which the data will be returned: CHECK for a code check, FATIGUE for CDRs, LOAD for internal loads, and STRESS for stresses.

The ACTION **DURATION** is different from all of the others. It returns information about the duration data used in computing fatigue. In particular it will return 4 numbers:

- The number of spectra used to define the duration,
- The length of time of the duration in days,
- The length of tow in in feet or meters, and
- The average speed of the tow in feet or meters per second.

The data returned for the other values of ACTION vary in detail, but in general will be:

- The name of the element/joint which was the most severe,
- The value which was most critical,
- The number of things which failed,
- The number of classes resized,
- The load case which produced the worst value, and
- The loads (six numbers) which produced the worst value.

The only ACTION which produces exactly what it says is **E_CHECK**. The others differ from the above as follows:

- **E_CHECK** – follows the above exactly,
- **E_FATIGUE** – has only the first three values,
- **E_LOADS** – does not have number of classes resized,
- **E_STRESS**– does not have number of classes resized,
- **J_CHECK** – has neither the number of classes resized nor the six loads,
- **J_FATIGUE** – has only the first three values,
- **C_CHECK** – does not have number of classes resized,
- **C_LOADS**– does not have number of classes resized,
- **C_FOUNDATION**– has neither the number of classes resized nor the six loads.

XXVIII.A Post–Processing Cases

In all commands in the Structural Post–Processing Menu, one has the option to specify precisely which cases to consider. One always has the "basic" cases obtained directly from the structural solution available, but one of the more powerful capabilities of MOSES is the ability to combine the results of the fundamental load cases to obtain new cases for post–processing. For most situations, this capability is exercised via the **CASES** command.

As mentioned above, most of the control of post–processing cases is accomplished with the command:

> **CASES_POST**, –OPTIONS

where the available options are:

> –**DELETE**, :CASE_SEL(1), ..., CASE_SEL(n)
> –**COMBINE**, NEWNAME, CASE(1), MULT_F(1), MULT_P(1) ...,
> –**AMOD**, MULT, NAME(1), ..., NAME(n)
> –**NOMINAL**, NAME, MULT
> –**PRELOAD**, NAME
> –**TIME**, ENV_NAME, NEWNAME(1), T(1), ..., NEWNAME(i), T(i)
> –**PROCESS**, PRC_NAME
> –**MEAN**, MLCNAME
> –**FATIGUE**, DURATION_NAME, TOTTIME, TOW_VEL
> –**SPECTRA**, ENV_NAME(1), ..., ENV_NAME(i)

If one has defined some cases which he no longer needs, he can delete them by using the option **–DELETE**, which will delete all cases which match any of the selectors CASE_SEL(i).

The **–COMBINE** option is used to define a simple linear combination of previously defined cases (either fundamental or combinations themselves). Here NEWNAME is the name the user wishes to give to this combination, CASE(i) is the name of the cases to be combined, and MULT_F(i) and MULT_P(i) are the multiplier for this case for loads and pressure respectively. One really should not have two multipliers here, but some codes require multiplying the static loads by a factor. If one also multiplies the pressures by the same factor, one gets unreasonable code pressure checks. If MULT_P is omitted, then it is set to the value of MULT_F.

Some codes allow one to employ an "allowable stress modifier". This is a number which is multiplied by the allowable stress in the computation of a code check. If the user takes no action, these factors are set to 1. To alter these multipliers, one should use the **–AMOD** option. Here, MULT is the new allowable stress modifier for cases which match any of the names NAME(i) and it can contain wild characters.

The allowable stress modifiers operate on "cases", either load cases or combine cases. They do not operate on the constituents of a combine. Thus, suppose one issues the following:

```
CASES –AMOD   1.3   @
CASES –COMBINE DOG CAT .5 BIRD .5
```

Since the combine case, DOG, was defined after the definition of the allowable stress modifiers, a default modifier will be assigned. If one wishes DOG to have a different modifier, he must either issue another **CASES −AMOD** command, or define DOG before the **−AMOD**.

The **−NOMINAL** and **−PRELOAD** options define load cases and multiplier to be considered as the nominal and pre–load cases during a foundation code check. They play the same role here as the –SET_STATE option on &CONNECTOR play for a Process Post–Processing FOUNDATION check. If either of these are not defined the values defined via &CONNECTOR will be used.

The **−TIME** option is used to produce a time domain "snap–shot" from results obtained via a frequency domain structural solution. Here, ENV_NAME is a seastate name which has been previously defined on an **&ENV** command, and NEWNAME(i) is the name given to the snapshot of the system at time T(i).

All of the other options deal with converting response operator solutions into something more meaningful. MOSES associates a process name with each set of response operator loads cases. The **−PROCESS** option tells MOSES to use the response operators associated with process PRC_NAME for all options which follow. Normally, one wishes to combine the response operators with the mean to obtain the results. Without any other action, MOSES will use the "frequency mean" load case for this process in the combination. Use of the **−MEAN** option instructs the program to use the case MLCNAME instead of FRQMEAN(i) for all cases defined after the **−MEAN** option is encountered until a new **−MEAN** is encountered.

The **−FATIGUE** option can be used to alter the total time of exposure to a given environment or the process to which it is applied. If this option is used, then whenever the duration DURATION_NAME is used to compute fatigue, the actual exposure for a given component will be WTIME * TOTTIME / SUMTIME where TOTTIME (days) is that specified with the option, WTIME is the time specified when the component was defined, and SUMTIME is the sum of WTIME(i) for the duration. If TOW_VEL is specified, the length of the tow will be computed as the tow velocity times the duration of the tow, where TOW_VEL is in ft/sec or m/sec, and the duration of the tow is determined by summing the durations. When this option is used, the *process associated with the duration is also changed* to be the one specified by the preceding –**PROCESS** option.

The –**SPECTRA** option allows one to combine the frequency domain results to obtain spectral ones. Here, ENV_NAME(i) are the names of environments which have been previously defined via **&ENV** commands or in the **&DATA** Menu. The cases defined by this option will have the same names as the environments. These cases will be formed as follows: First, the RMS of each force or deflection will be computed based on the spectrum. The statistic selected by the –**PROBABILITY** option on the **&ENV** command will then be computed. Finally, if the –**USE_MEAN** option was selected when the environment was defined, the statistical result will be added to the mean using the sign of the mean. In other words, the mean and the deviation will be combined by adding the absolute value of the two quantities and using the sign of the mean. Here, the mean is either the appropriate FRQMEAN or MLCNAME.

XXVIII.B Post–Processing & Pictures

Whenever one computes joint or element code checks or fatigue, or beam loads the last values computed will be stored for use with pictures. Storing only a single value is in keeping with the philosophy of checking all cases and using the maximum to size the structure.

With animations, however, we have a different problem. Here we may want to look at the deflections of the structure or at code checks as a function of time. Before you can do this, you must have load cases that are snapshots at times during a time domain simulation. If so, then you must tell MOSES to store the results so that you can use them for pictures. One does this with the option

–FOR_ANIMATION

on the Post–Processing command: commands:

BEAM_POST CODE_CHECK
JOINT_POST CODE_CHECK
JOINT_POST DISPLACEMENTS

If you use this option, then

- *no* output of the results will be generated.
- Results will be generated only for the current process. If you have more than one process, then you need to generate the data for each process; i.e. use &DESCRIBE PROCESS to change the current process.
- Once the data has been generated, it will be used in place of the "system wide" data for all events greater than 0.
- If you define a combination case, then it will inherit the time association of the *first* member of the combination.
- If you have a combination case generated with the **–TIME** option, then it will have the event behavior of the time specified. It will not, however, be useful unless you use the **FR_2TIME** to associate a process with these events.

Perhaps it is good to look at some examples. Suppose that you have a time domain simulation and that you generated load cases with:

```
&loop t 1 1000
   lcase –process %t &string(o_number T0000)
&endloop
```

Now, suppose that you wanted to look at the deflections as a function of time. You can do this with

```
strpost
```

```
        joint deflection –load t@
    end
    &picture iso –events 0 1000 –deflect 200 only –movie
```

Of course, once the deflections are associated with the events the making of the picture is the same as with any animation.

Now, suppose you have a combination case (suppose you are doing an LRFD code check, etc.) Then

```
    &loop t 1 1000
        &set num = &string(o_number 0000)
        cases –combine %(c)%num  %(t) 1 some –1
    &endloop
```

will have a time association while

```
    &loop t 1 1000
        &set num = &string(o_number 0000)
        cases –combine %(c)%num  some –1 %(t) 1
    &endloop
```

will not.

XXVIII.C Post–Processing Modes

Structural Post–Processing results for modes are obtained with the command:

MODES_POST, TYPE(1), ..., TYPE(i) –OPTIONS

where TYPE(i) must be chosen from **VALUES** or **VECTOR**. and the available options are:

–**LOAD**, :LSEL
–**NODE**, :NODE_SEL

With modes, the "load cases" are the names M00000001, M00000002, etc. where 1 designates the first mode, 2 the second mode, etc. The –**LOAD** option can be used to select modes based on this nomenclature. If TYPE is **VALUES**, then the eigenvalues (natural frequencies) for the selected modes will be reported. If TYPE is **VECTOR** then the eigenvectors (normal modes) for each node selected by –**NODE** :NODE_SEL will be reported.

With a TYPE of VECTOR, the command works the same way the joint deflections do. In particular, one can issue &PICTURE –TYPE STRUCT immediately after the command to plot the mode shape.

XXVIII.D Post–Processing Connectors & Restraints

To obtain post–processing results for restraints and connectors, one should issue the command:

RESTRAINT_POST, TYPE(1), ..., TYPE(i) –OPTIONS

where TYPE(i) must be chosen from **ENVELOPE**, **LOADS FOUNDATION**, or **PILE_CHECK**, and the available options are:

–**CLASS**, :CLS_SEL
–**NODE**, :NODE_SEL(1), :NODE_SEL(2), :NODE_SEL(3), :NODE_SEL(4)
–**ELEMENT**, :ELE_SEL
–**LOAD**, :LSEL
–**DETAIL**
–**STANDARD**, L(1), T(1), ... L(n), T(n)
–**SUMMARY**, L(1), T(1), ... L(n), T(n)

Here, one will only receive results for elements which match the selectors defined with the **–CLASS**, **–NODE**, and **–ELEMENT** options as described previously and for cases which match :LSEL which is defined with the **–LOAD** option. If no values are given for TYPE(i), then results for all TYPEs will be produced.

With a TYPE of **ENVELOPE**, the maximum and minimum restraint loads are summarized for all selected load cases and the **–DETAIL**, **–STANDARD**, and **–SUMMARY** options are ignored.

A TYPE of **LOADS** will yield a report of the loads acting in the elements which restrain and connect the system. A TYPE of **PILE_CHECK** will yield an API code check of any piles used in the solution, and a TYPE of **FOUNDATION** will produce a "foundation check". This check is the same as the FOUNDATION check in the Process Post–Processing menu. The only difference is that here the Nominal and Pre–load cases are those defined with the –NOMINAL and –PRELOAD options of the CASES command. If these are not defined, then the values defined with the &CONNECTOR command will be used.

The extent of the reports which will be produced is controlled by which of the report types were **–DETAIL**, **–STANDARD**, or **–SUMMARY** was selected and the report limits. With a **–STANDARD** or **–SUMMARY** report, L(i) and T(i) are used to specify a range of unity ratios for which a given report will be printed. One can specify as many ranges as he desires, or he can omit all data following the option. If no ranges are specified, one report for all ranges of unity ratio will be printed. An option of **–STANDARD** will result in a report of the results for the maximum unity ratio over all selected load cases for each member selected. If one specifies an option of **–SUMMARY**, this report will be reduced to the results for only the selected element in each class which has the greatest unity ratio. Finally, if one specifies **–DETAIL** as an option, the original report will be expanded to include checks of all members for all selected load cases at all load points. Notice that –**DETAIL**, –**STANDARD**, and –**SUMMARY** may all be used on the same command to produce reports of all three types. Also, if no options are specified, then a default of –**STANDARD** is assumed.

XXVIII.E Bending Moments and Shears

Bending moments and shears are somewhat different from the other commands in the Structural Post–Processing Menu in that one is placed in the Disposition Menu so that the results can be graphed, viewed, or saved. To obtain these results, one should issue the command:

BMOM_SHR, SELE_DATA –OPTIONS

here SELE_DATA is the data that selects the beams to be processed and is either:

BMOM_SHR, ELE(1), ELE(2), ... ELE(n–1) –OPTIONS

or

BMOM_SHR, *NODE(1), *NODE(2), ... *NODE(n) –OPTIONS

where the only option is:

–LOAD, :LSEL

With the first form of this command, one defines a string of beams which lie in a line by specifying their names, ELE(i). With the second, the elements are defined by the elements connecting a string of nodes. With this form of the command, the string function &NODE(N_2NODES *NODE(1), *NODE(n)) can be used here to find the intermediate nodes; i.e. an alternative to the above is:

BMOM_SHR, &NODE(N_2NODES *NODE(1), *NODE(n)) –OPTIONS

Please notice that SP_BEAMS are not real "elements" so that for this type of element, the first form of the command must be used. When either of these commands is issued, MOSES will find the beams connecting the nodes and store the element forces for each stress point of each beam according to the distance of the stress point from the first node of ELE(1) or *NODE(1) depending upon how the command was issued. The loads for up to 10 loads cases selected by :LSEL will be considered. After the data has been obtained, the user is placed in the Disposition Menu to dispose of it as he sees fit.

Another command which deals with moments and shears is:

COUNT_FM_POST, ELE(1), ELE(2), ... ELE(n–1) –OPTIONS

or

COUNT_FM_POST, *NODE(1), *NODE(2), ... *NODE(n) –OPTIONS

where the options are:

–**FM_BINS**, F(1), M(1), ..., F(n), M(n)
–**DURATION**, :DURATION_SEL

which allows one to have the number of cycles of the bending moments and shears broken down into "bins". The beams are selected for processing as with the BMOM_SHR command. The **–FM_BINS** option defines the values for force, F(i) (bforce), and moment, M(i) (bforce–blength) which are used to delimit the bins, and the **–DURATION** option selects the durations which will be used to count the cycles.

XXVIII.F Force Response Operators

Response operators of the forces in beams and connectors are obtained with the command:

F_RAO_POST, –OPTIONS

and the available options are:

–**CLASS**, :CLS_SEL
–**NODE**, :NODE_SEL(1), :NODE_SEL(2), :NODE_SEL(3), :NODE_SEL(4)
–**ELEMENT**, :ELE_SEL
–**DETAIL**
–**STANDARD**, L(1), T(1), ... L(n), T(n)
–**SUMMARY**, L(1), T(1), ... L(n), T(n)
–**REPORT**, YES/NO
–**FILE**, YES/NO

Here, one will only receive RAOs for elements which match the selectors defined with the **–CLASS**, **–NODE**, and **–ELEMENT** options as described previously.

The extent of the reports which will be produced is controlled by which of the three report types **–DETAIL**, **–STANDARD** or **–SUMMARY** were selected and the report limits. With a **–STANDARD** or **–SUMMARY** report, L(i) and T(i) are used to specify a range of values for which a given report will be printed. The value used for determining the range is the axial load. One can specify as many ranges as he desires, or he can omit all data following the option. If no ranges are specified, one report for all ranges of axial load will be printed. An option of **–STANDARD** will result in a report of the RAOs for the maximum axial force over all selected periods and headings for each member selected. If one specifies an option of **–SUMMARY**, this report will be reduced to the RAOs for only the selected element in each class which has the greatest axial force. Finally, if one specifies **–DETAIL** as an option, the original report will be expanded to include checks of all members for all periods and headings at all load points. Notice that **–DETAIL**, **–STANDARD**, and **–SUMMARY** may all be used on the same command to produce reports of all three types. Also, if no options are specified, then a default of **–STANDARD** is assumed.

The options **–REPORT** and **–FILE** are used to control whether or not the RAOs are written to a post–processing file, or the standard output file. The default is to write them only to the output file. If **–FILE YES** is specified, then the RAOs will be written to both places. If **–FILE YES –REPORT NO** is specified, then the RAOs will only be written to the post–processing file.

XXVIII.G Post–Processing Beams

Structural Post–Processing results for beams are obtained with the command:

> **BEAM_POST**, TYPE(1), ..., TYPE(i) –OPTIONS

where TYPE(i) must be chosen from **LOADS**, **CODE_CHECK**, **H_COLLAPSE**, **ENVELOPE**, **STRESS**, **COUNT**, **NT_FATIGUE**, or **FATIGUE**, and the available options are:

> –**CLASS**, :CLS_SEL
> –**NODE**, :NODE_SEL(1), :NODE_SEL(2), :NODE_SEL(3), :NODE_SEL(4)
> –**ELEMENT**, :ELE_SEL
> –**LOAD**, :LSEL
> –**DURATION**, :DURATION_SEL
> –**CODE**, TYPE, CCAT
> –**STANDARD**, L(1), T(1), ... L(n), T(n)
> –**SUMMARY**, L(1), T(1), ... L(n), T(n)
> –**DETAIL**
> –**DOF_SEL**, DOF
> –**REPORT**, YES/NO
> –**FILE**, YES/NO
> –**B_LOAD**, YES/NO
> –**EQUIVALENT**, YES/NO
> –**RESIZE**, CONT
> –**COSTS**, STCOST, RCOST
> –**UP_CLASS**, YES/NO
> –**S_BINS**, S(1), S(2), ..., S(n)
> –**SLA_COEFFICIENT**, S_COE
> –**SLA_DAF**, S_DAF
> –**SLA_CDAMP**, S_CDAMP
> –**SLA_FIXITY**, S_FIXITY
> –**SLA_MULTIPLIER** S_VEL(1), S_MUL(1), ... S_VEL(n), S_MUL(n)

Here, one will only receive results for elements which match the selectors defined with the –**CLASS**, –**NODE**, and –**ELEMENT** options as described previously and for cases which match :LSEL which is defined with the –**LOAD** option. For **COUNT**, **NT_FATIGUE** or **FATIGUE**, all durations which match :DURATION_SEL defined via the –**DURATION** option will be considered. If no values are given for TYPE(i), then results for all TYPEs will be produced.

Here, a TYPE of

- **LOADS** will produce the internal element loads,
- **CODE_CHECK** will produce a report of the elements checked against the code specified by the –**CODE** option,
- **H_COLLAPSE** will produce an API hydrostatic collapse check,

- **ENVELOPE** will produce an envelope of beam stresses or loads,
- **STRESS** will produce the stresses at "stress points" for each longitudinal location on the beam,
- **COUNT** will produce the number of cycles of the stresses in "bins", and
- **NT_FATIGUE** or **FATIGUE** will produce fatigue results. Here, **NT_FATIGUE** will consider only sections of beams which are not part of tubular joints and **FATIGUE** will consider all beams.

With the exception of a TYPE of **COUNT**, the extent of the reports which will be produced is controlled by which of the three report types were selected and the report limits. With a –**STANDARD** or –**SUMMARY** report, L(i) and T(i) are used to specify a range of values for which a given report will be printed. Remember, the beams for which results are computed is already restricted by the –**CLASS**, –**NODE**, and –**ELEMENT** selectors, and the load cases by the –**LOAD** selectors. Thus, here we are talking about restricting what is reported out of what is computed. The easiest of these to describe is –**DETAIL**. Here, all the results are printed. With –**STANDARD**, only the result with the greatest "value" is a candidate for reporting, and it will only be reported if the value is between the report limits specified for that report. What is meant by "value" depends on type and will be described later. With a report option of –**SUMMARY**, only the result with the greatest value for a class is a candidate for reporting. One can specify as many ranges as he desires, or he can omit all data following the option. If no ranges are specified, one report for all ranges of value will be printed. An option of –**STANDARD** will result in a report of the results for the maximum "value" over all selected load cases for each member selected. If one specifies an option of –**SUMMARY**, this report will be reduced to the results for only the selected element in each class which has the greatest "value". Notice that –**DETAIL**, –**STANDARD**, and –**SUMMARY** may all be used on the same command to produce reports of all three types. Also, if no options are specified, then a default of –**STANDARD** is assumed. For a type of **STRESS**, the reporting criteria is the Von Mises stress divided by the yield stress, for a type of **FATIGUE**, the value is the CDR, and for all others *except* **LOADS** it is the code unity value.

For a TYPE of **LOADS**, the value used for determining the range is the absolute value of DOF in bforce or bforce–blength units. For this type, The –**DOF_SEL** option is used to specify the degree of freedom to be used as the reporting criteria. With this option, DOF is selected from the list: **FX**, **FY**, **FZ**, **MX**, **MY**, **MZ**, **SHEAR** or **MOMENT**. Only one value may be selected, and the default value is **FX**, axial. The value for DOF determines which quantity of the beam is used as the criteria for selecting the load case to report. When using the –**STANDARD** or –**SUMMARY** options, only the load case providing the highest absolute value of DOF is reported. The options –**REPORT** and –**FILE** operate only with a TYPE of **LOADS**. These options are used to control whether or not the results are written to a post–processing file, or the standard output file. The default is to write them only to the output file. If –**FILE YES** is specified, then the results will be written to both places. If **–FILE YES –REPORT NO** is specified, then the results will be written to the post–processing file only.

The options **–CODE**, **–EQUIVALENT**, **–RESIZE**, **–COSTS**, **–UP_CLASS**, and **–B_LOAD** are applicable only to a TYPE of **CODE_CHECK**. The type of code which will be used depends upon the last **–CODE** option. Here, TYPE may be either **AISC**, **API**, **NORSOK**, or **ISO**. The value CCAT defines the class of check for AISC or API type checks. It should be omitted for **ISO** or **NORSOK** type checks and it must be either **WS** or **LRFD** for **AISC** or **API** checks. If it is omitted for these checks, **WS** will be assumed. If one wishes to use an LRFD check, it is his responsibility to build load cases which include the proper multipliers. This option is remembered between BEAM code checks and JOINT checks. Thus, if one has previously used the **–CODE** option he need not be re–issued. If one is checking **NORSOK** or **ISO** codes, non tube beams will be checked using EUROCODE 3. No National Annex changes are considered and class 4 sections are treated as failures.

None of the codes are clear in how to treat non–prismatic members with respect to buckling. If one uses the **–EQUIVALENT** option with NO, then each section is checked using a slenderness based on the geometrical properties of that section, the "k" factor, and the length. If, however, this option is used with YES, then the slenderness of each section will be based on an estimate of the true buckling load of the element; i.e. a Raleigh Quotient will be used to compute the Euler critical load in the element and then a slenderness (and hence a radius of gyration) will be computed for each section which when combined with the standard formulae yield the estimated Euler buckling load.

The **–RESIZE** option directs MOSES to resize the selected classes so that each member will have a unity ratio less than one for the load cases considered. If CONT is **UP** and a class has a unity ratio less than one, no change in the class will be made. If either the unity ratio is greater than one or CONT is **UPDOWN**, the class will be resized. The manner in which MOSES picks a new member depends upon the table selector specified on the **–RDES** option for the class definition command. The shapes selected by the selector are assigned a cost based on the two unit costs defined via the **–COSTS** option. Here, STCOST is the cost of steel in monetary units per bforce, and RCOST is the corresponding cost of adding a single hydrostatic ring. The selected shapes are then tried in order of increasing cost until a workable shape is found. If **–UP_CLASS** is specified with YES/NO equal **YES**, then the new sizes will be stored in the database. Otherwise, the original sizes will remain. If **–B_LOAD** is specified with YES/NO equal to **YES**, a third line will be added to the code check output. This line contains the loads which produced the reported stresses.

For a TYPE of **ENVELOPE**, the maximum and minimum internal element loads are summarized for all selected load cases. This is quite useful in understanding the nature of loadings in the beam elements of a structure. If **–DETAIL** is specified as an option, the envelope of loads is reported at all load points. With an option of **–STANDARD**, the maximum and minimum values are reported once, for any location on the beam.

The **–S_BINS** option is applicable only to a type of **COUNT**. It is used to define a set of "bins" by stress range to accumulate the cycle data. Here, S(1) (ksi or mpa) marks the "top" of the first bin, S(2) the top of the second bin, etc. The stresses which are accumulated include the stress concentration factor.

The options **–SLA_COEFFICIENT**, **–SLA_DAF**, **–SLA_CDAMP**, **–SLA_FIXITY**, or **–SLA_MULTIPLIER** are applicable only to a TYPE of **FATIGUE**. They define the parameters used in frequency domain slamming fatigue are are discussed in the section on Beam Fatigue Due to Slamming. If you do not want to consider slamming, you should use

–SLA_COEFFICIENT 0.

Also with **FATIGUE**, the meaning of **–DETAIL**, **–STANDARD**, and **–SUMMARY** is a bit different. **–DETAIL** is ignored. For **–STANDARD**, one receives the total CDRs for all computed points where the maximum of the CDRs lie between L(i) and T(i). Finally, for **–SUMMARY**, one only receives a report of the maximum CDR for all of the computed points which lie in the specified range.

XXVIII.H Post–Processing Generalized Plates

Structural Post–Processing results for generalized plates are obtained with the command:

PLATE_POST, TYPE(1), ..., TYPE(i) –OPTIONS

where TYPE(i) must be chosen from **STRESS**, **LOADS**, **FATIGUE**, or **COUNT**, and the available options are:

–**CLASS**, :CLS_SEL
–**NODE**, :NODE_SEL(1), :NODE_SEL(2), :NODE_SEL(3), :NODE_SEL(4)
–**ELEMENT**, :ELE_SEL
–**LOAD**, :LSEL
–**DURATION**, :DURATION_SEL
–**DETAIL**
–**STANDARD**, L(1), T(1), ... L(n), T(n)
–**SUMMARY**, L(1), T(1), ... L(n), T(n)
–**REPORT**, YES/NO
–**FILE**, YES/NO
–**S_BINS**, S(1), S(2), ..., S(n)
–**CDR_VONMISES**, FLAG

Here, one will only receive results for elements which match the selectors defined with the –**CLASS**, –**NODE**, and –**ELEMENT** options as described previously, and for cases which match :LSEL which is defined with the –**LOAD** option. For **FATIGUE**, all durations which match :DURATION_SEL defined via the –**DURATION** option will be considered. If no values are given for TYPE(i), then results for all TYPEs will be produced.

Here, a TYPE of **STRESS** will produce the average stress in the element, **LOADS** will produce the membrane tractions acting on the faces of the element and the average bending stresses, **FATIGUE** will produce spectral fatigue results, and **COUNT** will produce the number of cycles of the stresses in "bins". With the exception of **COUNT**, the extent of the reports which will be produced is controlled by which of the three report types –**DETAIL**, –**STANDARD** or –**SUMMARY** were selected and the report limits. With a –**STANDARD** ob –**SUMMARY** report, L(i) and T(i) are used to specify a range of values for which a given report will be printed. The value used for determining the range is the Von Mises stress divided by yield stress except for **FATIGUE** where the value is the CDR. One can specify as many ranges as he desires, or he can omit all data following the option. If no ranges are specified, one report for all ranges of value will be printed. An option of –**STANDARD** will result in a report of the results for the maximum unity ratio over all selected load cases for each member selected. If one specifies an option of –**SUMMARY**, this report will be reduced to the results for only the selected element in each class which has the greatest unity ratio. Finally, if one specifies **–DETAIL** as an option, the original report will be expanded to include checks of all members for all selected load cases at all load points. Notice that **–DETAIL**, **–STANDARD**, and **–SUMMARY** may all be used on the same command to produce reports of all three types. If no options are specified, then a default of **–STANDARD** is assumed.

The options **–REPORT** and **–FILE** operate only with a TYPE of **LOADS**. These options are used to control whether or not the results are written to a post–processing file, or the standard output file. The default is to write them only to the output file. If **–FILE YES** is specified, then the results will be written to both places. If **–FILE YES –REPORT NO** is specified, then the results will only be written to the post–processing file.

The last options are applicable only to a TYPE of **FATIGUE** or **COUNT**. The sets of duration data which will be used are those which match the selector :DURATION_SEL defined with the **–DURATION** option. The **–S_BINS** option is applicable only to a type of **COUNT**. It is used to define a set of "bins" by stress range to accumulate the cycle data. Here, S(1) (ksi or mpa) marks the "top" of the first bin, S(2) the top of the second bin, etc. The stresses which are accumulated include the stress concentration factor. If stress used in computing the CDR depends on the value of FLAG following the **–CDR_VONMISES** option. If it is **YES** the Von Mises stress will be used. If it is **NO** then the principle stresses will be used. Also with **FATIGUE**, the meaning of **–DETAIL**, **–STANDARD**, and **–SUMMARY** is a bit different. **–DETAIL** is ignored. For **–STANDARD**, one receives the total CDRs for all computed points where the maximum of the CDRs lie between L(i) and T(i).

XXVIII.I Post–Processing Joints

Structural Post–Processing results for joints are obtained with the command:

JOINT_POST, TYPE(1), ..., TYPE(i) –OPTIONS

where TYPE(i) must be chosen from **DISPLACEMENT**, **CODE_CHECK**, **S_FATIGUE**, **CRUSH**, **COUNT**, or **FATIGUE**, and the available options are:

–**NODE**, :NODE_SEL(1), :NODE_SEL(2), :NODE_SEL(3), :NODE_SEL(4)
–**ELEMENT**, :ELE_SEL
–**LOAD**, :LSEL
–**STANDARD**, L(1), T(1), ... L(n), T(n)
–**SUMMARY**, L(1), T(1), ... L(n), T(n)
–**DETAIL**
–**REPORT**, YES/NO
–**FILE**, YES/NO
–**LOCAL**, YES/NO
–**CLASS**, :CLS_SEL
–**CODE**, TYPE, CCAT, EDITION
–**SN**, CURVE, TYPE, S(1), N(1), ... S(n), N(n)
–**THICK_SN**, TO, POWER, MAXCOR, YES/NO
–**LIFE**, DLIFE
–**WL_RANGE**, –ELEV, +ELEV
–**DURATION**, :DURATION_SEL
–**S_BINS**, S(1), S(2), ..., S(n)
–**CSRV_JFAT**, YES/NO
–**CLS_MEAN**, YES/NO

Here, joints which match :NODE_SEL(1) will be considered. Further, if the class of the chord does not match :CLS_SEL, the joint will not be considered. Only braces with end nodes which match both :NODE_SEL(1) and :NODE_SEL(2), names which match :ELE_SEL, and classes which match :CLS_SEL will be considered. Results will only be considered for cases which match :LSEL which is defined with the **–LOAD** option. For **FATIGUE**, all durations which match :DURATION_SEL defined via the **–DURATION** option will be considered. If no values are given for TYPE(i), then results for all TYPEs will be produced.

Here, a TYPE of

- **DISPLACEMENT** will produce the deflections of the selected joints,
- **CODE_CHECK** will produce a report of the joint checked against API–RP2A,
- **S_FATIGUE** will produce a report of the API–RP2A simplified fatigue check,
- **CRUSH** will produce a report of the crush unities based on effective closed

ring analysis,

- **COUNT** will produce the number of cycles of the stresses in "bins", and finally,
- **FATIGUE** will produce CDRs computed using a stochastic fatigue approach.

The extent of the reports except for **DISPLACEMENT** and **COUNT** is controlled by which of the three report types were selected and the report limits. With a –**STANDARD** or –**SUMMARY** report, L(i) and T(i) are used to specify a range of code unity values for which a given report will be printed. For a TYPE of **FATIGUE**, the value is the CDR, and for all others it is the code unity value. One can specify as many ranges as he desires, or he can omit all data following the option. If no ranges are specified, one report for all ranges of value will be printed. An option of –**STANDARD** will result in a report of the results for the maximum unity ratio over all selected load cases for each member selected. If one specifies an option of –**SUMMARY**, this report will be reduced to the results for only the selected element in each class which has the greatest unity ratio. Finally, if one specifies –**DETAIL** as an option, the original report will be expanded to include checks of all members for all selected load cases at all load points. Notice that –**DETAIL**, –**STANDARD**, and –**SUMMARY** may all be used on the same command to produce reports of all three types. If no options are specified, –**STANDARD** is assumed.

For a TYPE of **DISPLACEMENT**, three additional options are available: –**REPORT**, –**FILE**, and –**LOCAL**. The first two of these options are used to control whether or not the results are written to a post–processing file, or to the standard output file. The default is to write them only to the output file. If –**FILE YES** is specified, then the results will be written to both places. If –**FILE YES** –**REPORT NO** is specified, then the results will only be written to the post–processing file. By default, the displacements are reported in the body part system. If one uses the option –**LOCAL NO**, then the displacements will be reported in the "current" global system. *Notice*, since the structural results can come from many different processes at many different events, the current global system may not be a good system in which to view the deflections produced at events with a different configuration.

The type of unity ratio which will be computed for types of **CRUSH** and **CODE_CHECK** depends upon the last –**CODE** option. The type of code which will be used depends upon the last –**CODE** option. Here, TYPE may be either **AISC**, **API**, **NORSOK**, or **ISO**. The value CCAT defines the class of check for AISC or API type checks. It should be omitted for **ISO** or **NORSOK** type checks and it must be either **WS** or **LRFD** for **AISC** or **API** checks. If it is omitted for these checks, **WS** will be assumed. If one wishes to use an LRFD check, it is his responsibility to build load cases which include the proper multipliers. This option is remembered between BEAM code checks and JOINT checks. Thus, if one has previously used the –**CODE** option he need not be re–issued. EDITION defines the edition of RP2A which will be used checked for the check. If it is **OLD**, then the first supplement of the 21st edition will be used otherwise the "new" method will be used. If one wishes to use an LRFD check, it is his responsibility to build load cases which include the proper multipliers.

For a type of **CRUSH** the results computed consist of the force and moment in the chord due to the braces, stresses in the chord as a function of angle around the chord, and a unity ratio. The unity ratio is the larger of the stress unity based on an allowable of .6Fy and the shear unity based on an allowable of .4Fy. The stress reported is the stress which produced the unity ratio. This angle is measured clockwise from the first brace. For the options **–STANDARD** and **–SUMMARY** only the results at the angle which corresponds to the highest unity check are reported, while if the option **–DETAIL** is selected, one will receive results for all 36 angles around the joint. The stresses are computed by superimposing stresses due to each brace load and using the formulae for circular rings in *Roark's Formulas for Stress and Strain*, Sixth Edition. Here, the length of the ring is taken to be the "effective length" of the joint according to API–RP2A. For each brace, two cases are added: first, a uniform load distributed over a segment of the chord equal the brace diameter, and second a shear around the circumference of the chord.

The **–SN**, **–F_STRESS**, and **–THICK_SN** options are used to define the SN curve which will be used in computing the cumulative damage ratios, and are discussed in detail in the section on Defining and Associating SN curves. The two options **–LIFE** and **–WL_RANGE** are applicable only to a TYPE of **S_FATIGUE**. Here, DLIFE is the design life, which must be either 20 or 40. The two parameters –ELEV and +ELEV define the distances below and above the water surface between which members will be considered to be "waterline" members. –ELEV should be a negative number and +ELEV should be a positive one.

The **–DURATION** option is applicable only to types of **FATIGUE** and **COUNT** and tells MOSES to use only the sets of duration data which match :DURATION_SEL. The **–S_BINS** option is applicable only to a type of **COUNT**. It is used to define a set of "bins" by stress range to accumulate the cycle data. Here, S(1) (ksi or mpa) marks the "top" of the first bin, S(2) the top of the second bin, etc. The stresses which are accumulated include the stress concentration factor, but not any stress concentration due to an SN curve.

The final options are applicable only to a TYPE of **FATIGUE**. Also with **FATIGUE**, the meaning of **–DETAIL**, **–STANDARD**, and **–SUMMARY** is a bit different. **–DETAIL** is ignored. For **–STANDARD**, one receives the total CDRs for all computed points where the maximum of the CDRs lie between L(i) and T(i). Finally, for **–SUMMARY**, one only receives a report of the maximum CDR for all of the computed points which lie in the specified range. The **–CSRV_JFAT** option controls the action taken with the brace/chord SCFs. MOSES computes CDRs at eight points around the intersection of the brace with the chord. If YES/NO is **NO**, both the brace and chord side are considered, and sixteen CDRs will be computed. If, however, YES/NO is **YES**, then the maximum of the two SCF values is taken and a CDR is computed for these values of SCF at the eight points. A value of **YES** will yield slightly conservative values and significantly reduce the computational effort. The **–CLS_MEAN** option controls the manner in which the joint is classified if the previous option is not used. IF YES/NO is **YES**, then the joint will be classified using the frequency mean load case; otherwise, MOSES will compute a new joint classification (and hence SCF) for each force response operator.

Index