TECHNICAL REPO[illegible]
884-120-04
28 FEBRUARY 19[illegible]

VERIFICATION OF SPREAD MOORING SYSTEMS FOR FLOATING DRILLING PLATFORMS

VOLUME IV: A STATIC MODEL FOR MOORING REVIEW

by

DAVID B. DILLON

Prepared for

TECHNOLOGY ASSESSMENT AND RESEARCH BRANCH
MINERALS MANAGEMENT SERVICE
RESTON, VIRGINIA 22091

Under Contract N00167-85-D-0002 with

DAVID W. TAYLOR NAVAL SHIP RESEARCH AND
DEVELOPMENT CENTER
BETHESDA, MARYLAND 20084

EG&G

DISCLAIMER

The views and conclusions contained in this document are those of the author and should not be interpreted as necessarily representing the official policies or recommendations of the United States Government.

PREFACE

As offshore oil exploration moves into ever deeper waters, greater demands are placed on mooring systems. Safety of the crew, preservation of the environment, and protection of the rig itself demand that mooring systems perform reliably during operations and storms alike. It is the responsibility of the Minerals Management Service (MMS) of the U.S. Department of the Interior to insure the satisfactory performance of mooring equipment aboard exploratory oil rigs in service in United States offshore oil fields. This work was commissioned to provide MMS personnel with a manual for the analytical and physical evaluation of rig mooring systems.

This is Volume IV of a four-volume set. The purpose of these manuals is to provide a procedural structure to support the activities of MMS described above. It does not purport to be a textbook of mooring analysis or design, nor a compendium of mooring design data. That ground has been well plowed by others. Rather, a procedure for evaluating the mooring gear for a drilling rig is described.

Volume I Methods for Spread Mooring Review
Volume II Methods for Spread Mooring Inspection
Volume III Dynamic Modeling in Spread Mooring Review
Volume IV A Static Model for Spread Mooring Review

Volume I describes five steps for evaluating a mooring design and illustrates the procedure by evaluating a sample semisubmersible mooring. Volume II is a review of mooring evaluation from the standpoint of the hardware itself - the components of a typical mooring, their inspection and testing. Volume III illustrates dynamic modeling of a spread-moored drilling platform.

This manual - Volume IV - documents a computer program called RIGMOOR which was prepared to simplify estimating the static holding power of spread moorings.

TABLE OF CONTENTS

Section	Description	Page
	INTRODUCTION	1
1	THE DEFINITION FUNCTIONS	1-1
2	THE COMPUTATIONAL FUNCTIONS	2-1
3	THE DISPLAY FUNCTIONS	3-1
4	RIGPLOT	4-1
	APPENDICES	
	HELP	HELP-2
	MOOR	MOOR-1
	CASE	CASE-1
	LOAD	LOAD-1
	LOOK	LOOK-1
	NAME	NAME-1
	PREL	PREL-1
	ROSE	ROSE-1
	SHOW	SHOW-1
	VERT	VERT-1
	RIGPLOT	RIGPLOT-1
	INDEX	INDEX-1

LIST OF FIGURES

Figure	Description	Page
1	Typical Spread Mooring for Floating Drilling Platform	1
2	RIGMOOR Control Screens	3
3	An Anchor Pattern with Circular Symmetry	4
1-1	Floating Platform Plan View Showing Fairlead Location Geometry	1-3
1-2	Case Selection on the Title Menu	1-4
3-1	H vs S Listing from Sample Problem by Function 6	3-4
3-2	Partial X vs H Listing from Sample Problem by Function 7	3-6
4-1	Spreadsheet for X vs H File	4-3
4-2	Partial Spreadsheet for H vs S File	4-4
4-3	Spreadsheet for Combined Rose Files	4-5

LIST OF TABLES

Table	Description	Page
3-1	Rose File Header Record Fields	3-2
3-2	Rose File Leg Table Record Fieldss	3-3
3-3	H vs S File Field Synopsis	3-5
3-4	X vs H File Header Record Fields	3-7
3-5	Synopsis of X vs H Mode Table Fields	3-8
4-1	Spreadsheet Column Identification for RIGPLOT File Conversion	4-2

INTRODUCTION

RIGMOOR is a computer model for evaluating the holding capacity of spread moorings (Figure 1). It emphasizes the needs of mooring evaluation rather than mooring design. The goal has been to provide a tool that is easy to use by people who are not mooring specialists, a tool that can be as convenient as a modern personal computer, yet a tool that encompasses the variety of design common to spread moorings. This model is not intended as a tool for monitoring spread moorings in real time. A mooring monitor program[1] supports such real-time functions as relocating the drilling platform within the operational watch circle, optimizing load sharing among legs for a specific weather load and the like.

---

1. Oppenheim, B.W., Manual for Computer Program "BOMOOR 2.00" for Interactive Mooring Analysis, B.W.Oppenheim, PhD & Associates, Inc., Santa Monica, CA, 1984.

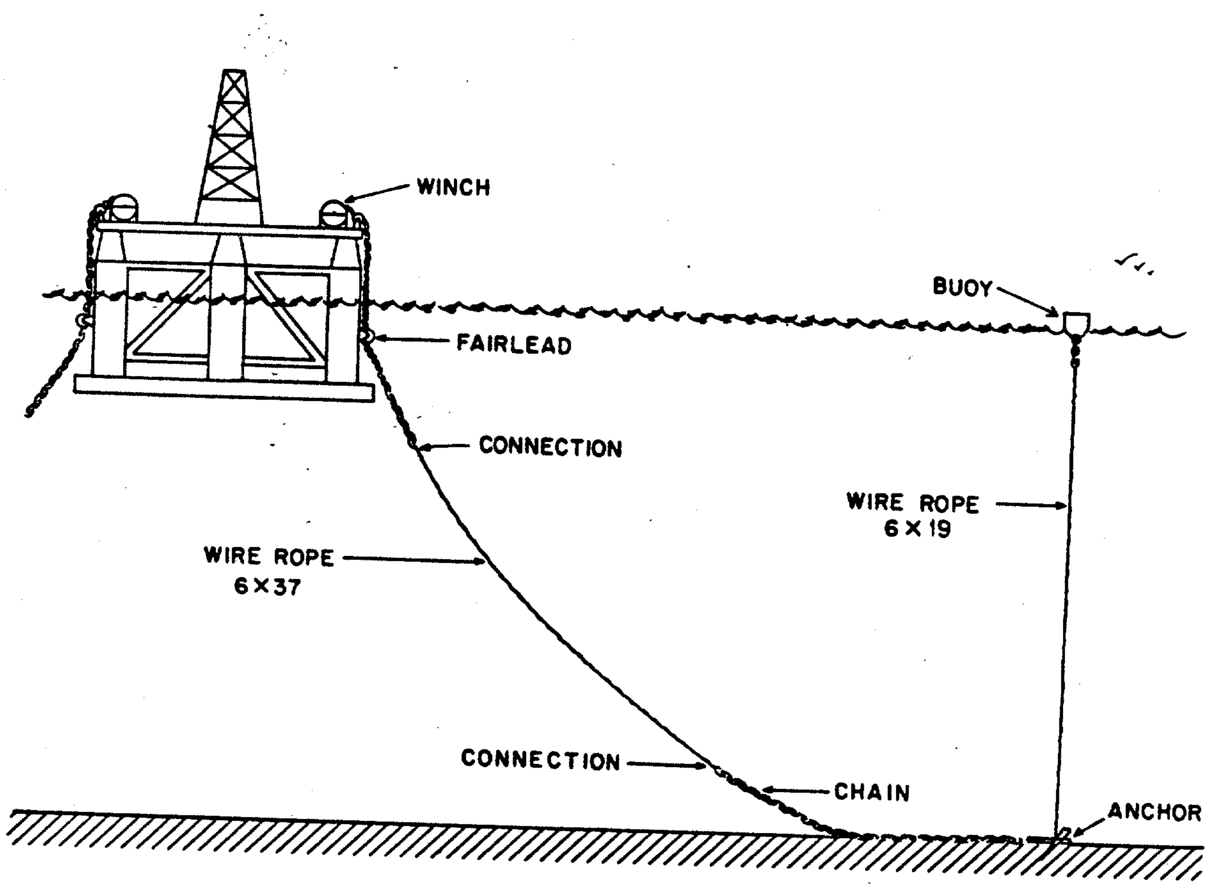

Figure 1. Typical Spread Mooring for Floating Drilling Platform

Consistent with the goal that RIGMOOR should be easy to use, the program is designed around a central menu (Figure 2). The menu lists ten functions performed by RIGMOOR clustered in groups. A summary directive is displayed below the menu: "Enter 0 through 9 or Q". That is, press any numeral or the "Q" key and then press the key marked "ENTER" or "RETURN". This directive is followed by the cryptic note "?=Help: ". This note appears throughout RIGMOOR when an operator entry is desired. It serves as a reminder that the operator may respond to any prompt by entering "?". In response RIGMOOR will display more explanation for the expected entry.

All functions end by re-displaying the function menu.

Functions 0 and 9 relate to defining moorng cases and selecting among them. Functions 1 through 4 perform mooring computations and functions 5 through 8 display results in tables. These groups are discussed in sections 1, 2, and 3.

RIGMOOR accomodates from two to twelve spread mooring legs, deployed from fairleads located around the perimeter of the platform. Spread moorings commonly use a single leg design for simplicity and symmetry. RIGMOOR accomodates this by the concept of leg type: the user describes one leg, and its properties are assigned to all similar legs.

Up to twelve individual types can be defined for a single mooring, which permits analysis of a mooring (probably absurd) in which every leg was different. The sample problem analysed in Volumes I and III illustrates a hypothetical mooring with ten legs and three leg types. The four legs at bow and stern are all chain, the two legs on the beam are wire rope with 9 shots of chain at the anchor and the four diagonal legs are all wire rope.

Since the weight, strength and elasticity of a leg element depend on its size and construction, only the size and construction are required entries. RIGMOOR will estimate the other parameters, using models for high strength stud-link chain and 6x37 Monitor AA wire rope (either fiber core or independent wire rope core). Unusual values can be specified directly.

Oil rigs commonly use one of three kinds of mooring legs - all chain, all wire rope, or a segment of chain near the anchor linked to a segment of wire rope at the upper end. RIGMOOR permits as many as five segments to be coupled in a single leg.

These three models are sufficient for most purposes. The definition of a mooring is stored in a computer disk file. The mooring definition file is a text file that can be edited using ordinary editor or word processing utilities. Punctilious users can change the derived parameters to reflect an unusual leg element, and expert users can create the entire definition file using the editor.

More common in spread moorings for ships and barges than for drilling platforms, bouyant or heavy objects may be placed at any junction between segments in a leg. The location of each buoy (weight) on or below the surface (above the bottom) is determined according to the load in the leg. Buoys and weights may be mingled in a leg, but surfaced buoys may not be nearer the anchor than bottomed clumps, a simplification in accord with common sense.

RIGMOOR

MULTI-LEG SURFACE MOORING DESIGN REVIEW

Version 1.00

DAVID B. DILLON

EG&G OCEAN SYSTEMS GROUP

9220 GAITHER ROAD

GAITHERSBURG, MD 20877

(301) 670-6464

Note: Enter ? in place of any entry to receive on-screen help.

Enter Drive and Rig Name
      A:RIGNAM
?=Help:

a. RIGMOOR Title Screen

EXPLORATORY OIL RIG MOORING LEG ANALYSIS

David B. Dillon   EG&G, Inc.   Current Rig Definition Root: SAMPLE

Entry	Operation
1	Compute operational and survival holding power rose for mooring
2	Adjust anchor leg lengths for mooring preload
3	Compute preload vs scope (H vs S) for each leg type
4	Compute offset vs load (X vs H) for each anchor leg
5	Display and print operational and survival holding power roses
6	Display and print preload vs scope tables
7	Display and print offset vs load tables
8	Display and print current rig definition
9	Select another rig definition file, old or new
0	Define a new rig
Q	Quit

Enter 0 through 9 or Q
?=Help:

b. Main Menu of Functions

Figure 2. RIGMOOR Control Screens

RIGMOOR assumes that the bottom is flat and that the leg fairleads are at a common level on the platform.

Anchor patterns for spread moorings are usually symmetric as shown in Figure 3, where the legs are uniformly spaced around a circle. Circular symmetry offers the most uniform holding power against loads from any direction. Another common pattern has left/right or bow/stern symmetry. This pattern can give greater holding power to resist storms from a prevailing direction. The sample problem has both right/left and bow/stern symmetry. RIGMOOR accepts anchor patterns with or without symmetry. RIGMOOR may be slick, but it won't slide uphill, and solutions may not be attained for highly unsymmetric mooring patterns.

RIGMOOR analyzes each leg and the anchor pattern to determine how the preload should be distributed among the legs in order to center the moonpool over the blowout preventer in the absence of external forces. With symmetric patterns and uniform leg construction, all the legs are loaded to the same preload; an unsymmetric pattern or leg construction will require the legs to have different preloads. RIGMOOR develops a table expressing the ratio of the actual preload in each leg to the average preload for the mooring. The user can enter the average preload or let RIGMOOR estimate it.

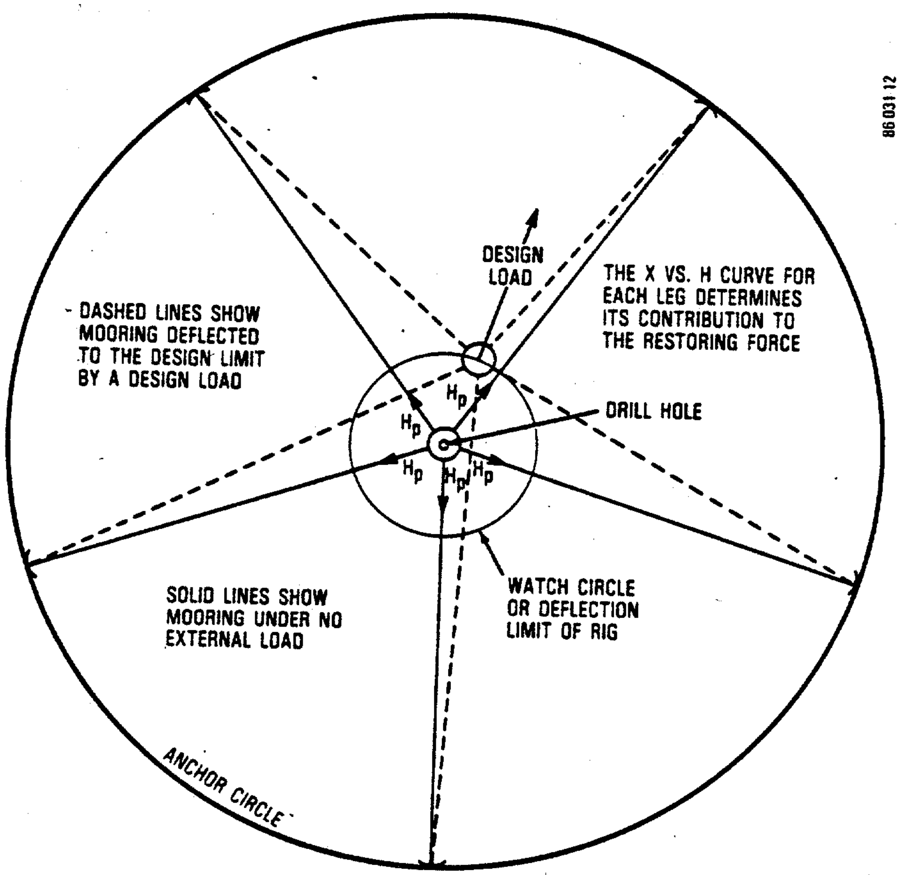

Figure 3. An Anchor Pattern with Circular Symmetry

RIGMOOR's preload minimizes the anchor radius on the philosophy that a leg overload should drag its anchor before parting the leg. Using the theory presented in Appendix B of Volume I, RIGMOOR estimates a preload such that:

The design load will deflect the leg to the design watch circle;

The design load will stress the leg to its working stress limit, i.e. tension divided by safety factor; and

The design load will lift all the leg from the bottom except a reserve of about five percent of the water depth.

RIGMOOR prepares a table of preload vs leg length for each different leg type so that the user can select other preloads.

RIGMOOR has no module to compute the dynamic response of the drilling vessel to ocean waves. These computations require specialized technical knowledge to match hydrodynamic theory with the geometry of the vessel. None of the large computer models that perform these computations will operate within the constraints of the present generation of personal computers. The assumption is justified by the prevalence of semi-submersible floating drilling platforms. The semi-submersible platform is noted for an inherently low response to wave action. RIGMOOR, however, does not ignore the motion of the platform. While it cannot make an estimate of the heave motion of the platform, it can show the quasi-static effects of heave on both holding power and tensile safety factor in the mooring legs. The static holding power can be re-calculated with the vessel displaced up or down from its nominal draft.

RIGMOOR is written in Fortran-77 in order to be easily installed on a variety of computers. Using the $STRICT metacommand of MicroSoft's Fortran compiler, version 3.31, ensures that no proprietary extensions to Fortran-77 are included. It is well-suited for use on any IBM-PC compatible machine with at least 192 Kb memory, one 360 Kb floppy disk drive, and 80 column printer. A second "floppy" makes separating case files from RIGMOOR program files more convenient. A "hard" disk accelerates file operations for any program. Execution time depends upon the number of leg types in a case and the number of segments in a leg. As a rule of thumb, it takes roughly as long to analyse a case (Function 1) as it does to define it (Function 0). A numerical co-processor (8087, etc.) speeds execution substantially, but without it case times are still measured in minutes.

RIGMOOR's preload minimizes the anchor radius on the philosophy that a leg overload should drag its anchor before parting the leg. Using the theory presented in Appendix B of Volume I, RIGMOOR estimates a preload such that:

The design load will deflect the leg to the design watch circle;

The design load will stress the leg to its working stress limit, i.e. tension divided by safety factor; and

The design load will lift all the leg from the bottom except a reserve of about five percent of the water depth.

RIGMOOR prepares a table of preload vs leg length for each different leg type so that the user can select other preloads.

RIGMOOR has no module to compute the dynamic response of the drilling vessel to ocean waves. These computations require specialized technical knowledge to match hydrodynamic theory with the geometry of the vessel. None of the large computer models that perform these computations will operate within the constraints of the present generation of personal computers. The assumption is justified by the prevalence of semi-submersible floating drilling platforms. The semi-submersible platform is noted for an inherently low response to wave action. RIGMOOR, however, does not ignore the motion of the platform. While it cannot make an estimate of the heave motion of the platform, it can show the quasi-static effects of heave on both holding power and tensile safety factor in the mooring legs. The static holding power can be re-calculated with the vessel displaced up or down from its nominal draft.

RIGMOOR is written in Fortran-77 in order to be easily installed on a variety of computers. Using the $STRICT metacommand of MicroSoft's Fortran compiler, version 3.31, ensures that no proprietary extensions to Fortran-77 are included. It is well-suited for use on any IBM-PC compatible machine with at least 192 Kb memory, one 360 Kb floppy disk drive, and 80 column printer. A second "floppy" makes separating case files from RIGMOOR program files more convenient. A "hard" disk accelerates file operations for any program. Execution time depends upon the number of leg types in a case and the number of segments in a leg. As a rule of thumb, it takes roughly as long to analyse a case (Function 1) as it does to define it (Function 0). A numerical co-processor (8087, etc.) speeds execution substantially, but without it case times are still measured in minutes.

SECTION 1

THE DEFINITION FUNCTIONS

FUNCTION 0. Enter the Parameters for a Spread Mooring Case.

Enter 0 to begin a new case. As illustrated in Volume I, Table C-1, you will be asked to enter various parameters that describe the mooring:

- A title for the case.

- The effective water depth. Be careful here. The effective water depth is the depth of water **beneath the mooring leg fairleads.**

- The design deflection limit for the mooring, otherwise known as the watch circle radius. It is usually a few percent of the water depth.

- The design safety factor. The tension in no leg will exceed this safety factor under either design or survival loading. (Loads imposed by heaving (vertical) motion of the rig infringe on the design safety factor.)

- The number of anchors. That is, the number of actual mooring legs. There must be at least two legs in a spread mooring; RIGMOOR can process as many as twelve.

- The number of different legs. All the legs are alike in most spread moorings. For these cases, enter 1. Legs are alike if the only difference is the length of the leg. At the other extreme, all the legs might be different, but the number of different legs cannot exceed the number of anchors.

If there is more than one different leg style, RIGMOOR repeats the following entry requests:

- **Number of segments** in leg type _n_. It is common for oil platform mooring legs to have one segment - all wire rope or all chain - but many rigs use two segments - wire rope plus chain near the anchor. RIGMOOR can process legs with as many as five segments.

  Segments count from the rig towards the anchor. Segment 1 passes through the fairlead; the last segment is attached to the farthest anchor on the leg. Do not declare segments that are not deployed - RIGMOOR always deploys segment 1 at least as far as the fairlead. Lengths inboard of the fairlead are ignored by RIGMOOR.

- Segments are described by five parameters. They are requested all together so you can type them in one entry (separated by commas). You may enter them singly or any other grouping and a new request will be displayed for the remainder. A full-screen prompt precedes the first entry request.

Material code is a number that identifies whether the segment is stud-link chain or wire rope (6x37). RIGMOOR estimates weight, strength and elasticity based on this selection. The coefficients used in this estimation assume the high-grade materials used for rig moorings. Exact weight, strength and elasticity for other grades and constructions can be edited into the case definition file if necessary for a special evaluation.

Diameter is the nominal size of the leg material: 3-inch chain, 3.5-inch wire rope, etc.

Length is important only for segment numbers greater than 1. The length of segment 1 will be adjusted according to preload during the analysis. RIGMOOR sets the length of segment 1 according to the mooring preload, but your entry fixes the length of segments outboard of segment 1. If there is only one segment in a leg, the length is adjusted during processing, so just enter 0 or 1.

Elasticity is the product of Young's Modulus, E, with the metallic cross section of the leg material. Enter 0 to specify an inextensible member. RIGMOOR will substitute a realistic value if you enter -1. Or you can enter your own value for special cases.

Inter-segment loads are buoys or clumps attached at the junctions between segments. The rig does not count as a "buoy", nor does the crown buoy often attached to the anchor to mark its position. Rigs rarely use this feature, but buoys can reduce line tension in deep moorings and clumps modify the displacement function of moorings.

The inter-segment load for a segment is attached to the end of the segment nearest the rig. Be careful to enter the load with the right segment! Remember to enter segments in order from the rig to the anchor, and enter an inter-segment load with the segment **beneath** it. Segment 1 never has an inter-segment load - always enter zero. If you use multiple anchors serially connected, include the weight of inboard anchors as inter-segment loads.

Finally, the location of the fairlead on the rig is requested for each actual anchor leg. This involves four parameters. Unlike the segment parameters, these must be entered in a group, separated by commas. Fairlead locations may be expressed from any reference point on the rig. For consistency, the following definitions are recommended, based on the center of the moon pool as reference point. Figure 1-1 illustrates this geometry.

Measure X from the center of the moon pool to starboard (fairleads on the port side have negative values for X.

Measure Y forward from the center of the moon pool, with aft fairleads negative.

The anchor direction is the angle, measured clockwise from forward, to a radius from the leg fairlead to its anchor.

With completion of the fairlead position and anchor direction entries, the mooring definition file is written and closed. The letters "DF.RIG" are appended to the current root name (title entry or Function 9) to form the name for the rig Definition File.

~~~~~~

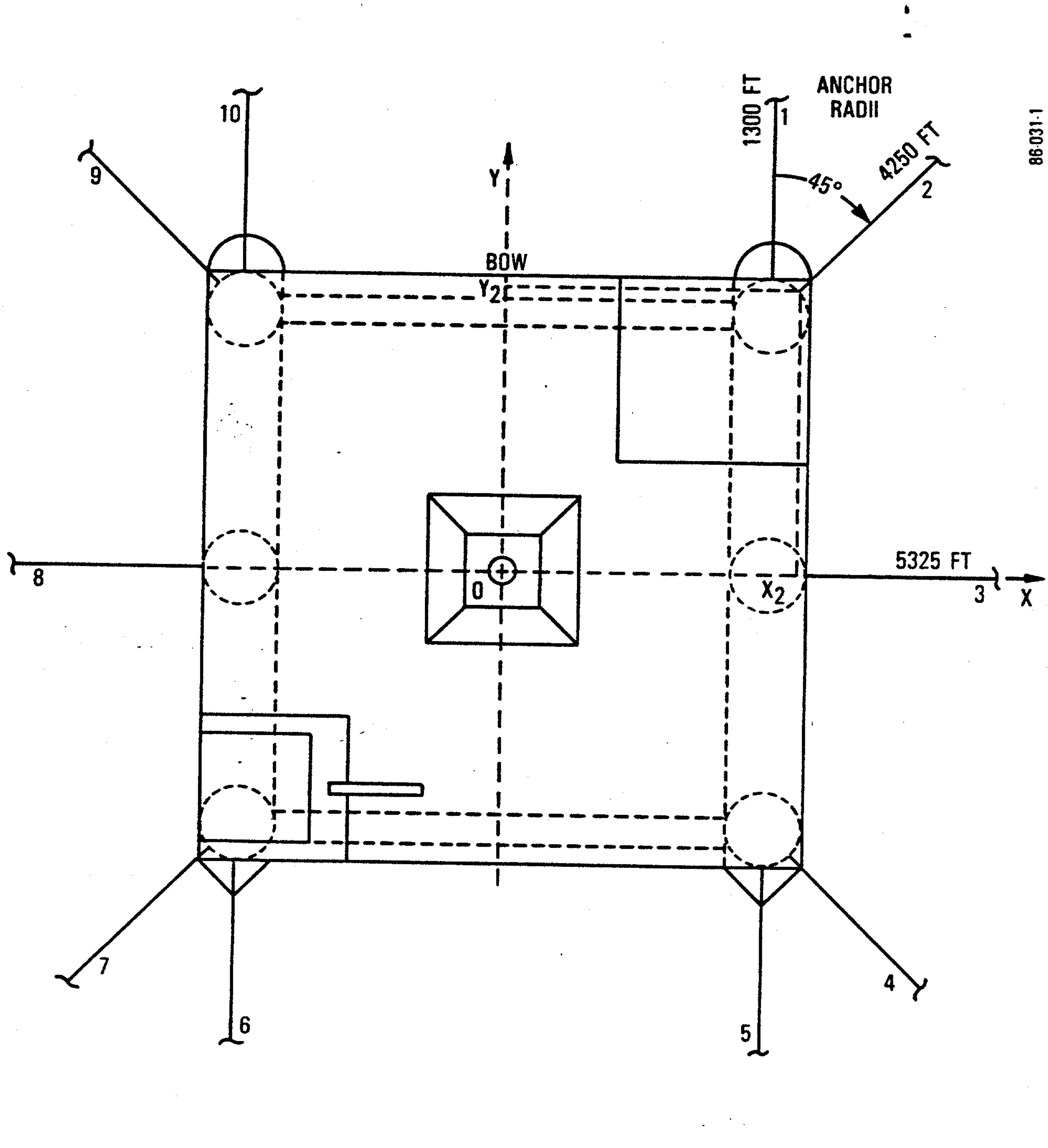

Figure 1-1. Floating Platform Plan View
Showing Fairlead Location Geometry
~~~~~~

FUNCTION 9. Enter another Root Name.

All the files associated with a case have names built around a "root name". The root name for a case has exactly six characters. A root name is entered at the beginning of a session, as part of the title screen (Figure 1-1). Use Function 9 to change the root name. This does not rename files; it allows you to work on more than one case in a RIGMOOR session.

~~~~~~

RIGMOOR

MULTI-LEG SURFACE MOORING DESIGN REVIEW

DAVID B. DILLON

EG&G OCEAN SYSTEMS GROUP

9220 GAITHER ROAD

GAITHERSBURG, MD 20877

(301) 840-3323

Note: Enter ? in place of any entry to receive on-screen help.

Enter Drive and Rig Name
A:RIGNAM
?=Help: C:SAMPLE <cr> [Root name for rig mooring case files may include MS-DOS drive and path.]

Figure 1-2. Case Selection on the Title Menu

~~~~~~

If you enter a root name that has no corresponding files, RIGMOOR will display the note, "New File". This warns you that no mooring has been defined for the new case. Function 0 is the only useful action to take after a "New File" warning.

You may also use Function 9 to change the disk drive (and MS-DOS subdirectory path). Simply precede the root name with a drive specification or drive and path specification. Suppose that "B:OLDCAS" is entered at the title menu beginning a RIGMOOR session and then Function 9 is selected.

Example 1:

Enter Drive and Rig Name
A:RIGNAM
?=Help: C:OLDCAS <cr>

[Prompt for new root name.]
[Example entry image, no path.]
[Change to drive C: use the version of OLDCAS found there, if any.]

Example 2:

Enter Drive and Rig Name
A:RIGNAM
?=Help: C:\MOORING\NEWCAS <cr>

[Change to MS-DOS subdirectory "MOORING" and use new root name "NEWCAS" for files there.]

SECTION 2

THE COMPUTATIONAL FUNCTIONS

FUNCTION 1. Analyse Case and Display Holding Power Rose Tables.

Function 1 is the normal successor to Function 0. It analyses the mooring and displays operational and survival holding power roses for the case. Table C-4 in Volume 1 is a transcript of the session in which the sample problem was analysed.

Function 1 prompts for a preload, for the limits of the holding power rose analysis and for rig heave values.

- **Preload.** RIGMOOR examines the mooring pattern for symmetry. If the pattern is not symmetric, then it may be necessary the preload to vary from leg to leg. This, incidentally, permits RIGMOOR to show the effects of errors in anchor placement, by deliberately mis-stating the anchor direction(s) in Function 0. For convenience, RIGMOOR expresses individual leg preloads as an average preload times a normalized scaling factor. The scaling factor is 1. for all legs of a symmetric mooring.

The list of scaling factors is displayed, along with RIGMOOR's estimate of an optimum preload and the range of average preloads that can be used without violating the bottom condition for fluked anchors (no vertical load on the anchor shackle):

```
Average Preload (U=Use  E=Estimate  Q=Quit)
          Minimum   Maximum   Current
          nnnnn.n   xxxxx.x   ccccc.c
?=Help:
```

You may respond by entering an average preload, "U", "E", or "Q". Entries outside the stated range are rejected. If you enter a preload within the stated range, it becomes the current value and the length of the top segment in each leg is adjusted accordingly. If you enter "U", the holding power roses are computed according to the current preload. "E" restores RIGMOOR's estimate, and "Q" aborts Function 1 and restores the main menu.

- **Holding Power Rose.** After the average preload is set, RIGMOOR requests a range of angles for printing the holding power rose table. This allows you to capitalize on symmetry in the mooring pattern to shorten the rose printout. In a symmetric mooring, you need display only one-half lobe, that is, half the angle between two adjacent legs. (Is this a case where half a lobe is better than no lobe?)

```
Initial, Final, Step Rig Deflection Angles
(Degrees Clockwise from Forward  Q=Quit)
?=Help:
```

To compute the holding power rose, the rig is displaced from its desired position with the moon pool centered over the drill hole to a point on the watch circle. Use the same angular reference as used for anchor directions: 0. degrees means the rig is displaced along a path straight ahead of the bow, 90. degrees is displacement to the starboard beam; and -90 (or 270.) is to the port beam.

Table C-4 in Volume I, pages C-15 through C-17 illustrate an angle entry and the resulting holding power rose table.

- **Rig Heave.** When the deflection angles are entered, RIGMOOR displays the holding power rose table assuming no heave. Heave is the upward displacement of the rig from its normal waterline. RIGMOOR is a static model, but quasi-dynamic results can be simulated by entering the vertical displacements expected in service.

```
    Rig Heave
     Downward: -1 thru -99 Feet
       Upward:  1 thru  99 Feet
Command Menu:  0
?=Help:                          [Enter 1 or 2 numerals,
                                  no decimal point]
```

The numerals for rig heave become part of the rose file name, so the heave must be a whole number. Enter a zero to restore the main menu, ending Function 1. Pages C-17 through C-19 illustrate heave entries.

Upward heave displacement may violate the fluke anchor condition. RIGMOOR does not currently issue a warning. You can prevent this by including upward heave in the value for depth. Then the normal waterline is analysed as a corresponding downward heave - awkward but effective.

FUNCTION 2. Select an Average Preload and Adjust Leg Lengths.

This function performs the first part of Function 1, selecting the preload and adjusting the top segment length in each leg, without computing the holding power.

FUNCTION 3. Write New Preload vs Leg Length Files.

Function 3 is also a subset of Function 1. It creates the load versus top segment length tables by leg type. No user entries are required.

FUNCTION 4. Write New Leg Displacement vs Horizontal Load Files.

Function 4 is a third subset of Function 1. It creates load versus deflection tables for all legs. No user entries are required.

SECTION 3

THE DISPLAY FUNCTIONS

RIGMOOR maintains four sets of files for each mooring case: They are named by a formal convention so that they can be identified in a disk file directory listing. They all begin with the six-character **root name**, followed by a two letter **file type** that identifies the content. The extension field (.ext in MS-DOS and many other systems) identifies the **data version**. Except for the case definition file, which is a common text file, the files are in binary format and cannot be listed directly to the screen or printer. This saves disk space as well as processing time, but requires special listing routines to convert their contents to readable form. Functions 5 through 8 perform this interpretation. These functions require a printer or spooler.

FUNCTION 5. Print Operational and Survival Holding Power Roses.

Function 5 displays the holding power rose files. The **file type** for rose files is OP for an operational rose or SV for a survival rose file. The **data version** shows the rig heave condition. "H00" indicates that the file represents a holding power rose for a rig floating at its normal waterline. "H" denotes upward heave displacement. the two numerals following the H (00 through 99) indicate the upward displacement in feet. "-nn" is the data version for downward heave, where "nn" are the numerals for the distance. Thus, BIGRIGOP.H09 is the OPerational rose file for case BIGRIG, with the rig heaved upward 9 feet and BIGWIGSV.-13 is the survival rose for case BIGWIG with the rig down 13 feet.

The rose files contain much more information than the run-time displays such as Tables 4-1, 4-2 and 4-3 in Volume I. Table C-4 in Volume I, pages C-22 through C-25, shows partial listings of operational and survival rose files for the sample problem. The rose file contains a header record and leg table for each line of the run-time display. The header record holds the eight values described in Table 3-1. Table 3-2 describes the eight fields in the table of leg loads. There is a leg load record for each leg of the mooring. Rose file listings run about ten pages each, with 3 - 5 angles per page.

FUNCTION 6. Print Preload *versus* Top Segment Length Tables.

These tables are produced by implementing the procedure described in Volume I, Appendix B, Figure B-3. An H *vs* S file is created for each different leg type by Function 1 or 3. The **file type** is HS and the **data version** is Lnn, where nn is leg type number. Thus BIGRIGHS.L02 is the H *vs* S file for leg type 2 of case BIGRIG. Only rarely do rig moorings have more than one leg type, so there will only be one HS file. Figure 3-1 is a listing of an HS file and Table 3-3 is the synopsis of its fields. H *vs* S listings are specific to a fixed depth and leg construction, except that the length of the top segment changes from line to line.

FUNCTION 7. Print Offset <u>versus</u> Horizontal Load Tables.

The **file type** for X vs H files is XH, and the **data version** is the same as for H vs S files. X vs H files are specific to depth and leg construction, including total length. The horizontal load is incremented from 0. until the tensile safety factor is exceeded. The load increment is chosen to provide about forty records in the file. Like the rose files, X vs H records are structured, with a header and a node table. Nodes represent ends of segments: there are at least two nodes in a leg plus another for each extra segment. Figure 3-2 is a fragment of an X vs H listing produced by by Function 7. Table 3-4 lists the header fields and Table 3-5 describes the node table. The X vs H listing typically runs about ten pages per leg.

~~~~~~

Table 3-1

Rose File Header Record Fields

Field Name	Content
Deflection Direction	This is the angular position of the rig clockwise around the watch circle in degrees, using the same reference as the anchor directions in the mooring definition file.
Holding Power	The vector sum of individual leg holding powers in pounds.
Weather Direction	The **downwind** direction of the total environmental force which the mooring can resist at that position on the watch circle, in degrees. It is the vector direction of the holding power.
X & Y Force Components	Components of holding power, resolved by weather direction. X and Y axis directions coincide with fairlead coordinates in mooring definition file. The forces are of **the mooring on the rig.** Their reverse image is the environmental force required to displace the rig to its current position on the watch circle.
Safety Factor	The least ratio of breaking strength to tension in any segment of any leg.
CW Moment	The residual clockwise torque of the mooring legs on the rig uncorrected in RIGMOOR's convergence algorithm. Zero for exact solution. The normalizing factor is the average preload times the watch circle radius.
Yaw Angle	The clockwise rotation of the rig required to cancel leg moments produced by the displacement to the current position on the watch circle.
~~~~~~

FUNCTION 8. Print Mooring Definition File.

The mooring definition file is produced by function 0. Its **file type** is DF and the **data version** is .RIG: BIGRIGDF.RIG. It is the only text file used by RIGMOOR. It is described in detail on Table C-3 in Volume I. Being a text file, experienced RIGMOOR users can create new cases by modifying existing DF.RIG files with a text editor or word processor. DF.RIG files rarely exceed a half-page listing. Table C-2 in Volume I shows how to build a DF file with even less typing: it is a text file, say QUICK.IN, containing the verbatim responses to every RIGMOOR prompt used to start and perform function 0. MS-DOS re-directs keyboard input with the command entry:

A>RIGMOOR < QUICK.IN <cr>.

MS-DOS replaces keyboard input to RIGMOOR with text from QUICK.IN, line by line. A similar procedure saves screen displays for a session:

A>RIGMOOR > SESSION.SAV <cr>.

That is how the sample sessions in Appendix C of Volume I were prepared. Both input and output options can be used at once.

Table 3-2

Rose File Leg Table Record Fields

Field	Content
Anchor	Anchor number.
Load	Horizontal load in leg, pounds.
Span	Horizontal radius from leg fairlead to anchor, feet.
X,Y Load	Components of horizontal load, using axis directions from fairlead positions in mooring definition file, pounds.
Safety	Least ratio of breaking strength to tension in any segment of the leg.
CW Moment	The clockwise moment exerted by the leg load on the rig, computed about the origin of fairlead positions, normalized by dividing it by the product of average preload times watch circle radius.
Active	T (True) or F (False). All legs are active in OPerational rose files; Leeward legs are slacked (not active) in SurVival files.

B:SAMPLE

H vs S for Leg Type 1

Top Scope	- Force/Length on Bottom - Slack	Preload	Design -	Pre-Span	Design Span	Pre-Load	Design Load	Holding Power
312.	0.	236.	203036.	2.	24.	27.	16234.	16207.
350.	-38.	3228.	178183.	139.	161.	5955.	98678.	92723.
400.	-88.	4090.	152267.	230.	252.	12433.	135282.	122849.
450.	-138.	4167.	131665.	304.	326.	18714.	155404.	136690.
500.	-188.	3866.	114794.	370.	392.	25004.	168251.	143247.
550.	-238.	3327.	100636.	432.	454.	31364.	177083.	145719.
600.	-288.	2611.	88512.	492.	513.	37800.	183444.	145643.
650.	-338.	1738.	77958.	549.	571.	44281.	188171.	143890.
700.	-388.	752.	68625.	605.	627.	50823.	191773.	140949.
750.	-438.	-5.	60279.	660.	682.	57353.	194557.	137204.
800.	-488.	-21.	52734.	715.	736.	63707.	196736.	133029.
850.	-538.	-41.	45847.	768.	790.	69776.	198454.	128678.
900.	-588.	-63.	39508.	821.	843.	75473.	199813.	124339.
950.	-638.	-89.	33633.	874.	896.	80707.	200885.	120177.
1000.	-688.	-117.	28148.	926.	948.	85436.	201727.	116291.
1050.	-738.	-148.	23000.	978.	999.	89576.	202378.	112802.
1100.	-788.	-183.	18143.	1029.	1051.	93081.	202871.	109791.
1150.	-838.	-221.	13543.	1080.	1102.	95899.	203229.	107330.
1200.	-888.	-262.	9157.	1131.	1153.	98068.	203475.	105407.
1250.	-938.	-306.	4968.	1182.	1203.	99456.	203621.	104165.
1300.	-988.	-353.	953.	1232.	1254.	100185.	203679.	103494.
1350.	-1038.	-402.	-38.	1282.	1304.	100402.	203681.	103280.
1400.	-1088.	-451.	-88.	1332.	1354.	100594.	203681.	103087.
1450.	-1138.	-500.	-138.	1382.	1404.	100769.	203681.	102912.
1500.	-1188.	-549.	-188.	1432.	1454.	100974.	203681.	102706.

Force/Length on Bottom:
Positive values are upward force on anchor,
Negative values are cable length on bottom.

Figure 3-1. H vs S Listing
from Sample Problem by Function 6

Table 3-3

H vs S File Field Synopsis

Field	Content
Top Scope	Length of segment 1 between its fairlead on the rig and its junction with segment 2 or the anchor shackle, feet. The file starts with the least possible value to reach from the fairlead to the anchor and increases this length until segment 1 lies along the bottom under the design load.
Vertical Force on Anchor or Length Lying on Bottom	
	This group of three fields uses a special notation to distinguish two mutually exclusive conditions. Positive values in any of these columns represent the upward force in pounds that the leg exerts on the anchor. A minus sign means that the value is the length of the leg in feet that is lying along the bottom. The length is positive; the - sign is just a flag.
- Slack	Leg is suspended with no horizontal load imposed.
- Preload	Leg is restraining the indicated horizontal preload.
- Design	Leg is restraining the indicated horizontal design load.
Pre-Span	Radius, feet, from fairlead to anchor under preload. Pre-Span is Design Span - watch circle radius.
Design Span	Radius, feet, from fairlead to anchor under design load.
Pre-Load	Horizontal load required to produce Pre-Span.
Design Load	Horizontal load required to produce design safety factor at some point in leg.
Holding Power	Design Load - Pre-Load.

Anchor 3 X vx H for Leg Type 3 Page 1

Horizontal Load	Horizontal Span	Length on Bottom	Stretched Length	Safety Factor	at Node	Nodes on Surface	Nodes on Bottom
.0	5688.0	5688.0	6000.0	166.064	1	1	2

Node	Node Position Span	Depth	Segment Span	Increment Depth	Stretched Length	Segment Downward Force Rig end	Anchor end
1	.00	.00	5148.00	-312.01	5460.01	4517.9	.0
2	5148.00	312.01	540.00	.00	540.00	.0	.0
3	5688.00	312.01					

Horizontal Load	Horizontal Span	Length on Bottom	Stretched Length	Safety Factor	at Node	Nodes on Surface	Nodes on Bottom
20000.0	5934.4	5020.8	6002.1	30.603	1	1	2

Node	Node Position Span	Depth	Segment Span	Increment Depth	Stretched Length	Segment Downward Force Rig end	Anchor end
1	.00	.00	5394.25	-312.00	5461.91	14179.0	.0
2	5394.25	312.00	540.14	.00	540.14	.0	.0
3	5934.39	312.00					

Horizontal Load	Horizontal Span	Length on Bottom	Stretched Length	Safety Factor	at Node	Nodes on Surface	Nodes on Bottom
40000.0	5955.5	4651.1	6004.1	16.854	1	1	2

Node	Node Position Span	Depth	Segment Span	Increment Depth	Stretched Length	Segment Downward Force Rig end	Anchor end
1	.00	.00	5415.20	-312.00	5463.80	19533.3	.0
2	5415.20	312.00	540.28	.00	540.28	.0	.0
3	5955.48	312.00					

Horizontal Load	Horizontal Span	Length on Bottom	Stretched Length	Safety Factor	at Node	Nodes on Surface	Nodes on Bottom
60000.0	5966.2	4362.9	6006.1	11.397	2	1	2

Node	Node Position Span	Depth	Segment Span	Increment Depth	Stretched Length	Segment Downward Force Rig end	Anchor end
1	.00	.00	5425.80	-312.03	5465.70	23706.3	.0
2	5425.80	312.03	540.42	.00	540.42	.0	.0
3	5966.22	312.03					

Figure 3-2. Partial X vs H Listing
from Sample Problem by Function 7

Table 3-4

X vs H File Header Record Fields

Field Name	Content
Horizontal Load	Leg load in pounds. It is the independent variable of the file.
Horizontal Span	Radius in feet from the leg fairlead to the anchor.
Length on Bottom	Length, feet, leg lying along bottom near anchor.
Stretched Length	Arc length of leg, fairlead to anchor. Same as nominal length if leg is inelastic. Stretch ignores bottom friction.
Safety Factor	Least safety factor in any segment of leg. Ratio of strength to tension.
at Node	End of segment where least safety factor occurs. Nearly always node 1, where segment 1 joins the rig, unless there is a subsurface buoy or the material size changes between segments.
Nodes on Surface	Number of nodes connected to buoys afloat. The rig always counts as 1. If a leg has buoys, they may be on the surface under slight load but pull under at larger loads. This field tracks buoy pullunder.
Nodes on bottom	Number of nodes lying on the bottom. The anchor node always counts as 1. If a leg has more than one segment, one or more of the segments may lie entirely on the bottom. If there is a clump weight at a node, it may be raised. This field tracks clump liftoff.

Table 3-5

Synopsis of X vs H Node Table Fields

Field Name	Content
Node	Segment junction index: 1 = rig fairlead end of segment 1; N+1 = anchor end of segment N.
Span	Horizontal radius of node from rig fairlead, feet. Must be 0. for node 1.
Depth	Depth of node below rig fairlead, feet. Must be 0. for node 1.
Segment Span	Horizontal span of segment "below" node, feet.
Increment Depth	Change in Z, feet, for segment below node. Z is positive upwards, so negative values mean node N+1 is below node N.
Stretched Length	Arc length, feet, of segment. Same as material length for inelastic segments.
Segment Downward Force	
- Rig End	Downward force of segment N acting at node N on segment N-1 ("Segment 0" is the rig). It is physically possible for this value to be negative, but never practiced in oil platform moorings. "Rig end" means the end of a segment nearest the rig.
- Anchor End	Downward force of segment N+1 acting at node N+1 on segment N. If there are K segments, "segment K+1" is the anchor at node K+1. This table is redundant for oil platform moorings, but if a leg has a buoy or clump at a node, then the downward force changes at the node. If the length lying along the bottom exceeds 0., then the downward force at the anchor node must be zero. If the length on the bottom is zero, there will be a downward force of the anchor node on the last segment. That is, the last segment pulls up on the anchor. This field also tracks the residual force of a bottom clump that is partially supported by the mooring upward force.

SECTION IV

RIGPLOT

RIGPLOT is a utility program that copies RIGMOOR case files into a form that the leading spreadsheet programs can read. This allows you to use the plotting commands of the spreadsheet program to view RIGMOOR results quickly. Using RIGPLOT is trivial. It asks you for the name of a RIGMOOR case file. After you enter it, the file is opened and copied into a second file in "Comma Separated Value" (CSV) format. The process is repeated until you enter N instead of a file name. The SuperCalc manual tells how to make plots from the CSV file. Several figures showing sample problem results in Volume I were prepared using RIGPLOT. RIGPLOT is coded in Fortran-77.

RIGMOOR maintains four sets of files for each case analysed. They are named using a formal code to distinguish them:

d:\path\rrrrrrXH.Lnn	Leg displacement vs Horizontal load,
d:\path\rrrrrrHS.Lnn	Preload vs Segment 1 length,
d:\path\rrrrrrOP.hnn	Operational holding power rose table, and
d:\path\rrrrrrSV.hnn	Survival holding power rose table.

The underlined portion of the name is MS-DOS specific, identifying the disk drive and subdirectory path. RIGPLOT places the CSV file in the same subdirectory as its source case file.

The letters, rrrrrr, represent the six character **root name** for the case. The two-letter **file type**, XH, HS, OP, or SV marks their content as listed above. The last three letters in each name hold the **data version**. For XH and HS files, the L stands for Leg and **nn** represents two numerals. L01 through L12 are possible. For OP and SV files, **nn** in the data version identifies the heave of the rig, in feet. The **h** will be H for upward heave or - for downward heave.

RIGPLOT checks your entry for X, H, O or S in the first position of the file type. If not, the entry is rejected. If X or H, it is replaced with P (for Plot), giving the following formal XH and HS names for RIGPLOT CSV output files:

d:\path\rrrrrrPH.Lnn	Leg displacement vs Horizontal load;
d:\path\rrrrrrPS.Lnn	Preload vs Segment 1 length.

OP and SV files are prepared by RIGMOOR in pairs. If you specify either of these names to RIGPLOT, the other is combined with it in a CSV file whose name form is:

d:\path\rrrrrrPP.hnn	Combined holding power rose tables.

Refer to your spreadsheet user's manual to see how to convert CSV files to spreadsheets and how to prepare plots from spreadsheet data.

Table 4-1 lists the field names (taken from the corresponding tables in Section 3) by spreadsheet column for each of the three CSV file types.

Figure 4-1 shows the spreadsheet version of an X vs H file taken from the sample problem. RIGPLOT only extracts the first five header fields from X vs H records. Figure B-3 in Volume I is plotted from columns A and B of this spreadsheet.

Figure 4-2 is the spreadsheet for the H vs S file for leg type 1 of the sample problem. Columns G, H and I are plotted against column A on Volume I, Figure B-4.

Operational and survival rose tables are combined on the spreadsheet shown as Figure 4-3. Volume I, Figures 4-3, 4-4, and 4-5 are plotted from this spreadsheet.

~~~~~~

Table 4-1
Spreadsheet Column Identification for RIGPLOT File Conversion

Column	Figure 4-1 Deflection rrrrrrPP.CSV	Figure 4-2 Preload rrrrrrPS.CSV	Figure 4-3 Holding Power rrrrrrPH.CSV
A	Horizontal Load	Top Scope	Deflection Dir.
B	Horizontal Span	Slack Force/Length	Oper. Holding Pwr.
C	Length on Bottom	Preload " "	Oper. Weather Dir.
D	Stretched Length	Design " "	Oper. X-Hold. Pwr.
E	Safety Factor	Pre-Span	Oper. Y-Hold. Pwr.
F		Design Span	Oper. Safety Factor
G		Preload	Oper. CW Moment
H		Design Load	Oper. Yaw Angle
I		Holding Power	Deflection Dir.
J			Surv. Holding Pwr.
K			Surv. Weather Dir.
L			Surv. X-Hold. Pwr.
M			Surv. Y-Hold. Pwr.
N			Surv. Safety Factor
O			Surv. CW Moment
P			Surv. Yaw Angle
~~~~~~

	A	B	C	D	E
1	0	1036.908	1036.908	1348.926	28.10354
2	20000	1209.688	841.7169	1349.288	15.42758
3	40000	1243.996	703.1199	1349.649	10.63209
4	60000	1261.525	589.4539	1350.008	8.110826
5	80000	1272.708	490.7368	1350.367	6.556046
6	100000	1280.703	402.3472	1350.724	5.50157
7	120000	1286.803	321.515	1351.081	4.739203
8	140000	1291.698	246.6978	1351.437	4.162487
9	160000	1295.75	176.6672	1351.793	3.710888
10	180000	1299.195	110.6512	1352.149	3.347703
11	200000	1302.172	47.92957	1352.504	3.049227
12	220000	1304.789	0	1352.859	2.7996
13	240000	1306.966	0	1353.214	2.587174
14	260000	1308.756	0	1353.569	2.404113
15	280000	1310.253	0	1353.925	2.244804
16	300000	1311.535	0	1354.281	2.104994
17	320000	1312.66	0	1354.638	1.981378
18	340000	1313.654	0	1354.995	1.871308
19	360000	1314.547	0	1355.351	1.772703
20	380000	1315.355	0	1355.708	1.683869
21	400000	1316.109	0	1356.066	1.603461
22	420000	1316.804	0	1356.423	1.530319
23	440000	1317.458	0	1356.78	1.463518
24	460000	1318.074	0	1357.137	1.402269
25	480000	1318.661	0	1357.495	1.345916
26	500000	1319.223	0	1357.852	1.293894
27	520000	1319.762	0	1358.21	1.245726
28	540000	1320.283	0	1358.567	1.201001
29	560000	1320.788	0	1358.925	1.159364
30	580000	1321.279	0	1359.283	1.120507
31	600000	1321.757	0	1359.64	1.084162
32	620000	1322.226	0	1359.998	1.050094
33	640000	1322.685	0	1360.356	1.018095
34	660000	1323.136	0	1360.713	.9879843
35	680000	1323.575	0	1361.071	.9595952
36					

Figure 4-1. Spreadsheet for X vs H File

	A	B	C	D	E	F	G	H	I
1	312	-.000410	235.5532	203035.9	1.719734	23.55973	26.93271	16234.23	16207.3
2	350	-38.0312	3228.099	178183.3	138.8512	160.6912	5954.666	98677.55	92722.88
3	400	-88.0312	4090.202	152266.5	230.0331	251.8731	12432.52	135281.5	122849
4	450	-138.031	4167.386	131665.2	303.8163	325.6563	18713.54	155403.8	136690.2
5	500	-188.031	3865.917	114794	370.1169	391.9568	25003.54	168250.9	143247.4
6	550	-238.031	3327.381	100635.7	432.2065	454.0465	31363.99	177083.2	145719.2
7	600	-288.031	2610.92	88511.95	491.5984	513.4384	37800.46	183443.9	145643.4
8	650	-338.031	1738.1	77957.8	549.1099	570.95	44281.07	188170.7	143889.7
9	700	-388.031	751.8186	68624.56	605.2629	627.1029	50823.4	191772.5	140949.1
10	750	-438.031	-4.62268	60279.33	660.3566	682.1966	57352.71	194557.1	137204.4
11	800	-488.031	-21.3362	52734.13	714.6165	736.4565	63706.77	196736.2	133029.4
12	850	-538.031	-40.8311	45847.47	768.1982	790.0383	69775.8	198454	128678.2
13	900	-588.031	-63.1975	39508.4	821.2162	843.0563	75473.24	199812.6	124339.4
14	950	-638.031	-88.6020	33632.86	873.7495	895.5895	80707.3	200884.7	120177.4
15	1000	-688.031	-116.976	28147.7	925.8668	947.7068	85435.76	201726.8	116291
16	1050	-738.031	-148.463	23000.22	977.6118	999.4518	89576.22	202378.4	112802.2
17	1100	-788.031	-183.088	18143.48	1029.023	1050.863	93080.77	202871.4	109790.7
18	1150	-838.031	-220.912	13542.68	1080.125	1101.964	95899.03	203229.2	107330.2
19	1200	-888.031	-261.604	9156.997	1130.954	1152.794	98068.44	203475	105406.6
20	1250	-938.031	-305.792	4968.466	1181.515	1203.355	99455.92	203620.5	104164.6
21	1300	-988.031	-352.693	952.5515	1231.826	1253.666	100184.6	203678.9	103494.3
22	1350	-1038.03	-401.780	-37.8527	1281.966	1303.806	100401.6	203681.4	103279.8
23	1400	-1088.03	-450.971	-87.8472	1332.096	1353.936	100594.3	203681.2	103086.9
24	1450	-1138.03	-500.264	-137.832	1382.226	1404.066	100768.7	203680.6	102912
25	1500	-1188.03	-549.324	-187.810	1432.355	1454.195	100973.7	203679.8	102706.1
26	1550	-1238.03	-598.633	-237.846	1482.491	1504.331	101163	203681.1	102518.1
27	1600	-1288.03	-647.640	-287.846	1532.623	1554.463	101375.5	203681.2	102305.7
28	1650	-1338.03	-696.940	-337.846	1582.755	1604.594	101557.5	203681.1	102123.6
29	1700	-1388.03	-746.168	-387.846	1632.886	1654.726	101752.6	203681.2	101928.6
30	1750	-1438.03	-795.303	-437.846	1683.018	1704.858	101948.1	203681.2	101733.1
31	1800	-1488.03	-844.505	-487.846	1733.15	1754.99	102138.2	203681.2	101543
32	1850	-1538.03	-893.690	-537.846	1783.281	1805.121	102333.6	203681.2	101347.6
33	1900	-1588.03	-942.884	-587.846	1833.413	1855.253	102526.6	203681.2	101154.5
34	1950	-1638.03	-992.092	-637.846	1883.545	1905.385	102717	203681.2	100964.1
35	2000	-1688.03	-1041.27	-687.846	1933.676	1955.516	102912	203681.2	100769.1

Figure 4-2. Partial Spreadsheet for H vs S File

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	0	494178.5	-4.53e-7	.0039063	-494179.	3.000766	0	0	0	683277	0	0	-683277	3.000766	0	-5.96e-8
2	5	493301.2	4.026693	-34640.2	-492084.	3.007529	.0070049	.0002968	5	682226.6	2.576881	-30672.8	-681537.	3.007529	.0144775	-.000110
3	10	491052.8	8.043487	-68710.4	-486222.	3.029809	.0141267	-.000090	10	679460.4	5.134543	-60808.2	-676734.	3.029809	.0286194	.0000741
4	15	486943.8	12.28184	-103583	-475799.	3.06837	.0214504	.0001125	15	674981.8	7.7528	-91054.6	-668812	3.06837	.0451629	-.000107
5	20	481860.2	16.45006	-136453.	-462136.	3.123965	.0275884	-.000024	20	686347.8	16.37327	-193477.	-658513.	3.123965	.0534939	.0000999
6	25	475493.6	20.77243	-168637.	-444585.	3.145724	.0325810	.0002789	25	680452.6	18.48321	-215722.	-645353.	3.145724	.0677326	.0000659
7	30	467834.2	25.17392	-199001.	-423400.	3.079854	.0354480	-.000049	30	672742.9	20.57674	-236443.	-629824.	3.079854	.0807813	.0001792
8	35	458954.8	29.93803	-229047.	-397714.	3.036567	.0399849	-.000128	35	636529.3	29.88024	-317112.	-551915.	3.036567	.0774678	-.000033
9	40	449942.7	34.60963	-255559.	-370321.	3.01121	.0414856	.0000558	40	628406.5	32.7682	-340120.	-528406.	3.01121	.0862053	.0000724
10	45	440231.7	39.53176	-280210	-339538.	3.002758	.0415789	.0001369	45	618932	35.76688	-361759.	-502203.	3.002758	.0934479	-.000055
11	50	431318.8	44.61435	-302929.	-307034.	3.01121	.0428256	-.000075	50	609441.6	38.80439	-381915.	-474932.	3.01121	.1060311	.0001121
12	55	423004.3	49.79022	-323042.	-273087.	3.036567	.0405377	.0000381	55	599673.9	41.83158	-399948.	-446822.	3.036567	.1110831	-.000039
13	60	415244.8	55.33841	-341549.	-236162.	3.079911	.0368404	-.000005	60	589133.4	45.03893	-416863.	-416297.	3.079911	.1142668	-.000200
14	65	409112.2	60.63084	-356532.	-200643.	3.145724	.0346613	.0002100	65	579531.2	48.01051	-430747.	-387703	3.145724	.1255718	-.000026
15	70	403312.3	66.34193	-369417.	-161840.	3.133374	.0276210	.0001385	70	580156.6	57.80841	-490970.	-309080.	3.133374	.1194584	-.000187
16	75	398157.8	72.40277	-379526.	-120373.	3.075299	.0219031	.0000753	75	571646.3	60.48253	-497450.	-281644.	3.075299	.1290231	.0000315
17	80	395433.4	78.23358	-387124.	-80637.7	3.035716	.0149666	.0001273	80	510868.2	81.77748	-505617.	-73063.3	3.035716	.0116723	-.000019
18	85	393146.1	84.04037	-391021.	-40819.5	3.011881	.0076251	.0002534	85	509894.6	85.83688	-508549.	-37016.5	3.011881	.0060913	.0000257
19	90	392432.2	90	-392432.	-.015625	3.003936	0	-.000001	90	509642	90	-509642	-.015625	3.003936	0	-.000001
20																

Figure 4-3. Spreadsheet for Combined Rose Files

! 1 PROGRAM INTRODUCTION ;MODULE RIGMOOR.MSG EG&G 1985

RIGMOOR

MULTI-LEG SURFACE MOORING DESIGN REVIEW

Version 1.00
14 February, 1986

DAVID B. DILLON

EG&G OCEAN SYSTEMS GROUP

9218 GAITHER ROAD

GAITHERSBURG, MD 20877

(301) 670-6464

Note: Enter ? in place of any entry to receive on-screen help.@

! 2 INTRODUCTORY HELP

A multi-leg surface mooring consists of a floating buoy, such as a semi-submersible oil drilling rig, restrained from drifting by two or more flexible members, called legs, whose outboard ends are anchored. RIGMOOR computes the equilibrium position of the rig under calm conditions as well as the displacement produced by the force of wind, waves and current acting on the rig.

You will be asked to supply the mooring geometry, the construction of each leg, and the range of forces to be applied to the mooring.

The symbol <cr> means "Press the key marked RETURN or ENTER." The word "enter" means "Type a value, then press the RETURN or ENTER key." If more than one value is requested, separate them by a comma. You may type leading spaces for clarity, but spaces must not follow whole numbers:

Entry	Interpreted as	
12, 34<cr>	12 and 34	OK
12 ,34<cr>	120 and 34	WRONG
12. , 34.5	12 and 34.5	OK

A help message will be displayed if you enter a ? in place of an entry.@

! 3 MAIN MENU, PART 2

Entry	Operation
1	Compute operational and survival holding power rose for mooring
2	Adjust anchor leg lengths for mooring preload
3	Compute preload vs scope (H vs S) for each leg type
4	Compute offset vs load (X vs H) for each anchor leg
5	Display and print operational and survival holding power roses
6	Display and print preload vs scope tables
7	Display and print offset vs load tables
8	Display and print current rig definition
9	Select another rig definition file, old or new
0	Define a new rig
Q	Quit

Enter 0 through 9 or Q@

! 4 WATER DEPTH HELP

The water depth is the vertical distance from the leg fairlead sheaves to the seafloor.

It must be entered in feet, and may not be zero or negative.@

! 5 ANCHOR COUNT HELP

A rig mooring has from 6 to 12 legs extending from fairleads on the rig to anchors on the sea floor.

The number of anchors corresponds to the number of actual mooring legs, one for each fairlead, even if two or more anchors are used to secure a single leg.@

! 6 LEG TYPE COUNT HELP

Multi-leg moorings often use one design for all the legs, but sometimes the legs are not all alike. In the extreme case, each leg of a mooring is different from the others. This complexity is rarely desirable.

You have specified the number of fairlead/anchor legs for the mooring. Now enter the number of different designs used among these legs.@

! 7 ANCHOR PATTERN HELP

A rig mooring has from 6 to 12 legs extending from fairleads on the rig to anchors on the sea floor. Locate each fairlead, measuring Y in feet forward of the rotary table and X in feet to starboard (right).

```
   Rig Mooring in Plan View
                 # <-- Anchor 1
     forward    / <--- Leg 1
        Y      /
  \     ^   |-/ <----- Direction Angle
   \    |   |/
    *---+---* <------- Fairlead at
    | y/|   |             (x,y)
        +---+--> X
    |  rig  |\x
    *-------*
   /  stern  \
  /           \ <--- Leg 2

   @
```

Enter the direction from the fairlead to its anchor, using 0 degrees for an anchor straight forward of its fairlead, 90 deg. for the starboard beam, -90 for the port beam, and 180 for an anchor astern of its leg fairlead.

You have specified the number of different leg designs for your RIGMOOR model, and the characteristics of the segments in each design. Enter a leg type for each fairlead/anchor.

! 8 LEG DEFINITION

Multi-leg moorings often use one design for all the legs, but sometimes the legs are different. It is rarely desireable to have all the legs unique.

A leg has 1 to 5 segments of wire rope and/or chain, connected end-to-end. A weight or buoy may attached at any intersegment junction. Two legs are identical if they have the same number of segments and the properties of each segment are the same:

- Material (Wire rope or Chain),
- Size (Nominal Diameter),
- Length (except for the topmost segment of a leg),
- Elasticty, and
- Intersegment weight or buoyant loads.

An intersegment load is entered with the parameters of the segment beneath it. The lengths of the topmost segments may differ between otherwise identical legs because their exact length is adjusted for preload before computing each case.

In this step, you are describing each different leg, segment by segment. You will assign a leg design to each anchor in a later step.@

! 9 SEGMENT DEFINITION

A mooring leg consists of 1 to 5 chain or wire rope segments connected end to end, counting downward from the fairlead on the drilling rig.

```
     o
   /_\   <=Rig
^-|_|-^- Waterline -^-^-
    |
    |    <=Seg. 5000 ft.
    |        1   3 in. IWRC
    |         EA = 58 Mlb.
   (#)   <=Clump 10000 lb.
    |
    0    <=Seg. 540 ft.
    |        2   3 in. Chain
    0         EA by RIGMOOR
_(-+___<=Anchor___________
XXXXXXXX Bottom XXXXXXXXX

Seg Entry_______________
 1  2,3,5000,58000000 <cr>
 2  1,3,540,-1,-10000 <cr>@
```

A code number identifies the segment material: 1=Stud-link Chain, 2=IWRC and 3=Fiber-core Wire Ropes

The segment diameter is in inches, using decimal fractions: 2.25, not 2-1/4. The segment length is in feet. The length of segment 1 will be reconciled with preload later on. Elasticity is EA in pounds. Use zero for inelastic catenaries. Use -1 to let RIGMOOR provide EA for you. Or enter your own EA as a positive number.

A buoy or weight linking segments is a load. A load is entered with the segment beneath it. Enter immersed displacement - air weight. Thus weights are negative. Segment 1 always has zero load.

The sketch illustrates a 2-segment leg.

!10 DISK DRIVE ID SELECTION

The RIGMOOR system creates several disk files for each mooring problem. You may assign the drive, path and root name for the current problem. These will be used to form the entire name for files used by RIGMOOR. The root name must have exactly six (6) letters and/or numerals.

Use a "root name" that identifies the mooring problem. Extensions that identify the purpose and content of each file are added by RIGMOOR. For example, the root name for an analysis of rig 113 of the Big Oil Co. might be BIG113. RIGMOOR would then create files named:

BIG113DF.RIG	to store the parameters that define the rig and its mooring;
BIG113XH.L01	to hold the force/displacement table for leg type 1, etc, and
BIG113HS.L02	for the preload vs scope for leg type 2, etc.

Lowercase letters are changed to UPPERCASE by RIGMOOR.@

!11 JOBNAME SELECTION

Type a title for your mooring study. Use at least 1 letter - or up to 72.@

!12 OFFSET SELECTION

The Design load is the horizontal force acting on a mooring leg that reduces the tensile safety factor to the design minimum. It represents the largest load that the leg can sustain on an operational basis. It is the load threshold at which remedial measures must be taken.

When a mooring is under no external load, the legs pull against each other. The leg load in that case is called the Preload. The horizontal deflection of the leg between the Preload and the Design load is called the Offset.

Oil rig moorings are usually designed to deflect about 5 to 7 percent of the water depth when environmental forces increase the leg load from the Preload to the Design load while drilling.

Enter the percent deflection for this mooring. For 5%, enter 5.@

!13 SAFETY FACTOR SELECTION

The Design load is the horizontal force acting on a mooring leg that reduces the tensile safety factor to the design minimum. It represents the largest load that the leg can sustain on an operational basis. It is the load threshold at which remedial measures must be taken.

The safety factor is usually based on the tension at the top of the leg, but the limiting tension may occur somewhere else on legs that have more than one segment diameter.

Oil rig moorings usually are designed to a safety factor of three while restraining a load that deflects the rig horizontally by 5% of the water depth.

Enter the safety factor at Design load and offset for this mooring.@

!14 RIGMOOR FUNCTION HELP

RIGMOOR supports 10 command functions, listed on the menu. Commands are selected by pressing a numeral (0 - 9) and the RETURN or ENTER key. Commands 1-4 perform mooring computations, 5-8 provide printed displays, 9 and 0 select a mooring definition. All computed tables are stored on disk.

Function 1 is the most general command. It finds the operational and survival holding power of a mooring as a function of storm direction.
Function 2 adjusts the length of each leg based on the mooring preload.
Function 3 relates preload, holding load and anchor radius to leg length.
Function 4 computes the displacement function for each unique leg.

Function 5 prints the leg loadings for the holding power roses.
Functions 6 and 7 print the H vs S and X vs H tables.
Function 8 prints the Rig Definition File currently active.

Function 9 allows you to change the file currently selected.
Function 0 prompts for the parameters of a new Rig Definition File.

Enter a Q (or q) to end the session.@

!15 EQUILIBRIUM AVERAGE PRELOAD ENTRY

At mooring equilibrium, the vector sum of the horizontal force in the mooring legs must be zero. With no environmental loads, the relative size of these forces is determined by the planview geometry of the mooring. The table shows the preload ratios for each anchor leg. The preload in a leg is scaled from the average by its preload ratio. Symmetric moorings have equal preloads.

RIGMOOR has estimated an average preload such that the shortest scope on the bottom at the DESIGN load is about five percent of the water depth. You may enter another value for the average preload, but use a value within the range shown.

Enter a U to use the current value;
E to recover the estimated value; or
Q to cancel go back to the main command menu.@

!16 PRELOAD DISTRIBUTION TABLE - UNREFERENCED

In the table above, your average preload has been scaled by the geometric preload ratio to an actual preload for each leg. This has been combined with the safety factor and deflection values specified in the rig mooring definition file to coordinate the length of the anchor leg and the design load.

The holding power of the leg is the difference between the design load and the preload. The table also shows the length that lies on the bottom under three conditions:

1) When the rig is displaced until the leg goes slack (without changing the length of the leg),

2) When the rig has no displacement - centered over the drill hole with all legs at their preload (no environmental load), and

3) When the rig is displaced until the leg reaches its design load (still without changing the length of the leg)

Negative values in the "Length on Bottom" columns are not lengths. They indicate that none of the leg lies on the bottom. The number (ignoring the - sign) is the vertical force of the leg on its anchor, indicating that a larger average preload is required if ordinary anchors are to be used with the leg.@

!17 SEGMENT PARAMETER PROMPT

Each segment is specified by five parameters:

Material Codes: 1=Stud-link Chain 2=IWRC Wire Rope 3=Fiber Core Wire Rope

Diameter in inches. Weight and strength will be scaled on diameter for you.

Length in feet. The length of the top segment will be adjusted by preload.

Elasticity, EA, in pounds. Use 0 for inextensible. RIGMOOR will provide a realistic value if you enter -1.

Intersegment Load in pounds, at top of segment. This is immersed displacement minus weight, positive for buoys, negative for weights. Use 0 if none. Neither the rig at the top nor the anchor at the bottom is an intersegment load.

You may enter these five values in one line, or one at a time.@

!18 SEGMENT DIAMETER ENTRY HELP

The diameter of a segment is its nominal size. For wire rope, it is the overall diameter. Six by 36 construction is assumed, using galvanized monitor AA-grade strands, whether Independent Wire Rope Core (IWRC) or Fiber Core construction has been selected. Chain - even in massive stud-links - is sized by the diameter of the side "wire" that forms the links.

RIGMOOR estimates the weight, strength and elasticity of a segment using diameter-dependent formulas derived from:

A Compendium of Tension Member Properties for Input to
Cable Structure Analysis Programs,
U.S.Naval Civil Engineering Laboratory,
Technical Report CR 82.017, April, 1982.

Quantity	Chain	IWRC	Fiber Core
Weight	Table 23	Table 9	Table 9
Strength	Table 23 (Proof)	Table 9 (Monitor AA)	Table 9 (Monitor)
Elasticity	Table 34	Table 30 (20-65% BS)	Table 30 (20-65% BS)
Metallic Area	--	Table 31 (6x37)	Table 31 (6x37)@

!19 SEGMENT LENGTH ENTRY HELP

Enter the length of this segment in feet. The length of the topmost segment is adjusted by RIGMOOR Function 7 according to complex relations among the preload, design load, and design offset in order to obtain:

* Equilibrium over the drill hole at the selected preload,
* Equilibrium at the design offset from the drill hole with the tautest leg at its design load and safety factor.@

!20 SEGMENT ELASTICITY ENTRY HELP

Elasticity, as used in RIGMOOR, means the product of effective Young's Modulus with metallic cross sectional area. The product is measured in pounds. Distortion under load - of the lays in wire rope and of the link shape in chain - reduces the effective Young's Modulus from the values obtained with the standard, solid tensile test specimens.

If you enter a zero, the segment will be treated as inelastic: incapable of elongation. A positive entry will be used to evaluate the segment's elongation under load using the elastic catenary equations. If you enter -1, or any negative value, RIGMOOR will estimate the elasticity for you, using diameter-dependent functions derived from:

A Compendium of Tension Member Properties for Input to
Gable Structure Analysis Programs,
U.S.Naval Civil Engineering Laboratory,
Technical Report CR 82.017, April, 1982.

Quantity	Chain	IWRC	Fiber Core
Elasticity	Table 34	Table 30 (20-65% BS)	Table 30 (20-65% BS)
Metallic Area	—	Table 31 (6x37)	Table 31 (6x37)@

!21 INTER-SEGMENT BUOYANCY ENTRY HELP

A buoy or weight may be attached between segments. Its buoyancy or weight is entered with the parameters for the segment BENEATH it. The rig - at the top of segment 1, not between segments - does not count as a buoy, and the anchor at the end of the last segment does not count as a weight. A maximum of 4 buoys and/or weights are possible in a 5-segment leg.

Enter the immersed displacement minus the air weight. This will be a postive number for buoys, but negative for weights. Enter a zero if there is no inter-segment load.

RIGMOOR will determine whether a buoy floats on the surface or is submerged, depending on the load in the leg. Likewise, weights may rest on the bottom or be suspended. Buoys and weights may be intermingled in any order along a leg, with a single exception:

> A buoy may not float on the surface if there is an inter-segment weight resting on the bottom between the buoy and the rig.

This is equivalent to saying that a clump may not rest on the bottom if there is a surfaced inter-segment buoy between the clump and its anchor.@

!22 HOLDING POWER ROSE HELP (GETRNG)

When no environmental forces disturb a moored rig, the leg preloads cancel each other so that no yaw moment is produced and the rig is centered over the drill hole. The operational holding power of a mooring is computed by deflecting the rig by a fixed percent of the depth without changing the length of any leg and rotating the rig to cancel any yaw moment that results from the deflection. The holding power is the net force required to hold the rig there. The direction of the holding force will roughly parallel the direction of the displacement.

The only way to increase the holding power of the mooring for survival purposes is to slack the leeward legs, since a leg aligned with the rig displacement will be loaded to its design safety factor by the displacement. To estimate survival holding power, RIGMOOR slacks legs that are within 75 deg. of the operational holding power.

Since holding power depends on the direction of displacement, they may be computed for a range of displacement directions in steps. You are asked to enter the first angle, last angle and angle step, all in degrees measured clockwise from the Y-axis, which extends from the center of the rig forward.@

!23 MORE HOLDING POWER ROSE HELP

The mooring holding power is computed by deflecting the rig from over the drill hole to points on a circle whose radius is the offset at design load (part of the rig definition). Deflected points on that circle are identified by their angle around the circle.

The point of zero angle coincides with a radius through the bow of the rig. Angles clockise from this point are positive; counter-clockwise angles are negative.

If the spread pattern and leg construction is symmetric, the holding power rose will be symmetric as well. For example, if a mooring has four equal legs spaced 90 degrees apart, then a 45 degree arc that begins (or ends) aligned with a leg gives information about the entire rose.

Enter an angle to mark the beginning of the arc, another to end the arc, and the angle step size. If the end angle equals the start angle, the step size may be omitted. Separate angle entries with a comma, space or <cr>.@

!24 RIG DEFINITION FILE HELP

The water depth, anchor pattern, and fairlead locations may be saved and/or recovered from a computer file in order to avoid repetitious entry of these values. This file and others created by RIGMOOR for this mooring problem will be stored together using a formal set of names built around a "root name".

You may assign the drive and MS-DOS path as well as the root name at this time. The root name must have exactly 6 letters and/or numerals. All told, the entry cannot exceed 63 characters under MS-DOS.

The Rig Definition file name is formed by adding "DF.RIG" to the root.

RIGMOOR has been unable to open the Rig Definition file in order to save the current rig. You may try again by typing a 'Y' (or 'y'). Otherwise the current rig definition will not be saved.@

!25 RIG HEAVE HELP
Heave is the vertical motion of the rig center of gravity caused by waves.

You have run a set of holding power roses with no heave. You may now run other roses with the rig displaced upward or downward, in order to simulate the motion of the rig during a storm. Neither the anchor placement nor the length of the mooring legs is altered for these cases. You may omit the heave analysis by entering a zero.

Heaving the rig upward reduces the safety factor. Note: the holding power will be larger for upward heave. This is because holding power is defined as the force required to produce a fixed deflection. The extra load required to produce the fixed deflection appears as less safety factor.

If you wish to maintain a specific safety factor including upward motion, add the heave to the water depth in the case definition. Then run negative heave displacements to determine the extra safety factor in still water.

Separate rose files are prepared for each value of heave. Heave cases are distinguished by the file name extension, .Hnn (upward) or .-nn (downward), where nn is the displacement in feet. Heave must be entered as a whole number.@

!26 ROSE DISPLAY HEAVE OPTION HELP
Heave is the vertical motion of the rig center of gravity caused by waves.

If you have computed holding power rose files for non-zero rig heave, you may print them by entering the heave value as a whole number between -99 and 99. Or enter a zero to restore the menu of commands.@

```
C$DEBUG
$STRICT                                           ;MODULE MOOR.FOR
C                                                 ;BLOCK DATA DEVICE
C                                                 ;PROGRAM RIGMOR
C                                                 ;SUBROUTINE DELAY
C                                                 ;LOGICAL FUNCTION GETRIG
C                                                 ;SUBROUTINE HLPMSG
C
C *** MS-FORTRAN-77 V. 3.2 REQUIRES BLOCK DATA ROUTINES TO LOAD FIRST
      BLOCK DATA DEVICE                           ;D.B.DILLON  EG&G  1985
C
C *** SET DEFAULT VALUES IN COMMON BLOCK VARIABLES
C
C     ------------------------------------------------------------------------
C     COMMON BLOCK VARIABLE DEFINITIONS:
C
      INTEGER*2 MTYPS                             ;/LEGMAP/    NO. MASTER TYPES
      INTEGER*2 MLEG                              ;MASTER LEG FOR TYPE I
      INTEGER*2 MTYP                              ;MASTER TYPE FOR LEG I
      REAL PI           ;/TRIG/ 3.1416
      REAL D2R          ;DEGREES TO RADIANS (.0174533)
      REAL R2D          ;RADIANS TO DEGREES (57.29578)
C
      REAL X            ;/HVSX/ SPAN TABLE   (I=FILE LINE  J=ANCHOR NO)
      REAL H            ;LOAD TABLE
      REAL S            ;SAFETY FACTOR TABLE
C
      LOGICAL*2 ACTIV   ;/TORX/ ACTIVE LEG FLAGS FOR ANALYZE MODULE
      REAL THETA        ;HOLDING POWER ROSE OFFSET DIRECTION
      REAL XRIG, YRIG   ;COMPONENTS OF RIG OFFSET
      REAL YAW          ;RIG YAW TO CANCEL MOMENTS WHEN OFFSET
      REAL NETMCW       ;NET CLOCKWISE MOMENT (ITERATION RESIDUAL)
      REAL NETFX, NETFY ;SUM OF X,Y LEG FORCES ON RIG
      REAL NETSF        ;LEAST SAFETY FACTOR IN ANY LEG WHEN OFFSET
      REAL XSPN         ;LIST OF SPANS BETWEEN ANCHORS AND OFFSET FAIRLEADS
      REAL LOAD         ;LIST OF OFFSET LEG H-LOADS
      REAL SAFAC        ;LIST OF SAFETY FACORS BY LEG IN OFFSET LOAD
      REAL TORQ         ;LIST OF MOMENTS OF LEG ON RIG
      REAL LEGX, LEGY   ;LISTS OF X,Y H-LOAD COMPONENTS BY LEG
C
      REAL HSN          ;/HSXTRM/ H VS S MINIMUMS BY COLUMN AND TYPE
      REAL HSX          ;MAXIMA BY COLUMN AND TYPE (XTREME FUNCTION)
C
      REAL SCOP         ;/HSTABL/ H VS S RECORD: LENGTH OF SEGMENT 1
      REAL XMIN         ;NO LOAD: + = LIFT ON ANCHOR  - = LENGTH ON BOTTOM
      REAL SBPL         ;PRELOAD:      DITTO
      REAL SBDL         ;DESIGN LOAD: DITTO
      REAL XPRE         ;SPAN AT PRELOAD
      REAL XDES         ;SPAN AT DESIGN LOAD
      REAL HPRE         ;PRELOAD FOR XPRE = XDES - OFFSET*DEPTH (SEE /ANCHOR/)
      REAL HDES         ;DESIGN LOAD FOR FMIN = SAFETY (SEE /ANCHOR/)
      REAL HDIF         ;HOLDING POWER, HDES - HPRE
C
      INTEGER*2 NOS     ;/XHTABL/ NO. NODES ON SURFACE
      INTEGER*2 NOB     ;NO. NODES ON BOTTOM
```

```
      INTEGER*2 NSF     ;NODE WHERE FMIN OCCURS
      REAL SPAN         ;TOTAL HORIZONTAL SPAN, FAIRLEAD TO ANCHOR
      REAL SB           ;TOTAL SCOPE ON BOTTOM
      REAL ST           ;TOTAL STRETCHED SCOPE
      REAL FMIN         ;LEAST SAFETY FACTOR IN LEG
      REAL DX           ;HORIZONTAL INCREMENT OF SEGMENT, +: AWAY FROM RIG
      REAL DY           ;VERTICAL INCREMENT, +: ANCHOR END ABOVE RIG END
      REAL V            ;UPWARD FORCE REQUIRED TO SUPPORT RIG END OF SEGMENT
      REAL U            ;UPWARD FORCE OF ANCHOR END SUPPORTING NEXT SEGMENT
      REAL Y            ;DEPTH OF NODE (SUM OF -DY, USUALLY)
      REAL SL           ;SEGMENT STRETCHED LENGTH
      REAL BL           ;SEGMENT BOTTOMED LENGTH
C
      REAL RIGX         ;/SEMI/ FAIRLEAD LOCATION, FEET STARBOARD OF ORIGIN
      REAL RIGY         ;LOCATION, FEET FORWARD OF ORIGIN
C
      INTEGER*2 ANCS    ;/ANCHOR/ NO. ANCHORS
      INTEGER*2 LTYP    ;LEG TYPE LIST BY ANCHOR
      REAL DEPTH        ;WATER DEPTH
      REAL OFFSET       ;DESIGN OFFSET, % OF DEPTH
      REAL SAFETY       ;DESIGN SAFETY FACTOR
      REAL ADIR         ;DIRECTION, FAIRLEAD TO ANCHOR, DEG. CW FROM FORWARD
      REAL ATOP         ;LENGTH OF TOPMOST SEGMENT BY ANCHOR
      REAL ARAD         ;LEG SPAN AT PRELOAD
      REAL APRE         ;LEG PRELOAD
      REAL ANCX, ANCY   ;ANCHOR LOCATION COORDINATES
C
      INTEGER*2 TYPS    ;/CABLES/ NO. LEG TYPES
      INTEGER*2 SEGS    ;NO. SEGMENTS, LIST BY TYPE
      INTEGER*2 MAT     ;SEGMENT MATERIAL CODE TABLE
      REAL DIA          ;SEGMENT DIAMETER TABLE
      REAL BRK          ;ELEMENT STRENGTH, TABLE BY SEGMENT AND TYPE
      REAL LEN          ;ELEMENT LENGTH, TABLE
      REAL WGT          ;ELEMENT LINEAR WEIGHT DESNITY, TABLE
      REAL EA           ;ELEMENT ELASTICITY TABLE
      REAL BNCY         ;BUOYANT LOAD AT ANCHOR END OF ELEMENT (NEG.=WEIGHT)
C
      REAL AW, BW                           ;/LEGMAT/ WEIGHT COEFFICIENTS
      REAL AB, BB                           ;STRENGTH
      REAL AE, BE                           ;ELASTICITY
C
      CHARACTER*16 MATNAM                   ;/MATLST/ MATERIAL NAME LIST
C
      INTEGER*2 CON     ;/UNITS/ LOGICAL UNIT NO. FOR USER I/O (KYBD/SCREEN)
      INTEGER*2 PTR     ;PRINTER
      INTEGER*2 MSG     ;HELP MESSAGE FILE
      INTEGER*2 AUX     ;AUXILIARY FILE: X VS H  OR  H VS S
      INTEGER*2 RIG     ;RIG DEFINITION FILE
C
      INTEGER*2 LTXT    ;/USRPTR/ LENGTH OF STRING IN TEXT
      INTEGER*2 LPTR    ;POINTER FOR TEXT STRING
C
      CHARACTER TEXT    ;/USRTXT/ USER INPUT TEXT BUFFER
C
      CHARACTER RIGNAM  ;/JOB/ TASK PATH AND ROOT NAME
```

```
      CHARACTER JOBNAM  ;TASK TITLE STRING
C
      INTEGER*2 RNL     ;/NAMLEN/ LENGTH OF RIGNAM STRING
      INTEGER*2 JNL     ;LENGTH OF JOBNAM STRING
C
C     ---------------------------------------------------------------
C     COMMON BLOCKS:
C
      COMMON /NAMLEN/ RNL, JNL                  ;NAME LENGTHS
      COMMON /JOB/ RIGNAM(72), JOBNAM(72)       ;PROBLEM ID
      COMMON /USRPTR/ LTXT, LPTR                ;ENTRY LENGTH, POINTER
      COMMON /USRTXT/ TEXT(80)                  ;ENTRY BUFFER
      COMMON /UNITS/ CON, PTR, MSG, AUX, RIG;LOGICAL UNITS
      COMMON /MATLST/ MATNAM(4)                 ;MATERIAL NAMES
      COMMON /LEGMAT/
     +    AW(3), BW(3),                         ;WEIGHT      AW * DIAM ** BW
     +    AB(3), BB(3),                         ;STRENGTH    AB * DIAM ** BB
     +    AE(3), BE(3)                          ;ELASTICITY  AE * BE * DIAM ** 2
      COMMON /CABLES/ TYPS, SEGS(12),
     +    MAT(5,12), DIA(5,12),
     +    BRK(5,12), LEN(5,12), WGT(5,12),
     +    EA(5,12), BNCY(5,12)                  ;LEG SEGMENT PARAMETERS
      COMMON /ANCHOR/ ANCS, LTYP(12),
     +    DEPTH, OFFSET, SAFETY,
     +    ADIR(12), ATOP(12), ARAD(12),
     +    APRE(12), ANCX(12), ANCY(12)          ;ANCHOR PARAMETERS
      COMMON /SEMI/ RIGX(12), RIGY(12)          ;BARGE BOLLARDS
      COMMON /XHTABL/
     +    NOS, NOB, NSF,
     +    SPAN, SB, ST,
     +    FMIN, DX(5), DY(5), V(5), U(5),
     +    Y(6), SL(5), BL(5)                    ;X VS H RECORD
      COMMON /HSTABL/ CASE(9)                   ;H VS S RECORD
      COMMON /HSXTRM/ HSN(9,12), HSX(9,12)      ;H VS S EXTREMES
      COMMON /TORX/
     +    THETA, XRIG, YRIG, YAW,               ;RIG DISPLACEMENT
     +    NETMCW, NETFX, NETFY, NETSF,          ;MOORING CAPACITY
     +    XSPN(12), LOAD(12), SAFAC(12),        ;LEG LOADING LIST
     +    TORQ(12), LEGX(12), LEGY(12),         ;MOMENT & FORCE LIST
     +    ACTIV(12)                             ;ACTIVE LEG FLAGS
      COMMON /HVSX/ X(80,12),                   ;SPAN TABLE
     +              H(80,12),                   ;LOADS
     +              S(80,12)                    ;SAFETY TABLES
      COMMON /TRIG/ PI, D2R, R2D                ;ANGLE CONVERSION
      COMMON /LEGMAP/
     +    MTYPS,                                ;NO. MASTER TYPES
     +    MLEG(12),                             ;MASTER LEG FOR TYPE I
     +    MTYP(12)                              ;MASTER TYPE FOR LEG I
C
C     ---------------------------------------------------------------
C     INITIALIZING DATA STATEMENTS FOR COMMON BLOCKS
C
      DATA RIGNAM /'N', 'o', 'n', 'e', 68*' '/
      DATA JOBNAM /72*' '/
      DATA CON, PTR, MSG, AUX, RIG              ;CON MUST BE 0 FOR MS-DOS
```

```
     + /   0,   2,   3,   4,   5/           ;/UNITS/
C *** MATERIAL PROPERTY COEFFICIENTS (REF. NAVAL CIVIL ENGINEERING LAB.
C     REPORT CR 82.017, APRIL 1982, "A COMPENDIUM OF TENSION MEMBER
C     PROPERTIES...")
C     CHAIN: TABLE 23, .87*AIR WGT, PROOF LOAD. FIT THRU 2.5 & 5.5 IN.
C     IWRC:  TABLE 9. .87*AIR WGT, MONITOR AA (6X37)     .25 - 3.5
C     FIBER: TABLE 9. .75*AIR WGT IN TABLE 9  (6X37)     .25 - 3.5
C     CHAIN AE: TABLE 34, STUD-LINK
C     IWRC  AE: TABLES 30&31, 6X37, 20-65% BS, FIT 1/4 THRU 3 IN.
C     FIBER AE: SAME AS IWRC
C     MATL CODE:  1         2         3
C               CHAIN     IWRC     FIBER     ;/MATLEG/
      DATA AW /8.45092, 1.60813, 1.26047/   ,WEIGHT COEFFICIENTS
      DATA BW /2.02293, 2.00046, 1.99997/
      DATA AB / 96370.,  87784.,  71068./   ;STRENGTH   "
      DATA BB /1.78365, 1.95296, 1.95333/
      DATA AE /8.595E6,   1.4E7,   1.1E7/   ;ELASTICITY "
      DATA BE /    1.0,    0.46,    0.40/   ;AREA FACTOR
      DATA MATNAM /'Stud-link Chain ',      ;/MATLST/
     +            ' IWRC Wire Rope ',
     +            'Fiber-Core W.R. ',
     +            'Unspecified Line'/       ;SEGMENT MATERIAL NAMES
      DATA TYPS, SEGS /0, 12*0/             ;/CABLES/
      DATA BRK,   LEN,    WGT,     EA,   BNCY
     +  /60*0., 60*0., 60*0., 60*0., 60*0./
      DATA ANCS, LTYP /12*0, 0/             ;/ANCHOR/
      DATA DEPTH, OFFSET, SAFETY
     +      /200.,       5.,      3./
      DATA ADIR, ATOP, ARAD /36*0./
      DATA APRE, ANCX, ANCY /36*0./
      DATA RIGX, RIGY /12*0., 12*0./        ;/SEMI/
      DATA NOS, NOB, NSF /3*0/              ;/XHTABL/
      DATA SPAN, SB, ST, FMIN /4*0./
      DATA DX, DY, V, U, Y, SL, BL /36*0./
      DATA CASE /9*0./                      ,/HSTABL/
      DATA HSN, HSX /108*0., 108*0./        ;/HSXTRM/
      DATA THETA, XRIG, YRIG, YAW /4*0./    ;/TORX/
      DATA NETMCW, NETFX, NETFY, NETSF /4*0./
      DATA XSPN, LOAD, SAFAC /36*0./
      DATA TORQ, LEGX, LEGY /36*0./
      DATA ACTIV /12*.TRUE./
      DATA X, H, S /2880*0./                ;/HVSX/
      DATA PI,        D2R,         R2D /    ;/TRIG/
     +     3.1415927, .017453292, 57.295780/
      DATA MTYPS, MLEG, MTYP /25*1/         ;/LEGMAP/ MASTER LEG LISTS
C
C     -------------------------------------------------------------------
C
      END
```

```
      PROGRAM RIGMOR                         ;D.B.DILLON  EG&G  1985
C
C *** SELECT MOORING ANALYSIS PROGRAMS FOR EXPLORATORY OIL RIGS
C
      LOGICAL*2 GETRIG,GETDEF,USRINP,GETSTR ;FUNCTIONS
      EXTERNAL  GETRIG,GETDEF,USRINP,GETSTR
      CHARACTER FNC, SPACE                  ;LOCALS
      LOGICAL*2 ERR                         ;GETPRE ERROR FLAG
      INTEGER*2 I, I1, J1                   ;SUBSCRIPTS
      INTEGER*2 ONE, THREE                  ;CALLING ARGUMENTS
C
C *** MS-FORTRAN-77 V.3.2 REQUIRES ALL COMMON BLOCKS IN MAIN PROGRAM
C
      INTEGER*2   MTYPS, MLEG, MTYP         ;/LEGMAP/
      REAL PI, D2R, R2D                     ;/TRIG/
      REAL X, H, S                          ;/HVSX/
      LOGICAL*2 ACTIV                       ;/TORX/
      REAL THETA, XRIG, YRIG, YAW,
     +     NETMCW, NETFX, NETFY, NETSF,
     +     XSPN, LOAD, SAFAC,
     +     TORQ, LEGX, LEGY
      REAL HSN, HSX                         ;/HSXTRM/
      REAL CASE                             ;/HSTABL/
      INTEGER*2 NOS, NOB, NSF               ;/XHTABL/
      REAL SPAN, SB, ST, FMIN, DX, DY,
     +     V, U, Y, SL, BL
      REAL RIGX, RIGY                       ;/SEMI/
      INTEGER*2 ANCS, LTYP                  ;/ANCHOR/
      REAL DEPTH, OFFSET, SAFETY,
     +    ADIR, ATOP, ARAD, APRE, ANCX, ANCY
      INTEGER*2 TYPS, SEGS, MAT             ;/CABLES/
      REAL DIA, BRK, LEN, WGT, EA, BNCY
      INTEGER*2 CON, PTR, MSG, AUX, RIG     ,/UNITS/
      INTEGER*2 LTXT, LPTR                  ;/USRPTR/
      CHARACTER TEXT                        ;/USRTXT/
      CHARACTER RIGNAM, JOBNAM              ;/JOB/
      INTEGER*2 RNL, JNL                    ;/NAMLEN/
C
      COMMON /NAMLEN/ RNL, JNL              ;NAME LENGTHS
      COMMON /JOB/ RIGNAM(72), JOBNAM(72)   ;PROBLEM ID
      COMMON /USRPTR/ LTXT, LPTR
      COMMON /USRTXT/ TEXT(80)              ;ENTRY BUFFER
      COMMON /UNITS/ CON, PTR, MSG, AUX, RIG;LOGICAL UNITS
      COMMON /CABLES/ TYPS, SEGS(12),
     +    MAT(5,12), DIA(5,12),
     +    BRK(5,12), LEN(5,12), WGT(5,12),
     +    EA(5,12), BNCY(5,12)              ;LEG SEGMENT PARAMETERS
      COMMON /ANCHOR/ ANCS, LTYP(12),
     +    DEPTH, OFFSET, SAFETY,
     +    ADIR(12), ATOP(12), ARAD(12),
     +    APRE(12), ANCX(12), ANCY(12)      ;ANCHOR PARAMETERS
      COMMON /SEMI/ RIGX(12), RIGY(12)      ;BARGE BOLLARDS
      COMMON /XHTABL/
     +    NOS, NOB, NSF,
     +    SPAN, SB, ST,
```

```
     +    FMIN, DX(5), DY(5), V(5), U(5),
     +    Y(6), SL(5), BL(5)                  ;X VS H RECORD
      COMMON /HSTABL/ CASE(9)                 ;H VS S RECORD
      COMMON /HSXTRM/ HSN(9,12), HSX(9,12)    ;H VS S EXTREMES
      COMMON /TORX/
     +    THETA, XRIG, YRIG, YAW,             ;RIG DISPLACEMENT
     +    NETMCW, NETFX, NETFY, NETSF,        ;MOORING CAPACITY
     +    XSPN(12), LOAD(12), SAFAC(12),      ;LEG LOADING LIST
     +    TORQ(12), LEGX(12), LEGY(12),       ;MOMENT & FORCE LIST
     +    ACTIV(12)                           ;ACTIVE LEG FLAGS
      COMMON /HVSX/ X(80,12),                 ;SPAN TABLE
     +              H(80,12),                 ;LOADS
     +              S(80,12)                  ;SAFETY TABLES
      COMMON /TRIG/ PI, D2R, R2D              ;ANGLE CONVERSION
      COMMON /LEGMAP/
     +    MTYPS,                              ;NO. MASTER TYPES
     +    MLEG(12),                           ;MASTER LEG FOR TYPE I
     +    MTYP(12)                            ;MASTER TYPE FOR LEG I
C
      DATA ONE, THREE /1, 3/                  ;I*2 ARGUMENTS
      DATA SPACE /´ ´/                        ;FOR CENTERING TITLES
C
C *** PREPARE CONSOLE AND PRINTER DEVICE DRIVERS
C
      OPEN (CON, FILE=´CON:´)                 ;UNDOCUMENTED DEVICE NAMES
      OPEN (PTR, FILE=´LPT1´)
      CALL HLPMSG (1)                         ;COMMERCIAL
      CALL DELAY (1)                          ;PAUSE AWHILE
  100 IF (GETRIG()) GOTO 200                  ;FORM RIG FILE NAME
      IF (GETDEF()) GOTO 400                  ;LOAD RIG DEF., IF ANY
C
C *** MAKE ROUTINE SELECTION
C
  200 I1 = RNL - 5                            ;ROOT NAME POINTER
      J1 = 40 - JNL/2                         ;CENTER JOB NAME
      WRITE (CON, 1000)
     +      (RIGNAM(I), I=I1,RNL),
     + (SPACE, I=1,J1), (JOBNAM(I), I=1,JNL);DISPLAY MENU
      CALL HLPMSG(3)                          ;MENU, PART 2
      IF (USRINP(14)) GOTO 410                ;GET FUNCTION SELECTION
      IF (GETSTR(FNC,ONE)) GOTO 200           ;BAD ENTRY
      IF (FNC .EQ. ´1´) CALL ANALYZ (.TRUE.);COMPUTE HOLDING POWER ROSES
      IF (FNC .EQ. ´A´) CALL ANALYZ(.FALSE.);ROSES USING OLD XvsH FILES
      IF (FNC .EQ. ´2´) CALL GETPRE (BAD)     ;PRELOAD VS SCOPE TRADEOFF
      IF (FNC .EQ. ´3´) CALL HVSS             ;OPTIMIZE SCOPE & PRELOAD
      IF (FNC .EQ. ´4´) CALL XVSH
     +                  (.FALSE., .TRUE.)     ;COMPUTE X VS H TABLES
      IF (FNC .EQ. ´5´) CALL SHOWHP           ;DISPLAY HOLDING POWER ROSES
      IF (FNC .EQ. ´6´) CALL SHOWHS           ;DISPLAY H VS S TABLES
      IF (FNC .EQ. ´7´) CALL SHOWXH           ;DISPLAY X VS H TABLES
      IF (FNC .EQ. ´8´) CALL PRTDEF (THREE)   ;DISPLAY DEFINITION
      IF (FNC .EQ. ´9´) GOTO 100              ;CHANGE CASE
      IF (FNC .EQ. ´0´) CALL NEWCAS           ;DEFINE NEW PROBLEM
      IF ((FNC .EQ. ´Q´) .OR.
     +    (FNC .EQ. ´q´)) STOP                ;QUIT
```

```
      GOTO 200                                          ;NEXT FUNCTION REQUEST
C
C *** HELP REQUESTS
C
  400 CALL HLPMSG(10)                                   ,RIG FILE
      PAUSE
      GOTO 100
C
  410 PAUSE                                             ;FUNCTION HELP
      GOTO 200
C
 1000 FORMAT (
     + 19X' EXPLORATORY OIL RIG MOORING LEG ANALYSIS'//
     + ' David B. Dillon   EG&G, Inc.',
     + 15X' Current Rig Definition Root:',
     +  1X6A1/1X80A1)                                   ;MAIN MENU, PART 1
C
 1002 FORMAT (A1)                                       ;USER INPUT
C
      END
```

```
      SUBROUTINE DELAY (I)
C
C *** DELAY LOOP
C
      IF (I .EQ. 0) RETURN                         ;NO DELAY
      DO 200 J=1,I
      DO 100 L=1,20000
  100 CONTINUE
  200 CONTINUE
      RETURN                                       ;DONE
C
      END
```

```
      LOGICAL*2 FUNCTION GETRIG()
C
C *** GET ROOT NAME BASE FOR PROBLEM
C
      LOGICAL*2 USRINP,GETSTR                  ;FUNCTIONS
      EXTERNAL  USRINP,GETSTR
      INTEGER*2 I, J, K, PATH                  ;LOCAL
      INTEGER*2 CON, PTR, MSG, AUX, RIG        ;/UNITS/
      CHARACTER RIGNAM, JOBNAM                 ;/JOB/
      INTEGER*2 RNL, JNL                       ;/NAMLEN/
C
      COMMON /NAMLEN/ RNL, JNL                 ;NAME LENGTHS
      COMMON /JOB/ RIGNAM(72), JOBNAM(72)      ;PROBLEM ID
      COMMON /UNITS/ CON, PTR, MSG, AUX, RIG;LOGICAL UNITS
C
C *** GET DRIVE NUMBER FROM USER CONSOLE
C
  200 RNL = 72                                 ;MAX. ROOT LENGTH
      WRITE (CON, 1001)                        ;ROOT NAME PROMPT
      IF (USRINP(10)) GOTO 200                 ;HELP REQUEST
      IF (GETSTR(RIGNAM,RNL)) GOTO 200         ;BAD ROOT
C
  400 K = 0                                    ;CHECK FOR SUBDIRECTORIES
      IF (RIGNAM(2) .EQ. ':') K = 2            ;DRIVE PREFIX INCLUDED
      I = RNL - K
      DO 410 J=1,I                             ;SCAN NAME STRING
      PATH = RNL - J                           ; BACKWARDS
      IF (RIGNAM(PATH) .EQ. '\') GOTO 500      ;MS-DOS SUBDIRECTORY MARK
  410 CONTINUE                                 ;FALL THRU WITH PATH = K
C
  500 IF (PATH-K .GT. 63) GOTO 600             ;SUBDIRECTORY OVERFLOW
      IF (RNL-PATH .NE. 6) GOTO 610            ;ROOT NOT 6 LETTERS
      GETRIG = .FALSE.                         ;GOOD CALL
      RETURN                                   ;MANUAL ENTRY COMPLETE
C
C *** ERROR RECOVERY
C
  600 WRITE (CON, 1003) RIGNAM                 ;PATH OVERFLOW
      GOTO 620
C
  610 WRITE (CON, 1004) RIGNAM                 ;ROOT LENGTH NOT 6
  620 GETRIG = .TRUE.                          ;SET ERROR FLAG
      RETURN
C
 1001 FORMAT (/
     + ' Enter Drive and Rig Name'/
     + '            A:RIGNAM')                 ;NAME PROMPT
C
 1003 FORMAT (' GETRIG@600:'/
     + ' Path exceeds 63 bytes.'/ 1X72A1/)     ,BAD PATH
C
 1004 FORMAT (' GETRIG@610:'/
     + ' Rig name not 6 letters.'/ 1X72A1/) ;BAD NAME ROOT
C
      END
```

```
      SUBROUTINE HLPMSG (MSGNO)
C
C *** GIVEN MSGNO = A MESSAGE NUMBER, FIND AND DISPLAY THE MESSAGE
C
      INTEGER*2 MSGNO                         ;ARGUMENT
      LOGICAL CLOSD                           ;LOCAL
      CHARACTER MARK, MSGLIN(80), ASCNO*2
      INTEGER*2 I, I1, I2, OLDNO
      INTEGER*2 CON, PTR, MSG, AUX, RIG       ;/UNITS/
C
      COMMON /UNITS/ CON, PTR, MSG, AUX, RIG
C
      DATA OLDNO,  CLOSD /0, .TRUE./
C
C *** CHECK FOR OPEN MESSAGE FILE AND CLEAR SCREEN
C
      IF (CLOSD) OPEN (MSG,
     +           FILE='RIGMOOR.MSG',
     +         STATUS='OLD')                  ;OPEN MESSAGE FILE
      CLOSD = .FALSE.                         ;FLAG STATUS
      IF (OLDNO .LT. MSGNO) GOTO 110          ;IS POINTER PAST MSG?
  100 REWIND MSG                              ;YES, RESET POINTER
C
C *** SEARCH FOR MESSAGE NO. IN LINE BEGINNING WITH A !
C
  110 READ (MSG, 1004, END=300) MARK, ASCNO ;MESSAGE RECORD
      IF (MARK .NE. '!') GOTO 110             ;START OF MESSAGE MARK?
      READ (ASCNO, 1006) OLDNO                ,ASCII TO BINARY
      IF (OLDNO-MSGNO) 110, 200, 310          ;YES: MSG NO'S MATCH?
C
C *** DISPLAY MESSAGE LINE BY LINE UNTIL @ AT END OF LINE
C
  200 READ (MSG, 1001) MSGLIN                 ;GET A MSG LINE
      DO 210 I=1,80                           ;FIND LINE LENGTH
      I1 = 81 - I
      IF (MSGLIN(I1) .NE. ' ') GOTO 220       ;LENGTH @NON-SPACE
  210 CONTINUE
  220 I2 = I1                                 ;SET LINE LENGTH
      IF (MSGLIN(I1) .EQ. '@') I2 = I1 - 1    ;LAST LINE?
      WRITE (CON, 1005) (MSGLIN(I), I=1,I2) ;DISPLAY MESSAGE LINE
      IF (I2 .EQ. I1) GOTO 200                ;NEXT LINE?
      RETURN
C
C *** ERROR RECOVERY PROCEDURES
C
  300 IF (OLDNO .GE. MSGNO) GOTO 100          ;NO ERROR
      WRITE (CON, 1002) MSGNO, OLDNO          ;MSG.NO. TOO LARGE
      RETURN
C
  310 WRITE (CON, 1003) MSGNO                 ;NO SUCH MSG.NO.
      RETURN
C
C *** FORMATS
C
 1001 FORMAT (80A1)                           ;MESSAGE FILE RECORD
```

```
C
 1002 FORMAT (' HLPMSG@300:'/I3,
     + ' Message request out of range', I3/);MSGNO TOO LARGE
C
 1003 FORMAT (' HLPMSG@310:'/I3,
     + ' Message number not in file'/)      ;MSG FILE OUT OF ORDER
C
 1004 FORMAT (A1, A2)                        ;START OF MESSAGE RECORD
C
 1005 FORMAT (1X80A1)                        ;MSG DISPLAY RECORD
C
 1006 FORMAT (I2)                            ;DECODE ASCNO TO OLDNO
C
      END
```

```
C$DEBUG
$STRICT
C                                                  ;MODULE CASE.FOR
C                                                  ;SUBROUTINE NEWCASE
C                                                  ,SUBROUTINE DEFLEG
C                                                  ;LOGICAL*2 FUNCTION USRINP
C                                                  ,LOGICAL*2 FUNCTION GETINT
C                                                  ;LOGICAL*2 FUNCTION GETFLT
C                                                  ;LOGICAL*2 FUNCTION GETDBL
C                                                  ;LOGICAL*2 FUNCTION GETSTR
C
C     ----------------------------------------------------------------
C
      SUBROUTINE NEWCAS                            ;D.B.DILLON  EG&G  1985
C
C *** DEFINE A NEW CASE AND STORE THE PARAMETERS IN A RIG FILE
C
      LOGICAL*2 USRINP, GETSTR, GETINT,
     +          GETFLT, PUTDEF                     ;FUNCTIONS
      EXTERNAL  USRINP, GETSTR, GETINT,
     +          GETFLT, PUTDEF
      INTEGER*2 ANC                                ;LOCAL: ANCHOR INDEX
      CHARACTER YN                                 ;YES/NO RESPONSE
      REAL RIGX, RIGY                              ;/SEMI/
      INTEGER*2 ANCS, LTYP                         ;/ANCHOR/
      REAL DEPTH, OFFSET, SAFETY,
     +    ADIR, ATOP, ARAD, APRE, ANCX, ANCY
      INTEGER*2 TYPS, SEGS, MAT                    ,/CABLES/
      REAL DIA, BRK, LEN, WGT, EA, BNCY
      INTEGER*2 CON, PTR, MSG, AUX, RIG            ;/UNITS/
      INTEGER*2 LTXT, LPTR                         ,/USRPTR/
      CHARACTER TEXT                               ;/USRTXT/
      CHARACTER RIGNAM, JOBNAM                     ;/JOB/
      INTEGER*2 RNL, JNL                           ;/NAMLEN/
C
      COMMON /NAMLEN/ RNL, JNL                     ;NAME STRING LENGTHS
      COMMON /JOB/ RIGNAM(72), JOBNAM(72)          ;PROBLEM ID
      COMMON /USRPTR/ LTXT, LPTR                   ;ENTRY LENGTH, POINTER
      COMMON /USRTXT/ TEXT(80)                     ;ENTRY BUFFER
      COMMON /UNITS/ CON, PTR, MSG, AUX, RIG,LOGICAL UNITS
      COMMON /CABLES/ TYPS, SEGS(12),
     +    MAT(5,12), DIA(5,12),
     +    BRK(5,12), LEN(5,12), WGT(5,12),
     +    EA(5,12), BNCY(5,12)                     ;LEG SEGMENT PARAMETERS
      COMMON /ANCHOR/ ANCS, LTYP(12),
     +    DEPTH, OFFSET, SAFETY,
     +    ADIR(12), ATOP(12), ARAD(12),
     +    APRE(12), ANCX(12), ANCY(12)             ;ANCHOR PARAMETERS
      COMMON /SEMI/ RIGX(12), RIGY(12)             ;BARGE BOLLARDS
C
C *** SPECIFY JOB NAME
C
      JNL = 72                                     ;MAX. JOBNAME LENGTH
   90 WRITE (CON, 1000)                            ;JOB PROMPT
      IF (USRINP(11)) GOTO 90                      ;GET JOB
      IF (GETSTR(JOBNAM,JNL)) GOTO 90              ;PARSE IT
```

```
C
C *** SPECIFY WATER DEPTH
C
  100 WRITE (CON, 1001)
      IF (USRINP(4)) GOTO 100              ;INPUT DEPTH
      IF (GETFLT(DEPTH)) GOTO 100          ;PARSE IT
      IF (DEPTH .LT. 0.) GOTO 100          ,INVALID
C
C *** SPECIFY DESIGN OFFSET
C
  110 WRITE (CON, 1003)
      IF (USRINP(12)) GOTO 110             ;INPUT OFFSET
      IF (GETFLT(OFFSET)) GOTO 110         ;PARSE IT
      IF (OFFSET .LT. 0.) GOTO 110         ;INVALID
C
C *** SPECIFY DESIGN SAFETY FACTOR
C
  120 WRITE (CON, 1004)
      IF (USRINP(13)) GOTO 120             ,INPUT SAFETY
      IF (GETFLT(SAFETY)) GOTO 120         ;PARSE IT
      IF (SAFETY .LT. 1.) GOTO 120         ,INVALID
C
C *** NUMBER OF ANCHORS
C
  200 WRITE (CON, 2001)                    ,NO. ANCHORS
      IF (USRINP(5)) GOTO 200              ;INPUT
      IF (GETINT(ANCS)) GOTO 200           ;PARSE
      IF (ANCS .LT. 2 .OR.
     +    ANCS .GT. 12) GOTO 200           ,INVALID
C
C *** NUMBER OF LEG TYPES
C
  300 WRITE (CON, 3001) ANCS
      IF (USRINP(6)) GOTO 300              ,GET NO. DISTINCT LEGS
      IF (GETINT(TYPS)) GOTO 300           ;PARSE
      IF (TYPS .LT. 1 .OR.
     +    TYPS .GT. ANCS) GOTO 300         ;INVALID
C
      CALL DEFLEG                          ;GET LEG DEFINITIONS
C
C *** ANCHOR LOCATIONS
C
      WRITE (CON, 4001) TYPS               ,MAIN PROMPT
      DO 410 ANC=1,ANCS                    ;SCAN ANCHORS
  400 WRITE (CON, 4002) ANC                ;SECONDARY PROMPT
      IF (USRINP(7)) GOTO 400
      IF (GETFLT(RIGX(ANC))) GOTO 400      ,PARSE FAIRLEAD X
      IF (GETFLT(RIGY(ANC))) GOTO 400      ;PARSE FAIRLEAD Y
      IF (GETFLT(ADIR(ANC))) GOTO 400      ,PARSE ANCHOR DIRECTION
      IF (GETINT(LTYP(ANC))) GOTO 400      ;PARSE LEG TYPE
      IF ((ADIR(ANC) .LT.-360.) .OR.
     +    (ADIR(ANC) .GT. 360.)) GOTO 400  ;CHECK RANGE
      IF ((LTYP(ANC) .LT. 1) .OR.
     +    (LTYP(ANC) .GT. TYPS)) GOTO 400  ;CHECK TYPE
      ATOP(ANC) = 0.                       ;CLEAR TOP SCOPE
```

```
      ARAD(ANC) = 0.                              ; ANCHOR RADIUS
      APRE(ANC) = 0.                              , LEG PRELOAD
      ANCX(ANC) = 0.                              ; AND ANCHOR X,
      ANCY(ANC) = 0.                              , Y LOCATION
  410 CONTINUE
C
  500 IF (PUTDEF()) GOTO 600                      ;SAVE RIG FILE
      RETURN                                      ;ENVIRONMENT DEFINED
C
C *** PUTDEF COULD NOT OPEN RIG FILE
C
  600 WRITE (CON, 1005)                           ;PROMPT FOR RE-TRY
      IF (USRINP(24)) GOTO 600                    ;RE-TRY HELP REQUEST
      ANC=1                                       ,RESPONSE STRING LENGTH
      IF (GETSTR(YN,ANC)) GOTO 600                ;RESPONSE ERROR
      IF (YN .EQ. 'Y' .OR.
     +    YN .EQ. 'y') GOTO 500                   ;TRY AGAIN
      RETURN
C
C
 1000 FORMAT (//' Job Name (1-72 char.)')         ;JOB NAME
C
 1001 FORMAT (//' Water Depth in feet (>0)');DEPTH
C
 1003 FORMAT (//' Offset at Design Load',
     + ' (percent of depth, 5-7 typical)')        ;WATCH CIRCLE
C
 1004 FORMAT (//' Tensile Safety Factor',
     + ' at design load (> 1, 3 typical)')        ;DESIGN SAFETY FACTOR
C
 1005 FORMAT (' Try again (Y/N)')                 ,RE-TRY SAVE PROMPT
C
 2001 FORMAT (//' No. of Anchors (2-12)')         ,ANCHOR COUNT
C
 3001 FORMAT (//' No. of different Legs (1 -',
     +   I2, ')')                                 ;LEG TYPE COUNT
C
 4001 FORMAT (//
     + ' Enter the Position (x,y) of',
     + ' each fairlead on the rig,'/
     + ' the Direction (Deg.) to its anchor,',
     + ' and the Leg Type (1 -', I2, ')'/)        ;MAIN ANCHOR PROMPT
C
 4002 FORMAT (' Position (X,Y in feet),',
     + ' Anchor Direction (Deg.), and',
     + ' Leg Type for fairlead', I3)              ;SECONDARY PROMPT
C
      END
```

```
      SUBROUTINE DEFLEG
C
C *** DEFINE THE PARAMETERS FOR EACH UNIQUE LEG
C
      INTEGER*2 MTL                          ;LOCALS: MATERIAL INDEX
      INTEGER*2 SEG                          ;SEGMENT INDEX
      INTEGER*2 SGS                          ;SEGMENT LIMIT
      INTEGER*2 TYP                          ;LEG TYPE INDEX
      INTEGER*2 I                            ;PARAMETER INDEX
      REAL BCY                               ;INTER-SEGMENT LOAD
      REAL DIM                               ;SEGMENT DIAMETER
      REAL ELS                               ;ELASTICITY
      REAL LNG                               ;LENGTH
      CHARACTER*36 PMT(4)                    ;SEGMENT PROMPTS
      CHARACTER*28 PMX(3,2)                  ;PROMPT EXTENSIONS
      LOGICAL*2 USRINP, GETINT, GETFLT       ;FUNCTIONS
      INTEGER*2 ANCS, LTYP                   ;/ANCHOR/
      REAL DEPTH, OFFSET, SAFETY,
     +    ADIR, ATOP, ARAD, APRE, ANCX, ANCY
      INTEGER*2 TYPS, SEGS, MAT              ;/CABLES/
      REAL DIA, BRK, LEN, WGT, EA, BNCY
      REAL AW, BW                            ;/LEGMAT/ WEIGHT COEFFICIENTS
      REAL AB, BB                            ;STRENGTH
      REAL AE, BE                            ;ELASTICITY
      INTEGER*2 CON, PTR, MSG, AUX, RIG      ;/UNITS/
      INTEGER*2 LTXT, LPTR                   ;/USRPTR/
      CHARACTER TEXT                         ;/USRTXT/
C
      COMMON /USRPTR/ LTXT, LPTR
      COMMON /USRTXT/ TEXT(80)               ;ENTRY BUFFER
      COMMON /UNITS/ CON, PTR, MSG, AUX, RIG;LOGICAL UNITS
      COMMON /LEGMAT/
     +    AW(3), BW(3),                      ;WEIGHT      AW * DIAM ** BW
     +    AB(3), BB(3),                      ;STRENGTH    AB * DIAM ** BB
     +    AE(3), BE(3)                       ;ELASTICITY  AE * BE * DIAM ** 2
      COMMON /CABLES/ TYPS, SEGS(12),
     +    MAT(5,12), DIA(5,12),
     +    BRK(5,12), LEN(5,12), WGT(5,12),
     +    EA(5,12), BNCY(5,12)               ;LEG SEGMENT PARAMETERS
      COMMON /ANCHOR/ ANCS, LTYP(12),
     +    DEPTH, OFFSET, SAFETY,
     +    ADIR(12), ATOP(12), ARAD(12),
     +    APRE(12), ANCX(12), ANCY(12)       ;ANCHOR PARAMETERS
C
      DATA PMT /                             ;SEGMENT PROMPTS
     + ' Diameter (Inches) . . .  of Segment',
     + ' Length (Feet) . . . . .  of Segment',
     + ' Elasticity, EA (Pounds)  of Segment',
     + ' Intersegment Load at top of Segment'/
      DATA PMX /                             ;PROMPT EXTENSIONS
     + ' -1: Use RIGMOOR Estimate,   ',
     + '  0: Use Inelastic Segment,  ',
     + ' >0: Use entry as EA value.  ',
     + ' <0: Entry is Clump Weight,  ',
     + '  0: No Intersegment Load,   ',
```

```
    + ' >0: Entry is Buoy.        '/
C
C *** ENTER PROPERTIES FOR EACH TYPE OF LEG
C
      DO 700 TYP=1,TYPS
  200 WRITE (CON, 2001) TYP
      IF (USRINP(8)) GOTO 200                ;GET NO. SEGMENTS IN TYPE
      IF (GETINT(SGS)) GOTO 200              ;PARSE
      IF (SGS .LT. 1 .OR.
     +    SGS .GT. 5) GOTO 200               ;INVALID
      SEGS(TYP) = SGS                        ;MOVE TO LIST
C
C *** ENTER PROPERTIES BY SEGMENT
C
      BNCY(1,TYP) = 0.                       ;RIG END
      CALL HLPMSG(17)                        ;HEADER PROMPT
      DO 600 SEG=1,SGS                       ;SCAN SEGMENTS
  300 WRITE (CON, 3001) SEG                  ;SEGMENT PROMPT
      IF (USRINP(9)) GOTO 300                ;GET SEGMENT PARAMETERS
      IF (GETINT(MTL)) GOTO 300              ;PARSE MATERIAL CODE
      IF ((MTL .LT. 1) .OR.
     +    (MTL .GT. 3)) GOTO 300             ;INVALID
      I = 1                                  ;DIAMETER INDEX
  310 IF (GETFLT(DIM)) GOTO 500              ;PARSE SIZE
      IF (DIM .LE. 0.) GOTO 500              ;INVALID SIZE
      I = 2                                  ;LENGTH INDEX
  320 IF (GETFLT(LNG)) GOTO 500              ;PARSE LENGTH
      IF (LNG .LE. 0.) GOTO 500              ;INVALID LENGTH
      I = 3                                  ;ELASTICITY INDEX
  330 IF (GETFLT(ELS)) GOTO 500              ;PARSE ELASTICITY
      IF (SEG .EQ. 1) GOTO 400               ;NO BUOY AT TOP END
      I = 4                                  ;INTERSEGMENT LOAD INDEX
  340 IF (GETFLT(BCY)) GOTO 500              ;PARSE BUOYANT NODE
C
C *** COMPUTE WEIGHT DENSITY AND BREAKING STRENGTH FOR SEGMENT
C
      BNCY(SEG,TYP) = BCY                    ;SET LOAD
  400 LEN(SEG,TYP) = LNG                     ;SET LENGTH
      WGT(SEG,TYP) = AW(MTL)*DIM**BW(MTL)    ,WEIGHT
      BRK(SEG,TYP) = AB(MTL)*DIM**BB(MTL)    ;STRENGTH
      DIA(SEG,TYP) = DIM                     ,DIAMETER
      MAT(SEG,TYP) = MTL                     ;MATERIAL CODE
      EA(SEG,TYP) = ELS                      ;ELASTICITY
      IF (ELS .LT. 0.) EA(SEG,TYP) =         ;USE RIGMOOR ESTIMATE
     +    AE(MTL) * BE(MTL) * DIM * DIM      ;ELASTICITY
      GOTO 600                               ;NEXT SEGMENT
C
C *** PROMPT FOR SEGMENT PARAMETERS INDIVIDUALLY
C
  500 WRITE (CON, 5001) PMT(I), SEG          ;PROMPT
      IF (I .GT. 2) WRITE (CON, 5002)        ;PROMPT EXTENSION
     + (PMX(J,I-2), J=1,3)
      J = I + 17                             ;HELP MESSAGE NUMBER
      IF (USRINP(J)) GOTO 500                ;NEXT PARAMETER ENTRY
      GOTO (310,320,330,340) I               ;VECTORED RETURN
```

```
C
  600 CONTINUE                                     ;NEXT SEGMENT
C
C *** COMPLETE FOR REST OF LEG TYPES AND QUIT
C
  700 CONTINUE
      RETURN
C
C
 2001 FORMAT (//
     + ' No. of segments (1-5) in leg type',
     +   I3)                                       ;SEGMENT COUNT
C
 3001 FORMAT (/
     + ' Code, Diameter, Length,',
     + ' Elasticity & Node buoyancy',
     + ' in segment', I2)                          ,SEGMENT PROMPT
C
 5001 FORMAT (/A36, I3)                            ;PARAMETER PROMPT
C
 5002 FORMAT (A28)                                 ;PROMPT EXTENSION
C
      END
```

```
      LOGICAL*2 FUNCTION USRINP(HLP)
C
C *** GET A LINE OF TEXT FROM USER´S CONSOLE
C     STRIP LEADING BLANKS AND FIND LENGTH
C     CALL FOR HELP MESSAGE NO. HLP IF ? OCCURS IN STRING
C     RETURN TEXT = STRIPPED INPUT STRING
C             LTXT = STRING LENGTH
C             LPTR = 0: POINTER BEFORE BYTE 1
C           USRINP = .FALSE. AFTER NORMAL ENTRY
C                    .TRUE. AFTER HELP REQUEST
C *** TYPICAL USAGE:
C  10 WRITE (1,1000)                --PROMPT MESSAGE
C     IF (USRINP(5)) GOTO 10        --GET RESPONSE
C     IF (GETINT(I)) GOTO 10        --PARSE INTEGER I FROM INPUT STRING
C     IF (GETFLT(R)) GOTO 10        --PARSE FLOATING POINT R
C     IF (GETDBL(D)) GOTO 10        --PARSE DOUBLE PRECISION D
C     IF (GETSTR(S,L)) GOTO 10      --PARSE L BYTES INTO STRING S
C
      INTEGER*2 HLP                              ;ARGUMENT
      INTEGER*2 I, J                             ;SUBSCRIPTS
      INTEGER*2 CON, PTR, MSG, AUX, RIG          ;/UNITS/
      INTEGER*2 LTXT, LPTR                       ;/USRPTR/
      CHARACTER*1 TEXT                           ;/USRTXT/
C
      COMMON /USRPTR/ LTXT, LPTR                 ;ENTRY BUFFER POINTERS
      COMMON /USRTXT/ TEXT(80)                   ;ENTRY BUFFER
      COMMON /UNITS/ CON, PTR, MSG, AUX, RIG;LOGICAL UNITS
C
C *** GET USER INPUT AS ASCII STRING
C
      DO 100 I=1,80                              ;ERASE BUFFER
      TEXT(I) = ´ ´
  100 CONTINUE
C
  200 WRITE (CON, 1000)                          ;ESC = HELP REQUEST
      READ (CON, 1001) TEXT                      ;USER INPUT
C
      DO 210 I=1,80                              ,SCAN FOR LENGTH
      LTXT = 81-I                                ;SET LENGTH
      IF (TEXT(LTXT) .NE. ´ ´) GOTO 300          ;SCAN RIGHT TO LEFT
  210 CONTINUE
      GOTO 200                                   ;REPEAT AFTER NULL ENTRY
C
  300 DO 310 I=1,LTXT
      IF (TEXT(I) .EQ. ´?´) GOTO 500             ,HELP REQUEST
  310 CONTINUE
C
      DO 320 I=1,LTXT                            ;DELETE LEADING SPACES
      IF (TEXT(I) .NE. ´ ´) GOTO 400             ;SCAN LEFT TO RIGHT
  320 CONTINUE
      GOTO 200                                   ;NULL ENTRY
C
  400 LPTR = 0
      IF (I .EQ. 1) GOTO 420                     ;NO LEADING SPACES
      DO 410 J=I,LTXT
```

```
      LPTR = LPTR + 1
      TEXT(LPTR) = TEXT(J)                    ;SHIFT OVER LEADING SPACES
  410 CONTINUE
      LTXT = LPTR                             ;CORRECT STRING LENGTH
C
  420 USRINP = .FALSE.                        ;CLEAR HELP FLAG
      RETURN                                  ;NORMAL CALL
C
C *** HELP REQUEST
C
  500 CALL HLPMSG (HLP)                       ;PROCESS HELP REQUEST
      USRINP = .TRUE.                         ;SET HELP FLAG
      RETURN
C
 1000 FORMAT (' ?=Help: '\)                   ;HELP REQUEST PROMPT
C
 1001 FORMAT (80A1)                           ;INPUT STRING
C
      END
```

```
      LOGICAL*2 FUNCTION GETINT (I)
C
C *** EXTRACT INTEGER VALUE I FROM USER INPUT TEXT
C     IGNORE LEADING NON-NUMERICS
C     RETURN I = INTEGER VALUE
C       GETINT = .FALSE. IF NORMAL INPUT   .TRUE. IF INPUT ERROR
C         LPTR = BYTE AFTER LAST CONSECUTIVE NUMERIC
C
C *** TYPICAL USAGE:
C  10 WRITE (1,1000)               --PROMPT MESSAGE
C     IF (USRINP(5)) GOTO 10       --GET RESPONSE
C     IF (GETINT(I)) GOTO 10       --PARSE INTEGER I FROM INPUT STRING
C
      LOGICAL*2 MINUS                       ;LOCAL: NEGATIVE FLAG
      INTEGER*2 I, J
      INTEGER*2 LTXT, LPTR                  ;/USRPTR/
      CHARACTER TEXT                        ;/USRTXT/
C
      COMMON /USRPTR/ LTXT, LPTR
      COMMON /USRTXT/ TEXT(80)              ;ENTRY BUFFER
C
      GETINT = .TRUE.                       ,SET ERROR FLAG
      MINUS = .FALSE.                       ;CLEAR NEGATIVE FLAG
      I = 0                                 ,SET NULL VALUE
C
C *** PRESCAN FOR LEADING NON-NUMERICS AND SIGN
C
  100 LPTR = LPTR + 1                       ;GET NEXT BYTE
      IF (LPTR .GT. LTXT) RETURN            ;NO VALUE ERROR
      IF (TEXT(LPTR) .EQ. '-') GOTO 400     ;SET MINUS FLAG
      IF (TEXT(LPTR) .LT. '0') GOTO 100     ;IGNORE LEADING
      IF (TEXT(LPTR) .GT. '9') GOTO 100     ; NON-NUMERICS
C
C *** MAIN SCAN OVER NUMERICS
C
  200 I = 10*I + ICHAR(TEXT(LPTR)) - 48     ;ACCUMULATE DIGITS
      GETINT = .FALSE.                      ;CLEAR ERROR
  210 LPTR = LPTR + 1                       ;NEXT BYTE
      IF (LPTR .GT. LTXT) GOTO 300          ;QUIT ON END OF LINE
      IF (TEXT(LPTR) .LT. '0') GOTO 300     ,QUIT ON NON-NUMERIC
      IF (TEXT(LPTR) .LT. ':') GOTO 200     ;CONTINUE ON NUMERIC
C
C *** SET SIGN AND RETURN VALID INTEGER
C
  300 IF (MINUS) I = -I                     ;SET NEGATIVE
      RETURN                                ;DONE
C
C *** SET MINUS FLAG
C
  400 MINUS = .TRUE.
      GOTO 210                              ;ENTER MAIN SCAN
C
      END
```

```
      LOGICAL*2 FUNCTION GETFLT (R)
C
C *** EXTRACT REAL VALUE R FROM USER INPUT TEXT
C     IGNORE LEADING NON-NUMERICS
C     RETURN R = REAL VALUE
C       GETFLT = .FALSE. IF NORMAL INPUT   .TRUE. IF INPUT ERROR
C         LPTR = BYTE AFTER LAST CONSECUTIVE NUMERIC
C
C *** TYPICAL USAGE:
C  10 WRITE (1,1000)              --PROMPT MESSAGE
C     IF (USRINP(5)) GOTO 10      --GET RESPONSE
C     IF (GETFLT(R)) GOTO 10      --PARSE FLOATING POINT R
C
      REAL R                                  ;ARGUMENT
      LOGICAL*2 GETDBL                        ;FUNCTION
      DOUBLE PRECISION D                      ;LOCAL
C
C *** USE DOUBLE PRECISION TO EXTRACT VALUE FROM STRING
C
      GETFLT = GETDBL(D)                      ;D=USER VALUE
      R = D                                   ;IN SINGLE PRECISION
      RETURN
C
      END
```

```
      LOGICAL*2 FUNCTION GETDBL (D)
C
C *** EXTRACT DOUBLE PRECISION VALUE D FROM USER INPUT TEXT
C     IGNORE LEADING NON-NUMERICS
C     RETURN D = DOUBLE PRECISION VALUE
C       GETDBL = .FALSE. IF NORMAL INPUT   .TRUE. IF INPUT ERROR
C         LPTR = BYTE AFTER LAST CONSECUTIVE NUMERIC
C
C *** TYPICAL USAGE:
C  10 WRITE (1,1000)                 --PROMPT MESSAGE
C     IF (USRINP(5)) GOTO 10         --GET RESPONSE
C     IF (GETDBL(D)) GOTO 10         --PARSE DOUBLE PRECISION D
C
      DOUBLE PRECISION D                         ,ARGUMENT
      DOUBLE PRECISION F, G                      ;LOCALS
      INTEGER*2 J                                ;DIGIT
      LOGICAL*2 MINUS                            ;MINUS FLAG
      INTEGER*2 LTXT, LPTR                       ,/USRPTR/
      CHARACTER TEXT                             ;/USRTXT/
C
      COMMON /USRPTR/ LTXT, LPTR
      COMMON /USRTXT/ TEXT(80)                   ;ENTRY BUFFER
C
      DATA F /1.D-01/                            ;DECIMAL FRACTION RATIO
C
      GETDBL = .TRUE.                            ,SET ERROR FLAG
      MINUS = .FALSE.                            ;CLEAR NEGATIVE FLAG
      D = 0.DO                                   ;NULL WHOLE VALUE
      G = F                                      ;SET FRACTION FACTOR
C
C *** PRESCAN FOR LEADING NON-NUMERICS AND SIGN
C
  100 LPTR = LPTR + 1                            ;GET NEXT BYTE
      IF (LPTR .GT. LTXT) RETURN                 ;NO VALUE ERROR
      IF (TEXT(LPTR) .EQ. '-') GOTO 400          ;SET MINUS FLAG
      IF (TEXT(LPTR) .EQ. '.') GOTO 500          ;POINT: FRACTION ONLY
      IF (TEXT(LPTR) .LT. '0') GOTO 100          ,IGNORE LEADING
      IF (TEXT(LPTR) .GT. '9') GOTO 100          ; NON-NUMERICS
C
C *** MAIN SCAN OVER NUMERICS
C
  200 J = ICHAR(TEXT(LPTR))                      ;ASCII CODE FOR NUMERAL
      D = 10*D + J - 48                          ;ACCUMULATE WHOLE NUMBER
      GETDBL = .FALSE.                           ;CLEAR ERROR
  210 LPTR = LPTR + 1                            ;NEXT BYTE
      IF (LPTR .GT. LTXT) GOTO 300               ;QUIT ON END OF LINE
      IF (TEXT(LPTR) .EQ. '.') GOTO 500          ;POINT: SCAN FRACTION
      IF (TEXT(LPTR) .LT. '0') GOTO 300          ,QUIT ON NON-NUMERIC
      IF (TEXT(LPTR) .LT. ':') GOTO 200          ;CONTINUE ON NUMERIC
C
C *** SET SIGN AND RETURN VALID INTEGER
C
  300 IF (MINUS) D = -D                          ,SET NEGATIVE
      RETURN                                     ;DONE
C
```

```
C *** SET MINUS FLAG
C
  400 MINUS = .TRUE.
      GOTO 210                                   ;ENTER MAIN SCAN
C
C *** SCAN OVER FRACTION
C
  500 LPTR = LPTR + 1                            ,NEXT BYTE
      IF (LPTR .GT. LTXT) GOTO 300               ;QUIT ON END OF LINE
      IF (TEXT(LPTR) .LT. '0') GOTO 300          ;QUIT ON NON-NUMERIC
      IF (TEXT(LPTR) .GT. '9') GOTO 300
      D = D + G*(ICHAR(TEXT(LPTR)) - 48)         ;ACCUMULATE FRACTION
      G = F*G                                    ;ADJUST FACTOR
      GETDBL = .FALSE.                           ,CLEAR ERROR
      GOTO 500
C
      END
```

```
      LOGICAL*2 FUNCTION GETSTR (S, L)
C
C *** EXTRACT L BYTES FROM USER INPUT TEXT INTO STRING S
C     RETURN S = STRING VALUE  L = ACTUAL STRING LENGTH PARSED
C       GETSTR = .FALSE. IF NORMAL INPUT   .TRUE. IF INPUT ERROR
C         LPTR = BYTE AFTER LAST CONSECUTIVE NUMERIC
C
C *** TYPICAL USAGE:
C  10 WRITE (1,1000)              --PROMPT MESSAGE
C     IF (USRINP(5)) GOTO 10      --GET RESPONSE
C     IF (GETSTR(S,L)) GOTO 10    --PARSE L BYTES INTO STRING S
C
      CHARACTER S(L)                            ;ARGUMENTS
      INTEGER*2 L
      INTEGER*2 I                               ;LOCAL
      INTEGER*2 LTXT, LPTR                      ;/USRPTR/
      CHARACTER TEXT                            ;/USRTXT/
C
      COMMON /USRPTR/ LTXT, LPTR
      COMMON /USRTXT/ TEXT(80)                  ,ENTRY BUFFER
C
      GETSTR = .TRUE.                           ;SET ERROR FLAG
      DO 90 I=1,L                               ;ERASE STRING
      S(I) = ' '
   90 CONTINUE
      I = 0                                     ;SET BYTE COUNTER
C
C *** PRESCAN FOR LEADING SPACES
C
  100 LPTR = LPTR + 1                           ;GET NEXT BYTE
      IF (LPTR .GT. LTXT) GOTO 210              ;ERROR: NOTHING TO PARSE
      IF (TEXT(LPTR) .EQ. ' ') GOTO 100         ;IGNORE LEADING SPACES
C
C *** MAIN SCAN OVER L BYTES
C
      GETSTR = .FALSE.                          ,CLEAR ERROR
  200 I = I + 1                                 ;SHIFT POINTER
      IF (I .GT. L) RETURN
      S(I) = TEXT(LPTR)                         ,EXTRACT BYTE
      LPTR = LPTR + 1                           ;NEXT BYTE
      IF (LPTR .LE. LTXT) GOTO 200              ;CHECK FOR END OF LINE
  210 L = I                                     ;RETURN ACTUAL LENGTH
      RETURN                                    ,DONE
C
      END
```

```
C$DEBUG
$STRICT                                       ;MODULE LOAD.FOR
C                                             ;SUBROUTINE XVSH
C                                             ;SUBROUTINE MAPLEG
C                                             ,SUBROUTINE HVSS
C                                             ;SUBROUTINE PRELOD
C                                             ;REAL FUNCTION STAN
C
C     -------------------------------------------------------------
C
      SUBROUTINE XVSH (TSOPT, WROPT)          ;D.B.DILLON  EG&G  1985
C
C ***  GIVEN: TSOPT=.TRUE. USE TOP SEGMENT LENGTH = ATOP(ANC)
C                    ELSE  USE LEN(1,LTYP(ANC)) INSTEAD
C             WROPT=.TRUE. WRITE RESULTS TO X VS H FILE
C      THEN   FILL X VS H ARRAY
C
      LOGICAL*2 MAKFIL                        ;FUNCTIONS
      REAL STPSIZ
      EXTERNAL MAKFIL, STPSIZ
      LOGICAL*4 TSOPT                         ;ARGUMENT: TOP SEGMENT OPTION
      LOGICAL*4 WROPT                         ;WRITE X VS H FILE OPTION
      INTEGER*2 I                             ;LOCALS: LOAD INDEX
      INTEGER*2 S                             ;SEGMENT INDEX
      INTEGER*2 SGS                           ,SEGMENT COUNT.
      INTEGER*2 NX                            ;NODE LIMIT
      INTEGER*2 TYP                           ;LEG TYPE
      INTEGER*2 ANC                           ;ANCHOR NO.
      INTEGER*2 MT                            ,MASTER LEG NO.
      REAL LGTH                               ;LEN(1,TYP) STORAGE
      REAL HX                                 ;MAXIMUM LOAD
      REAL DH                                 ;INCREMENT LOAD
      REAL X(6)                               ;NODE DISPLACEMENT
      CHARACTER XHFIL(6)                      ,FILE NAME SUFFIX
      INTEGER*2   MTYPS, MLEG, MTYP           ,/LEGMAP/
      REAL XTAB, HTAB, STAB                   ;/HVSX/
      INTEGER*2 NOS, NOB, NSF                 ;/XHTABL/
      REAL SPAN, SB, ST, FMIN, DX, DY,
     +     V, U, Y, SL, BL
      INTEGER*2 ANCS, LTYP                    ;/ANCHOR/
      REAL DEPTH, OFFSET, SAFETY,
     +    ADIR, ATOP, ARAD, APRE, ANCX, ANCY
      INTEGER*2 TYPS, SEGS, MAT               ;/CABLES/
      REAL DIA, BRK, LEN, WGT, EA, BNCY
      INTEGER*2 CON, PTR, MSG, AUX, RIG       ;/UNITS/
      CHARACTER RIGNAM, JOBNAM                ;/JOB/
      INTEGER*2 RNL, JNL                      ;/NAMLEN/
C
      COMMON /NAMLEN/ RNL, JNL                ;NAME LENGTHS
      COMMON /JOB/ RIGNAM(72), JOBNAM(72)     ;PROBLEM ID
      COMMON /UNITS/ CON, PTR, MSG, AUX, RIG;LOGICAL UNITS
      COMMON /CABLES/ TYPS, SEGS(12),
     +    MAT(5,12), DIA(5,12),
     +    BRK(5,12), LEN(5,12), WGT(5,12),
     +    EA(5,12), BNCY(5,12)                ;SEGMENT PARAMETERS
```

```
      COMMON /ANCHOR/ ANCS, LTYP(12),
     +    DEPTH, OFFSET, SAFETY,
     +    ADIR(12), ATOP(12), ARAD(12),
     +    APRE(12), ANCX(12), ANCY(12)       ;ANCHOR PARAMETERS
      COMMON /XHTABL/                        ;X VS H RECORD:
     +    NOS, NOB, NSF,                     ;SURFACE/BOTTOM/SAFETY NODE
     +    SPAN, SB, ST,                      ;X  BOTTOMED SCOPE  STRETCH
     +    FMIN,                              ;MIN. SAFETY FACTOR @ NSF
     +    DX(5), DY(5),                      ;SEGMENT OFFSETS
     +    V(5), U(5),                        ,VERTICAL TOP & BOTTOM LOAD
     +    Y(6),                              ;NODE DEPTH LIST
     +    SL(5), BL(5)                       ,STRETCHED/BOTTOMED LENGTH
      COMMON /HVSX/ XTAB(80,12),             ,SPAN(LOAD ITEM, ANCHOR)
     +              HTAB(80,12),             ;LOAD(ITEM, ANCHOR)
     +              STAB(80,12)              ;LEAST SAFETY FACTOR(ITEM, ANCHOR)
      COMMON /LEGMAP/
     +    MTYPS,                             ;NO. MASTER TYPES
     +    MLEG(12),                          ;MASTER LEG FOR TYPE I
     +    MTYP(12)                           ;TYPE FOR MASTER LEG I
C
      DATA XHFIL /'X', 'H', '.', 'L', 2*'0'/;RIGNAMXH.LNN
C
C *** OPEN DISPLACEMENT FILE BY LEG TYPE
C
      TYP = 1                                ;DUMMY PRINT FLAG
      IF (WROPT) CALL PRTDEF (TYP)           ;DISPLAY CASE
      CALL MAPLEG (TSOPT)                    ;BUILD MASTER LEG MAP
      DO 500 MT=1,MTYPS                      ;SCAN OVER MASTER LEGS
      ANC = MLEG(MT)                         ;ANC=MASTER LEG NO.
      I = 0                                  ;START RECORD COUNTER
      IF (.NOT. WROPT) GOTO 10               ;X VS H FILE SUPPRESSED
      IF (MAKFIL (XHFIL(1), ANC, AUX))
     +    GOTO 400                           ,OPEN XVSH FILE FOR ANCHOR
C
C *** SET UPPER LIMIT TO BREAK STRONGEST ELEMENT
C
   10 TYP = LTYP(ANC)                        ,SET LEG TYPE FOR ANCHOR
      LGTH = LEN(1,TYP)                      ;STORE LENGTH FOR TYPE
      IF (TSOPT) LEN(1,TYP) = ATOP(ANC)      ,.TRUE.: USE ADJUSTED LENGTH
      SGS = SEGS(TYP)                        ;DUMMY SIMPLE VARIABLE
      NX = SGS + 1                           ,INCLUDE BOTH END NODES
      HX = 0.                                ;START FORCE LIMIT
      DO 200 S=1,SGS                         ;SCAN ELEMENTS IN TYPE
      HX = AMAX1(HX,BRK(S,TYP))              ;SELECT LARGEST BREAK STR.
  200 CONTINUE
C
      DH = STPSIZ(HX/40)                     ;SELECT STEP SIZE
      H = 0.                                 ;START WITH NO LOAD
C
C *** COMPUTE AND RECORD THE X VS H TABLE IN NH RECORDS
C
  300 CALL GETXVH (SGS, DEPTH, H,
     + WGT(1,TYP), LEN(1,TYP),
     + EA(1,TYP), BNCY(1,TYP), BRK(1,TYP))   ;SOLVE FOR X(H)
C
```

```
      I = I + 1                                      ;COUNT RECORD
      IF (WROPT) WRITE (AUX)
     +     H, SPAN, SB, ST, FMIN,
     +     I, NOS, NOB, NSF,
     +     (DX(S), DY(S), V(S),
     +     U(S), SL(S), S=1,5)                      ;TABULATE X VS H ON FILE
C
      XTAB(I,MT) = SPAN                             ;FILL /HVSX/
      HTAB(I,MT) = H                                ; TABLES FOR
      STAB(I,MT) = FMIN                             ;  ANALYZ COMMAND
C
C *** ACCUMULATE SEGMENT OFFSETS FOR NODE POSITIONS
C
      X(1) = 0.                                     ;RADIUS ORIGIN AT FAIRLEAD
      Y(1) = 0.                                     ;DEPTH ORIGIN AT FAIRLEAD
      DO 310 S=1,SGS                                ;SUM INCREMENTS FOR
      X(S+1) = X(S) + DX(S)                         ;NODE LOCATIONS
      Y(S+1) = Y(S) - DY(S)                         ;INVERT DEPTH
  310 CONTINUE
C
      H = H + DH                                    ;NEXT H
      IF (H .LE. HX) GOTO 300                       ;DO IT
C
      IF (WROPT) ENDFILE AUX                        ;ELSE CLOSE X VS H FILE
      LEN(1,TYP) = LGTH                             ; RESTORE SEGMENT 1
C
  400 IF (I .GT. 79) GOTO 500                       ;  AND PAD UNUSED
      S = I + 1                                     ;   SPAN TABLE
      DO 410 I=S,80                                 ;    WITH -1'S
      XTAB(I,ANC) = -1.
  410 CONTINUE
C
  500 CONTINUE                                      ;NEXT ANCHOR
      RETURN
C
C *** FORCE DISPLACEMENT TABLE FILE FORM
C     NAME: RIGNAMXH.LNN, NN=LEG TYPE 01-12
C     FORM: 128 UNFORMATTED BYTES PER RECORD
C       BYTE  CONTENT . . .
C REAL:    0   HORIZONTAL FORCE IN LEG
C          4   HORIZONTAL SPAN OF LEG
C          8   MATERIAL LENGTH (UNSTRETCHED) LYING ON BOTTOM
C         12   STRETCHED LENGTH OF LEG
C         16   LEAST TENSILE SAFETY FACTOR IN LEG (TENSION/STRENGTH)
C INTEG. 20   RECORD NO.
C         22   NUMBER OF NODES ON SURFACE (BARGE NODE = 1)
C         24   NUMBER OF NODES ON BOTTOM (ANCHOR NODE = 1)
C         26   ELEMENT NUMBER HAVING LEAST SAFETY FACTOR
C REAL 4*5I+8 HORIZONTAL SPAN OF SEGMENT I    (I=1,2,3,4,5)
C        +12   VERTICAL SPAN
C        +16   VERTICAL LOAD AT "BARGE" END OF SEGMENT (+: SEGMENT SUPPORTED)
C        +20   VERTICAL LOAD AT "ANCHOR" END (+: SEGMENT SUPPORTING)
C        +24   STRETCHED LENGTH
C
      END
```

```
      SUBROUTINE MAPLEG (TSOPT)                       ;AUG-85 - D.B.DILLON - EG&G
C
C *** BUILD MAP OF UNIQUE LEGS, INCLUDING VARIATION IN TOP SCOPE
C
      LOGICAL*2   TSOPT                              ;ARGUMENT .TRUE.: ATOP=TOP SCOPE
      INTEGER*2   I                                  ;LOCAL: LEG INDEX
      INTEGER*2   J                                  ;MASTER LEG TYPE
      INTEGER*2   L                                  ;MASTER LEG NO.
      INTEGER*2   MTYPS, MLEG, MTYP                  ;/LEGMAP/
      INTEGER*2   ANCS, LTYP                         ;/ANCHOR/
      REAL        DEPTH, OFFSET, SAFETY,
     +            ADIR,  ATOP,   ARAD,
     +            APRE,  ANCX,   ANCY
      INTEGER*2 CON, PTR, MSG, AUX, RIG              ;/UNITS/
C
      COMMON /UNITS/ CON, PTR, MSG, AUX, RIG;LOGICAL UNITS
      COMMON /ANCHOR/
     +    ANCS,      LTYP(12),
     +    DEPTH,     OFFSET,    SAFETY,
     +    ADIR(12), ATOP(12), ARAD(12),
     +    APRE(12), ANCX(12), ANCY(12)
      COMMON /LEGMAP/
     +    MTYPS,                                     ;NO. MASTER TYPES
     +    MLEG(12),                                  ;MASTER LEG FOR TYPE I
     +    MTYP(12)                                   ;MASTER TYPE FOR LEG I
C
C *** CLEAR PRIOR MAPPING
C
      MTYPS = 1                                      ;ALWAYS AT LEAST 1 MASTER LEG
      DO 100 I=1,12                                  ;SCAN ARRAY DIMENSION
      MLEG(I) = 1                                    ;LEG 1 MASTERS ALL TYPES
      MTYP(I) = 1                                    ;ALL LEGS ARE TYPE 1
  100 CONTINUE
C
C *** LOCATE OTHER MASTER LEGS BY SCANNING REMAINING LEGS
C     A LEG IS A MASTER IF IT IS UNLIKE ANY OF THE EXISTING MASTERS
C
      DO 300 I=2,ANCS                                ;SKIP LEG 1, ALREADY FLAGGED
      DO 200 J=1,MTYPS                               ;SCAN EXISTING MASTERS
      L = MLEG(J)                                    ,MASTER LEG NO. FOR TYPE J
      IF ((LTYP(L) .NE. LTYP(I)) .OR.                ;NO MATCH ON TYPE
     +   (TSOPT .AND.
     +   (ATOP(L) .NE. ATOP(I))))                    ,NOR ON PRELOAD SCOPE OPTION
     +   GOTO 200                                    ;LEG J NOT MASTER FOR LEG I
      MTYP(I) = J                                    ;MATCH: LEG J REPRESENTS LEG I
      GOTO 300                                       ;END SUB-SCAN AFTER MATCH
  200 CONTINUE
C
      MTYPS = MTYPS + 1                              ;NO MATCH IN SUB-SCAN MEANS
      MLEG(MTYPS) = I                                ;LEG I IS NEXT MASTER IN LIST
      MTYP(I) = MTYPS                                ; AND USES NEXT TYPE NO.
  300 CONTINUE
      RETURN
C
      END
```

```
      SUBROUTINE HVSS                         ;RECOVERED  D.B.DILLON  EG&G
C
C *** GENERATE H VS S FILES FOR ALL DISTINCT LEG STYLES ON A RIG
C
      LOGICAL*2 MAKFIL                        ;FUNCTION
      EXTERNAL  MAKFIL
      CHARACTER HSFIL(6)                      ;LOCAL
      INTEGER*2 TYP
      INTEGER*2 TYPS, SEGS, MAT               ;/CABLES/
      REAL DIA, BRK, LEN, WGT, EA, BNCY
      INTEGER*2 CON, PTR, MSG, AUX, RIG       ;/UNITS/
      CHARACTER RIGNAM, JOBNAM                ;/JOB/
      INTEGER*2 RNL, JNL                      ;/NAMLEN/
C
      COMMON /NAMLEN/ RNL, JNL                ;NAME LENGTHS
      COMMON /JOB/ RIGNAM(72), JOBNAM(72)     ;PROBLEM ID
      COMMON /UNITS/ CON, PTR, MSG, AUX, RIG;LOGICAL UNITS
      COMMON /CABLES/ TYPS, SEGS(12),
     +    MAT(5,12), DIA(5,12),
     +    BRK(5,12), LEN(5,12), WGT(5,12),
     +    EA(5,12), BNCY(5,12)                ;LEG SEGMENT PARAMETERS
C
      DATA HSFIL /'H', 'S', '.', 'L', 2*'0'/;RIGNAMHS.LNN
C
C *** TABULATE THE FORCE VS SCOPE FUNCTION FOR EACH MOORING LEG TYPE
C
      DO 200 TYP=1,TYPS                       ;STYLE LOOP
      IF (MAKFIL(HSFIL(1),TYP,AUX)) GOTO 200;OPEN H VS S FILE
      CALL PRELOD(TYP)                        ;COMPUTE H VS S TABLE
      ENDFILE AUX                             ;CLOSE H VS S FILE
  200 CONTINUE
      RETURN                                  ;TASK COMPLETE
C
C *** LOAD/SPAN VS SCOPE FILE FORMAT
C     NAME:   rignamHS.Lnn   rigna    ;/UNITS/
C     FORM:   9 UNFORMATTED 4-BYTE REAL VALUES PER RECORD
C     ITEM  NAME  VALUE
C       1   SCOPE UNSTRETCHED LENGTH OF TOPMOST SEGMENT
C       2   XMIN  SLACK LOAD    +: UPWARD FORCE ON ANCHOR  -: LENGTH ON BOTTOM
C       3   SBPL  DITTO AT PRELOAD
C       4   SBDL  DITTO AT DESIGN LOAD
C       5   XPRE  SPAN AT THE PRELOAD CONSISTENT WITH DESIGN LOAD
C       6   XDES  SPAN AT THE LOAD CONSISTENT WITH SAFETY FACTOR
C       7   HPRE  PRELOAD WITH SPAN N%*DEPTH LESS THAN DESIGN SPAN
C       8   HDES  DESIGN LOAD CONSISTENT WITH SAFETY FACTOR
C       9   HDIF  HOLDING POWER = HDES - HPRE
C
      END
```

```
      SUBROUTINE PRELOD (TYP)                 ;6-DEC-84  D.B.DILLON  EG&G
C
C *** RECORD PRELOAD, PRESPAN, AND DESIGN LOAD VS TOP ELEMENT LENGTH
C
C *** GIVEN: RIG DEFINITION FILE OPENED AND READ INTO COMMON BLOCKS
C            OFFSET=LEG DEFLECTION AT DESIGN LOAD
C            SAFETY=LEG SAFETY FACTOR AT DESIGN LOAD
C            TYP=LEG TYP SUBSCRIPT
C
C *** RELIEVE TENSION IN LEG TO STAY WITHIN DESIGN LOAD CAPACITY
C
      INTEGER*2 TYP                           ;ARGUMENT
      REAL STAN, STPSIZ                       ;FUNCTIONS
C                                             ;LOCALS:
      INTEGFR*2 TRYS, STEP, TRYD, TRYP, TRYV;ITERATION INDEXES
      INTEGER*2 LSEG                          ,SEGMENT INDEX
      INTEGER*2 LSGS                          ;SEGMENTS IN LEG
      REAL LEN1                               ;ORIGINAL LENGTH OF SEGMENT 1
      REAL FIXD                               ;LENGTH IN SEGMENTS 2-
      REAL XTL                                ;LENGTH OF EXTENSIBLE SEGMENTS
      REAL HDES                               ;DESIGN HOLDING POWER
      REAL HPOS                               ;H LIMIT FOR + ERROR
      REAL HNEG                               ;"    "    "  - ERROR
      REAL HLIM                               ;"    "    "    CASE
      REAL HPRE                               ;DESIGN PRELOAD
      REAL HDIF                               ;HOLDING POWER, HDES - HPRE
      REAL HERR                               ;LOAD ERROR, (HPOS-HNEG)/H
      REAL HZRO                               ;SLACK LOAD = 0.
      REAL SBDL                               ;BOTTOM SCOPE, DESIGN LOAD
      REAL SBPL                               ;DITTO, PRELOAD
      REAL SCOPE                              ;LENGTH OF TOP SEGMENT
      REAL DSCOP                              ;SCOPE INCREMENT
      REAL DSC1                               ;ALTERNATE INCREMENT
      REAL SMAX                               ;SCOPE LIMIT
      REAL XMIN                               ;SLACK SPAN
      REAL XPRE                               ;DESIGN PRESPAN
      REAL XDES                               ;DESIGN SPAN
      REAL XERR                               ;PRESPAN ERROR
      REAL XOFF                               ;WORKING SPAN-PRESPAN SPEC.
      REAL FERR                               ;SAFETY FACTOR ERROR
      REAL EPOS                               ;POSITIVE ERROR LIMIT
      REAL ENEG                               ;NEGATIVE ERROR LIMIT
      REAL TOL                                ;NORMAL CONVERGENCE LIMIT
      REAL WTOL                               ;WORKING CONVERGENCE
      INTEGER*2 NOS, NOB, NSF                 ;/XHTABL/
      REAL SPAN, SB, ST, FMIN, DX, DY,
     +     V, U, Y, SL, BL
      INTEGER*2 ANCS, LTYP                    ;/ANCHOR/
      REAL DEPTH, OFFSET, SAFETY,
     +    ADIR, ATOP, ARAD, APRE, ANCX, ANCY
      INTEGER*2 TYPS, SEGS, MAT               ;/CABLES/
      REAL DIA, BRK, LEN, WGT, EA, BNCY
      INTEGER*2 CON, PTR, MSG, AUX, RIG       ;/UNITS/
C
      COMMON /UNITS/ CON, PTR, MSG, AUX, RIG;LOGICAL UNITS
```

```
      COMMON /CABLES/ TYPS, SEGS(12),
     +    MAT(5,12), DIA(5,12),
     +    BRK(5,12), LEN(5,12), WGT(5,12),
     +    EA(5,12), BNCY(5,12)                ;LEG SEGMENT PARAMETERS
      COMMON /ANCHOR/ ANCS, LTYP(12),
     +    DEPTH, OFFSET, SAFETY,
     +    ADIR(12), ATOP(12), ARAD(12),
     +    APRE(12), ANCX(12), ANCY(12)        ;ANCHOR PARAMETERS
      COMMON /XHTABL/
     +    NOS, NOB, NSF,
     +    SPAN, SB, ST,
     +    FMIN, DX(5), DY(5), V(5), U(5),
     +    Y(6), SL(5), BL(5)                  ;X VS H RECORD
C
      DATA TRYS /100/                         ;ITERATION LIMIT
      DATA TOL /2.5E-7/                       ;CONVERGENCE LIMIT
      DATA HZRO /0./                          ;SLACK LOAD
C
C *** SET CASE CONSTANTS
C
      TRYV = TRYS/2                           ;DELAYED DISPLAY
      WTOL = TOL                              ;SET WORKING TOLERENCE
      XOFF = OFFSET*DEPTH/100.                ;OFFSET IN FEET
      LSGS = SEGS(TYP)                        ;SEGMENTS IN LEG
      FIXD = 0.                               ;FIXED LENGTH SUM
      LEN1 = LEN(1,TYP)                       ;SAVE ORIGINAL LENGTH
      XTL = 0.                                ;EXTENSIBLE LENGTH SUM
      IF (EA(1,TYP) .GT. 0.) XTL=LEN(1,TYP)
      HDES = BRK(1,TYP)                       ;1ST DESIGN LOAD ESTIMATE
      IF (LSGS .EQ. 1) GOTO 110               ;UNIFORM LEG
C
      DO 100 LSEG=2,LSGS                      ;SCAN OTHER SEGMENTS
      FIXD = FIXD + LEN(LSEG,TYP)             ;SUM FIXED SCOPE
      IF (EA(LSEG,TYP) .GT. 0.)
     +            XTL = XTL + LEN(LSEG,TYP)   ;SUM EXTENSIBLE LENGTH
      HDES = AMIN1(HDES,BRK(LSEG,TYP))        ;USE MINIMUM STRENGTH
  100 CONTINUE
C
  110 HLIM = HDES/SAFETY                      ;LOAD UPPER BOUND
      HDES = HLIM/1.5                         ;ALLOW FOR ANGLE
      SCOPE = AMAX1(0.,DEPTH-FIXD)            ;START WITH LEAST SCOPE
      SMAX = STAN(BRK(1,TYP),                 ;SCOPE FOR SEG 1 TANGENT TO
     + WGT(1,TYP), EA(1,TYP), DEPTH, HNEG)    ; BOTTOM AT TENSION=B.S.
      SMAX = AMAX1(SMAX, 10.*DEPTH)           ;AT LEAST 10 TIMES DEPTH
      DSC1 = STPSIZ((SMAX - SCOPE)/40.)       ;SCAN IN ABOUT 40 STEPS
      DSCOP = DSC1 * (AINT(SCOPE/DSC1) + 1.);ADJUST 1ST STEP TO ALIGN
     +      - SCOPE                           ; INITIAL SCOPE WITH STEP
C#      WRITE (CON, 8001) XOFF, FIXD, HDES,     ;#####
C#     +                SMAX, SCOPE, DSC1       ;#####
C# 8001 FORMAT (' PRELOAD@110'/                ;#####
C#     + '       XOFF      FIXD      HDES',    ;#####
C#     + '       SMAX     SCOPE      DSC1'/    ;#####
C#     +  5F10.2, F10.3)                        ;#####
C
      WRITE (CON, 1000) TYP,
```

```
   + SCOPE, SMAX, DSC1, HLIM                  ;SCREEN TABLE HEADER
    IF (XTL/(FIXD+LEN(1,TYP)) .LT. 0.2)       ;SKIP VERTICAL CASE FOR
   +                                 GOTO 410 ;CABLES < 20% EXTENSIBLE
C
C *** SCAN THRU A RANGE OF TOP-ELEMENT SCOPES
C
  200 LEN(1,TYP) = SCOPE                      ;SET SCOPE
      STEP = TRYS                             ;SET ITERATION LIMIT
      TRYD = 0                                ;DES. LOAD ITERATION COUNT
      CALL GETXVH (LSGS, DEPTH, HZRO,         ;FILL /XHTABL/ @ ZERO LOAD
     +  WGT(1,TYP), LEN(1,TYP), EA(1,TYP),
     +  BNCY(1,TYP), BRK(1,TYP))
      EPOS = FMIN - SAFETY                    ;MAXIMUM RESERVE
      IF (FMIN .LE. SAFETY) GOTO 500          ;ABORT ON WEAK LEG
      HPOS = 0.                               ;H NEVER NEGATIVE
      XMIN = -SPAN                            ;SLACK SPAN
      IF (ABS(XMIN/DEPTH) .LT. .001)          ; ELSE VERTICAL
     +    XMIN = U(LSGS)                      ;  LOAD ON ANCHOR
      HDES = HLIM                             ;START WITH OVERLOAD
C
C *** PART 1: FIND DESIGN H FOR LEAST SAFETY FACTOR EQUAL TO GIVEN VALUE
C
  210 CALL GETXVH (LSGS, DEPTH, HDES,         ;FILL /XHTABL/ @TRIAL HDES
     +  WGT(1,TYP), LEN(1,TYP), EA(1,TYP),
     +  BNCY(1,TYP), BRK(1,TYP))
C
      TRYD = TRYD + 1                         ;COUNT ITERATIONS
      FERR = FMIN - SAFETY                    ;ERROR IN SAFETY FACTOR
      HERR = (HPOS - HNEG)/HDES               ;UNCONVERGED RANGE
C
      IF (TRYD - TRYV) 218,212,212            ;SLOW CONVERGENCE DISPLAY
  212 WTOL = 1.25*WTOL                        ;START RELAXING TOLERANCE
      WRITE (PTR, 1002)
     +    TRYD, HDES, HPOS, HNEG,
     +    HERR, FERR, EPOS, ENEG
C
  218 IF (ABS(HERR) .LT. WTOL) GOTO 300       ;DESIGN ERROR CONVERGED
C
      IF (FERR) 220,300,230                   ;RESET LIMIT ON SIGN
  220 HNEG = HDES                             ;MOVE NEG. LIMIT
      ENEG = FERR
      GOTO 240
  230 HPOS = HDES                             ;MOVE POS. LIMIT
      EPOS = FERR
  240 HDES = (HNEG + HPOS) / 2.               ;BINARY SEARCH
      GOTO 210                                ;ITERATE
C
C *** PART 2: FIND PRESPAN FROM DESIGN SPAN LIMIT ---------------------
C
  300 XDES = SPAN                             ;SPAN AT DESIGN LOAD
      SBDL = -SB                              ;ON BOTTOM AT DESIGN LOAD
      IF (ABS(SBDL/DEPTH) .LT. .001)          ; ELSE VERTICAL
     +    SBDL = U(LSGS)                      ;  LOAD ON ANCHOR
      XPRE = XDES - XOFF                      ;SPAN DUE TO PRELOAD
      IF (XPRE .LE. XMIN) GOTO 390            ;PRELOAD IS ZERO
```

```
      HPOS = HDES                          ;RESET LIMTS
      HNEG = 0.
      EPOS = XOFF                          ;+ ERROR AT HPRE=HDES
      ENEG = XMIN - XPRE                   ;- ERROR AT HPRE=0.
      TRYP = 0                             ;PRELOAD ITERATION COUNT
      WTOL = TOL                           ;SET WORKING TOLERENCE
C
C *** PART 3: FIND PRELOAD FROM PRESPAN --------------------------------
C
  310 HPRE = (HNEG + HPOS) / 2.            ;INTERPOLATE
      CALL GETXVH (LSGS, DEPTH, HPRE,      ;FILL /XHTABL/ @TRIAL HPRE
     +  WGT(1,TYP), LEN(1,TYP), EA(1,TYP),
     +  BNCY(1,TYP), BRK(1,TYP))
C
      TRYP = TRYP + 1                      ;COUNT ITERATIONS
      XERR = SPAN - XPRE                   ;PRESPAN ERROR
      HERR = (HPOS - HNEG)/HPRE            ;UNCONVERGED RANGE
C
      IF (TRYP - TRYV) 318,312,312         ;SLOW CONVERGENCE DISPLAY
  312 WTOL = 1.25*WTOL                     ;START RELAXING TOLERANCE
      WRITE (PTR, 1003)
     +    TRYD, HPRE, HPOS, HNEG,
     +    HERR, FERR, EPOS, ENEG
C
  318 IF (ABS(HERR) .LT. WTOL) GOTO 400    ;DESIGN ERROR CONVERGED
C
      IF (XERR) 320,400,330                ;BRANCH ON ERROR SIGN
  320 HNEG = HPRE
      ENEG = XERR
      GOTO 310                             ;ITERATE
  330 HPOS = HPRE
      EPOS = XERR
      GOTO 310                             ;ITERATE
C
C     PART 3A: PRESPAN IS SLACK
  390 HPRE = 0.
      XPRE = XMIN
      SB = XMIN
C
C *** PART 4: WRITE CASE SOLUTION --------------------------------------
C
  400 HDIF = HDES - HPRE                   ;HOLDING POWER
      SBPL = -SB                           ;PRELOAD BOTTOMED SCOPE
      IF (ABS(SBPL/DEPTH) .LT. .001)       ; ELSE VERTICAL
     +    SBPL = U(LSGS)                   ;  LOAD ON ANCHOR
      WRITE (AUX)
     +  SCOPE, XMIN, SBPL, SBDL,
     +   XPRE, XDES, HPRE, HDES, HDIF      ;STORE CASE
C
      WRITE (CON, 1001)
     +  SCOPE, XMIN, SBPL, SBDL,
     +   XPRE, XDES, HPRE, HDES, HDIF      ;DISPLAY CASE
C
      IF (TRYD .LT. TRYV .AND.
     +    TRYP .LT. TRYV) GOTO 410         ;NORMAL CONERGENCE
```

```
C
      WRITE (PTR, 1001)
     +        SCOPE, XMIN, SBPL, SBDL,
     +  XPRE, XDES, HPRE, HDES, HDIF       ;PRINT CASE
      WRITE (CON, 1004)
     + TRYD, FERR, TRYP, XERR              ;SLOW CONVERGENCE DISPLAY
C
  410 SCOPE = SCOPE + DSCOP                ;NEXT CASE
      DSCOP = DSC1                         ;USE REGULAR STEP SIZE
      IF (SCOPE .LT. SMAX) GOTO 200
  420 LEN(1,TYP) = LEN1                    ;RECOVER ORIGINAL LENGTH
      RETURN
C
C *** ERROR RECOVERY: WEAK SEGMENT FAILS IN SLACK LEG
C
  500 WRITE (CON, 1005)                    ;WEAK SEGMENT WARNING
     + NSF, TYP, FMIN, SAFETY, SCOPE
      GOTO 410
C
 1000 FORMAT (´1´,
     + 29X´H vs S for Leg Type´, I3//
     + 14X´First Scope   Last Scope  ´,
     + ´ Scope Step   Max. Load´/
     + 13X1P4E13.5//
     + ´     Top  - Force/Length on Bottom -´,
     + ´     Pre-    Design    Pre-´,
     + ´     Design  Holding´/
     + ´    Scope    Slack  Preload   Design´,
     + ´     Span     Span     Load´,
     + ´      Load     Power´)             ;DISPLAY HEADER
C
 1001 FORMAT (F8.0, 8F9.0)                 ;DISPLAY LINE
C
 1002 FORMAT (´ DESLOAD@210:´/
     + ´    TRY         HDES         HPOS         HNEG´,
     + ´         HERR         FERR         EPOS´,
     + ´         ENEG´/
     + I6, 3F12.1, 4F12.6)                 ;SLOW DES.LOAD CONVERGENCE
C
 1003 FORMAT (´ PRELOAD@310:´/
     + ´    TRY         HPRE         HPOS         HNEG´,
     + ´         HERR         FERR         EPOS´,
     + ´         ENEG´/
     + I6, 3F12.1, 4F12.6)                 ;SLOW PRELOAD CONVERGENCE
C
 1004 FORMAT (´ PRELOD@410:´/
     + ´ DESIGN TRIES   ERROR    ´,
     + ´PRELOAD TRIES   ERROR´/
     + 6XI6, 1PE14.6, 6XI6, E14.6)         ,SLOWLY CONVERGED SOLUTION
C
 1005 FORMAT (´ PRELOAD@500: Segment´, I3,
     + ´ fails in slack leg type´, I3/
     + F10.2, ´Safety Factor less than´,
     + F10.2, ´ when Segment 1 length =´,
     + F10.1)                              ;WEAK SEGMENT FAILURE
```

```
C
      END
```

```
      REAL FUNCTION STAN (T, W, E, Y, H)     ;D.B.DILLON  EG&G
C
C *** SINGLE ELASTIC SEGMENT TANGENT TO BOTTOM AT TENSION T
C     RETURN H=-1 AND STAN=T/W IF W*Y > T: WEAK CABLE
C            H=0 AND STAN=20*Y IF W<=0: NEUTRALY BUOYANT CABLE
C
C *** GIVEN:
      REAL T                                 ;TOP END TENSION
      REAL W                                 ;WEIGHT DENSITY
      REAL E                                 ;ELASTIC FACTOR, EA
      REAL Y                                 ;DEPTH
C *** RETURN:
C  *  REAL STAN                              ;MATERIAL LENGTH
      REAL H                                 ;HORIZONTAL LOAD
      REAL TQ                                ;LOCAL: SQUARE OF TENSION
      REAL YE                                ;SCOPE ESCESS
      INTEGER*2 CON, PTR, MSG, AUX, RIG      ;/UNITS/
C
      COMMON /UNITS/ CON, PTR, MSG, AUX, RIG;UNITS
C
C *** CHECK FOR NEUTRALLY BUOYANT SEGMENT
C
      H = 0.                                 ;SET N.B. ARGUMENTS
      STAN = 20.*Y
      IF (W .LE. 0.) RETURN
C
C *** CHECK FOR INVALID CASE: VERTICAL CABLE WEIGHT EXCEEDS TENSION
C
      TQ = T*T                               ,DUMMY: TENSION SQUARED
      YE = T/W - Y                           ;INELASTIC EXCESS SCOPE
      IF (E .GT. 0.) YE = YE + TQ*(E+E)/W    ;ELASTIC SCOPE EXCESS
      IF (YE .GT. 0.) GOTO 100               ;REACH BOTTOM?
      WRITE (CON, 1000) YE                   ;NO
      H=-1.                                  ;INVALID CALL
      RETURN
C
C *** COMPUTE CATENARY LOAD AND SCOPE
C
  100 H = T - Y*W                            ;INELASTIC LOAD
      IF (E .GT. 0.) H =                     ;ADJUST TO
     +   E*(SQRT(1 + (H+H)/E + TQ/E/E) - 1)  ; ELASTIC LOAD
      STAN = SQRT(TQ - H*H)/W                ;(W*S)^2 = T^2 - H^2
      RETURN
C
 1000 FORMAT (' STAN@90: Cable weight',
     + '"equals strength at depth excess of',
     +  f11.2)                               ;WEAK CABLE
C
      END
```

```
C$DEBUG
$STRICT                                             ;MODULE LOOK.FOR
C                                                   ;SUBROUTINE SETLEG
C                                                   ;SUBROUTINE LOOKXH
C                                                   ;SUBROUTINE LOADXH
C                                                   ;LOGICAL*2 FUNCTION PRANGE
C                                                   ;LOGICAL*2 FUNCTION XTREME
C                                                   ,LOGICAL*2 FUNCTION FINDHS
C                                                   ;LOGICAL*2 FUNCTION OPTPRE
C                                                   ;LOGICAL*2 FUNCTION TOPLEN
C
C     --------------------------------------------------------------
C
      SUBROUTINE SETLEG (STORM)                     ;D.B.DILLON  EG&G  MAY-85
C
C ***  GIVEN: STORM=.FALSE. ENABLE ALL LEGS IN /TORX/
C                   .TRUE.  DISABLE LEGS WITHIN 86 DEGREES OF HOLDING POWER
      LOGICAL*2 STORM                               ;ARGUMENT: ENABLING OPTION
      INTEGER*2 ANC                                 ;LOCALS: ANCHOR LEG INDEX
      REAL PHI                                      ,OPER. HOLDING POWER DIRECTION
      REAL PSI                                      ;OPER. LEG THRUST DIRECTION
      LOGICAL*2 ACTIV                               ;/TORX/
      REAL THETA, XRIG, YRIG, YAW,
     +     NETMCW, NETFX, NETFY, NETSF,
     +     XSPN, LOAD, SAFAC,
     +     TORQ, LEGX, LEGY
      INTEGER*2 ANCS, LTYP                          ,/ANCHOR/
      REAL DEPTH, OFFSET, SAFETY,
     +    ADIR, ATOP, ARAD, APRE, ANCX, ANCY
      COMMON /ANCHOR/ ANCS, LTYP(12),
     +    DEPTH, OFFSET, SAFETY,
     +    ADIR(12), ATOP(12), ARAD(12),
     +    APRE(12), ANCX(12), ANCY(12)              ;ANCHOR PARAMETERS
      COMMON /TORX/
     +    THETA, XRIG, YRIG, YAW,                   ;RIG DISPLACEMENT
     +    NETMCW, NETFX, NETFY, NETSF,              ;MOORING CAPACITY
     +    XSPN(12), LOAD(12), SAFAC(12),            ,LEG LOADING LIST
     +    TORQ(12), LEGX(12), LEGY(12),             ;MOMENT & FORCE LIST
     +    ACTIV(12)                                 ;ACTIVE LEG FLAGS
C
C *** SELECT OPERATIONAL OR SURVIVAL OPTION
C
      IF (STORM) GOTO 200                           ;SURVIVAL STORM
C
C *** OPERATIONAL STORM: ENABLE ALL LEGS
C
      DO 100 ANC=1,ANCS                             ;SCAN ALL LEGS
      ACTIV(ANC) = .TRUE.
  100 CONTINUE
      RETURN
C
C *** SURVIVAL STORM: DISABLE LEGS WITHIN 86 DEG. OF OPERATIONAL HOLDING POWER
C
  200 PHI = ATAN2(-NETFX,-NETFY)                    ;DIRECTION OF HOLDING POWER
      DO 210 ANC=1,ANCS                             ;SCAN ALL LEGS
```

```
      PSI = ATAN2(LEGX(ANC),LEGY(ANC))      ;DIRECTION OF LEG FORCE
      IF (ABS(PSI-PHI) .LT. 1.5) THEN       ;1.5 RAD. <=> 86 DEG.
         ACTIV(ANC) = .FALSE.               ;BACK-TENSION DISABLED FOR SURVIVAL
      ELSE
         ACTIV(ANC) = .TRUE.                ;NOT SLACKED FOR SURVIVAL
      ENDIF
  210 CONTINUE                              ;NEXT LEG
      RETURN
C
      END
```

```
      SUBROUTINE LOOKXH                               ;D.B.DILLON  EG&G  MAY-85
     +     (LEG, SPAN, LOAD, SAFAC)
C
C *** LOOKUP LOAD AND SAFETY FACTOR VS SPAN FOR LEG
C
      INTEGER*2 LEG                                   ;ARGUMENTS: LEG INDEX
      REAL SPAN                                       ;LEG OFFSET
      REAL LOAD                                       ;LEG LOAD
      REAL SAFAC                                      ;LEG SAFETY FACTOR
      INTEGER*2 TYP                                   ;LOCALS: MASTER TYPE FOR LEG
      INTEGER*2 I, J                                  ;LINE INDEXES
      REAL F                                          ;INTERPOLATION FRACION
      INTEGER*2   MTYPS, MLEG, MTYP                   ;/LEGMAP/
      REAL X, H, S                                    ,/HVSX/
C
      COMMON /HVSX/ X(80,12),                         ;SPAN TABLE
     +              H(80,12),                         ;LOADS
     +              S(80,12)                          ;SAFETY TABLES
      COMMON /LEGMAP/
     +    MTYPS,                                      ;NO. MASTER TYPES
     +    MLEG(12),                                   ;MASTER LEG FOR TYPE I
     +    MTYP(12)                                    ;MASTER TYPE FOR LEG I
C
      TYP = MTYP(LEG)                                 ;GET MASTER TYPE FOR LEG
      DO 100 I=2,80                                   ;SCAN TABLE
      IF (X(I,TYP) .GE. SPAN) GOTO 200                ,READY TO INTERPOLATE?
      IF (X(I,TYP) .GE. 0.) J = I                     ;NO - PAST END OF RECORDS?
  100 CONTINUE
      I = J                                           ;NOT FOUND: EXTRAPOLATE
C                                                     ; FROM LAST TWO RECORDS
  200 FR = (X(I,TYP) - SPAN) /                        ;INTERPOLATING FRACTION
     +     (X(I,TYP) - X(I-1,TYP))
      LOAD  = H(I,TYP) - FR *
     +        (H(I,TYP) - H(I-1,TYP))                 ;INTERPOLATE FOR LOAD
      SAFAC = S(I,TYP) - FR *
     +        (S(I,TYP) - S(I-1,TYP))                 ;INTERPOLATE FOR SAFETY FACTOR
C
      RETURN
C
      END
```

```
      SUBROUTINE LOADXH                         ;D.B.DILLON  EG&G  MAY-85
C
C *** FILL X VSH H ARRAYS IN /XVSH/ FROM DISK FILES
C
C
      LOGICAL*2 MAKFIL                          ,FUNCTION
      EXTERNAL  MAKFIL
      INTEGER*2 ANC                             ;LOCALS: ANCHOR NO.
      INTEGER*2 MT                              ;MASTER TYPE INDEX
      INTEGER*2 I, J                            ,RECORD INDEXES
      CHARACTER XHFIL(6)                        ;XVSH FILE NAME SUFFIX
      REAL SB, ST                               ;DUMMY SCOPE FIELDS
      INTEGER*2   MTYPS, MLEG, MTYP             ;/LEGMAP/
      REAL X, H, S                              ;/XVSH/
      INTEGER*2 ANCS, LTYP                      ;/ANCHOR/
      REAL DEPTH, OFFSET, SAFETY,
     +    ADIR, ATOP, ARAD, APRE, ANCX, ANCY
      INTEGER*2 CON, PTR, MSG, AUX, RIG         ,/UNITS/
C
      COMMON /UNITS/ CON, PTR, MSG, AUX, RIG;LOGICAL UNITS
      COMMON /ANCHOR/ ANCS, LTYP(12),
     +    DEPTH, OFFSET, SAFETY,
     +    ADIR(12), ATOP(12), ARAD(12),
     +    APRE(12), ANCX(12), ANCY(12)          ,ANCHOR PARAMETERS
      COMMON /HVSX/ X(80,12),                   ;SPAN TABLE
     +              H(80,12),                   ;LOADS
     +              S(80,12)                    ;SAFETY TABLES
      COMMON /LEGMAP/
     +    MTYPS,                                ;NO. MASTER TYPES
     +    MLEG(12),                             ;MASTER LEG FOR TYPE I
     +    MTYP(12)                              ;MASTER TYPE FOR LEG I
C
      DATA XHFIL /'X', 'H', '.', 'L', 2*'0'/;RIGNAMXH.LNN
C
C *** LOOP OVER ANCHOR LEGS
C
      DO 400 MT=1,MTYPS                         ;MASTER TYPE LOOP
      ANC = MLEG(MT)                            ;MASTER LEG NO.
      I = 1                                     ;FIRST RECORD
      IF (MAKFIL (XHFIL(1), ANC, AUX))
     +    GOTO 300                              ,OPEN XVSH FILE
C
C *** READ AND DISPLAY AN XVSH RECORD
C
  100 READ (AUX, END=200, ERR=210)
     + H(I,MT), X(I,MT), SB, ST, S(I,MT)        ;IGNORE REMAINDER OF RECORD
      I = I + 1                                 ;COUNT RECORDS
      IF (I .LT. 81) GOTO 100                   ;NEXT RECORD
C
C *** RECOVERY
C
      WRITE (CON, 1001) ANC                     ;ARRAY FULL
C
  200 IF (I .EQ. 1) GOTO 220                    ;NON-EXISTENT FILE
      GOTO 300
```

```
C
  210 IF (I .EQ. 1) GOTO 220                     ;NON-EXISTENT FILE
      WRITE (CON, 1003) I                        ;DAMAGED FILE
      GOTO 230
C
  220 WRITE (CON, 1004)                          ;NO SUCH FILE
  230 CALL DELAY (1)
C
C *** PAD UNFILLED SPAN TABLE WITH -1'S
C
  300 CLOSE (AUX)                                ;CLOSE X VS H FILE
      IF (I .GE. 80) GOTO 400                    ;ARRAY FULL
      J = I                                      ;1ST UNFILLED ELEMENT
      DO 310 I=J,80
      X(I,MT) = -1.                              ;MARK UNFILLED ELEMENTS
  310 CONTINUE
C
  400 CONTINUE                                   ;NEXT LEG
      RETURN                                     ;TASK COMPLETE
C
 1001 FORMAT (//' LOADXH@100:'/
     + ' /HVSX/ tables overflow before end',
     + ' of X vs H file for leg', I3/)           ;TABLE OVERFLOW
C
 1003 FORMAT (//' LOADXH@210:'/
     + ' Read error before record', I3/)         ;DAMAGED FILE
C
 1004 FORMAT (//' LOADXH@220:'/
     + ' X vs H file is empty'/)                 ;NEW FILE
C
      END
```

```
      LOGICAL*2 FUNCTION PRANGE                       ;1 FEB 86  D.B.DILLON  EG&G
     +          (PMIN, PMAX, SOB)
C
C *** GIVEN EQUILIBRIUM PRELOAD RATIOS IN APRE OF /ANCHOR/
C           MINIMUM SCOPE ON BOTTOM AT DESIGN LOAD
C     RETURN MINIMUM AND MAXIMUM AVERAGE PRELOADS CONSISTENT WITH H VS S
C           FILES IN ALL LEGS
C
      REAL PMIN                                       ;ARGUMENT: MIN. AVG. PRELOAD
      REAL PMAX                                       ;MAX. AVG. PRELOAD
      REAL SOB                                        ;SCOPE ON BOTTOM AT DESIGN LOAD
      LOGICAL*2 XTREME                                ;FUNCTIONS
      EXTERNAL XTREME
      INTEGER*2 A                                     ;LOCAL: ANCHOR INDEX
      INTEGER*2 T                                     ;LEG TYPE FOR ANCHOR
      REAL PN(12), PX(12)                             ;TRIAL MIN. & MAX
      REAL HSN, HSX                                   ;/HSXTRM/
      INTEGER*2 TYPS, SEGS, MAT                       ;/CABLES/
      REAL DIA, BRK, LEN, WGT, EA, BNCY
      INTEGER*2 ANCS, LTYP                            ;/ANCHOR/
      REAL DEPTH, OFFSET, SAFETY,
     +    ADIR, ATOP, ARAD, APRE, ANCX, ANCY
      INTEGER*2 CON, PTR, MSG, AUX, RIG               ;/UNITS/
C
      COMMON /UNITS/ CON, PTR, MSG, AUX, RIG;LOGICAL UNITS
      COMMON /ANCHOR/ ANCS, LTYP(12),
     +    DEPTH, OFFSET, SAFETY,
     +    ADIR(12), ATOP(12), ARAD(12),
     +    APRE(12), ANCX(12), ANCY(12)                ;ANCHOR PARAMETERS
      COMMON /CABLES/ TYPS, SEGS(12),
     +    MAT(5,12), DIA(5,12),
     +    BRK(5,12), LEN(5,12), WGT(5,12),
     +    EA(5,12), BNCY(5,12)                        ;LEG PARAMETERS
      COMMON /HSXTRM/ HSN(9,12), HSX(9,12)            ;H VS S TABLE EXTREMES BY TYPE
C
C *** NOW SCAN ANCHORS BY LEG TYPE AND WEIGHT BY EQUILIBRIUM RATIO
C
      PRANGE = XTREME (SOB)                           ;FILL /HSXTRM/
      IF (PRANGE) RETURN                              ;COULDN'T
      PMIN =-1.E26                                    ;SET EXTREMES
      PMAX = 1.E26                                    ; ON PRELOAD BY ANCHOR
      DO 100 A=1,ANCS                                 ;SCAN AROUND RIG
      T = LTYP(A)                                     ;SIMPLE VARIABLE
      PN(A) = HSN(7,T) / APRE(A)                      ;ADJUST ACTUAL PRELOADS
      PX(A) = HSX(7,T) / APRE(A)                      ; TO EQUIVALENT AVERAGES
      PMIN = AMAX1 (PMIN, PN(A))                      ;OVERLAPPING RANGE IS
      PMAX = AMIN1 (PMAX, PX(A))                      ;LARGEST MIN. THRU
  100 CONTINUE                                        ;  LEAST MAX.
C
      PRANGE = PMAX .LT. PMIN                         ;ERROR: INVERTED RANGE
      IF (PRANGE) THEN
           WRITE (CON, 1000)                          ;WARNING: INVERTED RANGE
           WRITE (CON, 1001) PRANGE,                  ;DISPLAY
     +     (PN(A), PX(A), A, A=1,ANCS),               ;  PRANGE
     +        PMIN,  PMAX                             ;    ARGUMENTS
```

```
          PAUSE                                      ;PAUSE TO READ
      ENDIF
      RETURN
C
 1000 FORMAT (/5X'Inverted Preload Range'/)  ;WARNING
C
 1001 FORMAT (/' PRANGE = ', L1/             ;PRANGE
     + '   PMIN          PMAX          LEG'/ ;  ARGUMENT
     + (2E14.7, I4))                         ;    DISPLAY
C
      END
```

```
      LOGICAL*2 FUNCTION XTREME (SOB)       ;1 FEB 86 D.B.DILLON  EG&G
C
C *** FIND MINIMUM AND MAXIMUM VALUE IN THE COLUMNS OF EACH H VS S FILE
C     FOR WHICH THE SCOPE ON BOTTOM AT DESIGN LOAD EXCEEDS SOB
C     GIVEN: SOB = SCOPE ON BOTTOM AT DESIGN LOAD, WITH - SIGN
C     NOTE: H VS S RESERVES + VALUES FOR LOAD ON ANCHOR, - VALUES FOR SOB
C
      LOGICAL*2 MAKFIL                      ;FUNCTION
      EXTERNAL  MAKFIL
      REAL SOB                              ;ARGUMENT: SCOPE ON BOTTOM
      INTEGER*2 RX                          ;LOCAL: RECORD INDEX
      INTEGER*2 COL                         ;ARGUMENTS: COLUMN NO.
      INTEGER*2 TYP                         ;LEG TYPE
      REAL CASE(9)                          ;H VS S RECORD
      CHARACTER HSFIL(6)                    ;H VS S SUFFIX HS.Lnn
      REAL HSN, HSX                         ;/HSXTRM/
      INTEGER*2 TYPS, SEGS, MAT             ;/CABLES/
      REAL DIA, BRK, LEN, WGT, EA, BNCY
      INTEGER*2 CON, PTR, MSG, AUX, RIG     ;/UNITS/
C
      COMMON /UNITS/ CON, PTR, MSG, AUX, RIG;LOGICAL UNITS
      COMMON /CABLES/ TYPS, SEGS(12),
     +    MAT(5,12), DIA(5,12),
     +    BRK(5,12), LEN(5,12), WGT(5,12),
     +    EA(5,12), BNCY(5,12)              ;LEG PARAMETERS
      COMMON /HSXTRM/ HSN(9,12), HSX(9,12)  ;H VS S TABLE EXTREMES BY TYPE
C
      DATA HSFIL /'H', 'S', '.', 'L', 2*'0'/;RIGNAMHS.LNN
C
C *** SCAN ALL LEG TYPES
C
      XTREME = .FALSE.
      DO 500 TYP=1,TYPS
      IF (MAKFIL(HSFIL(1), TYP, AUX))       ;OPEN drv:rignamHS.Ltyp
     +    GOTO 305                          ;ERROR ON OPEN
      RX = 0                                ;START RECORD COUNTER
      DO 90 COL=1,9
      HSN(COL,TYP) = 1.E26                  ;RESET MIN. FOR TYP
      HSX(COL,TYP) =-1.E26
   90 CONTINUE
C
C *** SCAN LINE BY LINE
C
  100 READ (AUX, END=200, ERR=300) CASE     ;GET H VS S LINE
      RX = RX + 1                           ;COUNT RECORDS
      IF (CASE(4) .GT. SOB) GOTO 100        ,REJECT VERTICAL LOAD ON ANCHOR
      DO 110 COL=1,9
      HSN(COL,TYP) =
     +    AMIN1(HSN(COL,TYP), CASE(COL))    ,NEW COLUMN MINIMUM
      HSX(COL,TYP) =
     +    AMAX1(HSX(COL,TYP), CASE(COL))    ;OR MAXIMUM
  110 CONTINUE
      GOTO 100                              ,NEXT H VS S LINE
C
  200 IF (RX .EQ. 0) GOTO 310               ;NON-EXISTENT FILE
```

```
      GOTO 400                                       ;DONE
C
C *** READ ERROR RECOVERY ------------------
C
  300 IF (RX .EQ. 0) GOTO 310                        ,NON-EXISTENT FILE
      WRITE (CON, 1004) RX                           ;DAMAGED FILE
      GOTO 320
C
  305 WRITE (CON, 1006) HSFIL, TYP                   ;OPEN ERROR
      GOTO 320
C
  310 WRITE (CON, 1005) AUX                          ,NO SUCH FILE
  320 XTREME = .TRUE.                                ;SET ERROR FLAG
C
C *** NEXT LEG TYPE -----------------------
C
  400 CLOSE (AUX)
  500 CONTINUE
C
      IF (XTREME) CALL DELAY(1)
      RETURN
C
C -------------------------------------------------------------------
C
 1004 FORMAT (//' XTREME@300:',
     + ' Read error after record', I3, '.')  ;DAMAGED FILE
C
 1005 FORMAT (//' XTREME@310:',
     + ' Empty AUX file on unit', I2)        ;NEW FILE
C
 1006 FORMAT (//' XTREME@305:',
     + ' Unable to open ', 6A1,
     + ' for leg type', I3)                  ;H VS S FILE OPEN ERROR
C
      END
```

```
      LOGICAL*2 FUNCTION FINDHS ()          ;D.B.DILLON  EG&G  MAY-85
C
C ***  GIVEN: ROOT NAME IN /JOB/
C     RETURN: FINDHS=.TRUE. IF ANY H VS S FILES ARE MISSING ELSE .FALSE.
C
      INTEGER*2 I                           ;LOCAL: BYTE INDEX
      INTEGER*2 TYP                         ;LEG TYPE INDEX
      LOGICAL*4 OLD                         ;OLD/NEW FILE FLAG
      CHARACTER AUXNAM*72                   ;A72 OVERLAY FOR 72A1 RIGNAM
      CHARACTER SUBNAM*6, SBNM(6)           ;H VS S SUFFIX
      INTEGER*2 TYPS, SEGS, MAT             ;/CABLES/
      REAL DIA, BRK, LEN, WGT, EA, BNCY
      CHARACTER RIGNAM, JOBNAM              ;/JOB/
      INTEGER*2 RNL, JNL                    ;/NAMLEN/
C
      COMMON /NAMLEN/ RNL, JNL              ;NAME LENGTHS
      COMMON /JOB/ RIGNAM(72), JOBNAM(72)   ;PROBLEM ID
      COMMON /CABLES/ TYPS, SEGS(12),
     +    MAT(5,12), DIA(5,12),
     +    BRK(5,12), LEN(5,12), WGT(5,12),
     +    EA(5,12), BNCY(5,12)              ;LEG SEGMENT PARAMETERS
C
      EQUIVALENCE (AUXNAM, RIGNAM),         ;OVERLAY STRING NAMES
     +            (SUBNAM, SBNM)
C
C *** FIND THE H VS S FILE FOR EACH LEG TYPE
C
      DO 200 TYP=1,TYPS                     ;SCAN LEGS BY TYPE
C
      WRITE (SUBNAM, 1000) TYP              ;ENCODE TYPE NUMBER IN EXTENSION
      DO 100 I=1,6                          ;COPY SUFFIX
      RIGNAM(RNL+I) = SBNM(I)               ; INTO RIG NAME
  100 CONTINUE                              ;  BYTE BY BYTE
C
      INQUIRE (FILE=AUXNAM, EXIST=OLD)      ;LOCATE H VS S FILE
      FINDHS = .NOT. OLD                    ;.TRUE. IF NEW
      IF (FINDHS) RETURN                    ; AND QUIT TO CREATE A NEW SET
C
  200 CONTINUE
      RETURN
C
 1000 FORMAT ('HS.L', I2.2)                 ;H VS S SUFFIX
C
      END
```

```
      LOGICAL*2 FUNCTION OPTPRE (DESOB, TYP);31-JAN-86 D.B.DILLON  EG&G
C
C *** LOOK UP THE OPTIMUM PRELOAD IN AN H VS S FILE BASED ON MINIMUM BOTTOM
C     SCOPE AT DESIGN LOAD.
C       DESOB: GIVEN SCOPE ON BOTTOM AT DESIGN LOAD
C         TYP: LEG TYPE INDEX TO BE OPTIMIZED
C      RETURN: INTERPOLATED OPTIMUM H VS S ENTRY IN /HSTABL/
C
      INTEGER*2 TYP                            ;ARGUMENTS: LEG TYPE
      REAL DESOB                               ;SCOPE ON BOTTOM, DESIGN LOAD
      LOGICAL*2 MAKFIL                         ;FUNCTION
      EXTERNAL  MAKFIL
      CHARACTER HSFIL(6)                       ;LOCAL
      REAL HVSS(9,2)                           ;INTERPOLATED ROWS
      REAL MU                                  ;INTERPOLATING FRACTION
      INTEGER*2 I, J, K, RX                    ;RECORD POINTERS
      REAL CASE                                ;/HSTABL/
      INTEGER*2 CON, PTR, MSG, AUX, RIG        ;/UNITS/
C
      COMMON /UNITS/ CON, PTR, MSG, AUX, RIG;LOGICAL UNITS
      COMMON /HSTABL/ CASE(9)                  ;H VS S FILE INTERPOLATION
C
      DATA HSFIL /'H', 'S', '.', 'L', 2*'0'/;RIGNAMHS.LNN
C
C ------------------------------------------------------------------
C *** VALIDATE ARGUMENTS AND INITIATE POINTERS
C
      IF (MAKFIL(HSFIL(1), TYP, AUX))          ;OPEN drv:rignamHS.Ltyp
     +    CALL PRELOD(TYP)                     ;COMPUTE H VS S TABLE
      RX = 0                                   ;RECORDS READ
      J = 1                                    ;START LINE TOGGLES
      HVSS(4,2) = 1.                           ;FORCE READ OF AT LEAST 2 ROWS
C
C *** SCAN FILE: SKIP OVER VERTICAL LOAD ROWS. DO NOT INTERPOLATE BETWEEN
C                A VERTICAL LOAD AND SOB ENTRIES; EXTRAPOLATE BACK INSTEAD
C
  100 READ (AUX, END=200, ERR=210)
     +        (HVSS(K,J), K=1,9)               ;GET CASE
      RX = RX + 1                              ;COUNT RECORDS
      I = J                                    ;TOGGLE LINE INDICES
      J = 3 - J                                ;I=NEWEST    J=PRIOR
      IF (HVSS(4,I) .GT. DESOB) GOTO 100       ;SOB=0: LINE NOT TANGENT
      IF (HVSS(4,J) .GT. 0.) GOTO 100          ;BOTH NEW AND OLD MUST HAVE SOB
C
C *** INTERPOLATE/EXTRAPOLATE FOR OPTIMUM PRELOAD
C
      MU = (DESOB - HVSS(4,J)) /
     +     (HVSS(4,I) - HVSS(4,J))             ,INTERPOLATING FRACTION
      DO 110 K=1,9                             ;INTERPOLATE AND
      CASE(K) = HVSS(K,J) + MU *               ;COPY RECORD INTO /HSTABL/
     +        (HVSS(K,I) - HVSS(K,J))
  110 CONTINUE
      OPTPRE = .FALSE.                         ;CONCLUDE GOOD LOOKUP
      GOTO 300                                 ;CLOSE FILE
C
```

```
C *** READ ERROR RECOVERY ------------------
C
  200 IF (RX .EQ. 0) GOTO 220                    ;NON-EXISTENT FILE
      WRITE (CON, 1003) RX                       ;END OF FILE
      GOTO 260                                   ;DISPLAY WARNING
C
  210 IF (RX .EQ. 0) GOTO 220                    ;NON-EXISTENT FILE
      WRITE (CON, 1004) RX                       ;DAMAGED FILE
      GOTO 260
C
  220 WRITE (CON, 1005) AUX                      ;NO SUCH FILE
C
  260 WRITE (CON, 1007) DESOB, TYP               ;NOT IN FILE
      PAUSE
      OPTPRE = .TRUE.                            ;SET ERROR FLAG
C
  300 CLOSE (AUX)
      RETURN
C
C ---------------------------------------------------------------------
C
C
 1003 FORMAT (//' OPTPRE@200:',
     + ' End of file after record', I3, '.');FINISHED
C
 1004 FORMAT (//' OPTPRE@210:',
     + ' Read error after record', I3, '.') ;DAMAGED FILE
C
 1005 FORMAT (//' OPTPRE@220:',
     + ' Empty AUX file on unit', I2)        ;NEW FILE
C
 1007 FORMAT (//' OPTPRE@260:', F10.2,
     + ' Scope on bottom not in H vs S',
     + ' file for leg type', I3)             ;NOT FOUND
C
      END
```

```
      LOGICAL*2 FUNCTION TOPLEN (PRELD, TYP);31-JAN-86 D.B.DILLON  EG&G
C
C *** LOOK UP THE TOP SEGMENT LENGTH CORRESPONDING TO A PRELOAD IN HVSS FILE
C     CONDITION: PRELOAD MUST PERMIT SCOPE ON BOTTOM AT DESIGN LOAD
c                TOPLEN RETURNS .T. IF PRELOAD REQUIRES VERTICAL ANCHOR PULL
C                AT DESIGN LOAD OR PRELOAD NOT FOUND IN H VS S FILE
C       PRELD: GIVEN PRELOAD FOR INTERPOLATION
C         TYP: LEG TYPE INDEX TO BE OPTIMIZED
C      RETURN: INTERPOLATED H VS S ENTRY IN /HSTABL/
C
      INTEGER*2 TYP                            ;ARGUMENTS: LEG TYPE
      REAL PRELD                               ;PRELOAD FOR INTERPOLATION
      LOGICAL*2 MAKFIL                         ;FUNCTION
      EXTERNAL  MAKFIL
      CHARACTER HSFIL(6)                       ;LOCAL
      REAL HVSS(9,2)                           ;INTERPOLATED ROWS
      REAL MU                                  ;INTERPOLATING FRACTION
      REAL DS                                  ;TOP LENGTH INTERPOLATION
      INTEGER*2 I, J, K, RX                    ,RECORD POINTERS
      REAL CASE                                ;/HSTABL/
      INTEGER*2 CON, PTR, MSG, AUX, RIG        ;/UNITS/
C
      COMMON /UNITS/ CON, PTR, MSG, AUX, RIG;LOGICAL UNITS
      COMMON /HSTABL/ CASE(9)                  ;H VS S FILE INTERPOLATION
C
      DATA HSFIL /'H', 'S', '.', 'L', 2*'0'/;RIGNAMHS.LNN
C
C ---------------------------------------------------------------------
C *** VALIDATE ARGUMENTS AND INITIATE POINTERS
C
      IF (MAKFIL(HSFIL(1), TYP, AUX))          ;OPEN drv:rignamHS.Ltyp
     +    CALL PRELOD(TYP)                     ;COMPUTE H VS S TABLE
      RX = 0                                   ;RECORDS READ
      HVSS(4,2) = 1.                           ;FORCES READ OF AT LEAST 2 ROWS
      J = 1                                    ;START LINE TOGGLES
C
C *** SCAN FILE: SKIP OVER VERTICAL LOAD ROWS. DO NOT INTERPOLATE BETWEEN
C                A VERTICAL LOAD AND SOB ENTRIES; EXTRAPOLATE BACK INSTEAD
C
  100 READ (AUX, END=200, ERR=210)
     +         (HVSS(K,J), K=1,9)              ;GET CASE
      RX = RX + 1                              ,COUNT RECORDS
      I = J                                    ;TOGGLE LINE INDICES
      J = 3 - J                                ;I=NEWEST    J=PRIOR
      IF (HVSS(4,J) .GT. 0.) GOTO 100          ;BOTH NEW AND OLD MUST HAVE SOB
C
C *** INTERPOLATE/EXTRAPOLATE FOR OPTIMUM PRELOAD
C
      MU = (PRELD - HVSS(7,J)) /
     +     (HVSS(7,I) - HVSS(7,J))             ;INTERPOLATING FRACTION
      DS = MU * (HVSS(1,I) - HVSS(1,J))        ;TOP LENGTH INTERPOLATION
      IF (DS .GT. 0.) GOTO 100                 ;ENTRY IS FARTHER DOWN TABLE
      CASE(1) = HVSS(1,J) + DS                 ;UPDATE TOP SCOPE
      DO 110 K=2,9                             ;INTERPOLATE AND
      CASE(K) = HVSS(K,J) + MU *               ;COPY RECORD INTO /HSTABL/
```

```
   +         (HVSS(K,I) - HVSS(K,J))
 110 CONTINUE
     TOPLEN = .FALSE.                         ;CONCLUDE GOOD LOOKUP
     GOTO 300                                 ;CLOSE FILE
C
C *** READ ERROR RECOVERY -----------------
C
 200 IF (RX .EQ. 0) GOTO 220                  ;NON-EXISTENT FILE
     WRITE (CON, 1003) RX                     ;END OF FILE
     GOTO 260                                 ;DISPLAY WARNING
C
 210 IF (RX .EQ. 0) GOTO 220                  ;NON-EXISTENT FILE
     WRITE (CON, 1004) RX                     ;DAMAGED FILE
     GOTO 260
C
 220 WRITE (CON, 1005) AUX                    ;NO SUCH FILE
C
 260 WRITE (CON, 1007) PRELD, TYP             ;NOT IN FILE
     PAUSE
     TOPLEN = .TRUE.                          ;SET ERROR FLAG
C
 300 CLOSE (AUX)
     RETURN
C
C -------------------------------------------------------------
C
C
1003 FORMAT (//' TOPLEN@200:',
    + ' End of file after record', I3, '.');FINISHED
C
1004 FORMAT (//' TOPLEN@210:',
    + ' Read error after record', I3, '.') ;DAMAGED FILE
C
1005 FORMAT (//' TOPLEN@220:',
    + ' Empty AUX file on unit', I2)       ;NEW FILE
C
1007 FORMAT (//' TOPLEN@260:', F10.0,
    + ' Preload not in H vs S',
    + ' file for leg type', I3)            ;NOT FOUND
C
      END
```

```
C$DEBUG
$STRICT                                             ;MODULE NAME.FOR
C                                                   ;LOGICAL*2 FUNCTION GETDEF
C                                                   ;LOGICAL*2 FUNCTION PUTDEF
C                                                   ;SUBROUTINE PRTDEF
C                                                   ;SUBROUTINE SHOLEG
C                                                   ;LOGICAL*2 FUNCTION MAKFIL
C                                                   ;LOGICAL*2 FUNCTION OPENUF
C                                                   ;LOGICAL*2 FUNCTION MAKDEF
C                                                   ;LOGICAL*2 FUNCTION GETRNG
C                                                   ;LOGICAL*2 FUNCTION MAKROS
C
C     ------------------------------------------------------------
C
      LOGICAL*2 FUNCTION GETDEF()                   ;D.B.DILLON  EG&G  1985
C
C *** LOAD PROBLEM DATA FROM DISK
C
      CHARACTER RIGFIL(6)                           ;LOCALS
      INTEGER*2 I, J, JX, TYP
      LOGICAL*2 MAKDEF                              ;FUNCTION
      EXTERNAL  MAKDEF
      REAL RIGX, RIGY                               ;/SEMI/
      INTEGER*2 ANCS, LTYP                          ;/ANCHOR/
      REAL DEPTH, OFFSET, SAFETY,
     +    ADIR, ATOP, ARAD, APRE, ANCX, ANCY
      INTEGER*2 TYPS, SEGS, MAT                     ;/CABLES/
      REAL DIA, BRK, LEN, WGT, EA, BNCY
      INTEGER*2 CON, PTR, MSG, AUX, RIG             ;/UNITS/
      CHARACTER RIGNAM, JOBNAM                      ;/JOB/
      INTEGER*2 RNL, JNL                            ;/NAMLEN/
C
      COMMON /NAMLEN/ RNL, JNL                      ;NAME LENGTHS
      COMMON /JOB/ RIGNAM(72), JOBNAM(72)           ;PROBLEM ID
      COMMON /UNITS/ CON, PTR, MSG, AUX, RIG,LOGICAL UNITS
      COMMON /CABLES/ TYPS, SEGS(12),
     +    MAT(5,12), DIA(5,12),
     +    BRK(5,12), LEN(5,12), WGT(5,12),
     +    EA(5,12), BNCY(5,12)                      ;LEG SEGMENT PARAMETERS
      COMMON /ANCHOR/ ANCS, LTYP(12),
     +    DEPTH, OFFSET, SAFETY,
     +    ADIR(12), ATOP(12), ARAD(12),
     +    APRE(12), ANCX(12), ANCY(12)              ,ANCHOR PARAMETERS
      COMMON /SEMI/ RIGX(12), RIGY(12)              ;BARGE BOLLARDS
C
      DATA RIGFIL /'D', 'F', '.', 'R', 'I', 'G'/
      DATA TYP /0/
C
C *** OPEN ENVIRONMENT FILE
C
      GETDEF = .FALSE.                              ;CLEAR ERROR FLAG
      IF (MAKDEF (RIGFIL(1), RIG)) GOTO 200 ;OPEN RIG DEFINITION
C
C *** LOAD TASK NAME AND LENGTH
C
```

```
      READ (RIG, 1000, ERR=210, END=210)
     + JOBNAM                                  ;PROBLEM NAME
      DO 80 I=1,72                             ;BACKSCAN FOR LENGTH
      JNL = 73 - I
      IF (JOBNAM(JNL) .NE. ' ') GOTO 90        ;AT 1ST NON-SPACE
   80 CONTINUE
C
C *** LOAD MOORING DEFINITION
C
   90 READ (RIG, 1001, ERR=200, END=220)
     +    DEPTH, OFFSET, SAFETY, ANCS, TYPS,
     +    (SEGS(I), I=1,TYPS)                  ;DIMENSIONS
C
      READ (RIG, 1002, ERR=200, END=220)
     + (RIGX(I), RIGY(I), ANCX(I), ANCY(I),
     +  ADIR(I), ARAD(I), APRE(I), ATOP(I),
     +  LTYP(I), I=1,ANCS)                     ;COORDINATES
C
      DO 100 I=1,TYPS                          ;LOOP OVER LEG STYLES
      JX = SEGS(I)                             ;DUMMY SIMPLE VARIABLE
      READ (RIG, 1003, ERR=200, END=220)
     + (MAT(J,I), DIA(J,I), LEN(J,I),
     +  WGT(J,I), BRK(J,I), BNCY(J,I),
     +   EA(J,I), J=1,JX)                      ;LOAD 1 LEG STYLE
  100 CONTINUE                                 ;NEXT STYLE
C
      CALL PRTDEF (CON)                        ;DISPLAY INPUT
      GOTO 240                                 ;NORMAL END
C
C *** RECOVER FROM DEFINITION FILE ERROR
C
  200 WRITE (CON, 1004) RIGNAM                 ;BAD FORMAT
      GOTO 230                                 ;QUIT
C
  210 WRITE (CON, 1005) RIGNAM                 ;NEW CASE
      GOTO 240
C
  220 WRITE (CON, 1006) RIGNAM                 ;INCOMPLETE DEFINITION
  230 GETDEF = .TRUE.                          ;SET ERROR FLAG
  240 CLOSE (RIG)                              ;CLOSE DEFINITION FILE
      CALL DELAY (1)                           ;TIMED PAUSE
      RETURN                                   ;CASE LOADED OK
C
 1000 FORMAT (72A1)                            ,PROBLEM NAME
C
 1001 FORMAT (3F10.2, 14I3)                    ;DIMENSIONS
C
 1002 FORMAT (2F9.2, 2F10.2, F9.3,
     +          F10.2, F10.0, F9.1, I3)        ;ANCHOR PARAMETERS
C
 1003 FORMAT (I5,  F10.3, F10.2,
     +      F10.4, 2F10.0, E10.4)              ;ELEMENT PARAMETERS
C
 1004 FORMAT (//' Bad Form: ', 72A1/)          ;BAD FORMAT WARNING
C
```

```
 1005 FORMAT (//' New File: ', 72A1/)          ;NEW FILE WARNING
C
 1006 FORMAT (//' Bad File: ', 72A1/)          ;SHORT FILE WARNING
C
      END
```

```
      LOGICAL*2 FUNCTION PUTDEF ()
C
C *** LOAD PROBLEM DATA UNTO DISK
C
      LOGICAL*2 MAKDEF                       ;FUNCTION
      EXTERNAL  MAKDEF
      CHARACTER RIGFIL(6)                    ;LOCAL
      REAL RIGX, RIGY                        ;/SEMI/
      INTEGER*2 ANCS, LTYP                   ;/ANCHOR/
      REAL DEPTH, OFFSET, SAFETY,
     +    ADIR, ATOP, ARAD, APRE, ANCX, ANCY
      INTEGER*2 TYPS, SEGS, MAT              ;/CABLES/
      REAL DIA, BRK, LEN, WGT, EA, BNCY
      INTEGER*2 CON, PTR, MSG, AUX, RIG      ;/UNITS/
      CHARACTER RIGNAM, JOBNAM               ;/JOB/
      INTEGER*2 RNL, JNL                     ;/NAMLEN/
C
      COMMON /NAMLEN/ RNL, JNL               ,NAME LENGTHS
      COMMON /JOB/ RIGNAM(72), JOBNAM(72)    ;PROBLEM ID
      COMMON /UNITS/ CON, PTR, MSG, AUX, RIG;LOGICAL UNITS
      COMMON /CABLES/ TYPS, SEGS(12),
     +    MAT(5,12), DIA(5,12),
     +    BRK(5,12), LEN(5,12), WGT(5,12),
     +    EA(5,12), BNCY(5,12)               ;LEG SEGMENT PARAMETERS
      COMMON /ANCHOR/ ANCS, LTYP(12),
     +    DEPTH, OFFSET, SAFETY,
     +    ADIR(12), ATOP(12), ARAD(12),
     +    APRE(12), ANCX(12), ANCY(12)       ;ANCHOR PARAMETERS
      COMMON /SEMI/ RIGX(12), RIGY(12)       ;BARGE BOLLARDS
C
C *** LOCAL DATA STATEMENTS
      DATA RIGFIL /'D', 'F', '.', 'R', 'I', 'G'/
      DATA TYP /0/
C
C *** OPEN ENVIRONMENT FILE
C
      PUTDEF = .FALSE.                       ;CLEAR ERROR FLAG
      IF (MAKDEF (RIGFIL(1), RIG)) GOTO 200 ;OPEN RIG DEFINITION FILE
C
C *** SAVE MOORING DEFINITION
C
      WRITE (RIG, 1000, ERR=200) JOBNAM,
     + DEPTH, OFFSET, SAFETY, ANCS, TYPS, SEGS,
     + (RIGX(I), RIGY(I), ANCX(I), ANCY(I),
     +  ADIR(I), ARAD(I), APRE(I), ATOP(I),
     +  LTYP(I), I=1,ANCS)                   ;MOORING ENVIRONMENT
C
      DO 110 I=1,TYPS                        ;LOOP OVER LEG STYLES
      JX = SEGS(I)                           ;DUMMY SIMPLE VARIABLE
      WRITE (RIG, 1001, ERR=200)
     + (MAT(J,I), DIA(J,I), LEN(J,I),
     +  WGT(J,I), BRK(J,I), BNCY(J,I),
     +   EA(J,I), J=1,JX)                    ;SAVE 1 LEG STYLE
  110 CONTINUE                               ;NEXT STYLE
C
```

```
  120 ENDFILE RIG                                   ;CLOSE DEFINITION FILE
      CALL PRTDEF (PTR)                             ;DISPLAY DEFINITION
      RETURN                                        ;CASE LOADED
C
C *** RECOVER FROM DEFINITION FILE ERROR
C
  200 WRITE (1, 1002) RIGNAM                        ;BAD FORMAT
      PUTDEF = .TRUE.                               ;SET ERROR FLAG
      GOTO 120                                      ;ABORT
C
 1000 FORMAT (72A1 / 3F10.2, 14I3 /
     + (2F9.2, 2F10.2, F9.3,
     +        F10.2, F10.0, F9.1, I3))              ;ANCHOR PARAMETERS
C
 1001 FORMAT (I5,  F10.3, F10.2,
     +     F10.4, 2F10.0, E10.4)                    ;ELEMENT PARAMETERS
C
 1002 FORMAT (/' Write error on', / 5X72A1)         ;BAD FORMAT WARNING
C
      END
```

```
      SUBROUTINE PRTDEF (ECHO)
C
C *** PRINT PROBLEM DATA FROM MEMORY
C     ECHO=0: NO DISPLAY      1: CONSOLE DISPLAY
C          2: PRINTED DISPLAY 3: BOTH
C
      INTEGER*2 ECHO                           ;ARGUMENT
      INTEGER*2 I, J, JX
      LOGICAL*2 PRT, DSP                       ;FLAGS
      CHARACTER SPACE
      REAL RIGX, RIGY                          ;/SEMI/
      INTEGER*2 ANCS, LTYP                     ;/ANCHOR/
      REAL DEPTH, OFFSET, SAFETY,
     +    ADIR, ATOP, ARAD, APRE, ANCX, ANCY
      INTEGER*2 TYPS, SEGS, MAT                ;/CABLES/
      REAL DIA, BRK, LEN, WGT, EA, BNCY
      INTEGER*2 CON, PTR, MSG, AUX, RIG        ;/UNITS/
      CHARACTER RIGNAM, JOBNAM                 ;/JOB/
      INTEGER*2 RNL, JNL                       ;/NAMLEN/
C
      COMMON /NAMLEN/ RNL, JNL                 ;NAME LENGTHS
      COMMON /JOB/ RIGNAM(72), JOBNAM(72)      ;PROBLEM ID
      COMMON /UNITS/ CON, PTR, MSG, AUX, RIG;LOGICAL UNITS
      COMMON /CABLES/ TYPS, SEGS(12),
     +    MAT(5,12), DIA(5,12),
     +    BRK(5,12), LEN(5,12), WGT(5,12),
     +    EA(5,12), BNCY(5,12)                 ;LEG SEGMENT PARAMETERS
      COMMON /ANCHOR/ ANCS, LTYP(12),
     +    DEPTH, OFFSET, SAFETY,
     +    ADIR(12), ATOP(12), ARAD(12),
     +    APRE(12), ANCX(12), ANCY(12)         ;ANCHOR PARAMETERS
      COMMON /SEMI/ RIGX(12), RIGY(12)         ;BARGE BOLLARDS
C
      DATA SPACE /' '/
C
C *** ECHO OPTIONS
C
      PRT = .FALSE.
      IF (ECHO .GT. 1) PRT = .TRUE.            ;PRINTOUT ENABLED
      DSP = .FALSE.
      IF (ECHO .EQ. 1 .OR. ECHO .EQ. 3)
     +                  DSP = .TRUE.           ;CONSOLE DISPLAY
C
C *** DISPLAY CENTERED ROOT NAME
C
      J = 40 - RNL/2                           ;TAB TO CENTER
      IF (DSP) WRITE (CON, 1000)
     + (SPACE, I=1,J), (RIGNAM(I), I=1,RNL)
      IF (PRT) WRITE (PTR, 1000)
     + (SPACE, I=1,J), (RIGNAM(I), I=1,RNL)
C
C *** DISPLAY CENTERED JOB NAME
C
      J = 40 - JNL/2                           ;TAB TO CENTER
      IF (DSP) WRITE (CON, 1001)
```

```
     + (SPACE, I=1,J), (JOBNAM(I), I=1,JNL)
      IF (PRT) WRITE (PTR, 1001)
     + (SPACE, I=1,J), (JOBNAM(I), I=1,JNL)
C
C *** DISPLAY RIG LIMITS
C
      IF (DSP) WRITE (CON, 1002)
     + DEPTH, OFFSET, SAFETY, ANCS, TYPS     ;DESIGN LIMITS
      IF (DSP) WRITE (CON, 1003)
     + (I, LTYP(I),
     +  RIGX(I), RIGY(I), ANCX(I), ANCY(I),
     +  ADIR(I), ARAD(I), APRE(I), ATOP(I),
     + I=1,ANCS)                             ;ANCHOR PARAMETERS
C
      IF (PRT) WRITE (PTR, 1002)
     + DEPTH, OFFSET, SAFETY, ANCS, TYPS     ;DESIGN LIMITS
      IF (PRT) WRITE (PTR, 1003)
     + (I, LTYP(I),
     +  RIGX(I), RIGY(I), ANCX(I), ANCY(I),
     +  ADIR(I), ARAD(I), APRE(I), ATOP(I),
     + I=1,ANCS)                             ;ANCHOR PARAMETERS
C
      DO 100 I=1,TYPS                        ;LOOP OVER LEG STYLES
      IF (DSP) CALL SHOLEG (CON, I)          ;DISPLAY LEG BY SEGMENTS
      IF (PRT) CALL SHOLEG (PTR, I)          ;PRINT LEG BY SEGMENTS
  100 CONTINUE                               ;NEXT STYLE
      IF (DSP) CALL DELAY(1)                 ;TIMED PAUSE
      RETURN                                 ;CASE LOADED
C
 1000 FORMAT ('1'/
     + 24X'Mooring Definition for Root Name'/
     + 80A1)                                 ;CENTERED NAME
C
 1001 FORMAT (/1X79A1)                       ;CENTERED JOB
C
 1002 FORMAT (/14X
     + '      Water    Design    Safety   ',
     + '   No.     No. Leg'/14X
     + '      Depth    Offset    Factor   ',
     + ' Anchors    Types'/14X
     + 3F10.2, 5XI2, 9XI2/)                  ;DESIGN LIMITS
C
 1003 FORMAT (/
     + '   Anchor  Fairlead Position',
     + '   Anchor Position   Anchor ',
     + ' Anchor    Anchor     Top'/
     + ' No. Type     X         Y  ',
     + '        X         Y  Direction',
     + ' Radius  Preload    Scope'/
     + (2I4, 2F9.2, 2F9.1, F9.2,
     +         F7.0, F10.0, F9.0))           ;DISPLAY/PRINT
C
      END
```

```
      SUBROUTINE SHOLEG (LUN, LEG)          ;D.B.DILLON  EG&G SEPT-85
C
C *** OUTPUT THE PARAMETERS OF A LEG TYPE BY SEGMENTS
C
      INTEGER*2 LUN                         ;ARGUMENTS: LOGICAL UNIT FOR OUTPUT
      INTEGER*2 LEG                         ;LEG TYPE CODE
      INTEGER*2 M                           ;LOCALS: SEGMENT MATERIAL CODE
      INTEGER*2 SEG                         ;SEGMENT INDEX
      INTEGER*2 SX                          ;SEGMENT LIMIT
      INTEGER*2 TYPS, SEGS, MAT             ;/CABLES/
      REAL DIA, BRK, LEN, WGT, EA, BNCY
      CHARACTER*16 MATNAM                   ;/MATLST/
C
      COMMON /MATLST/ MATNAM(4)             ;MATERIAL NAMES
      COMMON /CABLES/ TYPS, SEGS(12),
     +    MAT(5,12), DIA(5,12),
     +    BRK(5,12), LEN(5,12), WGT(5,12),
     +    EA(5,12), BNCY(5,12)              ;LEG SEGMENT PARAMETERS
C
C *** LEG HEADER
C
      WRITE (LUN, 1000) LEG
      SX = SEGS(LEG)                        ,DUMMY SIMPLE VARIABLE
C
C *** LOOP OVER SEGMENTS IN LEG
C
      DO 100 SEG=1,SX
      M = MAT(SEG,LEG)                      ;MATERIAL TYPE CODE
      IF (M .LT. 1 .OR. M .GT. 3) M = 4     ;FORCE BOUNDS
      WRITE (LUN, 1001) SEG, MATNAM(M),
     + DIA(SEG,LEG), LEN(SEG,LEG),
     + WGT(SEG,LEG), BRK(SEG,LEG),
     +  EA(SEG,LEG), BNCY(SEG,LEG)          ;DISPLAY SEGMENT
  100 CONTINUE                              ;NEXT SEGMENT
C
      RETURN                                ;FINISHED
C
 1000 FORMAT (/
     + ' Leg Type', I3/
     + ' Seg. Material          Size     ',
     + ' Length    Weight  Strength',
     + ' Elasticity Buoyancy')              ;HEADER
C
 1001 FORMAT (I4, 1XA16, F7.3, F10.2, F10.4,
     +    F10.0, 1P1E11.4, 0P1F10.0)        ;LEG ECHO
C
      END
```

```
      LOGICAL*2 FUNCTION MAKFIL (SUBNAM, TYP, LU)
C                                             ;D.B.DILLON  EG&G
C *** OPEN UNFORMATTED TABLE FILE ON DRIVE DU FOR LEG TYP
C
      CHARACTER SUBNAM(6)                     ;ARGUMENTS: NAME SUFFIX
      INTEGER*2 TYP                           ;LEG TYPE CODE
      INTEGER*2 LU                            ;FILE UNIT NUMBER
      LOGICAL*2 OPENUF                        ;FUNCTION
      EXTERNAL  OPENUF
      CHARACTER LNN*2, LXX(2)                 ;LOCAL: NAME EXTENSIONS
      EQUIVALENCE (LNN, LXX(1))               ;SHARE MEMORY
C
C *** FORM FILE SUBNAME AND EXTENSION FROM LEG TYPE
C     THE 'WRITE' IS FORTRAN-77 FOR 'ENCODE'
C     FORMAT I2.2 RETAINS LEADING ZEROS IN STRING: 1-9 -> 01-09
C     THIS IS A GOOD EXAMPLE OF HOW TORTUOUS FORTRAN-77'S STRING
C     HANDLING FACILITIES ARE. THE 1-BYTE ARRAYS OF FORTRAN-66 WEREN'T
C     PERFECT, BUT THEY AVOIDED THIS KIND OF SOPHISTRY.
C
      WRITE (LNN, 1000) TYP                   ;CONVERT TYP TO ASCII STRING
      SUBNAM(5) = LXX(1)                      ;COPY INTO FILE SUBNAME
      SUBNAM(6) = LXX(2)                      ; BYTE BY BYTE
      MAKFIL = OPENUF (SUBNAM(1), LU)         ;OPEN THE FILE
      RETURN
C
 1000 FORMAT (I2.2)                           ;USE LEADING ZERO
      END
```

```
      LOGICAL*2 FUNCTION OPENUF (SUBNAM, LU)
C                                                 ;D.B.DILLON  EG&G
C *** OPEN UNFORMATTED TABLE FILE ON DRIVE DU
C
      CHARACTER*1 SUBNAM(6)                       ;FILE CODE AND EXTENSION
      INTEGER*2 LU                                ;LOGICAL*2FILE UNIT NUMBER
      INTEGER*2 I                                 ;LOCAL
      CHARACTER*72 AUXNAM                         ;STRING EQUIV. FOR RIGNAM
      LOGICAL*4 OLD                               ;USED IN OPEN
      INTEGER*2 CON, PTR, MSG, AUX, RIG           ;/UNITS/
      CHARACTER RIGNAM, JOBNAM                    ;/JOB/
      INTEGER*2 RNL, JNL                          ;/NAMLEN/
C
      COMMON /NAMLEN/ RNL, JNL                    ;NAME LENGTHS
      COMMON /JOB/ RIGNAM(72), JOBNAM(72)         ;PROBLEM ID
      COMMON /UNITS/ CON, PTR, MSG, AUX, RIG;LOGICAL UNITS
C
      EQUIVALENCE (AUXNAM, RIGNAM)                ;LINK A72 TO 72A1 FOR OPEN
C
C *** OPEN AUXILIARY FILE FOR LEG
C
      DO 100 I=1,6
      RIGNAM(RNL+I) = SUBNAM(I)                   ;OVERLAY SUBNAME
  100 CONTINUE
C
      INQUIRE (FILE=AUXNAM, EXIST=OLD)            ;PRE-EXISTENT?
      IF (OLD) GOTO 110                           ;YES
      OPEN (LU, FILE=AUXNAM,                      ;NO
     +        STATUS='NEW',
     +          FORM='UNFORMATTED',
     +        IOSTAT=IO)                          ;OPEN NEW FILE
      GOTO 120
C
  110 OPEN (LU, FILE=AUXNAM,                      ;OPEN OLD FILE
     +        STATUS='OLD',
     +          FORM='UNFORMATTED',
     +        IOSTAT=IO)
C
  120 OPENUF = (IO .NE. 0)                        ;ERROR FLAG
      IF (OPENUF)
     +    WRITE (CON, 1001) IO, LU, RIGNAM        ;BAD OPEN
      REWIND LU                                   ;FIRST RECORD
      RETURN
C
 1001 FORMAT (' OPENUF@120:'/ I4,
     + ' Unable to open', I3/'  as ', 72a1) ;WARNING
C
      END
```

```
      LOGICAL*2 FUNCTION MAKDEF (SUBNAM, LU);D.B.DILLON  EG&G
C
C *** OPEN FORMATTED FILE ON DRIVE LU
C
      CHARACTER SUBNAM(6)                     ;ARGUMENTS: FILE EXTENSION
      INTEGER*2 LU                            ;LOGICAL UNIT
      INTEGER*2 I                             ;LOCAL
      LOGICAL*4 OLD                           ;INQUIRE ARGUMENT
      CHARACTER AUXNAM*72                     ,A72 OVERLAY FOR RIGNAM
      INTEGER*2 CON, PTR, MSG, AUX, RIG       ;/UNITS/
      CHARACTER RIGNAM, JOBNAM                ;/JOB/
      INTEGER*2 RNL, JNL                      ;/NAMLEN/
C
      COMMON /NAMLEN/ RNL, JNL                ,NAME LENGTHS
      COMMON /JOB/ RIGNAM(72), JOBNAM(72)     ;PROBLEM ID
      COMMON /UNITS/ CON, PTR, MSG, AUX, RIG;LOGICAL UNITS
C
      EQUIVALENCE (AUXNAM, RIGNAM)            ;LINK A72 TO 72A1 FOR OPEN
C
C *** FORM AUXILIARY FILE NAME
C
      DO 100 I=1,6
      RIGNAM(RNL+I) = SUBNAM(I)               ;OVERLAY SUBNAME
  100 CONTINUE
C
C *** OPEN FORMATTED FILE FOR RIG DEFINITIONS
C
      INQUIRE (FILE=AUXNAM, EXIST=OLD)        ;PRE-EXISTENT?
      IF (OLD) GOTO 110                       ;YES
      OPEN (LU, FILE=AUXNAM,                  ;NO
     +        STATUS='NEW',
     +        IOSTAT=IO)                      ;OPEN NEW FILE
      GOTO 120
C
  110 OPEN (LU, FILE=AUXNAM,                  ;OPEN OLD FILE
     +        STATUS='OLD',
     +        IOSTAT=IO)
C
  120 MAKDEF = (IO .NE. 0)                    ;ERROR FLAG
      IF (MAKDEF) WRITE (CON, 1001)
     +    IO, LU, RIGNAM                      ;BAD OPEN
      REWIND LU                               ;FIRST RECORD
      RETURN
C
 1001 FORMAT (' MAKDEF@120:'/ I4,
     + ' Unable to open', I3/'  as ', 72a1) ;WARNING
C
      END
```

```
      LOGICAL*2 FUNCTION GETRNG                  ;D.B.DILLON  EG&G  MAY-85
     +    (CON, FIRST, LAST, STEP)
C
C *** GET DEFLECTION ROSE ANGLE RANGE FROM USER INPUT
C
      INTEGER*2 CON                              ;ARGUMENTS: CONSOLE UNIT
      REAL FIRST                                 ;INITIAL ROSE ANGLE
      REAL LAST                                  ;FINAL ROSE ANGLE
      REAL STEP                                  ;ANGLE INCREMENT
      LOGICAL*2 USRINP, GETFLT                   ;FUNCTIONS
      EXTERNAL  USRINP, GETFLT
      REAL ASTP                                  ;LOCAL: STEP SIZE
      REAL ADIF                                  ;RANGE SIZE
      REAL TOL                                   ;LEAST STEP
      CHARACTER OPT                              ;ALIAS FOR 1ST USER ENTRY BYTE
      INTEGER*2 LTXT, LPTR                       ;/USRPTR/
      CHARACTER TEXT                             ;/USRTXT/
C
      COMMON /USRTXT/ TEXT(80)                   ;USER ENTRY BUFFER
      COMMON /USRPTR/ LTXT, LPTR                 ;BUFFER POINTERS: LENGTH, CURRENT
C
      EQUIVALENCE (OPT, TEXT(1))                 ;1ST BYTE, USER ENTRY
C
      DATA TOL /.001/
C
C *** PROMPT FOR THREE ANGLE ARGUMENTS
C
      GETRNG = .FALSE.                           ;NOT ABORTED
  100 WRITE (CON, 1001)                          ;PROMPT FOR ANGLE RANGE
      IF (USRINP(22)) GOTO 100                   ;GET RESPONSE OR HELP
      IF (OPT .EQ. 'Q' .OR.
     +    OPT .EQ. 'q') GOTO 300                 ;ABORT CASE
      IF (GETFLT(FIRST)) GOTO 100                ;PARSE INITIAL ANGLE
      FIRST = AMOD(FIRST,360.)                   ;RESTRICT RANGE -360 TO 360
C
  110 IF (GETFLT(LAST)) GOTO 200                 ;PARSE FINAL ANGLE
      LAST = AMOD(LAST,360.)                     ;FINAL ANGLE IN ENTRY
      GOTO 130
C
C *** CHECK FOR UNNECESSARY STEP
C
  130 ADIF = LAST - FIRST                        ;RANGE, + OR -
      STEP = SIGN(2. ADIF)*ADIF
      ADIF = ABS(ADIF)                           ;RANGE, +
      IF (ADIF .LT. TOL) GOTO 310                ;FIRST=LAST: NO STEP
      ADIF = ADIF*TOL                            ;MIN. STEP
C
  140 IF (GETFLT(STEP)) GOTO 150                 ;PARSE ANGLE STEP
      STEP = AMOD(STEP,360.)
      ASTP = ABS(STEP)                           ;STEP SIZE
      IF (ASTP .GT. ADIF) GOTO 310               ;OK
      WRITE (CON, 1003) ASTP, ADIF               ;STEP ERROR
C
C *** PROMPT FOR STEP ARGUMENT
C
```

```
  150 WRITE (CON, 1004) FIRST, LAST          ;PROMPT FOR STEP
      IF (USRINP(23)) GOTO 150               ;GET STEP OR HELP
      IF (OPT .EQ. 'Q' .OR.
     +    OPT .EQ. 'q') GOTO 300             ;ABORT CASE
      GOTO 140
C
C *** PROMPT FOR FINAL, STEP ARGUMENTS
C
  200 WRITE (CON, 1002) FIRST                ;PROMPT FOR FINAL, STEP ANGLES
      IF (USRINP(23)) GOTO 200               ;GET FINAL STEP OR HELP
      IF (OPT .EQ. 'Q' .OR.
     +    OPT .EQ. 'q') GOTO 300             ;ABORT CASE
      GOTO 110                               ; ELSE TRY AGAIN
C
C *** FINISHED
C
  300 GETRNG = .TRUE.                        ;ABORTED
  310 RETURN                                 ;RANGE SPECIFIED
C
 1001 FORMAT (/
     + ' Initial, Final, Step Rig',
     + ' Deflection Angles'/
     + ' (Degrees Clockwise from Forward',
     + '  Q=Quit)')                          ;THREE ARGUMENT PROMPT
C
 1002 FORMAT (/' Initial Rig Deflection: ',
     + F9.3, ' Deg.'/
     + ' Final, Step Rig Deflection Angles',
     + '  Q=Quit)')                          ;TWO ARGUMENT PROMPT
C
 1003 FORMAT (/E14.7,
     +     ' Step size must exceed', E14.7)  ;STEP WARNING
C
 1004 FORMAT (/' Initial Rig Deflection: ',
     + F9.3, ' Deg.'/
     +       '           Final Deflection: ',
     + F9.3, ' Deg.'/
     + ' Angle Step Size (Deg.)',
     + '  Q=Quit)')                          ;STEP PROMPT
C
      END
```

```
      LOGICAL*2 FUNCTION MAKROS                  ;D.B.DILLON  EG&G  MAY-85
     +                  (OPR, SRV, HEAVE)
C
C *** OPEN OPERATIONAL AND SURVIVAL ROSE FILES
C
      INTEGER*2 OPR, SRV                         ;ARGUMENTS: UNIT NOS.
      REAL HEAVE                                 ;RIG HEAVE, FEET, + UPWARD
      LOGICAL*2 OPENUF                           ;FUNCTION
      EXTERNAL  OPENUF
      INTEGER*2 HEAV                             ;INTEGER HEAVE (W/O DECIMAL POINT)
      CHARACTER*1 HPNAM(6)                       ;LOCAL: ROSE FILE NAME
      CHARACTER*3 HVNAM                          ;EQUIVALENCED DUMMY FOR ENCODE
      EQUIVALENCE (HVNAM, HPNAM(4))              ;HEAVE SUFFIX "Hnn" OR "-nn"
C                                                ;NOTE: ´+´ & ´ ´ NOT ALLOWED IN
C                                                ;      MSDOS FILE NAMES
      DATA HPNAM /
     +    ´O´, ´P´, ´.´, ´H´, ´0´, ´0´/          ;ROSE FILE NAME
C
      HEAV = HEAVE                               ;REAL TO INTEGER CONVERSION
      WRITE (HVNAM, 1000) HEAV                   ;EMBED HEAVE IN NAME SUFFIX
      IF (HPNAM(4) .NE. ´-´) HPNAM(4) = ´H´      ;FILL SIGN BYTE FOR + HEAVE
      HPNAM(1) = ´O´                             ;SET NAME
      HPNAM(2) = ´P´
      MAKROS = OPENUF (HPNAM(1), OPR)            ;OPEN OPERATIONAL ROSE FILE
C
      HPNAM(1) = ´S´                             ;SET SURVIVAL NAME
      HPNAM(2) = ´V´
      MAKROS = OPENUF (HPNAM(1), SRV)            ;OPEN FILE
C
      IF (MAKROS) PAUSE ´MAKROS aborting = TRUE´
      RETURN
C
 1000 FORMAT (I3.2)                              ;HEAVE SUFFIX W/O DECIMAL POINT
C
      END
```

```
C$DEBUG
$STRICT                                          ;MODULE PREL.FOR
C                                                ;SUBROUTINE GETPRE
C                                                ;LOGICAL*2 FUNCTION STEEP
C                                                ;LOGICAL*2 FUNCTION HPMAX
C
C     ------------------------------------------------------------
C
      SUBROUTINE GETPRE (BAD)                    ;D.B.DILLON  EG&G  1985
C
C *** GET AVG PRELOAD FROM USER OR USE OPTIONAL CALCULATED PRELOAD
C     RETURN: REQUIRED LENGTH OF TOPMOST SEGMENT IN ATOP
C             ACTUAL PRELOADS BY LEG IN APRE
C             BAD=.TRUE. IF FUNCTION CALL ERRORS OCCUR
C     OPTION: CALCULATE PRELOAD FOR LEAST SCOPE ON BOTTOM AT DESIGN LOAD
C
      LOGICAL*2 STEEP, PRANGE, OPTPRE,           ;FUNCTIONS
     + USRINP, GETFLT, GETSTR, PUTDEF,
     + FINDHS, HPMAX, TOPLEN
      EXTERNAL  STEEP, PRANGE, OPTPRE,
     + USRINP, GETFLT, GETSTR, PUTDEF,
     + FINDHS, HPMAX, TOPLEN
      LOGICAL*2 BAD                              ;ARGUMENT: ERROR FLAG
      INTEGER*2 ANC                              ;LOCAL: ANCHOR INDEX
      INTEGER*2 SBC4, PLC7                       ,HVSS FILE COLUMNS
      CHARACTER OPT                              ;ALIAS FOR 1ST USER ENTRY BYTE
      REAL PRE, PBAR, PTRY                       ;ACTUAL, AVERAGE, TRIAL PRELOADS
      REAL PMIN, PMAX, PN, PX                    ;EXTREME PRELOADS & DUMMIES
      REAL RGX, RGY, FBRG(12), FRAD(12)          ;RIG FAIRLEAD COORDINATES
      REAL SOB                                   ;SCOPE ON BOTTOM AT DESIGN LOAD
      REAL MU                                    ;DEPTH FRACTION FOR BOTTOM SCOPE
      REAL PI, D2R, R2D                          ;/TRIG/
      REAL SCOPE,                                ;/HSTABL/
     + XMIN, SBPL, SBDL, XPRE, XDES,
     +                 HPRE, HDES, HDIF
      REAL RIGX, RIGY                            ;/SEMI/
      INTEGER*2 ANCS, LTYP                       ;/ANCHOR/
      REAL DEPTH, OFFSET, SAFETY,
     +    ADIR, ATOP, ARAD, APRE, ANCX, ANCY
      INTEGER*2 CON, PTR, MSG, AUX, RIG          ;/UNITS/
      INTEGER*2 LTXT, LPTR                       ,/USRPTR/
      CHARACTER TEXT                             ;/USRTXT/
C
      COMMON /USRTXT/ TEXT(80)                   ,ENTRY BUFFER
      COMMON /USRPTR/ LTXT, LPTR                 ;BUFFER MARKERS: LENGTH, CURRENT
      COMMON /UNITS/ CON, PTR, MSG, AUX, RIG,LOGICAL UNITS
      COMMON /ANCHOR/ ANCS, LTYP(12),
     +    DEPTH, OFFSET, SAFETY,
     +    ADIR(12), ATOP(12), ARAD(12),
     +    APRE(12), ANCX(12), ANCY(12)           ;ANCHOR PARAMETERS
      COMMON /SEMI/ RIGX(12), RIGY(12)           ;RIG FAIRLEADS
      COMMON /HSTABL/ SCOPE,
     + XMIN, SBPL, SBDL, XPRE, XDES,
     +                 HPRE, HDES, HDIF          ;H VS S ENTRY
      COMMON /TRIG/ PI, D2R, R2D                 ;ANGLE CONVERSION
```

```
C
      EQUIVALENCE (OPT, TEXT(1))            ;1ST BYTE, USER ENTRY
      DATA SBC4, PLC7 /4, 7/                ;H VS S FILE COLUMNS
      DATA MU /0.05/                        ;DEPTH FRACTION FOR BOTTOM SCOPE
C
C *** CONVERT FAIRLEAD LOCATIONS TO POLAR COORDINATES
C
      IF (FINDHS()) CALL HVSS               ;MAKE H VS S FILES
      CALL PRTDEF (1)                       ;DISPLAY MOORING
      DO 100 ANC=1,ANCS
      RGX = RIGX(ANC)                       ;USE SIMPLE VARIABLES
      RGY = RIGY(ANC)
      FBRG(ANC) = ATAN2(RGX,RGY)            ;BEARING
      FRAD(ANC) = SQRT(RGX*RGX + RGY*RGY)   ;RADIUS
  100 CONTINUE
C
C *** DETERMINE PRELOAD RATIOS FOR EQUILIBRIUM PLANFORM GEOMETRY
C
      BAD = STEEP (APRE(1), FRAD(1),
     +             FBRG(1), ADIR(1), ANCS)  ;USE STEEPEST DESCENT
      IF (BAD) RETURN                       ;FAILED: SET ERROR FLAG
      SOB = -MU*DEPTH                       ;SCOPE ON BOTTOM AT DESIGN LOAD
      BAD = PRANGE (PMIN, PMAX, SOB)        ;PERMITTED PRELOADS
      IF (BAD) RETURN                       ;FAILED: SET ERROR FLAG
C
C *** ESTIMATE PRELOAD FOR SPECIFC SCOPE ON BOTTOM (SOB) AT DESIGN LOAD
C
  400 PTRY = -1.E26                         ;NEGATE GREATEST PRELOAD
      DO 410 ANC=1,ANCS                     ;LEG LOOP
      BAD = OPTPRE (SOB, LTYP(ANC))         ;OPTIMIZE PRELOAD ON SOB IN HVSS
      IF (BAD) BAD = HPMAX (SOB, LTYP(ANC)) ;ELSE MAX. H.POWER W/S.O.B.>SOB
      IF (BAD) GOTO 200                     ;BAD
      PTRY = AMAX1 (PTRY, HPRE/APRE(ANC))   ;SELECT GREATEST AVG. PRELOAD
  410 CONTINUE                              ; FOR S.O.B. AT LEAST SOB
      IF (PTRY .LT. PMIN) PMIN = PTRY       ;ADJUST FOR EXTRAPLOATED SOB
C
C *** LOOK UP PRELOAD IN H VS S TABLE
C
  500 PBAR = PTRY                           ;USE ENTRY
      IF (PBAR .LT. PMIN .OR.               ;VERIFY IT
     +    PBAR .GT. PMAX) GOTO 200          ;OUT OF LOOKUP RANGE
      WRITE (CON, 1002) PBAR                ;TABLE HEADER
      DO 510 ANC=1,ANCS                     ;LEG LOOP
      PRE = PBAR*APRE(ANC)                  ;LEG PRELOAD
      BAD = TOPLEN (PRE, LTYP(ANC))         ;LOOKUP TOP SCOPE IN H VS S
      IF (BAD) GOTO 200                     ;BAD
      WRITE (CON, 1003) ANC, SCOPE,
     + XMIN, SBPL, SBDL, XPRE, XDES,
     +                  HPRE, HDES, HDIF    ;DISPLAY CASE
      ATOP(ANC) = SCOPE                     ;ACCEPT TOP SCOPE
      ARAD(ANC) = XPRE                      ;ANCHOR RADIUS = PRELOAD SPAN
      ANCX(ANC) = RIGX(ANC) +
     +            XPRE * SIN(D2R*ADIR(ANC)) ;ANCHOR X COORDINATE
      ANCY(ANC) = RIGY(ANC) +
     +            XPRE * COS(D2R*ADIR(ANC)) ;ANCHOR Y COORDINATE
```

```
  510 CONTINUE
C
C *** GET USER AVERAGE PRELOAD REQUEST
C
  200 WRITE (CON, 1000)                       ;DISPLAY PRELOAD RATIOS
    +  (ANC, APRE(ANC), ANC=1,ANCS)
      WRITE (CON, 1001) PMIN, PMAX, PBAR      ;AVERAGE PRELOAD PROMPT
      IF (USRINP(15)) GOTO 200                ;GET AVG. PRELOAD
      IF (OPT .EQ. 'U' .OR.
    +     OPT .EQ. 'u') GOTO 300              ;USE CURRENT VALUE
      IF (OPT .EQ. 'E' .OR.
    +     OPT .EQ. 'e') GOTO 400              ;ESTIMATE A PRELOAD
      IF (OPT .EQ. 'Q' .OR.
    +     OPT .EQ. 'q') GOTO 610              ;ABORT THE CASE
      IF (GETFLT(PTRY)) GOTO 200              ; ELSE PARSE NEW PRELOAD
      GOTO 500                                ; AND TRY IT ON FOR SIZE
C
C *** ACCEPT CURRENT AVERAGE PRELOAD: QUIT
C
  300 DO 310 ANC=1,ANCS
      APRE(ANC) = PBAR*APRE(ANC)              ;RATIOS TO PRELOADS
  310 CONTINUE
      ANC = 2                                 ;CONSOLE DISPLAY ONLY
  320 BAD = PUTDEF ()
      IF (BAD) GOTO 600                       ;STORE CASE DEFINITION
      RETURN                                  ;QUIT
C
C *** ERROR RECOVERY: UNABLE TO SAVE DEFINITION
C
  600 WRITE (CON, 1004)
      IF (USRINP(24)) GOTO 600                ;HELP REQUEST
      IF (OPT .EQ. 'N' .OR.
    +     OPT .EQ. 'n') GOTO 610              ;ABORT UNSAVED DEFINITION
      CALL PRTDEF(ANC)                        ;PRINT ON PUTDEF ERROR
      GOTO 320
C
  610 BAD = .TRUE.                            ;SET ERROR FLAG
      RETURN
C
C ----------------------------------------
C
 1000 FORMAT (/
    + 29X'Anchor   Preload Ratio'/
    +  12(31XI2, 5XF11.6/))                   ;PRELOAD RATIOS
C
 1001 FORMAT (/
    + ' Average Preload',
    + '   (U=Use  E=Estimate  Q=Quit)'/
    + 9X'Minimum   Maximum   Current'/
    + 6X3F10.1)                               ;INPUT PROMPT
C
 1002 FORMAT (//
    + 19X'Anchor Properties for Average',
    + ' Preload =', F11.1//
    + ' Anc-    Top    Force/Length on Bottom',
```

```
     + '  Pre-   Design   Pre-',
     + '  Design Holding'/
     + ' hor  Scope  Slack  Hpre    Hdes',
     + '    Span    Span    Load',
     + '    Load    Power')                    ;DISPLAY HEADER
C
 1003 FORMAT (I4, 9F8.0)                      ,DISPLAY LINE
C
 1004 FORMAT (' Try again? (Y/N)')            ,RETRY SAVE DEFINITION
C
      END
```

```
      LOGICAL*2 FUNCTION STEEP                     ;26-FEB-85  D.B.DILLON EG&G
     +         (H, FR, FB, AB, N)
C
C *** RETURN RELATIVE EQUILIBRIUM PRELOAD VECTOR, H, USING THE METHOD
C      GIVEN FAIRLEAD RADIUS VECTOR, FR             OF STEEPEST DESCENT
C            FAIRLEAD BEARING LIST, FB
C            ANCHOR BEARING LIST, AB
C            NO. OF ANCHORS, N
C
      INTEGER*2 N                                  ;ARGUMENTS
      REAL H(N), FR(N), FB(N), AB(N)               ;SEE DESCRIPTION ABOVE
      INTEGER*2 I                                  ;LOCAL: LOOP INDEX
      INTEGER*2 TRY                                ;ITERATION COUNTER
      INTEGER*2 TRYS                               ;ITERATION LIMIT
      INTEGER*2 XP, XM, YP, YM, MP, MM             ;EFFECTIVESS FLAGS
      REAL RBAR                                    ;AVERAGE FAIRLEAD RADIUS
      REAL EX, EY, EM                              ;FORCE/MOMENT ERRORS
      REAL DEXDH(12), DEYDH(12), DEMDH(12)         ;ERROR PARTIAL DERIVATIVES
      REAL ERR                                     ;RMS FORCE/MOMENT ERROR
      REAL ERSQ                                    ;MEAN-SQUARED ERROR, ERR*ERR
      REAL EOLD                                    ;PRIOR ERROR
      REAL TOL                                     ;CONVERGENCE TOLERANCE
      REAL RATE                                    ;CONVERGENCE RATE FACTOR
      REAL P                                       ;LAPLACIAN OF RMS ERROR
      REAL DH(12)                                  ;PRELOAD UPDATE VECTOR
      REAL SDH                                     ;RMS PRELOAD UPDATE
      REAL ABG                                     ;ANCHOR BEARING, RADIANS
      REAL PI, D2R, R2D                            ;/TRIG/
      INTEGER*2 CON, PTR, MSG, AUX, RIG            ;/UNITS/
C
      COMMON /UNITS/ CON, PTR, MSG, AUX, RIG
      COMMON /TRIG/ PI, D2R, R2D                   ;ANGLE CONVERSION
C
      DATA TRYS /200/                              ;LIMIT ITERATION
      DATA TOL /.0001/                             ;CONVERGENCE TOLERANCE
C
C *** STEP 1: INITIATE CASE CONSTANTS
C
      RATE = 2.                                    ;BEGIN WITH OVER-RELAXATION
      RBAR = 0.                                    ;START AVERAGES
      DO 10 I=1,N                                  ;AVERAGE
      H(I) = 1.                                    ; PRELOADS AND
      RBAR = RBAR + FR(I)                          ; FAIRLEAD RADII
   10 CONTINUE
      RBAR = RBAR/N
C
C *** DETERMINE LEG SENSITIVITY AND EFFECTIVENESS. THERE MUST BE
C     OPPOSING EFFECTIVE LEGS FOR EACH COMPONENT X, Y, & MOMENT.
C
      XP = 0                                       ;NO + X-EFFECTIVENESS
      XM = 0                                       ;NO - X-EFFECT.
      YP = 0                                       ;ETC.
      YM = 0
      MP = 0
      MM = 0
```

```
C
      DO 20 I=1 N                              ;SCAN BY FAIRLEAD
      ABG = D2R*AB(I)                          ;ANCHOR BEARING IN RADIANS
      DEXDH(I) = SIN(ABG)                      ;NORMALIZE
      DEYDH(I) = COS(ABG)                      ; ERROR PARTIAL
      DEMDH(I) = FR(I)*SIN(ABG-FB(I))          ;  DERIVATIVES
      IF (RBAR .NE. 0.) DEMDH(I) =             ;TEST FOR SINGLE POINT
     +                  DEMDH(I)/RBAR          ; MOORING
      IF (DEXDH(I) .GT. TOL) XP = 1            ;LEG IS +X EFFECTIVE
      IF (DEXDH(I) .LT.-TOL) XM =-1            ;LEG IS -X EFFECTIVE
      IF (DEYDH(I) .GT. TOL) YP = 1            ;LEG IS +Y EFFECTIVE
      IF (DEYDH(I) .LT.-TOL) YM =-1            ;LEG IS -Y EFFECTIVE
      IF (DEMDH(I) .GT. TOL) MP = 1            ;LEG IS +M EFFECTIVE
      IF (DEMDH(I) .LT.-TOL) MM =-1            ;LEG IS -M EFFECTIVE
   20 CONTINUE
C
      IF ((XP+XM .NE. 0) .OR.
     +    (YP+YM .NE. 0) .OR.
     +    (MP+MM .NE. 0)) GOTO 410             ;INEFFECTIVE GEOMETRY
      EOLD = 1.E26                             ;"INFINITE" PRIOR ERROR
      TRY = 0                                  ;START ITERATION COUNTER
C
C *** STEP 2: TEST FOR CONVERGENCE
C
  100 TRY = TRY + 1                            ;COUNT TRYS
      IF (TRY .GT. TRYS) GOTO 400              ,UNCONVERGED
      EX = 0.                                  ;START ERROR SUMS
      EY = 0.
      EM = 0.
      DO 110 I=1,N                             ;SUM NORMALIZED ERRORS
      EX = EX + H(I)*DEXDH(I)                  ;X-FORCE FROM ANCHOR I
      EY = EY + H(I)*DEYDH(I)                  ;Y-FORCE
      EM = EM + H(I)*DEMDH(I)                  ;CLOCKWISE MOMENT/RBAR
  110 CONTINUE
C
      ERSQ = EX*EX + EY*EY + EM*EM             ,MEAN-SQUARED ERROR
      ERR = SQRT(ERSQ)                         ;NORMALIZED RMS ERROR
      IF (ERR .LT. TOL) GOTO 600               ;CONVERGED
      IF (ERR .LT. EOLD) GOTO 200              ;CONVERGING
C
      DO 120 I=1,N                             ;DIVERGING:
      H(I) = H(I) - DH(I)                      ;CANCEL STEP
      DH(I) = DH(I)/2.                         ; THEN USE
  120 CONTINUE                                 ;  BINARY SEARCH
      RATE = RATE/2.                           ;REDUCE RELAXATION
      IF (RATE .LT. 6.E-8) GOTO 420            ,OVERRELAXED
      GOTO 300                                 ;   AND TRY AGAIN
C
C *** STEP 3: ESTIMATE PRELOAD UPDATE BY METHOD OF STEEPEST DESCENT
C
  200 P = 0.                                   ;SUM DESCENT COMPONENTS
      DO 210 I=1,N
      DH(I) = EX*DEXDH(I) +
     +        EY*DEYDH(I) +
     +        EM*DEMDH(I)                      ;COMPONENT FOR ANCHOR I
```

```
      P = P + DH(I)*DH(I)                     ;SUM OF SQUARES
  210 CONTINUE
C
      P = ERSQ/P
      DO 220 I=1,N                            ;COMPUTE NORMALIZED
      DH(I) = -P*DH(I)*RATE                   ; UPDATE COMPONENTS
  220 CONTINUE                                ;  BY ANCHOR
      EOLD = ERR                              ;SAVE CURRENT AS PRIOR
C
C *** STEP 4: UPDATE PRELOAD ESTIMATE FOR NEXT TRY
C
  300 SDH = 0.                                ;START UPDATE VECTOR
      DO 310 I=1,N                            ;COMPUTE UN-NORMALIZED
      H(I) = H(I) + DH(I)                     ; PRELOAD COMPONENTS
      SDH = SDH + DH(I)*DH(I)                 ;  AND MS UPDATE
  310 CONTINUE
      IF (SDH .NE. 0.) GOTO 100               ;TRY AGAIN
C
C *** CONVERGENCE FAULT RECOVERY
C
  400 WRITE (CON, 1000)                       ;FAULT WARNING
     + TRY, EX, EY, EM, ERR, DH, H
      GOTO 500
C
  410 WRITE (CON, 1001)                       ;INEFFECTIVE GEOMETRY
     + XP, XM, YP, YM, MP, MM,
     + (I, DEXDH(I), DEYDH(I), DEMDH(I),
     +  I=1,N)
      GOTO 400
C
  420 WRITE (CON, 1002) RATE                  ;OVER-RELAXATION
      GOTO 400
C
  500 STEEP = .TRUE.                          ;FAULT OCCURRED
      PAUSE
      RETURN
C
C *** STEP 5: NORMALIZE RELATIVE PRELOADS TO AVERAGE 1.000
C
  600 RBAR = 0.                               ,USE FOR HBAR SUM
      DO 610 I=1,N                            ;SUM RELATIVE PRELOADS
      RBAR = RBAR + H(I)
  610 CONTINUE
      RBAR = RBAR/N                           ;AVERAGE RELATIVE PRELOAD
      DO 620 I=1,N                            ;NORMALIZE TO HBAR = 1.00
      H(I) = H(I)/RBAR
  620 CONTINUE
      STEEP = .FALSE.                         ;CLEAR ERROR FLAG
      RETURN                                  ;PRELOAD RATIOS GOOD
C
 1000 FORMAT (
     + ' STEEP/1000: CONVERGENCE FAILURE'/
     + ' TRY   EX          EY          ',
     + ' EM          ERROR' / I4, 1P4E12.4/
     + ' DH:', 6E12.4 / 4X6E12.4/
```

```
     + '  H:', 6E12.4 / 4X6E12.4)              ;FAULT DISPLAY
C
 1001 FORMAT (
     + ' STEEP/1001@410: INEFFECTIVE',
     + ' GEOMETRY: XP XM YP YM MP MM'/
     + 38X6I3/ ' LEG d(Ex)/d(Hi)  ',
     + ' d(Ey)/d(Hi)   d(Em)/d(Hi)'/
     + 12(I4,1P3E14.6/))                        ;INEFFECTIVE CASE
C
 1002 FORMAT (' STEEP/1002@420:',
     + ' OVER-RELAXED RATE=', 1P1E12.4)        ;OVER-RELAXED
C
      END
```

```
      LOGICAL*2 FUNCTION HPMAX (SOB, TYP)   ;15-MAR-85 D.B.DILLON  EG&G
C
C *** GIVEN A LEG TYPE, LOOK IN ITS H VS S FILE FOR THE LARGEST HOLDING POWER
C     FOR WHICH THE SCOPE ON BOTTOM IS MORE THAN SOB. RETURN HPMAX .TRUE. IF
C     SOB AT DESIGN LOAD IS ALWAYS LESS THAN ARGUMENT SOB.
C
      LOGICAL*2 MAKFIL                      ;FUNCTION
      EXTERNAL  MAKFIL
      INTEGER*2 TYP                         ;ARGUMENTS: LEG TYPE
      REAL SOB                              ;LEAST SCOPE ON BOTTOM
      CHARACTER HSFIL(6)                    ;LOCAL
      REAL TABL(9)                          ;WORKING LINE ON HY VS S FILE
      INTEGER*2 J, RX                       ;RECORD POINTERS
      REAL HSN, HSX                         ;/HSXTRM/
      REAL CASE                             ;/HSTABL/
      INTEGER*2 CON, PTR, MSG, AUX, RIG     ;/UNITS/
C
      COMMON /UNITS/ CON, PTR, MSG, AUX, RIG;LOGICAL UNITS
      COMMON /HSTABL/ CASE(9)               ;H VS S FILE INTERPOLATION
      COMMON /HSXTRM/ HSN(9,12), HSX(9,12)  ;H VS S FILE EXTREMES
C
      DATA HSFIL /'H', 'S', '.', 'L', 2*'0'/;RIGNAMHS.LNN
C
C ---------------------------------------------------------------------
C *** CHECK FOR POSSIBLE SOB (NOTE: SOB'S ARE NEGATIVE IN H VS S FILES)
C
      HPMAX = HSN(4,TYP) .GT. SOB           ;ERROR: NO SOB'S EXCEED ARGUMENT
      IF (HPMAX) RETURN
      IF (MAKFIL(HSFIL(1),TYP,AUX))
     +              CALL PRELOD(TYP)        ;OPEN H VS S FOR LEG TYPE
      CASE(9) = -1.E-26                     ;DUMMY MAXIMUM HOLDING POWER
      RX = 0                                ;RECORDS READ
C
C *** SCAN FILE
C
  100 READ (AUX, END=200, ERR=210) TABL     ;GET CASE
      RX = RX + 1                           ;COUNT RECORDS
      IF ((TABL(4) .GT. SOB) .OR.
     +    (TABL(9) .LT. CASE(9))) GOTO 100  ;NEXT RECORD
      DO 110 J=1,9                          ,YES:
      CASE(J) = TABL(J)                     ;COPY RECORD INTO /HSTABL/
  110 CONTINUE
      J = RX                                ;NOTE EXTREME RECORD
      GOTO 100
C
C *** READ ERROR RECOVERY -----------------
C
  200 IF (CASE(9) .GT. 0.) GOTO 300         ;GOOD LOOKUP
  210 WRITE (CON, 1000) TYP                 ;ERROR MESSAGE HEADER
      HPMAX = .TRUE.                        ;RETURN ERROR FLAG
      IF (RX .EQ. 0) THEN
        WRITE (CON, 1001) AUX               ;EMPTY FILE
      ELSE
        WRITE (CON, 1002) RX                ;DAMAGED FILE
      ENDIF
```

```
      PAUSE
C
  300 CLOSE (AUX)
      RETURN
C
C ----------------------------------------------------------------------
C
 1000 FORMAT (//' HPMAX Error for leg type', I3)
C
 1001 FORMAT (//' Empty AUX file on unit', I2)
C
 1002 FORMAT (//' Read error after record', I3)
C
      END
```

```
C$DEBUG
$STRICT                                            ;MODULE ROSE.FOR
C                                                  ;SUBROUTINE ANALYZ
C                                                  ;LOGICAL*2 FUNCTION GETYAW
C                                                  ;SUBROUTINE TORQUE
C                                                  ;INTEGER*2 FUNCTION TRIES
C                                                                            -
C     ----------------------------------------------------------------------
C
      SUBROUTINE ANALYZ (XHOPT)                    ;D.B.DILLON  EG&G  MAY-85
C
C *** GIVEN: XHOPT=.TRUE. MAKE NEW X VS H FILES, THEN WRITE THE HOLDING POWER
C                         ROSES FOR THE OP.ROS AND SV.ROS FILES
C                  .FALSE. USE OLD X VS H FILES FOR THE ROSES
C     USER INPUT: AVERAGE PRELOAD OR DEFAULT
C                 ROSE ANGLE RANGE
C
      LOGICAL*2
     +  USRINP, MAKROS,
     +  GETINT, GETRNG, GETYAW                     ;FUNCTIONS
      EXTERNAL
     +  USRINP, MAKROS,
     +  GETINT, GETRNG, GETYAW
      LOGICAL*4 XHOPT                              ;ARGUMENT: XVSH FILE OPTION
      LOGICAL*2 NORMAL, SURVIV                     ;LOCALS: MOORING CONDITION FLAGS
      LOGICAL*2 ERR                                ;GETPRE ERROR FLAG
      LOGICAL*4 WROPT                              ;WRITE OPTION FLAG TO SUBRTN XVSH
      CHARACTER*1 SPACE                            ;TAB FILLER
      INTEGER*2 OPR, SRV                           ;ROSE FILE UNITS
      INTEGER*2 I, I2, L                           ;INDEXES
      REAL HEAVE                                   ;RIG HEAVE, FEET POSITIVE UPWARD
      REAL NOHEV                                   ;SEAFLOOR TO FAIRLEAD W/O HEAVE
      REAL OPHOLD, SRVHLD                          ;HOLDING POWER OF MOORING
      REAL OPDIR, SRVDIR                           ;DIRECTION OF WEATHER
      REAL OPSF, OPYAW                             ;SAFETY FACTOR, RIG YAW
      REAL THET1, THET2, STEP                      ;DEFLECTION ROSE ANGLE LIMITS
      REAL PI, D2R, R2D                            ;/TRIG/
      LOGICAL*2 ACTV                               ;/TORX/
      REAL THETA, XRIG, YRIG, YAW,
     +     NETMCW, NETFX, NETFY, NETSF,
     +     XSPN, LOAD, SFAC,
     +     TORQ, LEGX, LEGY
      INTEGER*2 ANCS, LTYP                         ;/ANCHOR/
      REAL DEPTH, OFFSET, SAFETY,
     +   ADIR, ATOP, ARAD, APRE, ANCX, ANCY
      INTEGER*2 CON, PTR, MSG, AUX, RIG            ;/UNITS/
      CHARACTER RIGNAM, JOBNAM                     ;/JOB/
      INTEGER*2 RNL, JNL                           ;/NAMLEN/
C
      COMMON /NAMLEN/ RNL, JNL                     ;NAME LENGTHS
      COMMON /JOB/ RIGNAM(72), JOBNAM(72)          ;PROBLEM ID
      COMMON /UNITS/ CON, PTR, MSG, AUX, RIG;LOGICAL UNITS
      COMMON /ANCHOR/ ANCS, LTYP(12),
     +   DEPTH, OFFSET, SAFETY,
     +   ADIR(12), ATOP(12), ARAD(12),
```

```
      OPSF = NETSF                              ;LEAST SAFETY FACTOR IN ANY LEG
      OPYAW = R2D * YAW                         ,EQUILIBRIUM RIG YAW (DEG.)
      WRITE (OPR)                               ;OPERATIONAL ROSE FILE
     +   THETA,    OPHOLD,   OPDIR,    NETFX,   ; MOORING
     +   NETFY,      OPSF,   OPYAW,   NETMCW,   ;  STATE AND   (SEE NOTE BELOW)
     +(LOAD(L),  XSPN(L), LEGX(L), LEGY(L),     ; LEG DETAILS
     + SFAC(L),  TORQ(L), ACTV(L), L=1,12)      ;  FROM /TORX/
C
C *** NOTE: SAFETY FACTOR IS UNCHANGED BETWEEN OPERATIONAL AND SURVIVAL
C           BECAUSE IT IS FOUND IN THE MOST UP-WEATHER LEG
C
      CALL SETLEG(SURVIV)                       ;LEE LEGS SLACKED (LEE =
      ERR = GETYAW()                            ;DOWN-WEATHER +/- 86 DEG)
      SRVHLD = SQRT (NETFX*NETFX +
     +               NETFY*NETFY)               ;SURVIVAL HOLDING POWER
      SRVDIR = R2D * ATAN2(-NETFX,-NETFY)       ;WEATHER DIRECTION
      YAW = R2D * YAW                           ;SURVIVAL RIG YAW (DEG.)
      WRITE (SRV)                               ;SURVIVAL ROSE FILE
     +   THETA,    SRVHLD,   SRVDIR,   NETFX,   ; MOORING
     +   NETFY,      OPSF,      YAW,  NETMCW,   ;  STATE AND
     +(LOAD(L),  XSPN(L), LEGX(L), LEGY(L),     ; LEG DETAILS
     + SFAC(L),  TORQ(L), ACTV(L), L=1,12)      ;  FROM /TORX/
C
      WRITE (CON, 1002) THETA,
     +  OPHOLD,  OPDIR,   OPSF, OPYAW,
     +  SRVHLD, SRVDIR,   OPSF,    YAW          ;DISPLAY RESULTS
      WRITE (PTR, 1002) THETA,
     +  OPHOLD,  OPDIR,   OPSF, OPYAW,
     +  SRVHLD, SRVDIR,   OPSF,    YAW          ;PRINT RESULTS
C
      THETA = THETA + STEP                      ;NEXT DEFLECTION ROSE ANGLE
  100 CONTINUE
      DEPTH = NOHEV                             ;RESTORE NOMINAL DEPTH
      CLOSE (OPR)                               ;CLOSE ROSE FILES
      CLOSE (SRV)
C
C *** SET UP OPTIONAL HEAVED CASE
C
  200 WRITE (CON, 1003)                         ;PROMPT FOR HEAVE
      IF (USRINP(25)) GOTO 200                  ;RETRY AFTER HELP REQUEST
      IF (GETINT(I)) GOTO 200                   ;RETRY AFTER ENTRY ERROR
      IF (I .EQ. 0) RETURN                      ;END COMMAND
      IF (IABS(I) .GT. 99) GOTO 200             ;RETRY, ENTRY OUT OF RANGE
      HEAVE = I                                 ;ACCEPT HEAVE ENTRY
      DEPTH = NOHEV + HEAVE                     ;HEAVE RIG
      GOTO 10                                   ;RUN HEAVED CASE
C
 1000 FORMAT ('1'//7X
     + ' OPERATIONAL AND SURVIVAL',
     + ' HOLDING POWER ANALYSIS FOR', F5.0,
     + ' FEET HEAVE'//80A1/)                    ;CASE HEADER
C
 1001 FORMAT (
     + ' Direction  |',
     + '  Operational (All legs active)  |',
```

```
C *** GET NORMALIZING MOMENT FROM PRELOAD TIMES RIG OFFSET
C
      NORM = 0.                              ;START SUM
      DO 100 ANC=1,ANCS                      ;GET AVERAGE PRELOAD
      NORM = NORM + APRE(ANC)                ; BY SUMMING
  100 CONTINUE
      NORM = NORM*DR/ANCS                    ;NORMALIZING MOMENT
C
C *** SET UP ITERATION TO NULLIFY YAW MOMENT
C
      TRYS = TRYX                            ;ITERATION LIMIT
      YAWX =  1.5708                         ;-90 < YAW < 90 DEG.
      YAWN = -1.5708
      YAW = 0.                               ;INITIAL ESTIMATE
C
C *** SUM LEG MOMENTS AND FORCES ON RIG
C
  200 NETMCW = 0.                            ;CLOCKWISE MOMENT
      NETFX = 0.                             ; AND FORCES BY
      NETFY = 0.                             ;  LEGS ON RIG
C
      DO 210 ANC=1,ANCS                      ;LOOP OVER LEGS
      CALL TORQUE(ANC)                       ;GET LEG LOAD
      TORQ(ANC) = TORQ(ANC)/NORM             ;NORMALIZE TORQUES
      NETMCW = NETMCW + TORQ(ANC)            ;SUM MOMENTS
      NETFX = NETFX + LEGX(ANC)              ; AND FORCES
      NETFY = NETFY + LEGY(ANC)
  210 CONTINUE
C
      MERR = NETMCW                          ;NORMALIZE NET MOMENT
      IF (ABS(MERR) .LT. TOL) GOTO 300       ;CONVERGED: FIND LEAST SFACTOR
      TRYZ = TRIES(TRYS, YAW, YAWX, YAWN,    ;MANAGE NEWTON-RAPHSON
     +             MERR, TOL)
      IF (TRYZ .LT. TRYX) GOTO 200           ;TRY AGAIN
C
C *** UNCONVERGED
C
      GETYAW = .TRUE.                        ;SET ERROR FLAG
      PAUSE 'GETYAW aborting = TRUE'
      RETURN
C
C *** SELECT LEAST SAFETY FACTOR IN ANY ACTIVE LEG
C
  300 NETSF = SFAC(1)                        ;START SCAN
      DO 310 ANC=2,ANCS                      ; OF ALL LEGS
      IF ((SFAC(ANC) .LT. NETSF) .AND.
     +     ACTV(ANC)) NETSF = SFAC(ANC)      ;SELECT LEAST SAFETY FACTOR
  310 CONTINUE
      GETYAW = .FALSE.                       ;CLEAR ERROR FLAG
      RETURN
C
      END
```

```
      SUBROUTINE TORQUE (LEG)                 ;D.B.DILLON  EG&G  MAY-85
C
C ***  GIVEN: THE POSITION AND YAW OF THE RIG
C     RETURN: THE TORQUE AND HORIZONTAL FORCE EXERTED BY A LEG IN /TORQ/
C
      INTEGER*2 LEG                           ;ARGUMENT: LEG NUMBER
      INTEGER*2 TYP                           ;LOCAL:LEG TYPE
      REAL C, S                               ;SIN(YAW), COS(YAW)
      REAL XF, YF                             ;FAIRLEAD COORDINATES
      REAL XS, YS                             ;FAIRLEAD TO ANCHOR COMPONENTS
      REAL SLACK                              ;SLACKED TENSION/STRENGTH
      LOGICAL*2 ACTV                          ;/TORX/
      REAL THETA, XRIG, YRIG, YAW,
     +     NETMCW, NETFX, NETFY, NETSF,
     +     XSPN, LOAD, SFAC,
     +     TORQ, LEGX, LEGY
      REAL RIGX, RIGY                         ;/SEMI/
      INTEGER*2 ANCS, LTYP                    ;/ANCHOR/
      REAL DEPTH, OFFSET, SAFETY,
     +    ADIR, ATOP, ARAD, APRE, ANCX, ANCY
      INTEGER*2 TYPS, SEGS, MAT               ;/CABLES/
      REAL DIA, BRK, LEN, WGT, EA, BNCY
C
      COMMON /CABLES/ TYPS, SEGS(12),
     +    MAT(5,12), DIA(5,12),
     +    BRK(5,12), LEN(5,12), WGT(5,12),
     +    EA(5,12), BNCY(5,12)                ;LEG SEGMENT PARAMETERS
      COMMON /ANCHOR/ ANCS, LTYP(12),
     +    DEPTH, OFFSET, SAFETY,
     +    ADIR(12), ATOP(12), ARAD(12),
     +    APRE(12), ANCX(12), ANCY(12)        ;ANCHOR PARAMETERS
      COMMON /SEMI/ RIGX(12), RIGY(12)        ;BARGE BOLLARDS
      COMMON /TORX/
     +    THETA, XRIG, YRIG, YAW,             ;RIG DISPLACEMENT
     +    NETMCW, NETFX, NETFY, NETSF,        ;MOORING CAPACITY
     +    XSPN(12), LOAD(12), SFAC(12),       ;LEG LOADING LIST
     +    TORQ(12), LEGX(12), LEGY(12),       ;MOMENT & FORCE LIST
     +    ACTV(12)                            ;ACTIVE LEG FLAGS
C
      DATA SLACK /.02/                        ;BACK TENSION/STRENGTH IN
C                                               SLACKED LEG
C *** COMPUTE FAIRLEAD TO ANCHOR DISTANCE:
C
      C = COS(YAW)
      S = SIN(YAW)
      XF = RIGX(LEG)*C + RIGY(LEG)*S          ;YAWED FAIRLEAD OFFSETS
      YF = RIGY(LEG)*C - RIGX(LEG)*S          ; RELATIVE TO RIG CENTER (0,0)
      XS = ANCX(LEG) - XRIG - XF              ;DISTANCE TO ANCHOR
      YS = ANCY(LEG) - YRIG - YF
C
C *** LOOKUP FORCE VS DISPLACEMENT
C
      XSPN(LEG) = SQRT(XS*XS + YS*YS)         ;FAIRLEAD TO ANCHOR RADIUS
      IF (ACTV(LEG)) THEN                     ;ACTIVE LEG
          CALL LOOKXH(LEG, XSPN(LEG),         ;GET LEG LOADS
```

```
      INTEGER*2 FUNCTION TRIES
     + (STEP, XCUR, XMAX, XMIN, YERR, TOL)  ;D.B.DILLON EG&G
C
C *** MANAGE NEWTON-RAPHSON ITERATION USING ONE-STEP ITERATION METHOD
C      GIVEN: STEP>0 - SET TRYS=STEP AND SET UP INITIAL DIFFERENTIATION
C     RETURN: STEP=0  ELSE
C             TRIES=ITERATION COUNT
C
      INTEGER*2 STEP                        ;ARGUMENTS: 1ST ITERATION FLAG
      REAL XCUR                             ;CURRENT ESTIMATE OF UNKNOWN
      REAL XMAX                             ;UPPER BOUND ON XCUR
      REAL XMIN                             ;LOWER BOUND
      REAL YERR                             ;CURRENT ERROR RESULT
      REAL TOL                              ;CONVERGENCE TOLERANCE
      INTEGER*2 TRY                         ;LOCAL: ITERATION COUNTER
      INTEGER*2 TRYS                        ;ITERATION LIMIT
      INTEGER*2 TRYD                        ;DISPLAY KICKIN
      REAL MU                               ;KICKIN RATIO
      REAL DOLD                             ;PRIOR CORRECTION ESTIMATE
      REAL DX                               ;CORRECTION ESTIMATE
      REAL XOLD                             ;PRIOR SOLUTION ESTIMATE
      REAL YOLD                             ;PRIOR SOLUTION ERROR
      REAL XTRY                             ;TEMPORARY: NEXT SOLUTION ESTIMATE
      INTEGER*2 CON, PTR, MSG, AUX, RIG     ;/UNITS/
C
      COMMON /UNITS/ CON, PTR, MSG, AUX, RIG,UNITS
C
      DATA MU /.2/                          ;SLOW CONVERGENCE KICKIN
C
C *** STEP 1: FORM INCREMENTAL DERIVATIVE FOR 1ST TRY
C
      IF (STEP .EQ. 0) GOTO 110             ;1ST CALL
      TRYS = STEP                           ;ITERATION LIMIT
      TRYD = MU*STEP                        ;DISPLAY SLOW CONVERGENCE
      TRY = -1                              ;START COUNTER
      DOLD = 0.                             ;SET PRIOR INCREMENT
      STEP = 0                              ;CLEAR 1ST STEP FLAG
  100 DX = TOL*XCUR                         ;BASE UPDATE ON ESTIMATE
      IF (DX .EQ. 0.) DX = 1.
      GOTO 300                              ;SET UP NEXT ITERATION
C
C *** STEP 1B: ELSE EVALUATE FOR SMOOTH CONVERGENCE
C
  110 IF (TRY .LT. 2) GOTO 200              ;DERIVATIVE INCREMENT
      IF (YOLD .EQ. YERR) GO TO 400         ;INSENSITIVE STEP
      IF (ABS(YERR) .LT. ABS(YOLD)) GOTO 200;NORMAL ERROR REDUCTION
C
C *** BAD STEP: ERROR INCREASED
C
      DX = -DX/2.                           ;BACKUP HALFWAY
      IF (DX) 210,100,210                   ;AND TRY AGAIN
C
C *** STEP 2: NORMAL STEP - USE PRIOR CORRECTION TO FORM NEXT DERIVATIVE
C
  200 DX = YERR*DOLD/(YOLD-YERR)            ;ONE STEP UPDATE
```

```
      IF (TRY .LT. 2) GOTO 210                   ;SKIP MIN/MAX CHECK
      IF ((DX .LT. 0.) .AND.
     +    (DOLD .LT. 0.)) XMAX = XOLD            ;TIGHTEN LIMITS
      IF ((DX .GT. 0.) .AND.
     +    (DOLD .GT. 0.)) XMIN = XOLD
  210 XTRY = XCUR + DX                           ;TRIAL UPDATE
      IF (XTRY.LT.XMIN) DX = (XMIN-XCUR)/2.      ;TOO SMALL
      IF (XTRY.GT.XMAX) DX = (XMAX-XCUR)/2.      ;TOO LARGE
C
C *** SET UP ANOTHER TRY
C
  300 TRY = TRY + 1                              ;COUNT IT
      IF (TRY .GT. TRYD) WRITE (CON, 1000)       ;DURING SECOND HALF
     + TRY, DX, XCUR, YERR, XMAX, XMIN           ;DISPLAY ITERATION PROGRESS
      DOLD = DX                                  ;PRIOR UPDATE
      XOLD = XCUR                                ;SAVE CURRENT ESTIMATE
      YOLD = YERR
      XCUR = XCUR + DX                           ;NEXT EXTIMATE
      IF (TRY .LE. TRYS) GOTO 320                ;OK TO TRY AGAIN
C
C *** UNCONVERGED
C
  310 WRITE (CON, 1001) TRY                      ;UNCONVERGED WARNING
      PAUSE
  320 TRIES = TRY                                ;RETURN ITERATION COUNT
      RETURN
C
C *** ERROR ON DIGITAL INSENSITIVITY OR SINGULARITY
C
  400 WRITE (CON, 1002) TRY,
     +    XCUR, XOLD, YERR, YOLD, DX, DOLD       ;WARNING
      TRYD = TRY - 1                             ;START DISPLAY
      GOTO 100                                   ;USE FIXED INCREMENT
C
 1000 FORMAT (
     + ' TRY  UPDATE        ',
     + ' CURRENT       ERROR         ',
     + ' MAXIMUM        MINIMUM'/
     + I4, 1P5E14.6)                             ;SECOND HALF DISPLAY
C
 1001 FORMAT (' TRIES@310:'/
     + ' NO CONVERGENCE AFTER',
     + I4, ' TRIES.  PRESS RETURN.')             ,UNCONVERGED WARNING
C
 1002 FORMAT (' TRIES@400:'/
     + ' INSENSITIVE OR SINGULAR ITERATION STEP'/
     + ' TRY   CURRENT     PRIOR      ',
     + '  ERROR        PRIOR ERR. ',
     + ' UPDATE       PRIOR UPDATE'/
     + I4, 1P6E12.4)                             ,SINGULARITY WARNING
C
      END
```

```
C$DEBUG
$STRICT                                         ;MODULE SHOW.FOR
C                                               ;SUBROUTINE SHOWXH
C                                               ;SUBROUTINE LISTXH
C                                               ;SUBROUTINE SHOWHS
C                                               ;SUBROUTINE LISTHS
C                                               ;SUBROUTINE SHOWHP
C
C     ---------------------------------------------------
C
      SUBROUTINE SHOWXH                         ;D.B.DILLON  EG&G  1985
C
C *** DISPLAY X VS H FILES FOR ALL DISTINCT LEG STYLES ON A RIG
C
      LOGICAL*2 MAKFIL                          ;FUNCTION
      EXTERNAL  MAKFIL
      INTEGER*2 ANC                             ;LOCALS: ANCHOR LEG NO.
      INTEGER*2 AT                              ;MASTER LEG INDEX
      CHARACTER XHFIL(6)
      INTEGER*2   MTYPS, MLEG, MTYP             ;/LEGMAP/
      INTEGER*2 ANCS, LTYP                      ;/ANCHOR/
      REAL DEPTH, OFFSET, SAFETY,
     +    ADIR, ATOP, ARAD, APRE, ANCX, ANCY
      INTEGER*2 CON, PTR, MSG, AUX, RIG         ;/UNITS/
C
      COMMON /UNITS/ CON, PTR, MSG, AUX, RIG;LOGICAL UNITS
      COMMON /ANCHOR/ ANCS, LTYP(12),
     +    DEPTH, OFFSET, SAFETY,
     +    ADIR(12), ATOP(12), ARAD(12),
     +    APRE(12), ANCX(12), ANCY(12)          ;ANCHOR PARAMETERS
      COMMON /LEGMAP/
     +    MTYPS,                                ;NO. MASTER TYPES
     +    MLEG(12),                             ,MASTER LEG FOR TYPE I
     +    MTYP(12)                              ;MASTER TYPE FOR LEG I
C
      DATA XHFIL /'X', 'H', '.', 'L', 2*'0'/,RIGNAMXH.LNN
C
C *** LOOP OVER MASTER LEG TYPES
C
      CALL PRTDEF (3)                           ;DISPLAY MOORING
      DO 100 AT=1,MTYPS                         ;STYLE LOOP
      ANC = MLEG(AT)                            ;MASTER LEG NO
      IF (MAKFIL(XHFIL(1), ANC, AUX))
     +    GOTO 100                              ;OPEN XVSH FILE
      CALL LISTXH (ANC, LTYP(ANC))              ;DISPLAY X VS H TABLE
      CLOSE (AUX)                               ;CLOSE X VS H FILE
  100 CONTINUE
      WRITE (PTR, 1000)                         ;PAGE EJECT
      RETURN                                    ;TASK COMPLETE
C
 1000 FORMAT ('1')                              ;PAGE EJECT
C
      END
```

```
      SUBROUTINE LISTXH (ANC, TYP)           ;D.B.DILLON  EG&G
C
C *** TABULATE THE FORCE VS DISPLACEMENT FUNCTION FOR
C     ANCHOR ANC OF LEG TYPE ANC FROM UNIT AUX
C
      INTEGER*2 ANC                          ;ARGUMENTS: ANCHOR NO.
      INTEGER*2 TYP                          ;LEG TYPE
      INTEGER*2 I                            ;LOCALS: RECORD COUNT
      INTEGER*2 N                            ;NODE INDEX
      INTEGER*2 NS                           ;SEGMENT COUNT
      INTEGER*2 NX                           ;NODE COUNT
      INTEGER*2 PAGE                         ;PAGE INDEX
      INTEGER*2 PAGLEN                       ;PAPER SIZE, LINES
      INTEGER*2 TABLES                       ;RECORDS PER PAGE
      INTEGER*2 C                            ;TAB TO CENTER NAME
      CHARACTER SPACE                        ;FOR CENTERING
      REAL X(6)                              ;NODE DISPLACEMENT
      REAL H                                 ;LOAD
      INTEGER*2 NOS, NOB, NSF                ;/XHTABL/
      REAL SPAN, SB, ST, FMIN, DX, DY,
     +     V, U, Y, SL, BL
      INTEGER*2 TYPS, SEGS, MAT              ;/CABLES/
      REAL DIA, BRK, LEN, WGT, EA, BNCY
      INTEGER*2 CON, PTR, MSG, AUX, RIG      ;/UNITS/
      CHARACTER RIGNAM, JOBNAM               ;/JOB/
      INTEGER*2 RNL, JNL                     ;/NAMLEN/
C
      COMMON /NAMLEN/ RNL, JNL               ;NAME LENGTHS
      COMMON /JOB/ RIGNAM(72), JOBNAM(72)    ;PROBLEM ID
      COMMON /UNITS/ CON, PTR, MSG, AUX, RIG;LOGICAL UNITS
      COMMON /CABLES/ TYPS, SEGS(12),
     +    MAT(5,12), DIA(5,12),
     +    BRK(5,12), LEN(5,12), WGT(5,12),
     +    EA(5,12), BNCY(5,12)               ;LEG SEGMENT PARAMETERS
      COMMON /XHTABL/
     +    NOS, NOB, NSF,
     +    SPAN, SB, ST,
     +    FMIN, DX(5), DY(5), V(5), U(5),
     +    Y(6), SL(5), BL(5)                 ;X VS H RECORD
C
      DATA SPACE /' '/
      DATA PAGLEN /66/                       ;11 IN. AT 6 LINES/IN.
      DATA PAGE /0/                          ;DEFAULT 1ST PAGE
C
C *** DISPLAY THE X VS H RECORDS
C
      C = 40 - RNL/2                         ;CENTER TAB FOR 80 COLUMNS
      NS = SEGS(TYP)                         ;DUMMY FOR ARRAY
      NX = NS + 1                            ;TOTAL NODES
      X(1) = 0.                              ;ORIGIN AT
      Y(1) = 0.                              ; RIG FARILEAD
      IF (TYP .LT. 2) PAGE = 0               ;START PAGE COUNTER
      TABLES = (PAGLEN-2)/(9+NS)             ;WHOLE RECORDS/PAGE
      I = 0                                  ;NO RECORDS READ
C
```

```
C *** READ AND DISPLAY AN XVSH RECORD
C
  100 READ (AUX, END=300, ERR=310)
     +         H, SPAN, SB, ST, FMIN,
     +              I, NOS, NOB, NSF,
     +           (DX(N), DY(N), V(N),
     +           U(N), SL(N), N=1,5)        ;RETRIEVE X VS H
C
      DO 110 N=1,NS                         ;LOOP OVER SEGMENTS
      X(N+1) = X(N) + DX(N)                 ;ACCUMULATE SPAN
      Y(N+1) = Y(N) - DY(N)                 ; AND DEPTH
  110 CONTINUE
C
      IF (MOD(I,TABLES) .NE. 1) GOTO 120    ;SAME PAGE
      PAGE = PAGE + 1                       ;PAGE NUMBER
      WRITE (PTR, 1000)
     + (SPACE, N=1,C), (RIGNAM(N), N=1,RNL) ;NEW PAGE
      WRITE (PTR, 1005) ANC, TYP, PAGE      ; AND HEADER
C
  120 WRITE (PTR, 1001)
     +          H, SPAN, SB, ST, FMIN,
     +                  NSF, NOS, NOB,
     +   (N, X(N), Y(N), DX(N), DY(N),
     +     SL(N), V(N), U(N), N=1,NS),
     +               NX, X(NX), Y(NX)       ;PRINT XVSH RECORD
      WRITE (CON, 1001)                     ;V.1.01
     +          H, SPAN, SB, ST, FMIN,      ;V.1.01
     +                  NSF, NOS, NOB,      ;V.1.01
     +   (N, X(N), Y(N), DX(N), DY(N),      ;V.1.01
     +     SL(N), V(N), U(N), N=1,NS),      ;V.1.01
     +               NX, X(NX), Y(NX)       ;V.1.01 DISPLAY TOO
C
      GOTO 100                              ;NEXT RECORD
      RETURN                                ;LEG DONE
C
C *** READ ERROR RECOVERY
C
  300 IF (I .EQ. 0) GOTO 320                ;NON-EXISTENT FILE
      WRITE (CON, 1002) I                   ;END OF FILE
      GOTO 330
C
  310 IF (I .EQ. 0) GOTO 320                ;NON-EXISTENT FILE
      WRITE (CON, 1003) I                   ;DAMAGED FILE
      GOTO 330
C
  320 WRITE (CON, 1004)                     ;NO SUCH FILE
  330 CALL DELAY (1)
      RETURN
C
 1000 FORMAT ('1', 80A1)                    ;NEW PAGE
C
 1005 FORMAT (' Anchor', I3,
     + ' X vs H for Leg Type', I3,
     +  40X'Page', I3)                      ;PAGE HEADER
C
```

```
 1001 FORMAT (//
     + ´  Horizontal Horizontal Length on ´,
     + ´ Stretched  Safety   at   Nodes on´,
     + ´   Nodes on´/
     + ´     Load        Span      Bottom   ´,
     + ´  Length    Factor  Node   Surface´,
     + ´    Bottom´/
     +  4F11.1, F8.3, 3XI2, 6XI2, 9XI2//
     + ´            Node Position     ´,
     + ´ Segment Increment Stretched ´,
     + ´ Segment Downward Force´/
     + ´ Node    Span      Depth     Span ´,
     + ´     Depth    Length      Rig end´,
     + ´ Anchor end´,
     +  6(/I4, 1X5F10.2, 2F11.1))           ;XVSH RECORD DISPLAY
C
 1002 FORMAT (//
     + ´ End of file after record´, I3/)    ;END OF FILE
C
 1003 FORMAT (//´ LISTXH@310:´/
     + ´ Read error after record´, I3/)     ;DAMAGED FILE
C
 1004 FORMAT (//´ LISTXH@320·´/
     + ´ X vs H file is empty´/)            ;NEW FILE
C
      END
```

```
      SUBROUTINE SHOWHS                        ;D.B.DILLON  EG&G
C
C *** PRINT H VS S FILES FOR ALL DISTINCT LEG STYLES ON A RIG
C
      LOGICAL*2 MAKFIL                         ;FUNCTION
      EXTERNAL  MAKFIL
      CHARACTER HSFIL(6)                       ;LOCAL
      INTEGER*2 TYP
      INTEGER*2 TYPS, SEGS, MAT                ;/CABLES/
      REAL DIA, BRK, LEN, WGT, EA, BNCY
      INTEGER*2 CON, PTR, MSG, AUX, RIG        ;/UNITS/
C
      COMMON /UNITS/ CON, PTR, MSG, AUX, RIG,LOGICAL UNITS
      COMMON /CABLES/ TYPS, SEGS(12),
     +    MAT(5,12), DIA(5,12),
     +    BRK(5,12), LEN(5,12), WGT(5,12),
     +    EA(5,12), BNCY(5,12)                 ;LEG SEGMENT PARAMETERS
C
      DATA HSFIL /'H', 'S', '.', 'L', 2*'0'/;RIGNAMHS.LNN
C
C *** TABULATE THE FORCE VS SCOPE FUNCTION FOR EACH MOORING LEG TYPE
C
      CALL PRTDEF (3)                          ;DISPLAY MOORING
      DO 200 TYP=1,TYPS                        ;STYLE LOOP
      IF (MAKFIL (HSFIL(1), TYP, AUX))
     +    GOTO 200                             ;OPEN H VS S FILE
      CALL LISTHS(TYP)                         ;COMPUTE H VS S TABLE
      CLOSE (AUX)                              ;CLOSE H VS S FILE
  200 CONTINUE
      WRITE (PTR, 1000)                        ;EJECT LAST PAGE
      RETURN
C
 1000 FORMAT ('1')                             ;PAGE EJECT
C
      END
```

```
      SUBROUTINE LISTHS (TYP)                   ;MAY-85 D.B.DILLON  EG&G
C
C *** PRINT LOAD AND SPAN CONDITIONS VS TOP ELEMENT LENGTH
C     GIVEN: H VS S FILE OPENED AND TYP=LEG TYPE SUBSCRIPT
C
      INTEGER*2 TYP                             ;ARGUMENT: LEG TYPE
      INTEGER*2 C                               ;CENTERING TAB
      INTEGER*2 RC                              ;RECORD COUNTER
      INTEGER*2 PSIZ                            ;RECORDS/PAGE
      INTEGER*2 PAGE                            ;PAGE NO.
      INTEGER*2 I                               ;INDEX
      CHARACTER SPACE                           ;TAB FILL
      REAL SCOP                                 ;/HSTABL/: TOP SCOPE
      REAL XMIN, SBPL, SBDL                     ;BOTTOMED LENGTHS
      REAL XPRE, XDES                           ;PRE- & DESIGN SPANS
      REAL HPRE, HDES, HDIF                     ;PRE-& DESIGN LOAD, POWER
      INTEGER*2 TYPS, SEGS, MAT                 ;/CABLES/
      REAL DIA, BRK, LEN, WGT, EA, BNCY
      INTEGER*2 CON, PTR, MSG, AUX, RIG         ;/UNITS/
      CHARACTER RIGNAM, JOBNAM                  ;/JOB/
      INTEGER*2 RNL, JNL                        ;/NAMLEN/
C
      COMMON /NAMLEN/ RNL, JNL                  ;NAME LENGTHS
      COMMON /JOB/ RIGNAM(72), JOBNAM(72)       ;PROBLEM ID
      COMMON /UNITS/ CON, PTR, MSG, AUX, RIG,LOGICAL UNITS
      COMMON /CABLES/ TYPS, SEGS(12),
     +    MAT(5,12), DIA(5,12),
     +    BRK(5,12), LEN(5,12), WGT(5,12),
     +    EA(5,12), BNCY(5,12)                  ;LEG SEGMENT PARAMETERS
      COMMON /HSTABL/ SCOP, XMIN, SBPL, SBDL,
     +    XPRE, XDES, HPRE, HDES, HDIF          ;H VS S RECORD
C
      DATA SPACE /' '/
      DATA PSIZ  /50/
C
C *** TABLE HEADER
C
      PAGE = 1                                  ;START COUNTER
      C = 40 - RNL/2                            ;CENTER TAB
      WRITE (CON, 1000) PAGE,
     + (SPACE, I=1,C), (RIGNAM(I), I=1,RNL)     ;NEW PAGE
      WRITE (CON, 1001) TYP                     ,SCREEN TABLE HEADER
      RC = 0                                    ;START RECORD COUNTER
C
C *** SCAN FILE
C
  100 READ (AUX, END=200, ERR=210)
     +    SCOP, XMIN, SBPL, SBDL,
     +    XPRE, XDES, HPRE, HDES, HDIF          ;GET CASE
      IF (MOD(RC,PSIZ)) 120,110,120             ;NEW PAGE?
C
  110 WRITE (PTR, 1000) PAGE,                   ;YES:
     + (SPACE, I=1,C), (RIGNAM(I), I=1,RNL)     ;NEW PAGE
      WRITE (PTR, 1001) TYP                     ,PRINT TABLE HEADER
      PAGE = PAGE + 1
```

```
C
  120 RC = RC + 1                                ;COUNT RECORD
      WRITE (CON, 1002)
     +    SCOP, XMIN, SBPL, SBDL,
     +    XPRE, XDES, HPRE, HDES, HDIF           ;DISPLAY CASE
      WRITE (PTR, 1002)
     +    SCOP, XMIN, SBPL, SBDL,
     +    XPRE, XDES, HPRE, HDES, HDIF           ;PRINT CASE
      GOTO 100                                   ;NEXT SCOPE
C
C *** READ ERROR RECOVERY
C
  200 IF (RC .EQ. 0) GOTO 220                    ;NON-EXISTENT FILE
      WRITE (CON, 1003) RC                       ;END OF FILE
      WRITE (PTR, 1006)                          ;FOOTNOTE
      GOTO 230
C
  210 IF (RC .EQ. 0) GOTO 220                    ;NON-EXISTENT FILE
      WRITE (CON, 1004) RC                       ;DAMAGED FILE
      GOTO 230
C
  220 WRITE (CON, 1005)                          ;NO SUCH FILE
  230 CALL DELAY(1)                              ;TIMED PAUSE
      RETURN
C
 1000 FORMAT (´1´, 70X´Page´, I6 / 80A1)        ;NEW PAGE
C
 1001 FORMAT (//
     + 29X´H vs S for Leg Type´, I3//
     + ´     Top  - Force/Length on Bottom -´,
     + ´     Pre-    Design   Pre-´,
     + ´     Design  Holding´/
     + ´    Scope    Slack  Preload   Design´,
     + ´     Span     Span     Load´,
     + ´      Load    Power´)                    ;DISPLAY HEADER
C
 1002 FORMAT (F8.0, 8F9.0)                       ;DISPLAY LINE
C
 1003 FORMAT (//
     + ´ End of file after record´, I3/)         ,END OF FILE
C
 1004 FORMAT (//´ LISTHS@210:´/
     + ´ Read error after record´, I3/)          ;DAMAGED FILE
C
 1005 FORMAT (//´ LISTHS@220:´/
     + ´ Empty H vs S file´/)                    ;NEW FILE
C
 1006 FORMAT (/´ Force/Length on Bottom:´/
     + ´ Positive values are upward force´,
     + ´ on anchor,´/
     + ´ Negative values are cable length´,
     + ´ on bottom.´)                            ;FOOTNOTE
      END
```

```
      SUBROUTINE SHOWHP
C
C *** DISPLAY HOLDING POWER ROSE FILE FOR A RIG
C
      LOGICAL*2 USRINP, GETINT, MAKROS       ;FUNCTION
      EXTERNAL  USRINP, GETINT, MAKROS
      INTEGER*2 LU(2), LN                    ;LOCAL: ROSE FILE UNIT NO´S.
      INTEGER*2 IT1, IT2                     ;TAB SETTINGS
      INTEGER*2 LINE, PAGE                   ;PAGINATION INDEXES
      INTEGER*2 R                            ;ROSE INDEX
      REAL HEAVE                             ;RIG HEAVE (FILE SUFFIX)
      REAL HOLD                              ;HOLDING POWER
      REAL DIR                               ;WEATHER DIRECTION
      CHARACTER*12 ROSNAM(2)                 ;SUBTITLES
      CHARACTER SPC                          ;SPACE FOR TABS
      LOGICAL*2 ATCV                         ;/TORX/
      REAL THETA, XRIG, YRIG, YAW,
     +     NETMCW, NETFX, NETFY, NETSF,
     +     XSPN, LOAD, SFAC,
     +     TORQ, LEGX, LEGY
      INTEGER*2 ANCS, LTYP                   ;/ANCHOR/
      REAL DEPTH, OFFSET, SAFETY,
     +    ADIR, ATOP, ARAD, APRE, ANCX, ANCY
      INTEGER*2 CON, PTR, MSG, AUX, RIG      ;/UNITS/
      INTEGER*2 LTXT, LPTR                   ;/USRPTR/
      CHARACTER TEXT                         ;/USRTXT/
      CHARACTER RIGNAM, JOBNAM               ;/JOB/
      INTEGER*2 RNL, JNL                     ;/NAMLEN/
C
      COMMON /NAMLEN/ RNL, JNL               ;NAME LENGTHS
      COMMON /JOB/ RIGNAM(72), JOBNAM(72)    ;PROBLEM ID
      COMMON /USRPTR/ LTXT, LPTR
      COMMON /USRTXT/ TEXT(80)               ;ENTRY BUFFER
      COMMON /UNITS/ CON, PTR, MSG, AUX, RIG;LOGICAL UNITS
      COMMON /ANCHOR/ ANCS, LTYP(12),
     +    DEPTH, OFFSET, SAFETY,
     +    ADIR(12), ATOP(12), ARAD(12),
     +    APRE(12), ANCX(12), ANCY(12)       ;ANCHOR PARAMETERS
      COMMON /TORX/
     +    THETA, XRIG, YRIG, YAW,            ;RIG DISPLACEMENT
     +    NETMCW, NETFX, NETFY, NETSF,       ;MOORING CAPACITY
     +    XSPN(12), LOAD(12), SFAC(12),      ;LEG LOADING LIST
     +    TORQ(12), LEGX(12), LEGY(12),      ,MOMENT & FORCE LIST
     +    ATCV(12)                           ;ATCVE LEG FLAGS
C
      DATA ROSNAM /´ Operational´,
     +            ´  Survival  ´/
      DATA SPC /´ ´/
C
C *** PREPARE ROSE FILES FOR DISPLAY
C
      IT1 = 3
      CALL PRTDEF (IT1)                      ;DISPLAY DEFINITION
      LU(1) = MAXO(CON,PTR,MSG,AUX,RIG) + 1 ;SET ROSE FILE UNITS
      LU(2) = LU(1) + 1
```

```
      HEAVE = 0.
   10 IF (MAKROS (LU(1),LU(2),HEAVE)) RETURN;OPEN ERROR
      IT1 = 40 - RNL/2                      ;RIG NAME TAB
      IT2 = 40 - JNL/2                      ;JOB TITLE TAB
      PAGE = 0
C
C *** OPERATIONAL AND SURVIVAL ROSE TABLES
C
      DO 400 R=1,2                          ;DO 2 ROSE FILES
      LN = LU(R)                            ;SET LOGICAL UNIT NO.
      LINE = 99                             ;FORCE NEW PAGE
  100 IF (LINE+ANCS .LT. 56) GOTO 200       ;SAME PAGE?
C
C *** START NEW PAGE
C
      PAGE = PAGE + 1                       ;NEXT PAGE
      WRITE (CON, 1000) PAGE, (SPC, I=1,IT1),
     +   (RIGNAM(I), I=1,RNL)               ;HEADER 1
      WRITE (CON, 1001) (SPC, I=1,IT2),
     +   (JOBNAM(I), I=1,JNL)               ;HEADER 2
      WRITE (CON, 1002) ROSNAM(R), HEAVE    ;HEADER 3
C
      WRITE (PTR, 1004)                     ;PAPER EJECT
      WRITE (PTR, 1000) PAGE, (SPC, I=1,IT1),
     +   (RIGNAM(I), I=1,RNL)               ,HEADER 1
      WRITE (PTR, 1001) (SPC, I=1,IT2),
     +   (JOBNAM(I), I=1,JNL)               ;HEADER 2
      WRITE (PTR, 1002) ROSNAM(R), HEAVE    ;HEADER 3
      LINE = 4                              ;COUNT HEADER LINES
C
C *** DISPLAY A ROSE RECORD
C
  200 READ (LN, END=300, ERR=300)
     +   THETA,    HOLD,     DIR,    NETFX,
     +   NETFY,   NETSF,     YAW,   NETMCW, (
     + LOAD(L), XSPN(L), LEGX(L), LEGY(L),
     + SFAC(L), TORQ(L), ATCV(L), L=1,ANCS) ;GET RECORD
C
      WRITE (CON, 1003)
     +   THETA,    HOLD,     DIR,    NETFX,
     +   NETFY,   NETSF,  NETMCW,      YAW, (L,
     + LOAD(L), XSPN(L), LEGX(L), LEGY(L),
     + SFAC(L), TORQ(L), ATCV(L), L=1,ANCS) ;DISPLAY RECORD
C
      WRITE (PTR, 1003)
     +   THETA,    HOLD,     DIR,    NETFX,
     +   NETFY,   NETSF,  NETMCW,      YAW, (L,
     + LOAD(L), XSPN(L), LEGX(L), LEGY(L),
     + SFAC(L), TORQ(L), ATCV(L), L=1,ANCS) ;PRINT RECORD
C
      LINE = LINE + 8 + ANCS
      GOTO 100                              ;NEXT RECORD
C
  300 CLOSE (LN)
  400 CONTINUE                              ;NEXT ROSE
```

```
  410 WRITE (CON, 1005)                       ;PROMPT FOR HEAVED RIG
      IF (USRINP(26)) GOTO 410                ;RETRY AFTER HELP REQUEST
      IF (GETINT(R)) GOTO 410                 ;RETRY AFTER ENTRY ERROR
      IF (IABS(R) .GT. 99) GOTO 410           ;RETRY AFTER OUT OF RANGE
      IF (R .EQ. 0) THEN
           WRITE (PTR, 1004)                  ;EJECT PAGE
           RETURN                             ;USER QUITS
      ELSE
           HEAVE = R
           GOTO 10                            ;NEXT HEAVE CASE
      ENDIF
C
 1000 FORMAT (70X'Page', I2/80A1)             ;HEADER 1
C
 1001 FORMAT (80A1)                           ;HEADER 2
C
 1002 FORMAT (13XA12,
     + ' Holding Power Rose for', F5.0,
     + ' Ft Rig Heave')                       ;HEADER 3
C
 1003 FORMAT (//
     + '  Deflection  Holding  Weather ',
     + '   Force Components',
     + '  Safety  CW Moment     Yaw'/
     + '   Direction   Power  Direction',
     + '     X          Y    ',
     + '  Factor (Normalized)  Angle'/
     + F11.2, F11.0, F9.3, 2F10.0,
     +   F7.3, 2F10.4//
     + '   Anchor          Load       Span',
     + '   X-Load    Y-Load',
     + '  Safety CW Moment  Active'/
     + (I7, 5XF10.0, F9.1, 2F10.0, F7.3,
     +          F10.4, 5XL1))                 ;RECORD IMAGE
C
 1004 FORMAT ('1')                            ;PAGE EJECT
C
 1005 FORMAT (/
     + '     Rig Heave'/
     + '       Downward: -1 thru -99 Feet'/
     + '         Upward:  1 thru  99 Feet'/
     + ' Command Menu:  0')                   ;HEAVE PROMPT
C
      END
```

```
C$DEBUG
$STRICT                                           ;MODULE VERT.FOR
C                                                 ;SUBROUTINE GETXVH
C                                                 ;SUBROUTINE SURFACE
C                                                 ;LOGICAL*2 FUNCTION VVSH
C                                                 ;REAL FUNCTION DELY
C                                                 ;REAL FUNCTION DELX
C                                                 ;REAL FUNCTION STPSIZ
C
C     ---------------------------------------------------------
C
      SUBROUTINE GETXVH (NX,D,H,W,L,E,B,T)  ;D.B.DILLON  EG&G  1985
C
C *** GIVEN ARGUMENT LIST
C     ITERATE VERTICAL FORCE IN A LEG UNTIL ALL BUOY/CLUMPS/ANCHOR ARE
C     AT OR BETWEEN THE SURFACE/BOTTOM BOUNDARIES,
C *** RETURN X VS H TABLE AND PARAMETERS IN COMMON /XHTABL/
C
C *** RESTRICTION: THERE MAY NOT BE A BUOY ON THE SURFACE AT A NODE
C                  BETWEEN A CLUMP ON THE BOTTOM AND THE ANCHOR. THIS
C                  IS EQUIVALENT TO SAYING THAT THERE MAY NOT BE A
C                  CLUMP ON THE BOTTOM AT A NODE BETWEEN A BUOY ON THE
C                  SURFACE AND THE MOORED HULL.
C
C *** NOTE: NODES ARE COUNTED FROM 1=RIG NODE TOWARDS THE ANCHOR NODE
C           ELEMENT 1 EXTENDS FROM NODE 1 TOWARDS THE ANCHOR NODE.
C           THE Y-AXIS IS POSITIVE UPWARDS, SO DY>0 DECREASES DEPTH.
C
      INTEGER*2 NX                                ;ARGUMENTS: SEGMENTS IN LEG
      REAL D                                      ,WATER DEPTH
      REAL H                                      ;HORIZONTAL LOAD
      REAL W(NX)                                  ;SEGMENT WEIGHT LIST
      REAL L(NX)                                  ;SEGMENT LENGTHS
      REAL E(NX)                                  ;SEGMENT ELASTICITY
      REAL B(NX)                                  ;NODE BUOYANCY
      REAL T(NX)                                  ;SEGMENT STRENGTH
      INTEGER*2 NOS, NOB, NSF                     ;/XHTABL/
      REAL SPAN, SB, ST, FMIN, DX, DY,
     +     V, U, Y, SL, BL
      INTEGER*2 CON, PTR, MSG, AUX, RIG           ;/UNITS/
C
      COMMON /UNITS/ CON, PTR, MSG, AUX, RIG;LOGICAL UNITS
      COMMON /XHTABL/
     +    NOS, NOB, NSF,
     +    SPAN, SB, ST,
     +    FMIN, DX(5), DY(5), V(5), U(5),
     +    Y(6), SL(5), BL(5)                      ,X VS H RECORD
C
      DATA TOL /1.E-4/                            ;TOLERANCE LIMIT
C
C *** NULL ARRAYS BY SEGMENT
C
      DO 90 N=1,5
      DX(N) = 0.                                  ;SPAN
      DY(N) = 0.                                  ;DEPTH
```

```
      V(N) = 0.                                  ;TOP-END VERTICAL LOAD
      U(N) = 0.                                  ;BOTTOM-END "
      Y(N) = 0.                                  ;NODE DEPTH
      SL(N) = 0.                                 ;STRETCHED LENGTH
      BL(N) = 0.                                 ;LENGTH ON BOTTOM
   90 CONTINUE
      G = TOL*D                                  ;DEPTH TOLERANCE
C
C *** FIND THE NODE NEAREST THE ANCHOR THAT HAS A BUOY, AND SET THE
C     VERTICAL FORCE UPDATE LIMIT TO 20% OF LEAST TENSILE STRENGTH
C
      Y(1) = D                                   ;HULL IS ON SURFACE
      Y(6) = 0.                                  ;ANCHOR IS ON BOTTOM
      NBY = 1                                    ;MOORED HULL IS BUOYANT
      F = T(1)/5.                                ;START V-UPDATE LIMIT
      IF (NX .EQ. 1) GOTO 110                    ;SKIP SCAN FOR 1 SEGMENT
      DO 100 N=2,NX                              ;SCAN ALL NODES
      IF (B(N) .GT. 0.) NBY=N                    ;MOVE BUOY NODE MARK
      F = AMIN1(F,T(N)/5.)                       ;LIMIT 20% LEAST STRENGTH
  100 CONTINUE
C
C *** FIND VERTICAL FORCE EQUILIBRIUM.
C
  110 IF (H .EQ. 0.) V(1) = W(1)*D               ;INITIAL ESTIMATE
      NB=NX + 1                                  ;MARK ANCHOR NODE
      CALL SURFAC (NX, NBY, H, W, L, E, B, F, G, V, U, Y, DY)
C
C *** SOLUTION COMPLETE: FINAL SCAN TO COMPUTE SPAN OF EACH ELEMENT
C
  300 SPAN = 0.                                  ;START SPAN
      SB = 0.                                    ;MATERIAL LENGTH ON BOTTOM
      ST = 0.                                    ;STRETCHED LENGTH
      NOS = 0                                    ;NODES ON SURFACE COUNTER
      NOB = 1                                    ;NODES ON BOTTOM INCL ANCH.
      FMIN = 1.E26                               ;MIN. SAFETY FACTOR
      DO 310 N=1,NX                              ;SCAN ALL ELEMENTS
      IF (ABS(Y(N)-D) .LT. G)     NOS=NOS+1      ;COUNT NODES ON SURFACE
      IF (ABS(Y(N)-Y(NB)) .LT. G) NOB=NOB+1      ; AND  NODES ON BOTTOM
      DX(N) = DELX(H, Y(NB),
     +     W(N), L(N), E(N), T(N), G, FS,
     +     V(N), U(N), Y(N), BL(N), SL(N))       ;HORIZONTAL SEGMENT OFFSET
      SPAN = SPAN + DX(N)                        ;SUM SPAN, X
      ST = ST + SL(N)                            ;SUM STRETCHED LENGTH
      SB = SB + BL(N)                            ;SUM BOTTOMED SCOPE
      IF (FMIN .LT. FS) GOTO 310                 ;NEW MINIMUM?
      NSF = N                                    ;YES, NOTE NODE
      FMIN = FS                                  ;AND SAFETY FACTOR
  310 CONTINUE
C
      RETURN
C
C *** INVALID CASE: BUOY BETWEEN CLUMP AND ANCHOR
C
  400 WRITE (CON, 1002) N, NBY
      RETURN
```

```
C
 1002 FORMAT (/
     + ' BUOY AT NODE', I3,
     + ' BETWEEN CLUMP AT NODE', I3,
     + ' AND ANCHOR NODE.'/)                      ;ERROR TERMINATION
C
      END
```

```
      SUBROUTINE SURFAC (NS, NB,             ;D.B.DILLON  EG&G  1985
     +     H, W, L, E, B, F, G, V, U, Y, DY)
C
C *** ITERATE VERTICAL FORCES FOR SURFACE BUOYS
C     METHOD: MOVE HIGHEST BUOY ABOVE SURFACE TO SURFACE
C GIVEN:
      INTEGER*2 NS                           ;NUMBER OF SEGMENTS IN LEG
      INTEGER*2 NB                           ;LARGEST BUOYANT NODE NO.
      REAL H                                 ;HORIZONTAL LOAD IN LEG
      REAL W(NS)                             ;SEGMENT WEIGHT LIST
      REAL L(NS)                             ;SEGMENT LENGTH LIST
      REAL E(NS)                             ;SEGMENT ELASTICITY LIST
      REAL B(NS)                             ;NODE BUOYANCY LIST
      REAL F                                 ;V-UPDATE LIMIT
      REAL G                                 ;LENGTH TOLERANCE
C RETURN:
      REAL V(NS)                             ;VERTICAL LOADS, RIG END
      REAL U(NS)                             ;VERTICAL LOADS, ANCHOR END
      REAL Y(NS)                             ;NODE POSITIONS, RIG END
      REAL DY(NS)                            ;SEGMENT DEPTH SPAN LIST
C
      LOGICAL*2 VVSH                         ;FUNCTION
      INTEGER*2 MAP(5)                       ;LOCALS: HIGHEST BUOY MAP
      INTEGER*2 J, JS, JB                    ;NODE INDEX/SURFACE/BOTTOM
      INTEGER*2 K                            ;SURFACED BUOY MAP INDEX
      INTEGER*2 M, M1, M2                    ;SUB-LEG NODE INDEX, LIMITS
      INTEGER*2       CON, PTR, MSG, AUX, RIG;/UNITS/
C
      COMMON /UNITS/ CON, PTR, MSG, AUX, RIG
C
      DATA MAP /5*0/                         ;CLEAR MAP
C
C *** GET SOLUTION WITHOUT REGARD TO BOUNDARIES BETWEEN JS AND JB
C
      JS = 1                                 ;SURFACE NODE AT RIG
      JB = NS + 1                            ;BOTTOM NODE AT ANCHOR
      K = 1                                  ;1ST MAP LEVEL INCLUDES
      MAP(1) = JB                            ; THE ENTIRE LEG
  100 J = JS
      IF (VVSH(H, JB-J, NB-J+1,
     + W(J), L(J), E(J), B(J), F, G,
     + V(J), U(J), Y(J), DY(J)))
     + WRITE (CON, 1000)                     ;FIND V(JS) FOR YC=Y(JB)
      IF (NB .LT. 2) RETURN                  ;NO INTERSEGMENT BUOYS
C
C *** FIND_BUOY HIGHEST ABOVE SURFACE, IF ANY
C
      M1 = JS + 1                            ;DEFINE SCAN LIMITS
      M2 = MIN0(JB-1,NB)                     ;LIMIT SCAN
      IF (M1 .GT. M2) GOTO 200               ;NO BUOYS IN SUB-LEG
      DO 110 M=M1,M2
      IF (Y(M) .GE. Y(J)) J = M              ;FLAG HIGHER NODE
  110 CONTINUE
      IF (J .EQ. JS) GOTO 200                ;NO NODES ABOVE SURFACE
C
```

```
C *** MOVE HIGHEST BUOY TO SURFACE AND MAP
C
      K = K + 1                              ;MAP ENTRY
      MAP(K) = J                             ;HIGHEST NODE
      Y(J) = Y(1)                            ;MOVE IT TO SURFACE
      JB = J                                 ;NEW "ANCHOR" END
      GOTO 100
C
C *** WORK BACK THRU MAP, IF ANY
C
  200 IF (K .EQ. 0) RETURN                   ;NONE WERE ABOVE SURFACE
      JS = MAP(K)                            ;NEW "RIG" END
      K = K - 1                              ;WORK BACK
      JB = MAP(K)                            ;NEW "ANCHOR" END
      IF (JS .GT. 1) V(JS) = V(JS)-.9*B(JS) ;REDUCE BUOYANCY ESTIMATE
      IF (K .GT. 0) GOTO 100                 ;SOLVE THIS SUB-LEG
C
      RETURN                                 ;ALL DONE
C
 1000 FORMAT (' SURFACE: VVSH ERROR')
C
C SURFACE IS AN ALGORITHM THAT IMPOSES THE SURFACE BOUNDARY LIMIT ON
C LINE BUOYS. THE PROCEDURE WORKS BY IDENTIFYING SUB PORTIONS OF THE
C MOORING LEG WHOSE END POSITIONS ARE FIXED, STARTING WITH THE WHOLE
C LEG, WHICH HAS ITS "RIG" END ON THE SURFACE AND ITS "ANCHOR" END
C ON THE BOTTOM.
C
C SUBROUTINES VVSH AND DELY ADJUST FOR INTERSEGMENT WEIGHTS THAT FALL
C ON OR BELOW THE BOTTOM, BUT IMPOSE NO RESTRICTIONS ON THE POSITION
C OF BUOYS. SURFACE DETECTS BUOYS ABOVE THE SURFACE, AND SELECTS THE
C HIGHEST ONE, WHICH IT MOVES TO THE SURFACE. THE NODE NUMBER IS RE-
C CORDED IN A MAP LIST. VVSH IS USED AGAIN TO RESOLVE THE VERTICAL
C FORCE BALANCE BETWEEN THE "RIG" END AND THE MAPPED NODE.
C
C THE SEARCH FOR BUOYS ABOVE THE SURFACE IS REPEATED AND THE HIGHEST
C SUCH BUOY MAPPED UNTIL NO FURTHER BOUYS ARE FOUND. THIS GIVES THE
C FINAL SOLUTION BETWEEN THE "RIG" END AND THE LATEST MAPPED NODE.
C
C THEN THE PROCEDURE WORKS BACK UP THROUGH THE MAP LIST, RESOLVING THE
C VERTICAL FORCES BETWEEN SURFACE NODES IN THE MAP LIST. THESE SUBLEGS
C ARE TREATED AS THE WHOLE LEG WAS TREATED, SO THAT IF BUOYS ARE ABOVE
C THE SURFACE THE MAP LIST IS EXTENDED, AND SO ON.
C
      END
```

```
      LOGICAL*2 FUNCTION VVSH (H, NS, M,     ;4-APR-85  D.B.DILLON  EG&G
     + W, L, E, B, F, G, V, U, Y, DY)
C
C *** GIVEN H>=0: HORIZONTAL FORCE ACTING ON LEG,
C                 POSITIVE IN POSITIVE SENSE OF X-AXIS
      INTEGER*2 NS                           ;>0: NO. SEGMENTS IN LEG
      INTEGER*2 M                            ;BUOY/CLUMP NODE LIMIT
      REAL W(NS)                             ;WEIGHT DENSITY LIST
      REAL L(NS)                             ;SEGMENT LENGTH LIST
      REAL E(NS)                             ;ELASTICITY LIST, 0=INELAS.
C                                            ;>0: YOUNG'S MOD.*MET.XSEC.
      REAL B(NS)                             ;BUOY/CLUMP LIST
      REAL F                                 ;VERTICAL LOAD UPDATE LIMIT
      REAL G                                 ;CONVERGENCE TOLERANCE
      REAL V(NS)                             ;V(1)=ESTIMATE VERTICAL LOAD
C ** RETURN:
      REAL U(NS)                             ;VERTICAL LOAD ON LOWER END
      REAL Y(NS)                             ;POSITION OF NODE I
      REAL DY(NS)                            ;VERTICAL SPAN LIST
C ** ALSO:  V(I)  COMPUTED VERTICAL FORCE ON SEGMENT I AT Y(I) END
C    RETURN VVSH  .FALSE. IF THE CORRECTION IS SUCCESSFUL
C                 .TRUE.  IF ERRORS OCCUR
C
      LOGICAL*2 VFX                          ;LOCAL
      INTEGER*2 I                            ;NODE INDEX
      INTEGER*2 LAST                         ;LAST NODE
      INTEGER*2 TRY                          ;ITERATION COUNT
      INTEGER*2 TRYS                         ;ITERATION LIMIT
      INTEGER*2 TRYD                         ;DISPLAY TRIGGER
      REAL DV                                ;FORCE UPDATE
      REAL DVI                               ;INITIAL UPDATE
      REAL DVO                               ;PRIOR UPDATE
      REAL EY                                ;DEPTH ERROR
      REAL EYO                               ;PRIOR ERROR
      REAL VMAX                              ;FORCE LIMIT
      REAL VMIN                              ;DITTO
      REAL VT                                ;TRIAL FORCE
      REAL VI                                ;INITIAL ESTIMATE
      REAL YW                                ;TOTAL DEPTH SPAN
      INTEGER*2 CON, PTR, MSG, AUX, RIG      ;/UNITS/
C
      COMMON /UNITS/ CON, PTR, MSG, AUX, RIG
C
      DATA TRYS, TRYD /90, 75/
C
C *** SET UP INITIAL TRY
C
      VI = V(1)                              ;PRESERVE INITIAL ESTIMATE
      DVO = G/Y(1)                           ;DUMMY FOR TOL
      DVI = AMAX1(DVO*H,G*W(NS),DVO*ABS(VI)) ;INITIAL FORCE UPDATE
      VVSH = .FALSE.                         ;ASSUME CONVERGENCE
      LAST = NS + 1                          ;LAST NODE
   90 TRY = 0                                ;ITERATION COUNTER
      IF (TRYD .EQ. 0) WRITE (PTR, 8005)     ;#####
     + H, F, G, DVI, NS, M,                  ;#####
```

```
     + (W(I), L(I), E(I), B(I),                  #####
     +  V(I), U(I), Y(I), I=1,NS)                ;##### DISPLAY ARGUMENTS
 8005 FORMAT (' VVSH: H, F, G, DVI, NS, M/',;#####
     + ' W, L, E, B, V, U, Y'/ 4F10.2, 2I3,     ;#####
     +   (/F8.4, F8.0, 4E13.6, F8.3))            ;#####
      V(1) = VI                                  ,RESTORE INITIAL ESTIMATE
      VMIN = -1.E25                              ;MINIMUM VERTICAL FORCE
      VMAX =  1.E25                              ;MAXIMUM
C
C *** NEXT TRY
C
  100 YW = Y(1)                                  ;START DEPTH
C
C *** SEGMENT LOOP
C
      DO 200 I=1,NS
      IF (I .GT. 1) V(I) = U(I-1) + B(I)         ;XSFER LOAD ACROSS NODE
      VFX = I .LT. M                             ;SET VERTEX PERMISSION
      Y(I) = YW                                  ;SAVE DEPTH FOR RETURN
      DY(I) = DELY(V(I), H,
     +  W(I), L(I), E(I), U(I), VFX)             ;CATENARY DEPTH CHANGE
      YW = YW + DY(I)                            ;NEXT NODE POSITION
      IF (TRYD .EQ. 0) WRITE (PTR, 8001)         ;#####
     + TRY, I, V(I), U(I), YW                    ;#####
 8001 FORMAT (2I3, 3F10.2)                       ;#####
  200 CONTINUE
C
C *** CHECK DEPTH CONVERGENCE
C
      EY = YW - Y(LAST)                          ;DEPTH ERROR
      IF (ABS(EY) .LT. G) GOTO 500               ;CONVERGED?
      IF (EY .LT. 0.) VMAX=AMIN1(V(1),VMAX)      ;ADJUST UPPER LIMIT
      IF (EY .GT. 0.) VMIN=AMAX1(V(1),VMIN)      ; AND LOWER
C
C *** DEPTH UNCONVERGED: ADJUST VERTICAL FORCE
C
      DV = SIGN(DVI,EY)                          ;INCREMENT FOR TRY 0
      IF (TRY .EQ. 0) GOTO 300                   ;WITHOUT ADJUSTMENT
      IF (EYO .NE. EY)
     + DV = AMOD(EY*DVO/(EYO-EY),F)              ;1 STEP NEWTON METHOD
C
C *** STAY WITHIN RANGE VMIN < V(1) < VMAX
C
      VT = V(1) + DV                             ;TRIAL UPDATE
      IF (VT .LT. VMIN) DV =                     ;NEXT V(1) = AVERAGE OF
     +              0.5*(VMIN - V(1) - DVO)      ; VMIN AND PRIOR V(1)
      IF (VT .GT. VMAX) DV =                     ;NEXT V(1) = AVERAGE OF
     +              0.5*(VMAX - V(1) - DVO)      ; VMAX AND PRIOR V(1)
C
C *** PREPARE NEXT ITERATION
C
  300 EYO = EY                                   ;SAVE ERROR
      DVO = DV                                   ;AND UPDATE
      V(1) = V(1) + DV                           ;UPDATE FORCE
      TRY = TRY + 1                              ;COUNT ITERATIONS
```

```
      IF (TRY .LT. TRYD) GOTO 310
      WRITE (CON, 8003) TRY, DV, V(1), EY, G;DISPLAY SLOW CONVERGENCE
 8003 FORMAT (I3, 4E14.7)
      IF (TRYD .EQ. 0) WRITE (PTR, 8002)    ;#####
     + TRY, EY, VMIN, VMAX, DV, V(1), U(1)  ;#####
 8002 FORMAT (                              ;#####
     + ' TRY   EY          VMIN        VMAX',;#####
     + '       DV          V              U'/;#####
     + I4, 6G12.5)                          ;#####
  310 IF (TRY .LE. TRYS) GOTO 100           ;NEXT ITERATION
C
C *** CONVERGENCE FAILURE
C
      WRITE (CON, 1001) TRYS, V(1)
      VVSH = .TRUE.                         ;ERROR IS TRUE
      IF (TRYD .EQ. 0) GOTO 500             ;REPEAT FINISHED
      TRYD = 0                              ;REPEAT IN DETAIL
      TRYS = 25                             ;BUT TRUNCATED
      GOTO 90
  500 TRYS = 90                             ;RESTORE LIMITS
      TRYD = 75
      RETURN                                ;DONE
C
 1001 FORMAT (
     + ' No convergence after', I3,
     + ' tries. Vertical force =', G14.7)
C
      END
```

```
      REAL FUNCTION DELY                              ;D.B.DILLON  EG&G  1985
     +           (V, H, W, S, EA, U, VXFLAG)
C
C *** GIVEN:
      REAL V  ;VERTICAL FORCE ACTING ON END OF CABLE, POSITIVE UPWARDS
      REAL H  ;EXTERNAL HORIZONTAL FORCE ACTING ON CABLE,
C              POSITIVE IN THE POSITIVE SENSE OF THE X-AXIS
      REAL W  ;LINEAR (IMMERSED) WEIGHT DENSITY OF CABLE
      REAL S  ;UNSTRETCHED MATERIAL LENGTH OF CABLE
      REAL EA ; =0: INELASTIC CABLE
C             ; >0: EFFECTIVE YOUNG'S MODULUS X METALLIC CROSS-SECTION
C ** RETURN:
C     DELY    ;HEIGHT DIFFERENCE, YU-YB
      REAL U  ;VERTICAL FORCE ON RESTRAINT AT OPPOSITE END FROM V
      LOGICAL*2 VXFLAG  ; .TRUE.: CABLE MAY VERTEX
C                       ;.FALSE.: DELY=DEPTH TO VERTEX
      INTEGER*2 CON, PTR, MSG, AUX, RIG      ;/UNITS/
C
      COMMON /UNITS/ CON, PTR, MSG, AUX, RIG
C
C *** TEST FOR CABLE ON/BELOW BOTTOM
C
      SU=S                                    ;ASSUME NO VERTEX
      IF (VXFLAG) GOTO 200                    ;VERTEX PERMITTED?
      DELY = 0.                               ;NO
      U = 0.                                  ;FORCE HORIZONTAL
      IF (V .GT. 0.) GOTO 100                 ;PARTIAL CATENARY
      V = 0.                                  ;CANCEL NEGATIVE CATENARY
      RETURN                                  ;CABLE ON/BELOW BOTTOM
C
C *** DETERMINE SCOPE OF CATENARY
C
  100 H2 = V/W                                ;SCOPE TO VERTEX
      SU = AMIN1(S, H2)                       ;LESS THAN LENGTH
C
C *** COMPUTE INELASTIC CATENARY OFFSETS
C
  200 U = V - W*SU
      H2 = H*H
      DELY = (SQRT (H2 + U*U)
     +     -  SQRT (H2 + V*V))/W
C
C *** ADJUST FOR ELASTIC EFFECTS
C
      IF (EA .NE. 0.) DELY =
     +    DELY - (U + V)*SU/(EA+EA)
      RETURN
C
      END
```

```
      REAL FUNCTION DELX (H, YB, W, S, E, T,;D.B.DILLON  EG&G  1985
     +        G, F, V, U, Y, BL, SL)
C
C *** COMPUTE THE SPAN OF AN ELASTIC OR INELASTIC CATENARY
C     GIVEN:
      REAL H                                 ;HORIZONTAL LOAD IN CATENARY
      REAL YB                                ;DEPTH OF BOTTOM
      REAL W                                 ;WEIGHT DENSITY OF ELEMENT
      REAL S                                 ;MATERIAL LENGTH UNSTRETCHED
      REAL E                                 ;ELASTICITY (EA)
      REAL T                                 ;TENSILE STRENGTH
      REAL G                                 ;LENGTH TOLERANCE
      REAL V                                 ;VERTICAL FORCE AT NODE END
      REAL U                                 ;VERTICAL LOAD AT ANCHOR END
      REAL Y                                 ;DEPTH "NODE" END
C    RETURN:
C *   REAL DELX                              ;SPAN OF MATERIAL LENGTH L
      REAL F                                 ;LEAST SAFETY FACTOR
      REAL BL                                ;BOTTOMED MATERIAL LENGTH
      REAL SL                                ;STRETCHED LENGTH
      REAL H2                                ;LOCALS: LOAD H SQUARED
      REAL E2                                ;2XEA
      REAL TV                                ;TENSION AT RIG END
      REAL TU                                ;TENSION AT ANCHOR END
      REAL SB                                ;STRETCH ALONG BOTTOM
      REAL SW                                ; AND STRETCH IN WATER
      REAL SV                                ;LENGTH TO VERTEX
      REAL YX                                ;VERTEX DEPTH
      REAL DX                                ;SPAN INCREMENT
      REAL VT, UT                            ;TEMPORARIES
      INTEGER*2 CON, PTR, MSG, AUX, RIG      ;/UNITS/
C
      COMMON /UNITS/ CON, PTR, MSG, AUX, RIG
C
C *** INITIAL VALUES
C
      H2 = H*H
      E2 = E+E
      TV = SQRT(H2+V*V)                      ;NODE END TENSION
      TU = SQRT(H2+U*U)                      ;ANCHOR END TENSION
      F = T/AMAX1(1., TV, TU)                ;LEAST SAFETY FACTOR
      SL = S                                 ;ALL IN CATENARY
C
C *** COMPUTE SCOPE TO VERTEX FROM EITHER END
C
      IF (W .EQ. 0.) GOTO 100                ,NO ELEMENT VERTEX
      IF (V .LT. 0.) GOTO 100                ;NO ELEMENT VERTEX
      SV = V/W                               ;LENGTH TO VERTEX
      IF (SV .GT. S) GOTO 100                ;NO ELEMENT VERTEX
      YX = Y - (TV-H)/W                      ;IS VERTEX ON BOTTOM?
      IF (E .NE. 0.) YX = YX - V*SV/E2       ;ADJUSTED FOR EA
      IF (YX-YB .LT. G) SL = SV              ;SL=MAT'L SCOPE TO VERTEX
C                                               ON BOTTOM
C *** CATENARY SPAN
C
```

```
  100 BL = S - SL                              ;LENGTH ON BOTTOM
      SB = 0.                                  ;INELASTIC ON BOTTOM
      SW = 0.                                  ;INELASTIC IN WATER
      DX = 0.                                  ;ASSUME VERTICAL
      VT = V + TV                              ;TEMPORARY
      UT = U + TU
      IF (TV .NE. 0.) DX = S*H/TV              ;PROJECT WEIGHTLESS LINE
      IF ((VT .NE. 0.) .AND.                   ;INELASTIC
     +    (UT .NE. 0.) .AND.                   ; CATENARY
     +    (W  .NE. 0.)) DX=H/W*ALOG(VT/UT)     ;  SPAN
      IF (E .EQ. 0.) GOTO 110                  ;DONE IF INELASTIC
      SB = BL*H/E                              ;STRETCH ALONG BOTTOM
      IF (W .EQ. 0.) THEN                      ;STRETCH IN WATER
          SW = TV*S/E                          ; WEIGHTLESS CASE
      ELSE
          SW = DX*H/E2 + (V*TV-U*TU)/W/E2      ; HEAVY CASE
      ENDIF
      DX = DX + H*S/E                          ;ELASTIC SPAN
  110 SL = S + SB + SW                         ;STRETCHED LENGTH
      DELX = BL + DX                           ;TOTAL SPAN
      RETURN
C
      END
```

```
      REAL FUNCTION STPSIZ (STEP)              ,D.B.DILLON  EG&G  1985
C
C *** GIVEN: STEP   = EXACT ARGUMENT
C    RETURN: STPSIZ = STEP ROUNDED TO SEQUENCE 1, 2, 2.5, 5 X 10^N
C
      REAL STEP                                ;ARGUMENT
      REAL P                                   ;LOCAL: POWER
      REAL X                                   ;PARAMETER
      REAL S                                   ;STEP SIZE
C
C *** RETURN 1 ON BAD CALL
C
      STPSIZ = 1.
      IF (STEP .LT. 1.) RETURN                 ;LEAST STEP SIZE
C
C *** SPLIT STEP INTO A BASE AND A MULTIPLIER
C
      X = ALOG10(STEP)                         ;EXACT POWER
      P = AINT(X)                              ;INTEGER POWER
      X = 10.**(X - P)                         ;1 < BASE STEP < 10
      P = 10.**P                               ;10^N MULTIPLIER
C
C *** COMPARE BASE STEP TO STEP SEQUENCE
C
      S = 1.0                                  ;1ST CHOICE
      IF (X .GT. 1.50) S =  2.0                ;2ND
      IF (X .GT. 2.25) S =  2.5                ;3RD
      IF (X .GT. 3.75) S =  5.0                ;4TH
      IF (X .GT. 7.50) S = 10.0                ;5TH
      STPSIZ = S*P                             ;RETURN SCALED STEP
      RETURN
C
      END
```

```
      PROGRAM PLTRIG
C
C *** EXTRACT VALUES FOR QMPLOT FROM RIGMOOR FILES
C
      IMPLICIT INTEGER*2 (A-Z)
      LOGICAL*2 RIGNAM                              ;FUNCTION
      INTEGER*2 DP                                  ;POINTS TO ´:´
      INTEGER*2 PP                                  ;    "   TO LAST ´\´ OF PATH
      INTEGER*2 NP                                  ;    "   TO ´.´
      INTEGER*2 XP                                  ;    "   TO LAST BYTE
      CHARACTER*1 ONAME(64), ONAM*64                ;OUTPUT FILE NAME
      CHARACTER*1 INAME(64)                         ;INPUT DITTO
      CHARACTER*1 OK(4), CASE                       ;FILE FORM CODES
      EQUIVALENCE (ONAM, ONAME)                     ;DUAL NAMES FOR OPEN
      DATA OK /´H´, ´X´, ´O´, ´S´/                  ;HVSS XVSH OPROS SVROS
C
100   IF (RIGNAM(INAME(1),DP,PP,NP,XP)) STOP        ;OPEN SOURCE FILE
      DO 110 I=1,XP
110   ONAME(I) = INAME(I)                           ;COPY NAME FOR OUTPUT
      I = 64                                        ;ONAME LENGTH
      CALL PADNAM (ONAME, XP, I)                    ;APPEND SPACES
      ONAME(NP-2) = ´P´                             ;PLOT FLAG IN FILE NAME
      CASE = INAME(NP-2)                            ;H X S OR O RIGMOOR FILE ID
      DO 120 I=1,4                                  ;TEST FOR VALID NAME
120   IF (CASE .EQ. OK(I)) GOTO 130                 ;YES
      CLOSE (3)                                     ;NO
      GOTO 100                                      ;TRY AGAIN
C
130   OPEN (5,  FILE=ONAM, STATUS=´NEW´)            ;OPEN OUTPUT FILE
      IF (I .EQ. 1) CALL GETHS (REC)                ;CONVERT H VS S
      IF (I .EQ. 2) CALL GETXH (REC)                ;CONVERT X VS H
      IF (I .EQ. 3 .OR. I .EQ. 4) CALL GETROS
     +         (I, REC, INAME(1), DP, PP, NP, XP);CONVERT ROSE FILES
      WRITE (*, 1010) REC, INAME, ONAME             ;NOTE PROGRESS
      CLOSE (3)                                     ;END OF INPUT
      CLOSE (5)                                     ;END OF OUTPUT
      GOTO 100                                      ;NEXT CASE
C
1010  FORMAT (/I3, ´ RECORDS COPIED FROM / TO´/
     +             1X64A1/1X64A1/)
C
      END
```

```
      SUBROUTINE GETHS (REC)
C
C *** COPY H VS S FILE FROM BINARY IN UNIT 3 TO ASCII IN UNIT 5
C
      IMPLICIT INTEGER*2 (A-Z)
      REAL HVSS(9)
C
      DO 100 REC=1,127
      READ (3, END=200, ERR=200) HVSS             ;GET A LINE
100   WRITE (5, 1000) HVSS                        ;COPY IT IN ASCII
200   RETURN
C
1000  FORMAT (E14.7, 8(',', E13.7))               ;COMMA-SEPARATED-VALUE FORM
      END
C
C *** ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
C
      SUBROUTINE GETXH (REC)
C
C *** COPY X VS H FILE FROM BINARY IN UNIT 3 TO ASCII IN UNIT 5
C
      IMPLICIT INTEGER*2 (A-Z)
      REAL XVSH(5)                                ;LOAD SPAN SOB STRS FMIN
C
      DO 100 REC=1,127
      READ (3, END=200, ERR=200) XVSH             ;GET A LINE
100   WRITE (5, 1000) XVSH                        ;COPY IT IN ASCII
200   RETURN
C
1000  FORMAT (E14.7, 4(',', E13.7))               ;COMMA-SEPARATED-VALUE
      END
```

```
      SUBROUTINE GETROS (I, REC, INAM3, DP, PP, NP, XP)
C
C *** COPY ROSE FILES FROM BINARY IN UNITS 3 AND 4 TO ASCII IN UNIT 5
C
      IMPLICIT INTEGER*2 (A-Z)
      CHARACTER*1 INAM3(64), INAM5(64), NAME5*64
      CHARACTER*1 OVLY(2,2)                        ;NAME OVERLAY
      REAL OROSE(8), SROSE(8)                      ;THETA HOLD WIND
C                                                  ;FX FY SF YAW CWM
      EQUIVALENCE (NAME5, INAM5(1))
      DATA OVLY /'S','V', 'O','P'/
      DO 100 J=1,XP
100   INAM5(J) = INAM3(J)                          ;COPY SOURCE NAME
      K = NP-2
      DO 110 J=1,2
      INAM5(K) = OVLY(J,I-2)                       ;INAM5 & INAM3 ARE PAIRS
110   K=K+1
      K = 64                                       ;INAM5 LENGTH
      CALL PADNAM (INAM5, XP, K)                   ;PAD INAM5 WITH SPACES
      OPEN (4,  FILE=NAME5,
     +          STATUS='OLD',
     +            FORM='UNFORMATTED')              ;INPUT OP/SRV COMPLEMENT
      J = 7 - I                                    ;COMPLEMENT OF I=3,4
      DO 200 REC=1,127
      READ (I, END=300, ERR=300) OROSE             ;READ OPERATIONAL ROSE
      READ (J, END=300, ERR=300) SROSE             ;READ SURVIVAL ROSE
200   WRITE (5, 1010) OROSE, SROSE                 ;COPY IT IN ASCII
300   CLOSE (5)                                    ;CLOSE EXTRA FILE
      RETURN
C
1010  FORMAT (E14.7, 15(',', E13.7))               ;COMMA-SEPARATED-VALUE FORM
      END
C
C *** ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
C
      SUBROUTINE PADNAM (NAME, AL, CL)
C
C *** PAD NAME FROM END OF STRING TO END OF VARIABLE WITH SPACES
C
      INTEGER*2   AL                               ;STRING LENGTH
      INTEGER*2   CL                               ;NAME LENGTH
      CHARACTER*1 NAME(CL)                         ;STRING TO PAD
      INTEGER*2   J, K                             ;POINTERS
C
      K = AL+1
      DO 140 J=K,CL
140   NAME(J) = ' '
      RETURN
C
      END
```

```
      LOGICAL*2 FUNCTION RIGNAM (NAME, DRV, PTH, NAM, EXT)
C
C *** GET AND OPEN A FILE NAME FROM USER
C       RETURN NAME, AND POINTERS TO :, LAST \, ., AND LAST BYTE
C
      IMPLICIT INTEGER*2 (A-Z)
      CHARACTER*1 NAME(64), NAM3(64), ONAME*64   ;DRIVE+PATH+NAME+EXTENSION
      EQUIVALENCE (ONAME, NAM3(1))
C
C *** GET FILE NAME
C
      RIGNAM = .TRUE.
100   WRITE (*, 1000)                            ,PROMPT FOR FILE NAME
1000  FORMAT (/' RIGMOOR file name: (N=None)'/)
      READ (*, 1010) ONAME                       ;ACCEPT FILE NAME
1010  FORMAT (A)
C
C *** IGNORE TRAILING AND LEADING SPACES
C
      DO 200 J=64,1,-1
      IF (NAM3(J) .GT. ' ') GOTO 210             ;SCAN FOR NON-BLANK
      NAME(J) = ' '                              ;ERASE ARGUMENT
200   CONTINUE
      GOTO 100                                   ;RE-ENTER ON NULL ENTRY
210   JF = J                                     ;MARK FINAL BYTE
C
      DO 220 J=1,JF                              ;SCAN LEADING SPACES
      IF (NAM3(J) .GT. ' ') GOTO 230             ;FIRST NON-BLANK
220   CONTINUE
      GOTO 100                                   ;RE-ENTER SCREW-UP
C
230   JS = J                                     ;START OF NAME
      IF (JS .EQ. JF .AND.
     +            NAM3(JS) .EQ. 'N') RETURN      ,CHECK FOR USER ABORT
C
      K=0
      DO 300 J=JS,JF
      K=K+1
      IF (K .LT. J) NAM3(K) = NAM3(J)            ;LEFT-JUSTIFY NAME
      NAME(J) = ' '                              ;ERASE TRAILING NAME
      NAME(K) = NAM3(K)                          ;COPY TO ARGUMENT
      IF (NAME(K) .EQ. ':') DRV = K              ;END DRIVE NAME
      IF (NAME(K) .EQ. '\') PTH = K              ;END PATH NAME
      IF (NAME(K) .EQ. '.') NAM = K              ;END MAIN NAME
300   CONTINUE
      EXT = K                                    ;END EXTENSION
      IF (NAM .EQ. 0) NAM = EXT + 1              ;NO EXTENSION IMPLIES .
C
400   OPEN (3,  FILE=ONAME,
     +            STATUS='OLD',
     +              FORM='UNFORMATTED')          ;OPEN THE FILE
      RIGNAM = .FALSE.
      RETURN
C
      END
```

INDEX TO ROUTINES BY ROUTINE NAME

ROUTINE NAME		MODULE	PAGE
ANALYZ	SUBROUTINE	ROSE -	1
DEFLEG	SUBROUTINE	CASE -	4
DELAY	SUBROUTINE	MOOR -	8
DELX	REAL FUNCTION	VERT -	10
DELY	REAL FUNCTION	VERT -	9
DEVICE	BLOCK DATA	MOOR -	1
FINDHS	LOGICAL FUNCTION	LOOK -	10
GETDBL	LOGICAL FUNCTION	CASE -	11
GETDEF	LOGICAL FUNCTION	NAME -	1
GETFLT	LOGICAL FUNCTION	CASE -	10
GETINT	LOGICAL FUNCTION	CASE -	9
GETPRE	SUBROUTINE	PREL -	1
GETRIG	LOGICAL FUNCTION	MOOR -	9
GETRNG	LOGICAL FUNCTION	NAME -	12
GETSTR	LOGICAL FUNCTION	CASE -	13
GETXVH	SUBROUTINE	VERT -	1
GETYAW	LOGICAL FUNCTION	ROSE -	5
HLPMSG	SUBROUTINE	MOOR -	10
HPMAX	LOGICAL FUNCTION	PREL -	9
HVSS	SUBROUTINE	LOAD -	5
LISTHS	SUBROUTINE	SHOW -	6
LISTXH	SUBROUTINE	SHOW -	2
LOADXH	SUBROUTINE	LOOK -	4
LOOKXH	SUBROUTINE	LOOK -	3
MAKDEF	LOGICAL FUNCTION	NAME -	11
MAKFIL	LOGICAL FUNCTION	NAME -	9
MAKROS	LOGICAL FUNCTION	NAME -	14
MAPLEG	SUBROUTINE	LOAD -	4
NEWCAS	SUBROUTINE	CASE -	1
OPENUF	LOGICAL FUNCTION	NAME -	10
OPTPRE	LOGICAL FUNCTION	LOOK -	11
PRANGE	LOGICAL FUNCTION	LOOK -	6
PRELOD	SUBROUTINE	LOAD -	6
PRTDEF	SUBROUTINE	NAME -	6
PUTDEF	LOGICAL FUNCTION	NAME -	4
RIGMOR	PROGRAM	MOOR -	5
SETLEG	SUBROUTINE	LOOK -	1
SHOLEG	SUBROUTINE	NAME -	8
SHOWHP	SUBROUTINE	SHOW -	8
SHOWHS	SUBROUTINE	SHOW -	5
SHOWXH	SUBROUTINE	SHOW -	1
STAN	REAL FUNCTION	LOAD -	11
STEEP	LOGICAL FUNCTION	PREL -	5
STPSIZ	REAL FUNCTION	VERT -	12
SURFAC	SUBROUTINE	VERT -	4
TOPLEN	LOGICAL FUNCTION	LOOK -	13
TORQUE	SUBROUTINE	ROSE -	7
TRIES	INTEGER FUNCTION	ROSE -	9
USRINP	LOGICAL FUNCTION	CASE -	7
VVSH	LOGICAL FUNCTION	VERT -	6
XTREME	LOGICAL FUNCTION	LOOK -	8
XVSH	SUBROUTINE	LOAD -	1

INDEX TO ROUTINES BY MODULE NAME

MODULE PAGE	ROUTINE NAME	CALLING FORMAT
CASE - 1	NEWCAS	SUBROUTINE
CASE - 4	DEFLEG	SUBROUTINE
CASE - 7	USRINP	LOGICAL FUNCTION
CASE - 9	GETINT	LOGICAL FUNCTION
CASE - 10	GETFLT	LOGICAL FUNCTION
CASE - 11	GETDBL	LOGICAL FUNCTION
CASE - 13	GETSTR	LOGICAL FUNCTION
LOAD - 1	XVSH	SUBROUTINE
LOAD - 4	MAPLEG	SUBROUTINE
LOAD - 5	HVSS	SUBROUTINE
LOAD - 6	PRELOD	SUBROUTINE
LOAD - 11	STAN	REAL FUNCTION
LOOK - 1	SETLEG	SUBROUTINE
LOOK - 3	LOOKXH	SUBROUTINE
LOOK - 4	LOADXH	SUBROUTINE
LOOK - 6	PRANGE	LOGICAL FUNCTION
LOOK - 8	XTREME	LOGICAL FUNCTION
LOOK - 10	FINDHS	LOGICAL FUNCTION
LOOK - 11	OPTPRE	LOGICAL FUNCTION
LOOK - 13	TOPLEN	LOGICAL FUNCTION
MOOR - 1	DEVICE	BLOCK DATA
MOOR - 5	RIGMOR	PROGRAM
MOOR - 8	DELAY	SUBROUTINE
MOOR - 9	GETRIG	LOGICAL FUNCTION
MOOR - 10	HLPMSG	SUBROUTINE
NAME - 1	GETDEF	LOGICAL FUNCTION
NAME - 4	PUTDEF	LOGICAL FUNCTION
NAME - 6	PRTDEF	SUBROUTINE
NAME - 8	SHOLEG	SUBROUTINE
NAME - 9	MAKFIL	LOGICAL FUNCTION
NAME - 10	OPENUF	LOGICAL FUNCTION
NAME - 11	MAKDEF	LOGICAL FUNCTION
NAME - 12	GETRNG	LOGICAL FUNCTION
NAME - 14	MAKROS	LOGICAL FUNCTION
PREL - 1	GETPRE	SUBROUTINE
PREL - 5	STEEP	LOGICAL FUNCTION
PREL - 9	HPMAX	LOGICAL FUNCTION
ROSE - 1	ANALYZ	SUBROUTINE
ROSE - 5	GETYAW	LOGICAL FUNCTION
ROSE - 7	TORQUE	SUBROUTINE
ROSE - 9	TRIES	INTEGER FUNCTION
SHOW - 1	SHOWXH	SUBROUTINE
SHOW - 2	LISTXH	SUBROUTINE
SHOW - 5	SHOWHS	SUBROUTINE
SHOW - 6	LISTHS	SUBROUTINE
SHOW - 8	SHOWHP	SUBROUTINE
VERT - 1	GETXVH	SUBROUTINE
VERT - 4	SURFAC	SUBROUTINE
VERT - 6	VVSH	LOGICAL FUNCTION
VERT - 9	DELY	REAL FUNCTION
VERT - 10	DELX	REAL FUNCTION
VERT - 12	STPSIZ	REAL FUNCTION

INDEX TO ROUTINES BY CALLING FORM

ROUTINE NAME	CALLING FORM	MODULE	PAGE
DEVICE	BLOCK DATA	MOOR	- 1
TRIES	INTEGER FUNCTION	ROSE	- 9
FINDHS	LOGICAL FUNCTION	LOOK	- 10
GETDBL	LOGICAL FUNCTION	CASE	- 11
GETDEF	LOGICAL FUNCTION	NAME	- 1
GETFLT	LOGICAL FUNCTION	CASE	- 10
GETINT	LOGICAL FUNCTION	CASE	- 9
GETRIG	LOGICAL FUNCTION	MOOR	- 9
GETRNG	LOGICAL FUNCTION	NAME	- 12
GETSTR	LOGICAL FUNCTION	CASE	- 13
GETYAW	LOGICAL FUNCTION	ROSE	- 5
HPMAX	LOGICAL FUNCTION	PREL	- 9
MAKDEF	LOGICAL FUNCTION	NAME	- 11
MAKFIL	LOGICAL FUNCTION	NAME	- 9
MAKROS	LOGICAL FUNCTION	NAME	- 14
OPENUF	LOGICAL FUNCTION	NAME	- 10
OPTPRE	LOGICAL FUNCTION	LOOK	- 11
PRANGE	LOGICAL FUNCTION	LOOK	- 6
PUTDEF	LOGICAL FUNCTION	NAME	- 4
STEEP	LOGICAL FUNCTION	PREL	- 5
TOPLEN	LOGICAL FUNCTION	LOOK	- 13
USRINP	LOGICAL FUNCTION	CASE	- 7
VVSH	LOGICAL FUNCTION	VERT	- 6
XTREME	LOGICAL FUNCTION	LOOK	- 8
RIGMOR	PROGRAM	MOOR	- 5
DELX	REAL FUNCTION	VERT	- 10
DELY	REAL FUNCTION	VERT	- 9
STAN	REAL FUNCTION	LOAD	- 11
STPSIZ	REAL FUNCTION	VERT	- 12
ANALYZ	SUBROUTINE	ROSE	- 1
DEFLEG	SUBROUTINE	CASE	- 4
DELAY	SUBROUTINE	MOOR	- 8
GETPRE	SUBROUTINE	PREL	- 1
GETXVH	SUBROUTINE	VERT	- 1
HLPMSG	SUBROUTINE	MOOR	- 10
HVSS	SUBROUTINE	LOAD	- 5
LISTHS	SUBROUTINE	SHOW	- 6
LISTXH	SUBROUTINE	SHOW	- 2
LOADXH	SUBROUTINE	LOOK	- 4
LOOKXH	SUBROUTINE	LOOK	- 3
MAPLEG	SUBROUTINE	LOAD	- 4
NEWCAS	SUBROUTINE	CASE	- 1
PRELOD	SUBROUTINE	LOAD	- 6
PRTDEF	SUBROUTINE	NAME	- 6
SETLEG	SUBROUTINE	LOOK	- 1
SHOLEG	SUBROUTINE	NAME	- 8
SHOWHP	SUBROUTINE	SHOW	- 8
SHOWHS	SUBROUTINE	SHOW	- 5
SHOWXH	SUBROUTINE	SHOW	- 1
SURFAC	SUBROUTINE	VERT	- 4
TORQUE	SUBROUTINE	ROSE	- 7
XVSH	SUBROUTINE	LOAD	- 1