TABLE OF CONTENTS PAGE

1.0 INTRODUCTION

This report documents the improvements and enhancements made to the ULSLEA program during Phase III of the "Screening Methodologies for Use in Platform Assessments and Requalification" project conducted by the Marine Technology and Management Group (MTMG) at U. C. Berkeley. Phase III had the following goals:

- Develop and document earthquake analysis procedures for offshore platforms suitable for use with the ULSLEA program. This includes both strength-level analysis and determination of deck accelerations for the design of mounted equipment.
- Develop and document fatigue analysis procedures for main load-carrying member joints and attachments suitable for use with the ULSLEA program.
- Adapt the ULSLEA program to allow for storm, fatigue and earthquake analyses of four additional platform configurations: multi 4-leg jackets, tripod jackets, guyed and braced monopods/caissons.
- Apply the ULSLEA program to the task of performing a series of parameter studies of a typical jacket-type platform, to evaluate how global capacity changes in response to local damage.

The earthquake analysis procedures development effort has been documented in a companion report, "Earthquake Analysis of Offshore Platforms" (Stear, Bea, 1997). The parameter studies findings have also been documented in a companion report, "ULSLEA: Parametric Studies of the Effects of Local Damage and Repairs on Global Lateral Load Capacity of a Typical Offshore Platform" (Aviguetero, Bea, 1996). This report documents the fatigue analysis procedures and use of the fatigue analysis option, use of the earthquake analysis option, details of the adaptation of the ULSLEA program to allow for the analysis of additional platform configurations, as well as several additional program enhancements.

The ULSLEA program has undergone some important changes which may not be obvious to the user. These changes are described in Appendix A. Program code (Visual Basic, Excel 5.0) for the sub-procedures and functions used by the new version, v3.0 beta, may also be found in Appendix A.

This project has been sponsored by ARCO Exploration and Production Technology, Exxon Production Research Company, Mobil Technology Company, Shell Offshore Incorporated, and Unocal Corporation. Special appreciation is expressed to Mr. Allen Brooks (Shell), Mr. Brad Campbell (Exxon), Mr. Kris Digre (Shell), Mr. Steve Guynes (ARCO), Dr. Mehrdad Mortazavi (Exxon), and Mr. J. Ward Turner (Exxon) for information and assistance in the development, testing and verification of v3.0 beta of the ULSLEA program.

2.0 GENERAL ENHANCEMENTS

In addition to the effort devoted to developing and documenting procedures to allow an ULSLEA program user to perform fatigue analysis and earthquake analysis of jacket-type structures, and expanding the applicability of the ULSLEA program to include additional platform configurations (multi 4-leg jacket structures, tripod jackets, guyed and braced monopods/caissons), several additional features have been added to the program. These features are:

- Drag coefficient scaling for near-surface wave forces
- Grouting of jacket legs
- The use of corner skirt piles, which act in both the end-on and broadside direction
- The absence of an unbraced deck bay above the jacket

These features are briefly discussed in this section.

2.1 Drag Coefficient Scaling for Hydrodynamic Force Calculation

Previous versions (pre-v3.0 beta) of the ULSLEA program have used a drag coefficient C_d which is invariant with depth to calculate wave-induced forces. While a constant value of C_d is recommended for conservative design purposes, it should be recognized that there are significant differences between the actual drag force on members near the surface and the drag forces calculated through application of the Morison equation with a constant drag coefficient due to near-surface flow distortion effects (McDonald, et al., 1990). In recognition of these effects, an option by which C_d is scaled linearly from zero at the wave crest to full value to a depth of two velocity heads (U_{crest}^2 / g) below the crest has been added to the program. This option is now used as the default procedure for determining C_d. If a user desires to use an unscaled C_d, the **PRELIMINARY DESIGN** menu item of the **INPUT** menu should be chosen; this will bring up a screen of analysis

options (Figure 2.1). To use the unscaled C_d, the user simply checks the "Use Design C_d" checkbox.

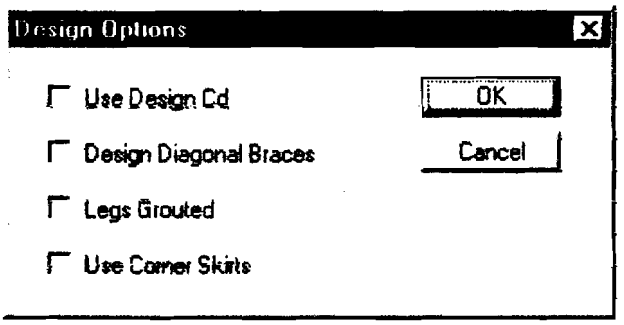

Figure 2.1: Design Options Screen

To demonstrate the difference in the estimated wave forces using scaled versus unscaled C_d, an ULSLEA analysis of the South Pass 62 A platform (characteristics of this structure have been documented by Stear and Bea, 1996A) was conducted using both options. The results are shown in Figures 2.2 and 2.3 for the case of 82 ft broadside wave attack.

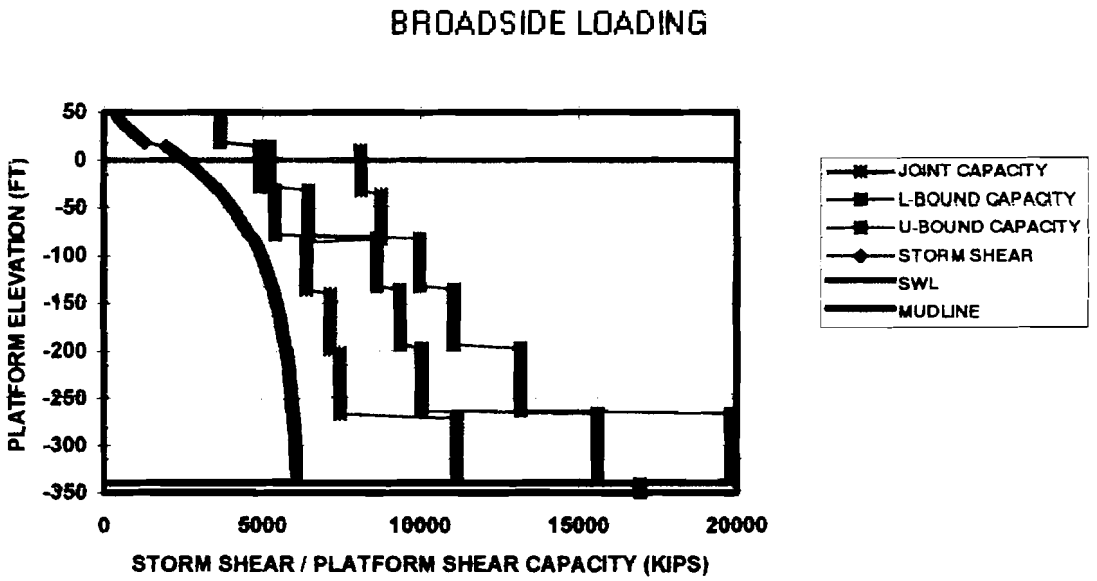

Figure 2.2: SP 62 A, Broadside Loading (82 ft wave), C_d Scaling

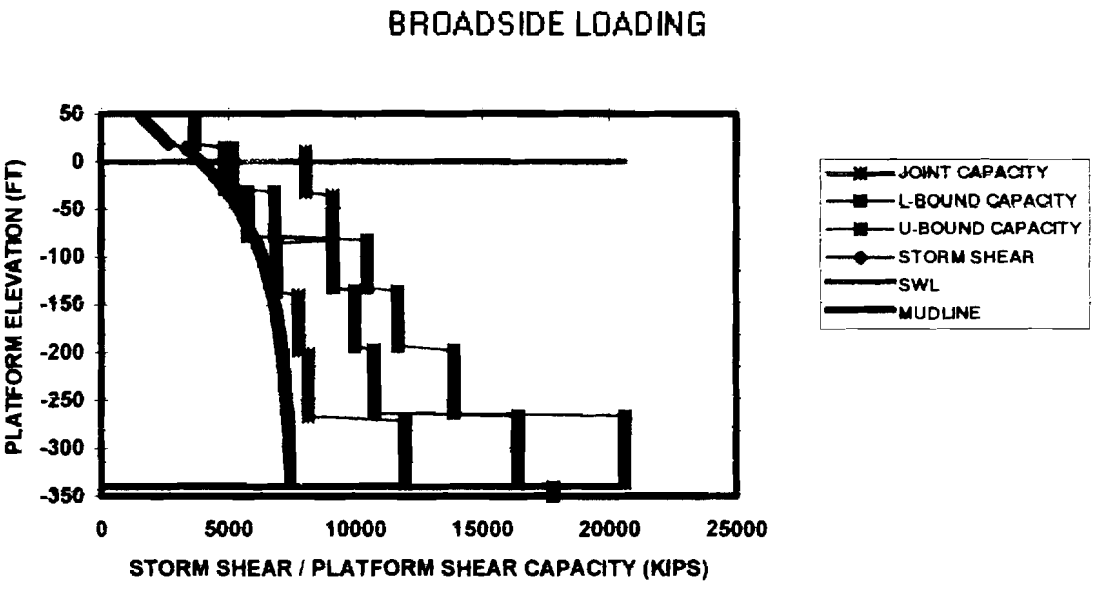

Figure 2.3: SP 62 A, Broadside Loading (82 ft wave), No C_d Scaling

The estimated base shear for the scaled case is approximately 6,100 kips, while the base shear for the unscaled case is 7,500 kips. Use of C_d scaling has resulted in a 19% reduction in the estimated waves forces. Obviously, this is fairly significant, and should bear some consideration when the user is faced with the task of reassessing a platform. This reduction could also explain why two structures (Amoco ST 161 A, Chevron ST 151 K) previously examined during the Screening Methodologies project survived extreme events with little or no damage when analyses (USFOS, documented by Loch and Bea, 1995; ULSLEA, documented by Mortazavi, 1996; SUS, documented by Stear and Bea, 1996B) indicated that failure was likely (ST 161 A) or definite (ST 151 K).

2.2 Grouted Jacket Legs

Users may now also indicate whether or not the main piles are grouted in the jacket legs. This will increase the local bending stiffness of the jacket legs (important for assessing deck bay displacements), and it has a significant influence on the global vertical and bending stiffnesses of the jacket structure. When this option is selected (via the **PRELIMINARY DESIGN** screen, Figure 2.1), local rotational stiffnesses will be based

on the combined cross sections of the jacket legs and piles and global vertical and bending stiffnesses of the jacket will be calculated based on the axial stiffnesses of both the jacket legs and main piles. If the option is left unchecked, the local stiffnesses are calculated based on the jacket legs, while the global stiffnesses are calculated based only on the axial stiffness of the main piles.

2.3 Corner Skirt Piles

Previous versions (pre-v3.0 beta) of the ULSLEA program assume that skirt piles on a particular face of a platform only act to resist overturning forces perpendicular to that platform face. For cases in which skirt piles are located near the corners of the jacket, these piles should act to resist overturning forces on both platform principal axes. By checking the "Corner Skirts" box on the **PRELIMINARY DESIGN** screen (Figure 2.1), piles input on each face will resist overturning forces perpendicular to both principal axes. To avoid duplication, when piles exist only at the corners, half the total number of piles should be specified as acting against overturning in the end-on direction, and the other half should be specified as acting against overturning in the broadside direction.

2.4 Braces in Deck Bay

For situations in which there is no unbraced deck bay, and the jacket batter is carried to the deck level, by entering a deck bay height of "1", the top jacket bay will be treated as the deck bay, allowing for the input of bracing in that bay. Deck leg thickness and diameter values equal to the jacket leg thickness and diameter should still be input as local parameters; while these are ignored for capacity purposes, they are still used to create very stiff elements constraining the deck mass to the deck bay.

3.0 FATIGUE ANALYSIS OF MAIN DIAGONAL TUBULAR JOINTS

3.1 Overview

ULSLEA was originally developed for use in evaluating the ultimate lateral load capacity for template-type offshore platforms. However, a decision as to the serviceability of a structure should be based not only upon strength, but also upon durability (resistance to fatigue damage). Hence, it is desirable to devise some means of evaluating the fatigue damage potential of critical platform structural elements within the framework of a simplified analysis process.

Platform components particularly vulnerable to fatigue damage are the tubular joints between members. If improperly detailed, they can give rise to very high stress concentrations, which in turn will lead to a rapid accumulation of fatigue damage.

A simplified approach to estimate fatigue damage for tubular joints has been implemented in the ULSLEA program. This approach is based upon the one outlined in API RP 2A-LRFD (1993), by which the fatigue damage in a structural element is related to the distribution of wave heights affecting the structure over a fixed period of time. The stress demand on the joint due to the fatigue design wave load is calculated; this value is then combined with information on the long-term distribution of waves in the area (a combined spectrum of both typical and extreme waves) together with S-N parameters in order to give a generic estimate of accumulated damage. This estimate is then used to rank the joints so that a qualitative assessment of potential problem areas can be made.

This approach assumes fatigue damage will most likely occur in those joints to which vertical diagonal braces are attached; hence, its application is limited to that class of joint. Only principal directions of loading are considered.

3.2 Analytical Approach

The approach taken to calculate fatigue damage for tubular joints is based on four assumptions:

1. The maximum stress range at the critical area in the joint is dependent only on the wave heights, and is related to them as follows:

$$S = CH^{g_{stress}}$$

where:

S = maximum stress range
C = calibrated constant
H = wave height
g_{stress} = calibrated exponent

2. The S-N curve characterizing the fatigue behavior is given by:

$$NS^m = K$$

where:

N = number of cycles to failure at a given S
m = empirical constant
K = empirical constant

3. Miner's rule applies. This is given by:

$$D=\sum_i \frac{n_i}{N_i}$$

where:

D = total fatigue damage
n_i = number of cycles of stress range S_i
N_i = number of cycles to failure given stress range S_i

4. The long-term wave-height distribution can be represented as a sum of two Weibull distributions, one of which represents typical waves while the other represents storm waves. These distributions are given by their cumulative distribution functions:

$$F_{H_0}(h)=1-\exp\left[-\left(\frac{h}{H_0}\right)^{\xi_0}\ln N_0\right]$$

and:

$$F_{H_1}(h)=1-\exp\left[-\left(\frac{h}{H_1}\right)^{\xi_1}\ln N_1\right]$$

where:

H_i = maximum wave height for time considered
N_i = number of waves for time considered
ξ_i = Weibull distribution parameter

Based on these assumptions, a closed-form solution for the fatigue damage may be obtained (Nolte, Hansford, 1976):

$$D_d = \frac{T_d C^m}{K}\left(Y_0 + Y_1\right)$$

where:

$$Y_0 = \frac{N_0}{T_{spectrum}} H_0^{g_{stress}m}\left(\ln N_0\right)^{\frac{g_{stress}m}{\xi_0}} \Gamma\left(1 + \frac{g_{stress}m}{\xi_0}\right)$$

and:

$$Y_1 = \frac{N_1}{T_{spectrum}} H_1^{g_{stress}m}\left(\ln N_1\right)^{\frac{g_{stress}m}{\xi_1}} \Gamma\left(1 + \frac{g_{stress}m}{\xi_1}\right)$$

D_d is the fatigue damage accumulated over the specified duration of service T_d.

C should be determined from "fatigue design wave" conditions; this may be done as follows:

$$C = \frac{S_f}{H_f^{g_{stress}}}$$

where:

H_f = fatigue design wave height

S_f = maximum stress range corresponding to H_f

S_f may be evaluated from:

$$S_f = S_p(1-R)$$

where:

S_p = peak stress in component due to static application of forces from H_f

R = stress cycle ratio

With some rearranging, the accumulated fatigue damage may then be expressed by:

$$D_d = \frac{T_d}{K}\left(\frac{S_p(1\text{-}R)}{H_f^{g_{stress}}}\right)^m \left(Y_0 + Y_1\right)$$

This accumulated damage value may then be used to make qualitative comparisons of joint performance. A high damage value for a joint may indicate it is a potential problem, and hence may need careful inspection.

3.3 Fatigue Damage Parameters

To evaluate the accumulated fatigue damage for a given joint, the parameters K, m, g_{stress}, R, H_0, H_1, ξ_0, ξ_1, N_0, N_1, H_f, and $T_{spectrum}$ must be determined. The calculation of S_p is discussed in Section 3.4.

K and m will be specified by the S-N curve chosen for the joint being analyzed. The X S-N curve (for welds with profile control) and the X′ S-N curve (for welds without profile control) provided by API RP 2A-LRFD (1993) may be used for a conservative assessment.

g_{stress} is a constant relating the wave height to the stress range at a particular location. Luyties and Geyer (1987) indicate that for structures having natural periods of three seconds or less, g_{stress} may be taken as 1.2 for components at the waterline, and as 1.3 for all other members. R is typically taken to be the total base shear cycle ratio (i.e. maximum base shear divided by minimum base shear for a single wave cycle), as described by API RP 2A-LRFD (1993) and Luyties and Geyer (1987). Typical values of R range from -0.15 to -0.5 and are dependent upon water depth.

H_0, H_1, ξ_0, ξ_1, N_0, N_1 and $T_{spectrum}$ must be specified according to the joint distribution of typical and storm waves to be found in the region where the structure is sited. This distribution must reflect wave encounters for the specified duration T_d. Similarly, H_f should be selected as the maximum wave height expected for the duration T_d. It is worth noting that for most cases this specified wave will not inundate the decks of the platform; should situations arise in which the specified wave does inundate the decks, the calculated stress S_p due to this wave will be conservatively high due to the inclusion of deck inundation forces in the component load calculation.

3.4 Calculating the Peak Stress

Calculation of the stresses in a tubular joint can be a very time consuming and complicated process. Much effort may be expended in trying to refine estimates of the forces affecting the joint, and considerable detail may be applied in constructed a model which relates the stresses in the joint to the applied forces. A simplified approach to capturing the relationship between the forces acting on the structure and the stress at the joint is therefore proposed, in order to reduce the complexity of input information and calculation effort required.

The stresses existing in a joint are assumed to be dominated by two components:

1. Stresses induced by the axial force carried by the attached diagonal bracing member

2. Stresses due to bending induced in the attached diagonal bracing member by local hydrodynamic forces

Therefore, the following steps must be taken:

1. The global load or story shear due to the application of H_f must be calculated
2. The component of the story shear carried axially by the brace attached to the joint must be found
3. The bending moment due to local hydrodynamic forces from H_f at the brace end connected to the joint must be calculated

With the axial force and the moment known, the stress at the brace-joint interface may be estimated using simple mechanics:

$$\sigma \frac{F_{axial}}{A_{brace}} + \frac{M_{end} r_{brace}}{I_{brace}}$$

The stresses due to axial force and bending may be further modified by stress concentration factors in order to relate the simply-calculated stresses to the peak stress which exists in the welded region:

$$S_p = (SCF)\sigma_{axial} + (SCF)\sigma_{bending}$$

3.4.1 Axial Force in Brace

To find the axial force in the brace attached to the joint under evaluation, it is first necessary to find the load carried by the jacket bay in which the brace is located. The amount of load actually carried by the brace itself is estimated by considering the brace

to be part of a parallel system of elements which all share the same lateral displacement imposed by the load on the bay.

The lateral forces imposed by H_f on the structure may be calculated as described by Mortazavi (1996). The story shear existing at the midpoint of the jacket bay in which the brace is located is taken as the force carried by that bay.

The bay in which the attached brace is located is considered to resist the applied load as a system of parallel elements, those elements being the braces in the bay. Assuming that rigid framing exists above and below the bay, all braces in the bay will therefore share the same imposed lateral displacement due to the load. The lateral stiffness for the bay may be formulated as:

$$k_{bay} = \sum k_i \cos^2 \vartheta_i$$

where:

k_i = axial stiffness of each individual brace, i.e. $\frac{EA}{L}$

θ_i = angle between each brace and the horizontal

The lateral displacement of the bay may then be calculated as:

$$\Delta_{bay} = \frac{V_{bay} - F_L}{k_{bay}}$$

where:

V_{bay} = lateral load carried by bay

F_L = lateral load carried by battered legs

The axial force in an individual brace is then calculated according to:

$$F_{axial\,i} = k_i \Delta_{bay} \cos \vartheta_i$$

3.4.2 Moment at Brace End

To find the moment at the brace end, i.e. where it connects to the joint being evaluated, it is necessary to find the local distributed load along the length of the brace. Then, together with assumptions as to the end fixity of the brace at the joint, this may be used to calculate the moment at the end of the brace by simple beam theory.

The distributed load w on individual braces due to H_f may be calculated as described by Mortazavi (1996). By conservatively assuming the ends of the brace to be completely fixed (and hence may develop large bending moments), the moment at the brace end may be found from:

$$M_{end} = \frac{wL^2}{12}$$

Together with the estimate of axial force in the brace, the moment may be used to calculate the stress at the brace-joint interface.

3.5 Using the Fatigue Analysis Feature of the ULSLEA Program

The procedures described in the previous section have been implemented in the ULSLEA program to allow damage estimates and fatigue life to be calculated for the main diagonal tubular joints of jacket-type structures. The user must supply all relevant platform global and local input data, as when conducting a **STORM** analysis. For environmental conditions, the user should assign the recommended fatigue design wave

height (taken from Figure C.3.7-3 of API RP 2A-LRFD or specified by user) and appropriate wave period, together with zero wind and zero current.

When the user selects **FATIGUE** from the **ANALYSIS** menu, an input screen (Figure 3.1) will appear, prompting the user to enter data on K and m (S-N curve parameters), H_0, H_1, ξ_0, ξ_1, N_0, N_1 and $T_{spectrum}$ (wave height distribution parameters), g_{stress} and R (stress response parameters), and the exposure period T_d.

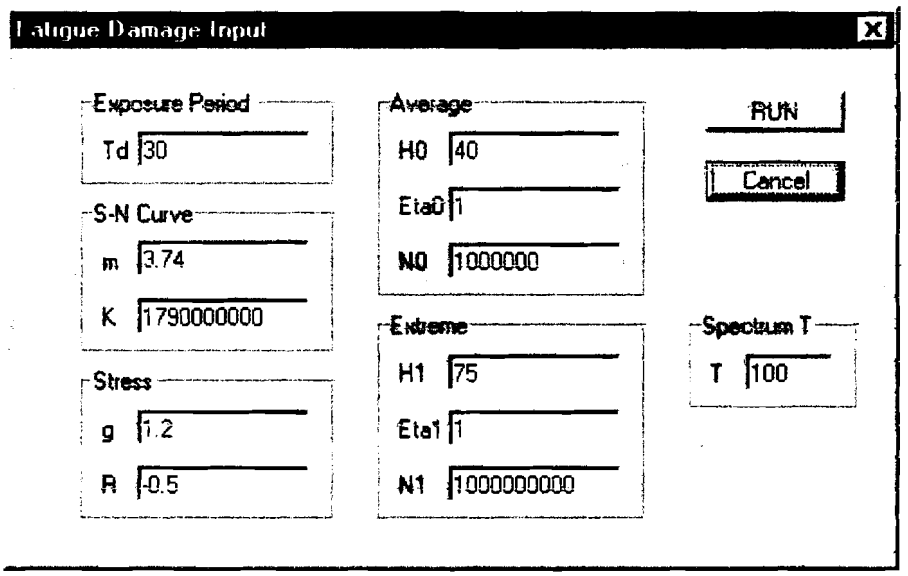

Figure 3.1: Fatigue Analysis Input Screen

The data which the user inputs using this screen is not stored to the platform data file; this will be changed in future program versions. A set of default data based on Commentary F from API RP 2A-LRFD (1993) has been assigned to the screen; the user may change this data as desired. Stress concentration factors for both axial load and in-plane bending are automatically determined by the program according to Table Commentary F.1-1 from API RP 2A-LRFD (1993); the "Joint Bias" value in the **BIASES AND UNCERTAINTIES** menu option from the **INPUT** menu (Figure 3.2) may be used to scale these stress concentration factors up or down (see Table 3.1).

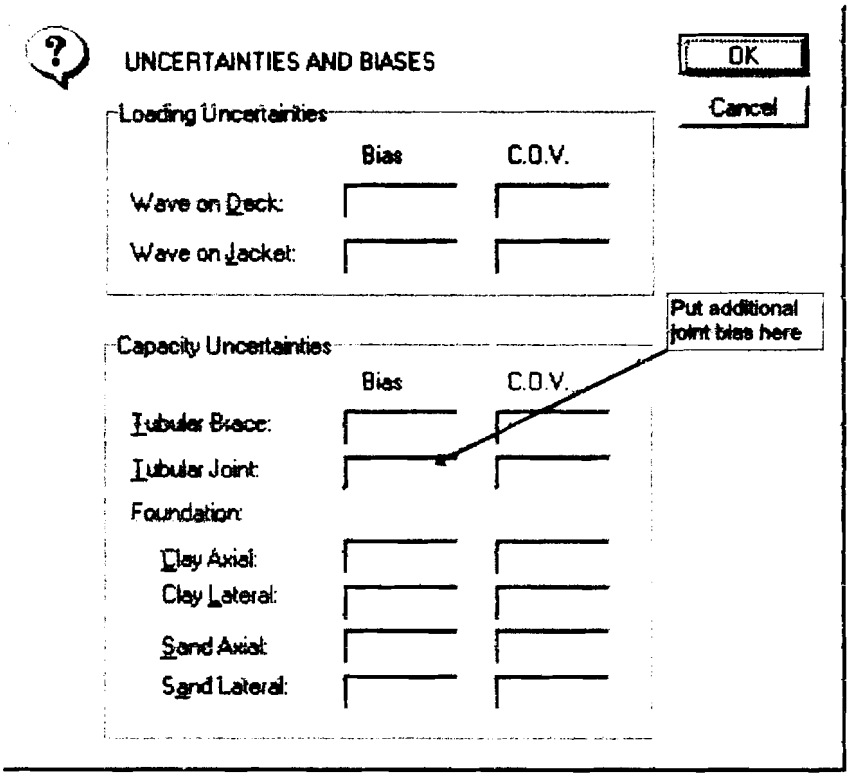

Figure 3.2: Biases and Uncertainties Input Screen

Connection	Axial *SCF*	In-Plane Bending *SCF*
Chord K	$1.8\sqrt{\gamma\tau}\sin\vartheta$	$1.2\sqrt{\gamma\tau}\sin\vartheta$
Chord T and Y	$3.06\sqrt{\gamma\tau}\sin\vartheta$	$2.04\sqrt{\gamma\tau}\sin\vartheta$
Chord X, $\beta < 0.98$	$4.32\sqrt{\gamma\tau}\sin\vartheta$	$2.88\sqrt{\gamma\tau}\sin\vartheta$
Chord X, $\beta \geq 0.98$	$3.06\sqrt{\gamma\tau}\sin\vartheta$	$2.04\sqrt{\gamma\tau}\sin\vartheta$
All Braces	$1.0+0.375\left(1+\sqrt{\tau/\beta}SCF_{chord}\right)\geq 1.8$	

$$\beta = d/D \qquad \tau = t/T \qquad \gamma = D/(2T)$$

where: d = brace diameter

D = chord diameter

t = brace thickness

T = chord thickness

θ = angle between chord and brace axes

Table 3.1: Tubular Joint *SCFs*

When the "RUN" option button is selected from the fatigue analysis input screen, the program determines the accumulated fatigue damage for all main diagonal joints, as well as the expected fatigue life of those joints. This information is output in tabular form on Sheet5 of ULSLEA.xls; it may be accessed by selecting **FATIGUE DAMAGE** from the **OUTPUT** menu. ULSLEA v3.0 beta does not have a provision for printing this data; however, this will be addressed in future versions.

4.0 STRENGTH-LEVEL EARTHQUAKE ANALYSIS OF PLATFORMS

4.1 Overview

Given the increasing importance of seismic considerations for offshore structures, effort has been directed towards developing procedures by which earthquake analysis of jacket-type structures can be conducted using the ULSLEA program. The scope of the initial effort has been directed towards strength-level analysis of platforms, together with providing the means to find accelerations relevant to the design of deck-mounted equipment. A detailed report on the ULSLEA program earthquake analysis methodology has been provided in a companion report by Stear and Bea (1997); the principal results of the development effort are summarized below:

1. Procedures have been implemented which allow for determination of up to three platform horizontal mode shapes and periods for the principal directions of response, as well as for estimation of the fundamental period for vertical response. No torsional effects are considered; it is assumed the platforms analyzed have mass and stiffness symmetry. Foundation effects are assumed to be concentrated in the first mode; a period-lengthening approach proposed by Veletsos (1977) is used to account for foundation flexibility. Foundation axial and lateral stiffnesses are modeled in accordance with suggestions made by Penzien (1975). Added mass is determined by considering all piles, jacket legs, and conductors to be flooded, and by considering a variable external hydrodynamic mass scaled both by a coefficient provided by the user, and by proximity to the free surface. Braces are assumed to be unflooded; the mass of grout in braces is included in the mass determination.
2. The periods and mode shapes determined by the program are used in response spectrum analysis to estimate the peak load demands on the structure and foundation. The response spectrum from API RP 2A-LRFD (1993) is used to find the spectral acceleration appropriate to each mode; vertical zero period acceleration is assumed to

be 50% of that for horizontal. The user is allowed to specify either a square-root-sum-of-the-squares (SRSS) or absolute sum (ABS) modal combination rule.

3. A procedure developed by Bowen and Bea (1995) has been implemented to allow for the determination of appropriate deck accelerations for use in designing equipment. For an equipment period provided by the user, acceleration for both principal directions and the vertical will be calculated.
4. Output is provided in terms of graphical display of horizontal shear demands for the principal directions of response, axial load demands on piles from combined horizontal and vertical response (an SRSS combination is used), and safety indices of the structure and foundation components. In addition, the platform horizontal and periods and mode shapes and the vertical fundamental period are presented in tabular form, along with the accelerations appropriate to deck-mounted equipment.

4.2 Using the Earthquake Analysis Feature of the ULSLEA Program

To perform a strength-level earthquake analysis of an offshore platform, or to determine deck accelerations for mounted equipment, the user must supply all global and local platform data. Water depth and surge depth should be entered under environmental conditions; wind, wave and current data should be left blank (combined storm/earthquake cases are not considered). When the user selects **EARTHQUAKE** from the **ANALYSIS** menu, an input screen appears (Figure 4.1), prompting the user to input data on the modal combination rule to be used (SRSS or ABS), the API response spectrum (soil type selection, zero-period acceleration *ZPA*, and spectrum coefficient of variation), the external hydrodynamic mass coefficient C_m, the effective shear modulus G and poisson's ratio υ of the foundation medium, and any foundation stiffness biases desired by the user.

When the user selects the "OK" option button, another input screen appears (Figure 4.2), prompting the user to input data for any conductors or risers the platform supports, along with the period of any mounted equipment for which deck accelerations are desired. The

data which the user inputs using these screens is not stored to the platform data file; this will be changed in future program versions.

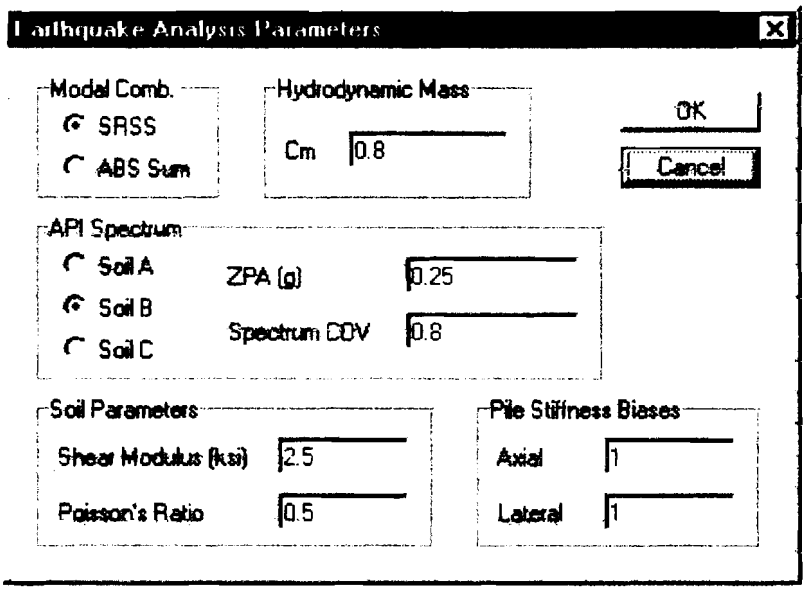

Figure 4.1: Earthquake Analysis Input Screen

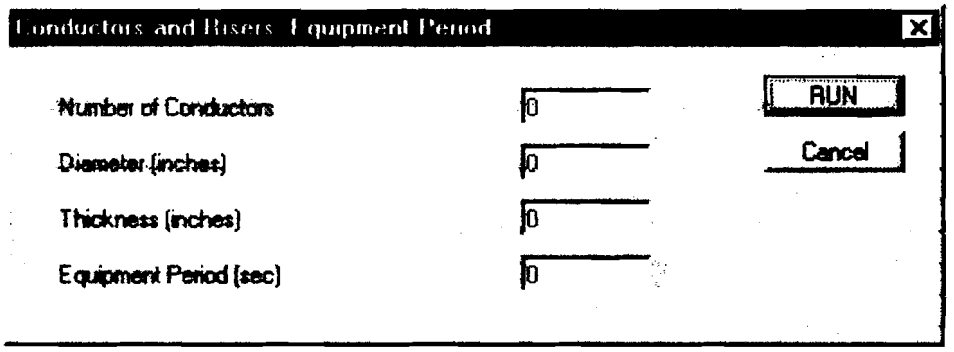

Figure 4.2: Appurtenances and Equipment Period Input Screen

When "RUN" is selected, the program determines the horizontal mode shapes and periods of the platform, the vertical period, the horizontal load demands on the structure

and foundation, and the axial load demands on the foundation. In addition, safety indices for the structure and foundation are also calculated. The horizontal load demand graphs may be seen by selecting the **END-ON RESPONSE** and **BROADSIDE RESPONSE** menu items from the **OUTPUT** menu. Vertical loads are expressed as reserve strength ratios on the piles, which are displayed by selecting the **PILE AXIAL** menu item from the **OUTPUT** menu. Platform component safety indices may be seen by selecting the **END-ON RELIABILITY** or **BROADSIDE RELIABILITY** menu items from the **OUTPUT** menu. These safety indices are developed by assuming the spectral ordinates of the API (1993) spectrum used to calculate loads during the response spectrum analysis are mean (unbiased) values. Finally, platform horizontal mode shapes and periods, the vertical period, and the acceleration of any deck-mounted equipment may be seen in tabular form (located on Sheet5 of ULSLEA.xls) by selecting the **PERIODS AND MODE SHAPES** menu item from the **OUTPUT** menu. ULSLEA v3.0 beta does not have a provision for printing the period and mode shape data; however, this will be addressed in future versions.

5.0 PROGRAM ADAPTATION FOR OTHER PLATFORM CONFIGURATIONS

In addition to developing procedures for fatigue and earthquake analysis, effort has also been devoted to adapting the ULSLEA program to allow for the assessment of four additional offshore platform configurations: multi 4-leg jackets, tripod jackets, and guyed and braced monopods/caissons. In the following sections, each of these types of structure are described, and the special aspects of their analysis are summarized. Finally, procedures for assessing these structures using ULSLEA v3.0 beta are given.

5.1 Multi 4-Leg Jacket Structures

The multi 4-leg jacket structures addressed by ULSLEA v3.0 beta consist of a large deck structure supported by four identical 4-leg jackets, one located at each corner (see Figure 5.1). It is assumed that the jackets share the deck loads (both vertical and lateral) equally; however, local forces on the individual jackets may differ depending on the load. For example, considering the large distance between the individual jackets, there will be significant spatial variation in the water depth and water particle kinematics acting on the jackets as a wave passes the structure.

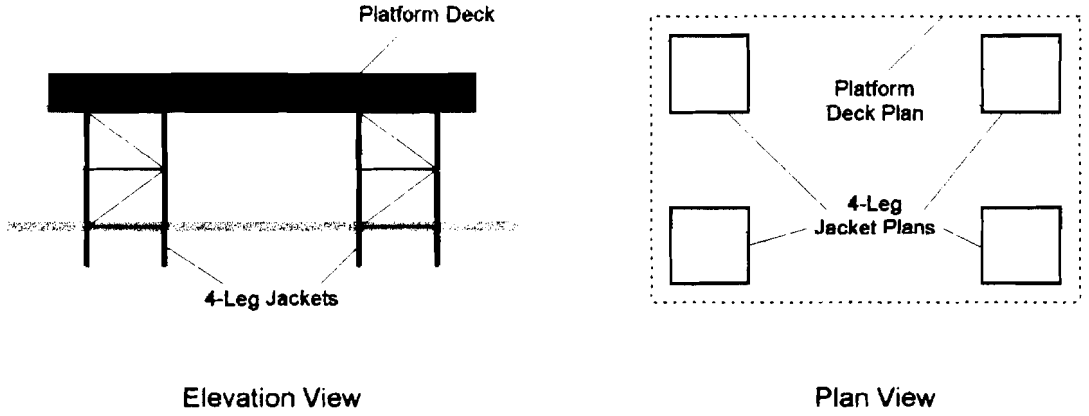

Figure 5.1: Multi 4-Leg Jacket Platform

5.1.1 Special Considerations

The platform capacity is assumed to be governed by failure in the supporting jackets. If one jacket fails, the entire platform is considered to have failed. Hence, an ULSLEA analysis of a multi jacket platform will entail the analysis of one of the supporting 4-leg jackets. The capacities of the individual jackets may be readily determined using the jacket bay and pile capacity procedures already contained in the ULSLEA program. However, there are several important aspects of load transfer and load sharing between the jackets which need to be addressed, in order to determine appropriate loads for the individual jackets.

Considering loading only on the principal axes of the platform, the structure may be considered to act as a portal frame as shown in Figure 5.2.

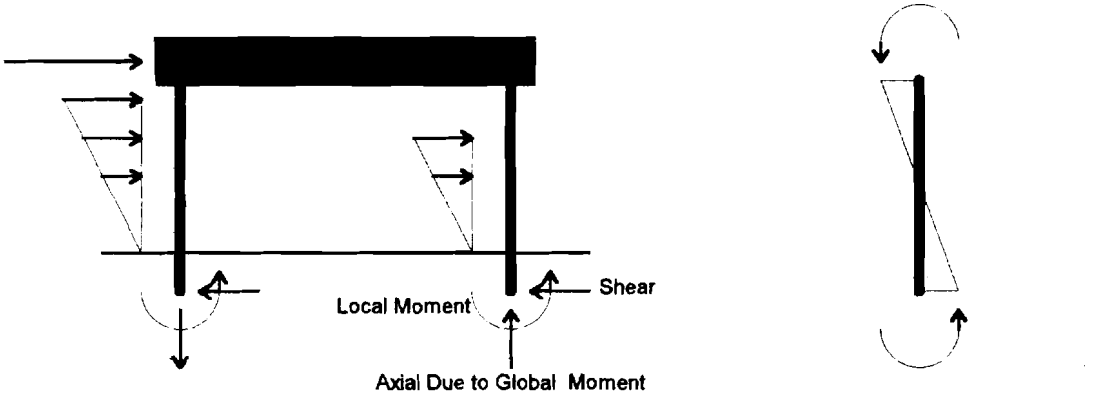

Figure 5.2: Loads on Multi Jacket Platform

In addition to local overturning moments and horizontal shears at the base of each jacket, there will also be axial forces induced by the global overturning moment of the entire platform. Furthermore, the distribution of bending moment in each jacket will be

changed by the presence of moment-resisting connections at the deck level. Also, if the spatial variation in water particle kinematics is taken into account, the hydrodynamic loads on the individual jackets will not be the same. Due to the presence of the rigid deck constraining the jackets together, there will be sharing of these local jacket forces, which will result in changes to the imposed shear on the individual jackets. However, this sharing may be conservatively neglected.

As the majority of these structures are in fairly shallow water, and the supporting jackets tend to be unbattered, the local moment effects will be neglected. This also ensures the foundation overturning moment is not underestimated. However, the effective jacket bending stiffness (used in modifying first mode periods for modal analysis, see Stear and Bea, 1997), will be based on $12EI/L$ as opposed to $3EI/L$ to represent the assumed fixed-fixed global behavior of the jacket. The global platform overturning forces are easily calculated providing the length between supporting jackets is provided by the user. Hence, the only adaptations necessary are to modify the effective jacket bending stiffness, allow for user input of the length separating the individual jackets, and developing program code to find the axial forces resulting from global overturning. This has been accomplished in ULSLEA v3.0 beta.

5.1.2 Analyzing a Multi 4-Leg Jacket with the ULSLEA Program

To analyze a multi 4-leg jacket platform using the ULSLEA program, the user must first input all global and local parameters appropriate to the definition of an individual jacket. In addition, the following must be done:

- The user should enter only one-quarter of the deck weight of the entire platform. The jackets are considered to share the deck load equally.
- The user should enter only one-quarter of the projected area of the platform deck structure in each direction. This is again due to the fact that the jackets are

considered to share the deck load equally. Also, the user should divide up projected areas due to boatlandings and appurtenances if they are not local to any given jacket.

- In the first **GLOBAL PARAMETERS** input screen (Figure 5.3), the user should enter the distance between the centerlines of the individual jackets in the "Middle Section Width" input box. This value will be used to determine the foundation reactions to global overturning.

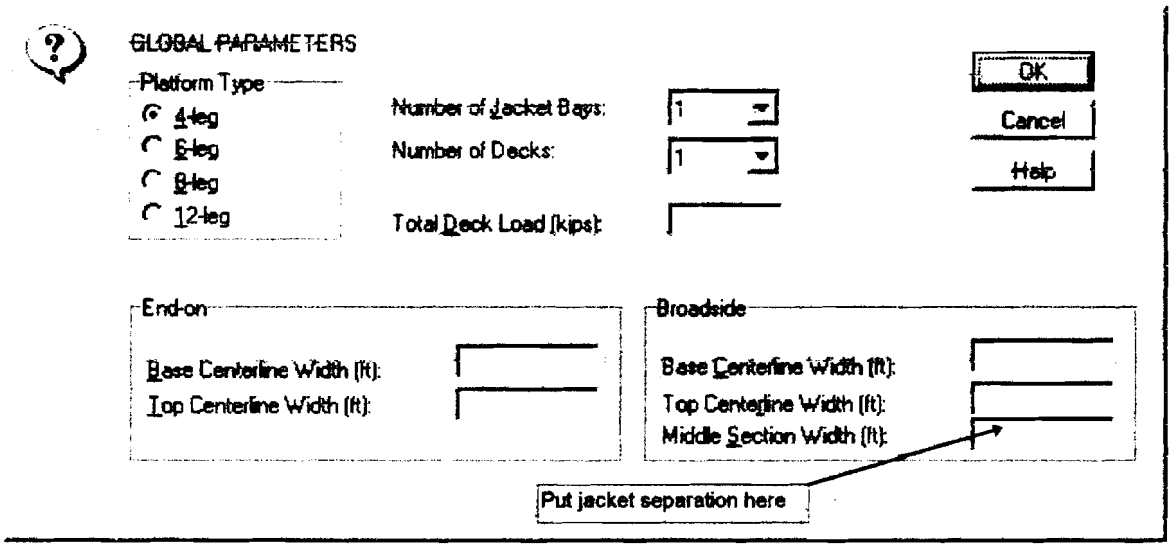

Figure 5.3: Global Parameters Input Screen

- The user should enter only one-quarter of the number of conductors when performing an earthquake analysis, as the dynamic properties of the platform are based upon one quarter of the mass and stiffness of the entire platform.

Prior to performing an analysis, the user must declare the structure type by accessing the **STRUCTURE CLASS** input screen from the **INPUT** menu (see Figure 5.4). As the structure class is not saved with the platform data file, the user must declare it each time the file is opened. The user may exercise any of the three calculation options, once all

necessary input has been provided. In this fashion, the suitability of the individual supporting jackets can be assessed.

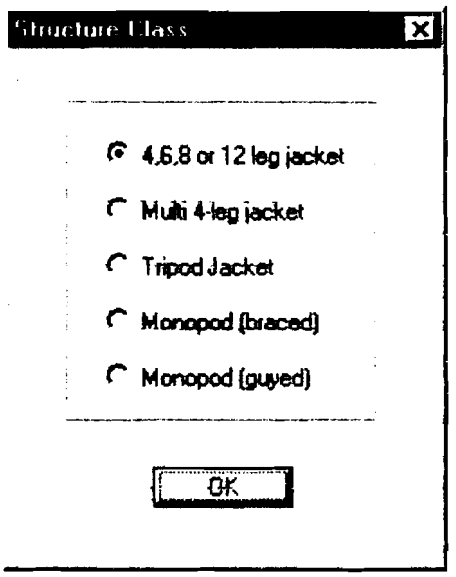

Figure 5.4: Structure Class Declaration Screen

5.2 Tripod Jackets

ULSLEA v3.0 beta has been adapted to allow for the analysis of tripod jacket structures which possess vertical faces or equal batter on all legs (i.e. the structure cannot have a single vertical leg and two battered legs). The plan may be either equilateral or isosceles.

The capacity and dynamic characteristics of these structures are determined for two directions of load: parallel to one face, and perpendicular to one face (see Figure 5.5). It is assumed that for the purposes of analyzing the broadside direction, the mass centers, aerodynamic and hydrodynamic load centers, and stiffness centers all coincide so that there are no torsion effects. It is important that the user is aware of the implications of this assumption; by neglecting torsion effects, some significant sources of load on the structure may be absent.

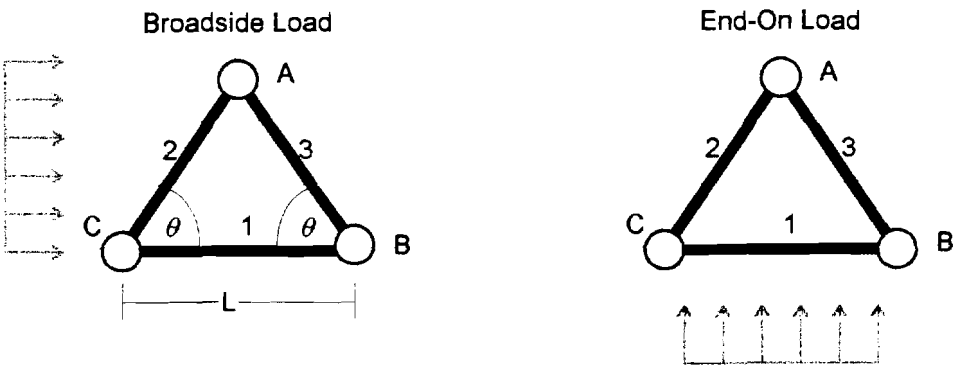

Figure 5.5: Tripod Jacket Load Directions

5.2.1 Special Considerations

Jacket bay capacity for the case of broadside loading will be assumed to be controlled by the brace or braces in Frame 1. However, the braces in Frames 2 and 3 will also carry some load. Hence, the capacity of each bay will be given by:

$$P_{uH_{bay}} = P_{uH_{MLTF\ Frame\ 1}} + \left(\frac{P_{uH_{MLTF\ Frame\ 1}}}{k_{H_{MLTF\ Frame\ 1}}}\right) k_{H_{Frame\ 1}} + \sum_{Frames} \left(\frac{P_{uH_{MLTF\ Frame\ 1}}}{k_{H_{MLTF\ Frame\ 1}}}\right) k_{H_{Frames\ 2,3}} \cos^2\theta$$

where:

$P_{uH_{MLTF\ Frame\ 1}}$ = horizontal component of ultimate axial strength of MLTF main diagonal member in Frame 1

$k_{H_{MLTF\ Frame\ 1}}$ = horizontal stiffness of MLTF member in Frame 1

$k_{H_{Frame\ i}}$ = horizontal stiffness of other members in Frames 1 (if not single-braced), 2, and 3

For the case of end-on loading, jacket bay capacity will be given by:

$$P_{uH_{bay}} = P_{uH_{MLTF\ Frames\ 2,3}} \sin\theta + \sum \left(\frac{P_{uH_{MLTF\ Frames\ 2,3}}}{k_{H_{MLTF\ Frames\ 2,3}}} \right) k_{H_{Frames\ 2,3}} \sin\theta$$

Similar adjustments may also be made for bay capacity estimates based on joint strength. The upper-bound capacity of these structures is set equal to their lower-bound, given the low redundancy.

The piles are assumed to share horizontal load equally. It is assumed that broadside overturning is resisted entirely by piles B and C. End-on overturning is resisted by all three piles; however, the axial load on pile A will be twice the individual axial loads on piles B and C.

Along with adjusting the bay capacity calculations and the foundation axial load determination procedures, substantial effort must be devoted to ensuring hydrodynamic forces on members and added mass on members are calculated correctly, given that many members do not lie in planes parallel or perpendicular to the directions of loading.

5.2.2 Analyzing a Tripod with the ULSLEA Program

When analyzing a tripod jacket using ULSLEA, the structure is considered to have one broadside load-resisting frame (Frame 1 in Figure 5.5) and two end-on load-resisting frames of equal size (Frames 2 and 3 in Figure 5.5). Frame 1 dimensions should be entered for end-on base and top dimensions in the first **GLOBAL PARAMETERS** input screen (Figure 5.6), while Frame 2/3 dimensions should be entered for base and top broadside dimensions. "Platform Type" should be set to 4-leg.

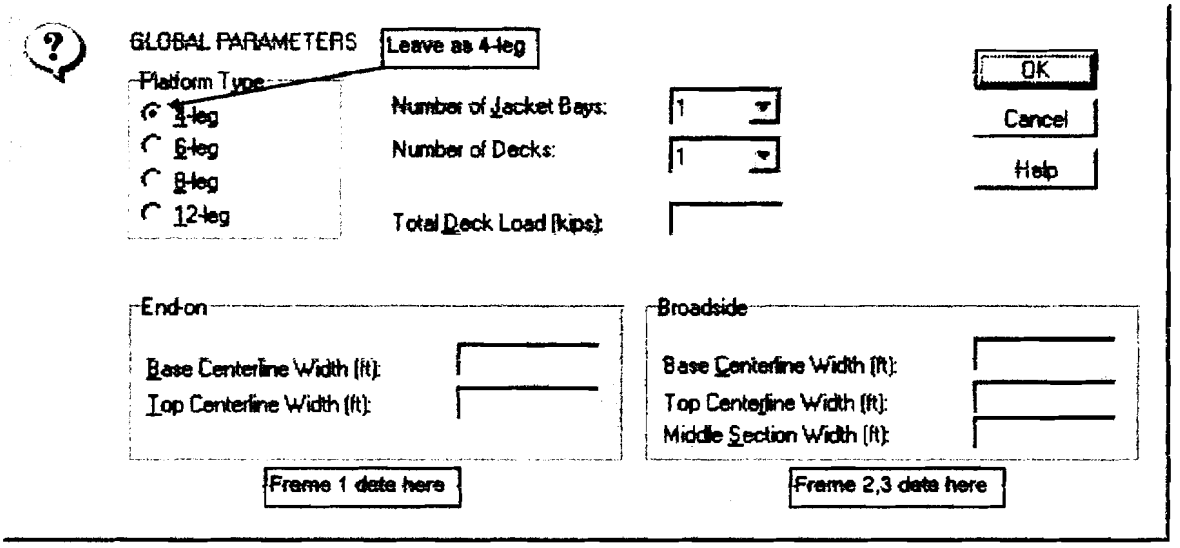

Figure 5.6: Inputting Tripod Global Dimensions

Similarly, Frame 1 braces should be entered as end-on elevation braces in the "Total # of Diagonal Braces" section of the third **GLOBAL PARAMETERS** input screen (Figure 5.7), while all Frame 2 and 3 braces should be entered as broadside elevation braces.

The angle (≤90°) needed to define to orientation of horizontal braces should be measured relative to an axis parallel to Frame 1 (the end-on face of the platform). When defining skirt piles for tripods, it is assumed the piles are grouped in equal numbers at the corners of the tripod. Hence, the user should enter the number of piles at one corner in the "End-On" box on the **SKIRT PILES** input screen (Figure 5.8). It is not necessary to check the "corner skirts" box on the **PRELIMINARY DESIGN** input screen when analyzing a tripod.

Prior to performing any calculations, however, the user must be sure to first assign the structure type as "Tripod" using the **STRUCTURE TYPE** menu item from the **INPUT** menu (Figure 5.4). Output is provided the same as for standard jackets.

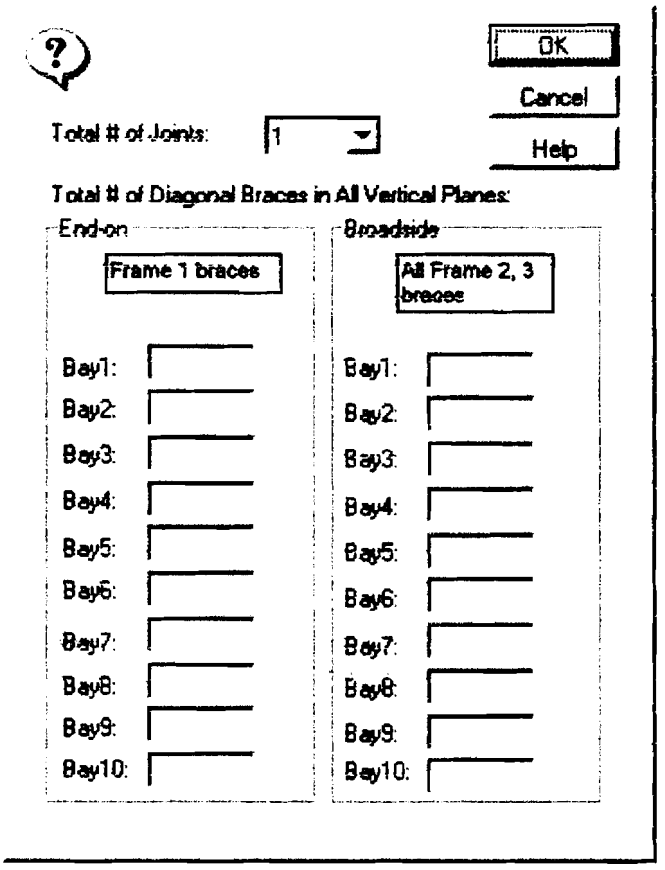

Figure 5.7: Declaring Number of Braces

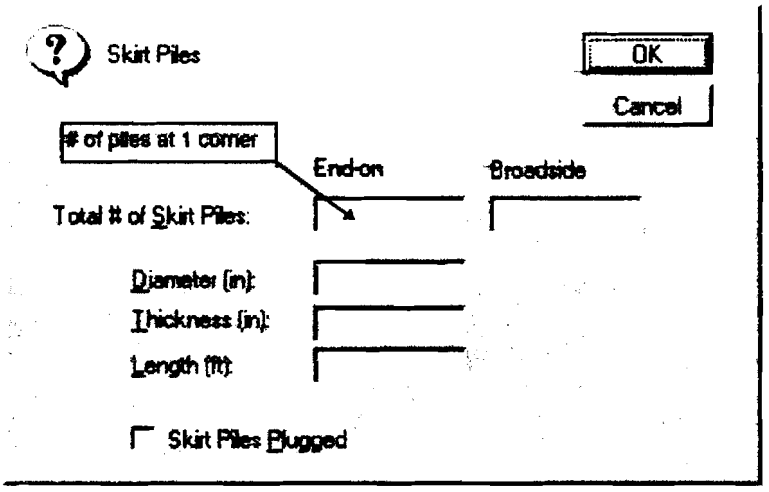

Figure 5.8: Declaring Tripod Skirt Piles

5.3 Monopods-Braced

ULSLEA v3.0 allows for the analysis of simple braced monopods (also referred to as braced caissons). Analysis is limited to the class of structures which consist of a single main pile (unjacketed) supported in orthogonal directions by single bracing piles (see Figure 5.9). It is assumed the braces are angled in such a way as to resist load primarily through axial action, and not by bending (hence, capacity estimates for platforms which have shallow batter braces will be very conservative). The bracing piles are connected to the main pile by short tubular sections.

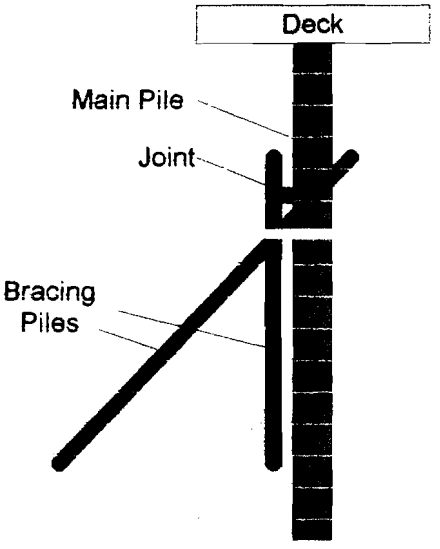

Figure 5.9: Braced Monopod

5.3.1 Special Considerations

- For the purpose of analysis, it is assumed that the directions of load coincide with the axes defined by the bracing piles. Furthermore, it is assumed that the lateral load center will be at or above the point of bracing, so that the platform may be treated as a supported cantilever, by which the majority of the horizontal load is presumed to be carried by the bracing member (see Figure 5.10).

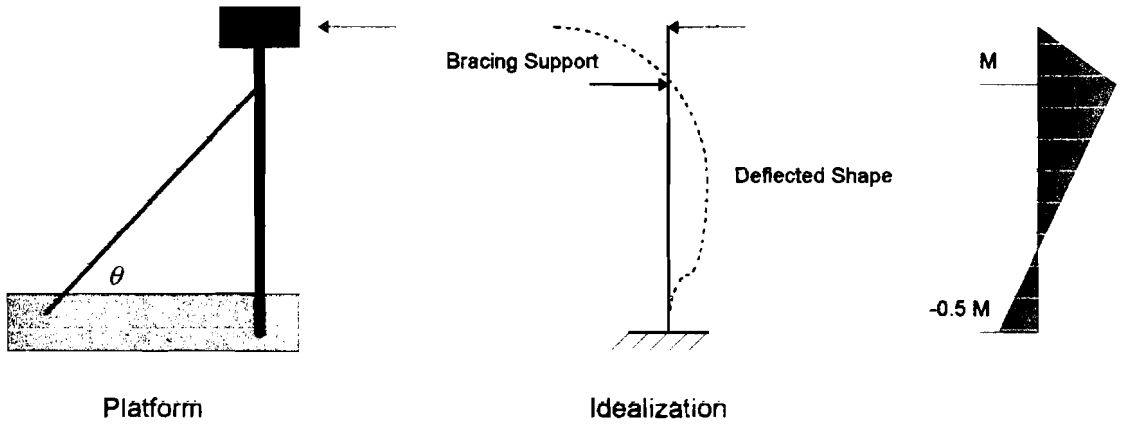

Figure 5.10: Platform Structural Idealization

From Figure 5.10, it can be seen that the platform has three modes of failure:

- Hinging of the main pile above the point at which the bracing piles are connected
- Failure of a bracing pile (either by yielding or buckling, by failure of the embedded end, or by failure of the connecting tubular section)
- Pull-out or plunging of the main pile from overturning

The ultimate capacity of the main pile above the point of bracing is assumed to be reached when a plastic hinge forms just above the bracing attachment point. The equivalent horizontal load capacity is thus:

$$P_{uH} = \frac{(M_u - Q\Delta)}{H_d}$$

where: M_u = ultimate moment (refer to Mortazavi, 1996)

Q = vertical deck load

H_d = distance from deck to bracing point

Δ = deck displacement at collapse

The deck displacement may be estimated from:

$$\Delta = M_u\left(\frac{H_d^2}{3EI_d}+\frac{H_d}{C_r}\right)$$

where: I_d = moment of inertia of main pile cross section

C_r = equivalent rotational stiffness coefficient representing main pile below bracing point and bracing pile (refer to Mortazavi, 1996)

The ultimate capacity of the braced section will be determined by yielding or buckling of the bracing pile, failure of the imbedded portion of the bracing pile, or by failure of the connecting tubular section. This capacity will be further reduced by the bending action of the section above the bracing attachment point, but will be increased somewhat by the horizontal stiffness contribution of the main pile. The effective horizontal capacity is given by the following:

$$P_{uH} = P_{uH_{brace}} + \frac{P_{uH_{brace}}}{k_{H_{brace}}} k_{H_{main\,pile}} - \frac{3M}{2H_u}$$

where: H_u = distance from bracing point to depth of ten pile diameters below mudline (equivalent point of base fixity for main pile)

M = moment induced by external loads at point of bracing attachment

$k_{H_{main\,pile}}$ = cantilever stiffness of main pile, taken as $\frac{3EI_d}{H_u^3}$

The ultimate horizontal load capacity of the bracing pile, $P_{uH_{brace}}$, should be the minimum of the following:

$\frac{A_c \sigma_y}{2}$ = shearing capacity of connecting tubular section

$P_{u_{brace}} \cos\theta$ = horizontal component of brace axial capacity (see Mortazavi, 1996); θ is the angle between the bracing pile axis and the horizontal

$P_{u_{axial}} \cos\theta$ = horizontal component of imbedded portion axial capacity (see Mortazavi, 1996)

Assuming the pile is subjected primarily to axial load, the effective horizontal stiffness of the bracing pile may be approximated by considering the imbedded and unimbedded sections as a series system:

$$k_{H_{brace}} = \left(\frac{\left(\frac{EA_{brace}}{L_{unimbedded}}\right)\left(\frac{iEA_{brace}}{L_{imbedded}}\right)}{\left(\frac{EA_{brace}}{L_{unimbedded}}\right) + \left(\frac{iEA_{brace}}{L_{imbedded}}\right)} \right) \cos^2\theta$$

where: i = bias on axial stiffness of imbedded portion

The last concern is that of pull-out or plunging of the main pile due to overturning. The axial capacity of this pile can be determined from relations documented by Mortazavi, 1996; the axial load on this pile may be approximated by:

$$F_{axial} = \frac{0.5M + M_{global\ overturning}}{L_{unimbedded} \cos\theta}$$

5.3.2 Analyzing a Braced Monopod with the ULSLEA Program

To analyze a braced monopod using ULSLEA v3.0 beta, the user should follow the steps listed below:

1. Enter global parameters for the structure as shown in Figure 5.11. The structure type should be left as 4-leg. The user should declare one jacket bay, the height of which goes from the brace attachment point to the mudline. The user should declare one main diagonal in each direction, and one joint type.

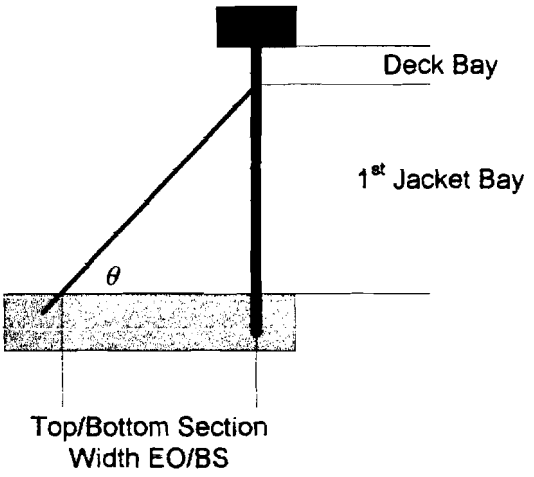

Figure 5.11: Input Data for Braced Monopod

2. Enter the diameter and thickness of the main pile when prompted for "deck legs" and "jacket legs." The user may enter data on the main diagonals as in a standard jacket analysis.
3. Enter the diameter, thickness and imbedment of the main pile when prompted for foundation information. The "Skirt Piles" box should be checked.

"End-On Skirts" should be assigned a value of "1," while any desired pile axial stiffness bias should be assigned to "Broadside Skirts" (see Figure 5.12). The length should be the imbedded length of the braces.

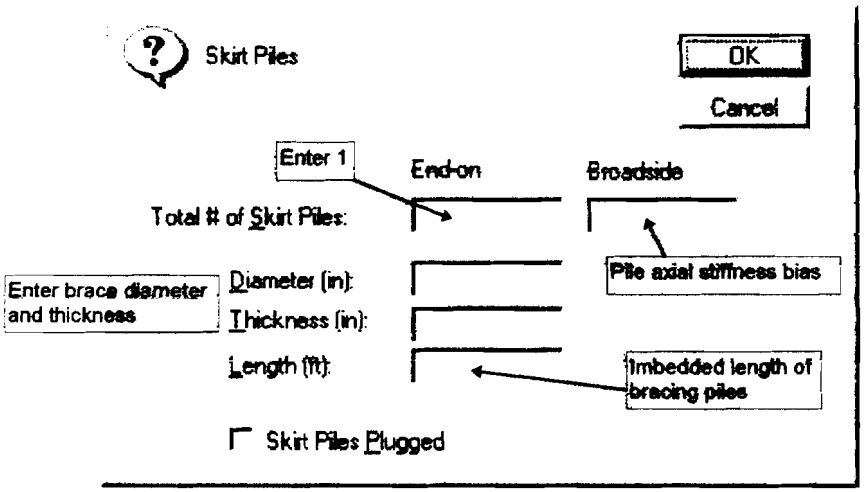

Figure 5.12: Bracing Pile Imbedded Characteristics

4. The diameter and thickness of the tubular connection between the bracing piles and the main pile should be entered as "Chord Diameter" and "Chord Thickness" on the **TUBULAR JOINTS** input screen. The joint type may be left as "K" (see Figure 5.13).

Prior to performing an analysis, the user must assign the structure type by accessing the **STRUCTURE TYPE** input menu (Figure 5.4) and choosing "Monopod (braced)." The braced monopod may then be analyzed using any of the three calculation options. Only lower-bound output is provided for the braced portion of the platform, and no distinction is made as to where the weakness in the joint-brace-imbedment system is. The horizontal foundation capacity represents the horizontal foundation capacity of the main pile. Similarly, the axial load diagrams also refer to the main pile. When performing a fatigue analysis, the fatigue life of the connecting tubular is checked. The shearing stresses in

this element are determined by considering all horizontal load to be transferred through this joint to the attached bracing pile. No inherent *SCF* is used; if one is desired, the user may assign one using the "Tubular Joints Bias."

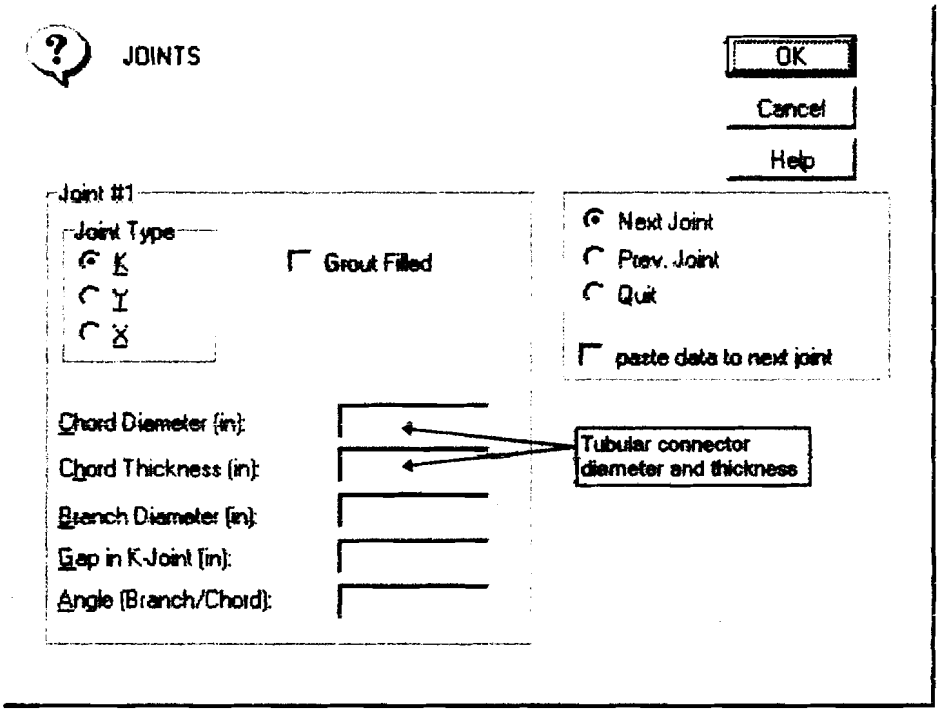

Figure 5.13: Entering Connection Data for Braced Monopod

5.4 Monopods-Guyed

ULSLEA v3.0 allows for the analysis of simple guyed monopods (also referred to as guyed caissons). Analysis is limited to the class of structures which consist of a single main pile (unjacketed) supported in three directions (120° apart) by single guy wires anchored by vertical piles (see Figure 5.14).

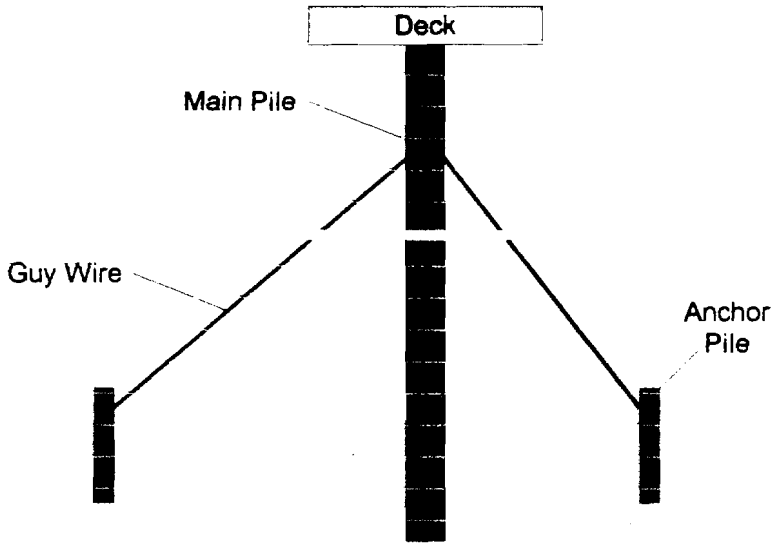

Figure 5.14: Guyed Monopod

5.4.1 Special Considerations

For the purpose of analysis, it is assumed that the direction of load coincides with the axis of one of the guy wires. In terms of behavior, the guyed monopod is similar to the braced monopod discussed in Section 5.3. The platform may be treated as a supported cantilever, by which the majority of the horizontal load is presumed to be carried by the guy wire. The principal failure modes are:

- Hinging of the main pile above the point at which the guy wires are attached
- Failure of a guy wire (either by yielding of the wire or attachment point, or by failure of the anchor pile)
- Plunging of the main pile from overturning

The capacity of the unsupported section of the guyed monopod may be expressed in a similar fashion as to the unsupported section of the braced monopod (see Section 5.3).

The horizontal capacity of the guy wire will be given by:

$$P_{uH} = P_{uH_{guy\ wire}} + \frac{P_{uH_{guy\ wire}}}{k_{H_{guy\ wire}}} k_{H_{main\ pile}} - \frac{3M}{2H_u} - F_{pretension} \cos\theta$$

The ultimate horizontal load capacity of the guy wire, $P_{uH_{guy\ wire}}$, should be the minimum of the following:

$A_{guy\ wire}\sigma_y \cos\theta$	=	tension capacity of guy wire
$\frac{A_c\sigma_y}{2\sin\theta}\cos\theta$	=	shearing capacity of connection to main pile; θ is the angle between the guy wire axis and the horizontal
$A_c\sigma_y$	=	tension capacity of connection to main pile
$\frac{P_{u_{axial}}\cos\theta}{\sin\theta}$	=	anchor pile tensile axial capacity (see Mortazavi, 1996)
$P_{u_{horizontal}}$	=	anchor pile horizontal capacity (see Mortazavi, 1996)

The effective stiffness of the guy wire-anchor system may be approximated by considering the wire and anchor pile to act as a series system:

$$k_{H_{guy\ wire}} = \left(\frac{\left(\frac{EA_{wire}}{L_{wire}}\right)\cos^2\theta\left(k_{H_{anchor}}\right)\left(\frac{k_{Z_{anchor}}}{\sin^2\theta\cos\theta}\right)}{\left(\frac{EA_{wire}}{L_{wire}}\right)\cos^2\theta\left(k_{H_{anchor}}\right)+\left(k_{H_{anchor}}\right)\left(\frac{k_{Z_{anchor}}}{\sin^2\theta\cos\theta}\right)+\left(\frac{EA_{wire}}{L_{wire}}\right)\cos^2\theta\left(\frac{k_{Z_{anchor}}}{\sin^2\theta\cos\theta}\right)} \right)$$

where: $k_{Z_{anchor}} = iEA_{anchor} / L_{anchor}$

$$k_{H_{anchor}} = 18.2Gr_{anchor}\frac{(1-\upsilon^2)}{(2-\upsilon)^2}$$

and: i = pile axial stiffness bias

G = soil effective shear modulus of elasticity

υ = poisson's ratio of soil

The anchor vertical and horizontal stiffnesses are based on approximations suggested by Penzien (1975); these are documented by Stear and Bea (1997). If the wires are pre-tensioned, the effective stiffness will be doubled.

The last concern is that of plunging of the main pile due to overturning. The axial capacity of this pile can easily be determined from relations documented by Mortazavi, 1996; the axial load on this pile may be approximated by:

$$F_{axial} = \frac{0.5M + M_{global\ overturning}}{L_{wire}\cos\theta}$$

It should be noted that in addition to forces due to deck vertical loads, this pile will also have loads due to the pre-tensioning of the supporting guy wires.

5.4.2 Analyzing a Guyed Monopod with the ULSLEA Program

To analyze a guyed monopod using ULSLEA v3.0 beta, the user should follow the steps listed below:

1. Enter global parameters for the structure as shown in Figure 5.15. The structure type should be left as 4-leg. The user should declare one jacket bay, the height of which goes from the wire attachment point to the mudline. The user should declare one main diagonal in each direction, and one joint type.
2. The characteristics of the main pile should be entered in the **FOUNDATION** input screen, whereas the characteristics of the anchor pile (it is assumed the anchors are identical) should be entered in the **SKIRT PILES** screen. "End-On Skirts" should be

assigned a value of "1," while any desired pile axial stiffness bias should be assigned to "Broadside Skirts."

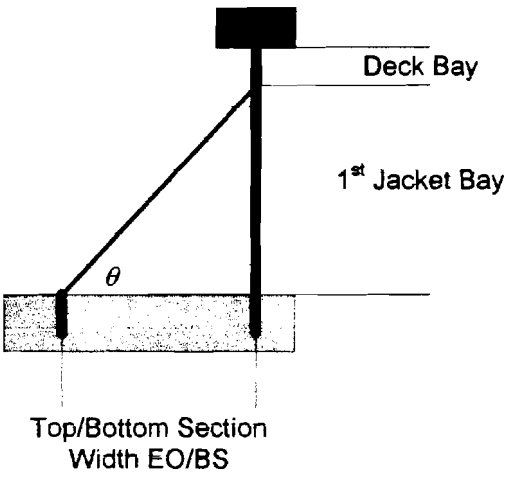

Figure 5.15: Input Data for Guyed Monopod

3. Due to the special nature of the guyed monopod with regards to input, additional foundation information is collected using the **JOINTS** input screen, in lieu of using an extra input screen (this will be changed in future versions of the program). The effective shear modulus and poisson's ratio should be entered as "Chord Thickness" and "Branch Diameter" respectively on the **JOINTS** input screen. The cross section area of the guy wire attachment point should be entered as "Chord Diameter" on the **JOINTS** input screen. Any pre-pretension force in the wires should be input as "Gap in K-Joint" on the **JOINTS** input screen (see Figure 5.16).
4. Enter the diameter and thickness of the main pile when prompted for "deck legs" and "jacket legs." The diameter of the supporting wire should be entered as main diagonal brace diameter.

Prior to performing an analysis, the user must assign the structure type by accessing the **STRUCTURE TYPE** input menu (Figure 5.4) and choosing "Monopod (guyed)." The guyed monopod may be analyzed using any of the three calculation options.

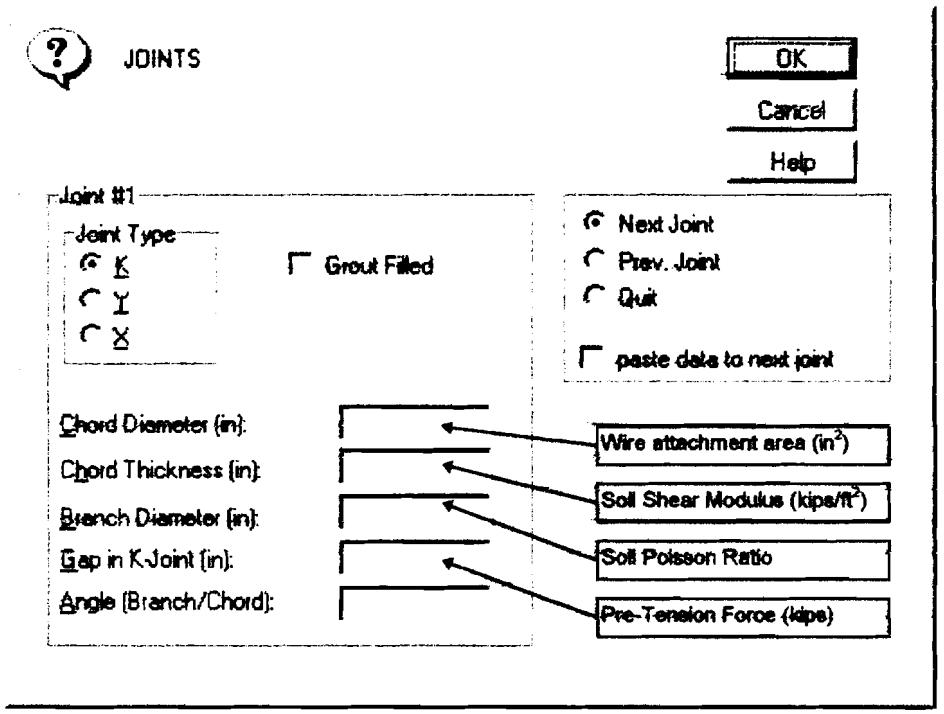

Figure 5.16: Additional Guyed Monopod Input

Only lower-bound output is provided for the braced portion of the platform, and no distinction is made as to where the weakness in the connection-wire-anchor system is. The horizontal foundation capacity represents the horizontal foundation capacity of the main pile. Similarly, the axial load diagrams also refer to the main pile. When performing a fatigue analysis, the fatigue life of the connection is checked. The stresses in this element are determined by considering all horizontal load to be transferred through the connection to the attached wire. No inherent *SCF* is used; if one is desired, the user may assign one using the "Tubular Joints Bias."

6.0 FUTURE ENHANCEMENTS

The following tasks are tentatively scheduled for the next phase (Phase IV) of the Screening Methodologies project:

- A series of parameter studies aimed at finding the dependence of global strength on local damage and repairs using the ULSLEA program. This study will be performed using three platforms.
- Development of ductility-level earthquake analysis routines.
- Diagonal loading formulations and capacity formulations for elements sensitive to diagonal loads.
- Adaptation of the program to allow for the analysis of 4-leg structures with one vertical face, and tripod structures with one vertical leg.
- Developing and documenting updated biases and uncertainties for joints, braces, and piles.
- Adapting the program interface to allow for the input of separate yield strengths for individual components of the structure (joint cans, braces).
- Improved graphical and printed output.
- Foundation elements allowing for layered soils, and contributions to stiffness and capacity from mud mats, mudline braces, and conductors.
- Algorithms allowing a user to calculate reliability sensitivity factors.
- Spatial variation of wave forces for platforms with large dimensions.
- Shallow water wave kinematics.
- Vertical loads and loading capacities for deck structures.

As part of the Phase IV Screening Methodologies effort, it is intended to add two features not included in the Phase IV project scope: the ability to analyze unbraced pile-supported platforms, and the ability to analyze platforms which have only skirt piles.

7.0 REFERENCES

American Petroleum Institute, "Recommended Practice for Planning, Designing, and Constructing Fixed Offshore Platforms - Load and Resistance Factor Design," API Recommended Practice 2A-LRFD (RP 2A-LRFD), 1st Edition, Washington, D. C., July 1993.

Aviguetero, T. A., and Bea, R. G., "ULSLEA: Parametric Studies of the Effects of Local Damage and Repairs on Global Lateral Load Capacity of a Typical Offshore Platform," Report to Joint Industry Project Sponsors, Department of Civil and Environmental Engineering, University of California at Berkeley, CA, December 1996.

Bowen, C. M., and Bea, R. G., "Simplified Earthquake Floor Response Spectra for Equipment on Offshore Platforms," Proceedings of the International Workshop on Wind and Earthquake Engineering for Coastal and Offshore Facilities, Department of Civil Engineering, University of California at Berkeley, CA, January 17-19 1995.

Luyties, W. H., and Geyer, J. F., "The Development of Allowable Fatigue Stresses in API RP2A,", Proceedings of the Offshore Technology Conference, OTC 5555, Houston, TX, 1987.

McDonald, D. T., Bando, K., Bea, R. G., Sobey, R. J., "Near Surface Wave Forces on Horizontal Members and Decks of Offshore Platforms," Final Report, Coastal and Hydraulic Engineering, Department of Civil Engineering, University of California at Berkeley, CA, December 1990.

Mortazavi, M., "A Probabilistic Screening Methodology for Use in Assessment and Requalification of Steel, Template-Type Offshore Platforms," Ph.D. Thesis, Department of Civil Engineering, University of California at Berkeley, CA, January 1996.

Nolte, K. G., and Hansford, J. E., "Closed-Form Expressions for Determining the Fatigue Damage of Structures Due to Ocean Waves," Proceedings of the Offshore Technology Conference, OTC 2606, Houston, TX, 1976.

Penzien, J., "Seismic Analysis of Platform Structure-Foundation Systems," Proceedings of the Offshore Technology Conference, OTC 2352, Houston, TX, May 1975.

Stear, J. D., and Bea, R. G., "Earthquake Analysis of Offshore Platforms," Report to Joint Industry Project Sponsors, Department of Civil and Environmental Engineering, University of California at Berkeley, CA, June 1997.

Stear, J. D., and Bea, R. G., "Using Static Pushover Analysis to Determine the Ultimate Limit States of Two Gulf of Mexico Steel Template-Type Platforms Subjected to Hurricane Wind and Wave Loads," Report to Minerals Management Service and Joint Industry Project Sponsors, Department of Civil and Environmental Engineering, University of California at Berkeley, CA, March 1996(A).

Stear, J. D., and Bea, R. G., "Comparison of Two Screening Methodologies for Steel Template-Type Platforms," Report to Joint Industry Project Sponsors, Department of Civil and Environmental Engineering, University of California at Berkeley, CA, June 1996(B).

Veletsos, A. S., "Dynamics of Structure-Foundation Systems," Structural and Geotechnical Mechanics: A Volume Honoring Nathan M. Newmark, editor: W. J. Hall, Prentice-Hall, Englewood Cliffs, NJ, 1977.

APPENDIX A: ULSLEA Program Development Update and Computer Code

The ULSLEA program is currently undergoing modification to increase its usability. In addition to new algorithmic changes over the past year, the basic structure of the program has been changed, with the emphasis being upon the creation of small modules to do specific tasks. The program has been organized into seven Excel 5.0 modules. These modules contain the following:

1. Variable declaration and data collection
2. Reliability calculation, printing input echo and graphing data
3. Fatigue analysis, tripod horizontal capacity, monopod capacities
4. Menu items, analysis control algorithms
5. Geometry definition, structure weights, preliminary design, tubular joint capacity, diagonal brace capacity, deck bay capacity, jacket bay capacity, foundation capacity, foundation stiffness, capacity profiles
6. Earthquake analysis: added mass, API response spectrum function, modal analysis routine, response spectrum analysis routine for load calculations
7. Hydrodynamics: kinematics determination, projected area determination, deck forces calculation, member forces calculation

Currently, data storage is performed using a linked Excel sheet INP.xls. Program development is currently oriented towards removing the dependence of the program on this sheet, and doing data collection and storage entirely within the ULSLEA.xls workbook. Effort will be made, however, to see that there is backwards compatibility with data files from ULSLEA v2.1 beta.

The best results have been obtained running ULSLEA on machines with Pentium 90 or better processors, and a minimum of 32 MB RAM.

ULSLEA
Ultimate Limit State Limit Equilibrium Analysis

This software is provided "as is" by the Marine Technology and Management Group (MTMG) at the University of California at Berkeley to sponsors of the research project "Screening Methodologies for Use in Offshore Platform Assessments and Requalifications." Any express or implied warranties of mechantability and fitness for a particular purpose are disclaimed. In no event shall the MTMG be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods and services; loss of use, data, or profits; or business interruption) however caused by any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of this software, even if advised of the possibility of such damage.

Program password (v3.0): ulsleamtmg

```
'
'          Module 1:
'
'  Variable declaration
'
Public wdep, sdep, vrh, wavh, wavp, cswl, cmdl, cprof, nleg, nbay, qdeck, pltype
Public ecdw(2), ecdh(2), edw(2), edh(2), bcw(2), tcw(2), msw
Public ndeck, deckw(2, 6), ok(6), uk(6), vcrest, cdd(6), wsc(6), decka(2, 6)
Public fhydro(2, 6), fhydroh(6), faero(2, 6), faeroh(6), deckh(6), faerobar(2), fhydrobar(2)
Public njoint, ndb(2, 30), bayh(30)
Public dld, dlt, jld(2, 30), jlt(2), dbtype(2, 30, 30), dbpos(2, 30, 30), theta(2, 30, 30), thetax(2, 30, 30)
Public dbconf(2, 30, 30), dbd(2, 30, 30), dbt(2, 30, 30), dbcond(2, 30, 30), dbjointi(2, 30, 30)
Public dbjointj(2, 30, 30), dbdam(2, 30, 30), dbrep(2, 30, 30), ddep(2, 30, 30), oos(2, 30, 30)
Public jtype(100), jgrout(100), jchd(100), jcht(100), jbrd(100), jbrt(100), jgap(100), jang(100), jpu(2, 30, 30), jput(100),
    jpuc(100), jpuh(2, 30, 30), jcap(2, 30), jcapbar(2, 100)
Public fy, e, kbuck, bres, stype, su1, su2, sphi, gammas, scour, cdj, cb, ds, lf
Public marineg, mg(30), pilet, piled, pilel, pgrout, plug, skirt, boatl(2), dequapp(30), dequ(2, 30)
Public spilet, spiled, spilel, splug, nskirt(2), nsk
Public nhb(30), hbd(30, 28), hbt(30, 28), hbl(30, 28), hbang(30, 28), hbequa(2, 30)
Public wdbias, wdcov, wjbias, wjcov, btbias, btcov, bcbias, bccov, jtbias, jtcov, jcbias, jccov
Public cabias, cacov, clbias, clcov, sabias, sacov, slbias, slcov
Public lh(2, 30), lht(2, 30), alpha(2), htotal
Public dbl(2, 30, 30), dblx(2, 30, 30), dbalpha(2, 30, 30)
Public phi(5), eta(5), wvel(100), cvel(100), vel(100), wvcrest, elev(100), velocity(30)
Public elevation(31), interval, f(2, 100), cumf(2, 100)
Public jbeta(100), jgamma(100), jqg(100), jdummy(100), jqbeta(100)
Public h(100, 30), m(2, 100, 30), mbar(2, 30), legf(2, 30), legfh(2, 30)
Public dla, dli, ibay1(2), dlzp, dlr, dlmp, dlmcr, dlm, dlcs(2), dlcm(2), dldelta(2), dlcap(2)
Public shear(2), dlmbar(2, 2)
Public dba(2, 30, 30), dbzp(2, 30, 30), dblam(2, 30, 30), dbdeque(2, 30, 30), dbdequb(2, 30, 30), dbdequebar(2, 30),
    dbdequbbar(2, 30), dbpe(2, 30, 30)
Public dbpy(2, 30, 30), dbpcr(2, 30, 30), dbmp(2, 30, 30), dbmcr(2, 30, 30), dbpcrl(2, 30, 30), dbr(2, 30, 30)
Public dbi(2, 30, 30)
Public dummye(2, 30), dummyb(2, 30)
Public dbw(30, 30), dbeps0(2, 30, 30), dbdelta(2, 30, 30), dbki(2, 30, 30)
Public dbpu(2, 30, 30), dbpuh(2, 30, 30), dbpuhu(2, 30, 30), epsilon, Formula, Formula1
Public dbalhpa(2, 30, 30), dbes(2, 30, 30), dbag(2, 30, 30), dbeg(2, 30, 30), dbatr(2, 30, 30), dbesdbeg(2, 30, 30), dbetr(2, 30,
    30), dbis(2, 30, 30)
Public dbig(2, 30, 30), dbitr(2, 30, 30), dbrtr(2, 30, 30), dbztr(2, 30, 30), dbastr(2, 30, 30)
Public dblamp(2, 30, 30), dbm(2, 30, 30), dbk(2, 30, 30)
Public dbpcrld(2, 30, 30), dbmcrd(2, 30, 30), dblamd(2, 30, 30), dbpcrd0(2, 30, 30), dbpcrd(2, 30, 30)
Public pilea, pilepy, pilezp, piler, pilemp, pilemcr, pilem, n(2)
Public spilea, spilepy, spilezp, spiler, spilemp, spilemcr, spilem
Public ff, wp, dummy, qcc, qtc, bca, a, b, c, cbar, seta, psi, pucbar, puc(2), bcl, nq, qmax, ffmax, fas, qcs, qts, bsa, bsl
Public kp, pusbar, pus(2)
Public swp, sqcc, sqtc, spuc(2), sqcs, sqts, spus(2)
Public cap(2, 30), capu(2, 30), rsrcc(2), rsrtc(2), rsrcs(2), rsrts(2), rsrc(2), rsrt(2)
Public lbcap(2, 31), lbcapbar(2, 100), ubcap(2, 31), ubcapbar(2, 100), fcap(2)
Public meanload(2, 30), meanmarg(2, 30), covload(2, 30), covcap(2, 30), covlfcap(2), covafcap(2)
Public sigmacap(2, 30), sigmaalpha(2, 30), sigmamarg(2, 30), sigmapu(2, 30), sumksq(2, 30)
Public meanmargacf(2), meanmargatf(2), sigmamargaf(2), betaacf(2), betaatf(2), beta(2, 30)
Public prelim, steelg, steelw, steelv, decklegsw, jacketw, pilew
Public kbay(2, 30), deckk(2)
Public masshb(30), addedmasshb(30, 3), massdb(30), addedmassdb(30, 3), pilelegmass(30)
Public pilelegaddedmass(30), skirtmass, skirtaddedmass, foundationmass, sfoundationmass
Public decklegmass, rho, Pi, elevcheck, cmused, g
Public mass(2, 30), zmass, groutg, zforce, crest, structuretype, offangle
'
'
'
Sub get_data()
'
'  This procedure collects user input from INP.XLS.
'
'  Created by: Merhdad Mortazavi
'  Created on: 1/22/96
'
'  Last Modified by:  James Stear
'  Last Modified on:  5/15/97
'
```

```
'
'  First begin assignment of global parameters
'
'  Assign user inputs to storm conditions
'
wdep = Workbooks("INP.xls").ActiveSheet.Cells(5, 8)
sdep = Workbooks("INP.xls").ActiveSheet.Cells(7, 8)
vrh = Workbooks("INP.xls").ActiveSheet.Cells(10, 8)
wavh = Workbooks("INP.xls").ActiveSheet.Cells(13, 8)
wavp = Workbooks("INP.xls").ActiveSheet.Cells(15, 8)
cswl = Workbooks("INP.xls").ActiveSheet.Cells(22, 8)
cmdl = Workbooks("INP.xls").ActiveSheet.Cells(24, 8)
cprof = Workbooks("INP.xls").ActiveSheet.Cells(18, 8)
'
'  Assign user inputs to global parameters for platform definition
'
pltype = Workbooks("INP.xls").ActiveSheet.Cells(52, 8)
If pltype = 1 Then
   nleg = 4
ElseIf pltype = 2 Then
   nleg = 6
ElseIf pltype = 3 Then
   nleg = 8
Else
   nleg = 12
End If
nbay = Workbooks("INP.xls").ActiveSheet.Cells(72, 8)
qdeck = Workbooks("INP.xls").ActiveSheet.Cells(76, 8)
bcw(1) = Workbooks("INP.xls").ActiveSheet.Cells(59, 8)
tcw(1) = Workbooks("INP.xls").ActiveSheet.Cells(61, 8)
bcw(2) = Workbooks("INP.xls").ActiveSheet.Cells(64, 8)
tcw(2) = Workbooks("INP.xls").ActiveSheet.Cells(66, 8)
msw = Workbooks("INP.xls").ActiveSheet.Cells(68, 8)
ndeck = Workbooks("INP.xls").ActiveSheet.Cells(73, 8)
'
'  Assign user inputs to deck areas and other deck characteristics
'
For i = 1 To ndeck
   uk(i) = Workbooks("INP.xls").ActiveSheet.Cells(815 + 7 * i, 8)
   ok(i) = Workbooks("INP.xls").ActiveSheet.Cells(816 + 7 * i, 8)
   deckw(2, i) = Workbooks("INP.xls").ActiveSheet.Cells(817 + 7 * i, 8)
   deckw(1, i) = Workbooks("INP.xls").ActiveSheet.Cells(818 + 7 * i, 8)
   cdd(i) = Workbooks("INP.xls").ActiveSheet.Cells(819 + 7 * i, 8)
   wsc(i) = Workbooks("INP.xls").ActiveSheet.Cells(820 + 7 * i, 8)
Next i
'
'  Assign user input to # of joints, # of main diagonals, and bay heights
'
njoint = Workbooks("INP.xls").ActiveSheet.Cells(666, 8)
For i = 1 To 2
   For j = 1 To 10
      ndb(i, j) = Workbooks("INP.xls").ActiveSheet.Cells(668 + 2 * (i - 1) + 4 * (j - 1), 8)
      If nbay > 10 Then ndb(i, j + 10) = Workbooks("INP.xls").ActiveSheet.Cells(715 + 2 * (i - 1) + 4 * (j - 1), 8)
      If nbay > 20 Then ndb(i, j + 20) = Workbooks("INP.xls").ActiveSheet.Cells(762 + 2 * (i - 1) + 4 * (j - 1), 8)
   Next j
Next i
For i = 0 To nbay
   bayh(i) = Workbooks("INP.xls").ActiveSheet.Cells(477 + 2 * i, 8)
Next i
If nbay > 14 Then
   For i = 0 To 15
      bayh(i + 15) = Workbooks("INP.xls").ActiveSheet.Cells(513 + 2 * i, 8)
   Next i
Else
End If
'
'  Now begin assignment of local parameters
'
'  Assign user input to deck legs and main diagonals
'
```

```
dld = Workbooks("INP.xls").ActiveSheet.Cells(343, 8)
dlt = Workbooks("INP.xls").ActiveSheet.Cells(345, 8)
For i = 1 To 2
jlt(i) = Workbooks("INP.xls").ActiveSheet.Cells(6 + 1399 * (i - 1), 16)
   For j = 1 To nbay
   jld(i, j) = Workbooks("INP.xls").ActiveSheet.Cells(8 + 1399 * (i - 1), 16 + 8 * (j - 1))
      For k = 1 To ndb(i, j)
      dbd(i, j, k) = Workbooks("INP.xls").ActiveSheet.Cells(14 + 1399 * (i - 1) + 46 * (k - 1), 16 + 8 * (j - 1))
      dbt(i, j, k) = Workbooks("INP.xls").ActiveSheet.Cells(16 + 1399 * (i - 1) + 46 * (k - 1), 16 + 8 * (j - 1))
      dbpos(i, j, k) = Workbooks("INP.xls").ActiveSheet.Cells(18 + 1399 * (i - 1) + 46 * (k - 1), 16 + 8 * (j - 1))
      dbtype(i, j, k) = Workbooks("INP.xls").ActiveSheet.Cells(23 + 1399 * (i - 1) + 46 * (k - 1), 16 + 8 * (j - 1))
      dbconf(i, j, k) = Workbooks("INP.xls").ActiveSheet.Cells(36 + 1399 * (i - 1) + 46 * (k - 1), 16 + 8 * (j - 1))
      dbjointi(i, j, k) = Workbooks("INP.xls").ActiveSheet.Cells(27 + 1399 * (i - 1) + 46 * (k - 1), 16 + 8 * (j - 1))
      dbjointj(i, j, k) = Workbooks("INP.xls").ActiveSheet.Cells(29 + 1399 * (i - 1) + 46 * (k - 1), 16 + 8 * (j - 1))
      dbcond(i, j, k) = Workbooks("INP.xls").ActiveSheet.Cells(48 + 1399 * (i - 1) + 46 * (k - 1), 16 + 8 * (j - 1))
      ddep(i, j, k) = Workbooks("INP.xls").ActiveSheet.Cells(32 + 1399 * (i - 1) + 46 * (k - 1), 16 + 8 * (j - 1))
      oos(i, j, k) = Workbooks("INP.xls").ActiveSheet.Cells(34 + 1399 * (i - 1) + 46 * (k - 1), 16 + 8 * (j - 1))
      Next k
   Next j
Next i
'
'  Assign user input to horizontal braces
'
For i = 1 To nbay + 1
   nhb(i) = Workbooks("INP.xls").ActiveSheet.Cells(2846, 16 + 8 * (i - 1))
   For j = 1 To 7
      hbd(i, j) = Workbooks("INP.xls").ActiveSheet.Cells(2806 + 5 * (j - 1), 16 + 8 * (i - 1))
      hbt(i, j) = Workbooks("INP.xls").ActiveSheet.Cells(2807 + 5 * (j - 1), 16 + 8 * (i - 1))
      hbl(i, j) = Workbooks("INP.xls").ActiveSheet.Cells(2808 + 5 * (j - 1), 16 + 8 * (i - 1))
      hbang(i, j) = Workbooks("INP.xls").ActiveSheet.Cells(2809 + 5 * (j - 1), 16 + 8 * (i - 1))
      hbd(i, j + 7) = Workbooks("INP.xls").ActiveSheet.Cells(2857 + 5 * (j - 1), 16 + 8 * (i - 1))
      hbt(i, j + 7) = Workbooks("INP.xls").ActiveSheet.Cells(2858 + 5 * (j - 1), 16 + 8 * (i - 1))
      hbl(i, j + 7) = Workbooks("INP.xls").ActiveSheet.Cells(2859 + 5 * (j - 1), 16 + 8 * (i - 1))
      hbang(i, j + 7) = Workbooks("INP.xls").ActiveSheet.Cells(2860 + 5 * (j - 1), 16 + 8 * (i - 1))
      hbd(i, j + 14) = Workbooks("INP.xls").ActiveSheet.Cells(2905 + 5 * (j - 1), 16 + 8 * (i - 1))
      hbt(i, j + 14) = Workbooks("INP.xls").ActiveSheet.Cells(2906 + 5 * (j - 1), 16 + 8 * (i - 1))
      hbl(i, j + 14) = Workbooks("INP.xls").ActiveSheet.Cells(2907 + 5 * (j - 1), 16 + 8 * (i - 1))
      hbang(i, j + 14) = Workbooks("INP.xls").ActiveSheet.Cells(2908 + 5 * (j - 1), 16 + 8 * (i - 1))
      hbd(i, j + 21) = Workbooks("INP.xls").ActiveSheet.Cells(2955 + 5 * (j - 1), 16 + 8 * (i - 1))
      hbt(i, j + 21) = Workbooks("INP.xls").ActiveSheet.Cells(2956 + 5 * (j - 1), 16 + 8 * (i - 1))
      hbl(i, j + 21) = Workbooks("INP.xls").ActiveSheet.Cells(2957 + 5 * (j - 1), 16 + 8 * (i - 1))
      hbang(i, j + 21) = Workbooks("INP.xls").ActiveSheet.Cells(2958 + 5 * (j - 1), 16 + 8 * (i - 1))
   Next j
Next i
'
'  Assign user input to tubular joints
'
For i = 1 To njoint
   jtype(i) = Workbooks("INP.xls").ActiveSheet.Cells(3015, 8 + 8 * i)
   jgrout(i) = Workbooks("INP.xls").ActiveSheet.Cells(3027, 8 + 8 * i)
   jchd(i) = Workbooks("INP.xls").ActiveSheet.Cells(3005, 8 + 8 * i)
   jcht(i) = Workbooks("INP.xls").ActiveSheet.Cells(3007, 8 + 8 * i)
   jbrd(i) = Workbooks("INP.xls").ActiveSheet.Cells(3009, 8 + 8 * i)
   jgap(i) = Workbooks("INP.xls").ActiveSheet.Cells(3011, 8 + 8 * i)
   jang(i) = Workbooks("INP.xls").ActiveSheet.Cells(3013, 8 + 8 * i)
   If njoint > 30 Then
      jtype(i + 30) = Workbooks("INP.xls").ActiveSheet.Cells(3046, 8 + 8 * i)
      jgrout(i + 30) = Workbooks("INP.xls").ActiveSheet.Cells(3058, 8 + 8 * i)
      jchd(i + 30) = Workbooks("INP.xls").ActiveSheet.Cells(3036, 8 + 8 * i)
      jcht(i + 30) = Workbooks("INP.xls").ActiveSheet.Cells(3038, 8 + 8 * i)
      jbrd(i + 30) = Workbooks("INP.xls").ActiveSheet.Cells(3040, 8 + 8 * i)
      jgap(i + 30) = Workbooks("INP.xls").ActiveSheet.Cells(3042, 8 + 8 * i)
      jang(i + 30) = Workbooks("INP.xls").ActiveSheet.Cells(3044, 8 + 8 * i)
   ElseIf njoint > 60 Then
      jtype(i + 60) = Workbooks("INP.xls").ActiveSheet.Cells(3077, 8 + 8 * i)
      jgrout(i + 60) = Workbooks("INP.xls").ActiveSheet.Cells(3089, 8 + 8 * i)
      jchd(i + 60) = Workbooks("INP.xls").ActiveSheet.Cells(3077, 8 + 8 * i)
      jcht(i + 60) = Workbooks("INP.xls").ActiveSheet.Cells(3079, 8 + 8 * i)
      jbrd(i + 60) = Workbooks("INP.xls").ActiveSheet.Cells(3071, 8 + 8 * i)
      jgap(i + 60) = Workbooks("INP.xls").ActiveSheet.Cells(3073, 8 + 8 * i)
```

```
      jang(i + 60) = Workbooks("INP.xls").ActiveSheet.Cells(3075, 8 + 8 * i)
   Else
End If
Next i
'
'  Assign user input to foundation, both main and skirt piles
'
pilel = Workbooks("INP.xls").ActiveSheet.Cells(209, 8)
piled = Workbooks("INP.xls").ActiveSheet.Cells(211, 8)
pilet = Workbooks("INP.xls").ActiveSheet.Cells(213, 8)
plug = Workbooks("INP.xls").ActiveSheet.Cells(214, 8)
skirt = Workbooks("INP.xls").ActiveSheet.Cells(215, 8)
spilet = Workbooks("INP.xls").ActiveSheet.Cells(226, 8)
spiled = Workbooks("INP.xls").ActiveSheet.Cells(224, 8)
spilel = Workbooks("INP.xls").ActiveSheet.Cells(228, 8)
splug = Workbooks("INP.xls").ActiveSheet.Cells(233, 8)
nskirt(2) = Workbooks("INP.xls").ActiveSheet.Cells(221, 8)
nskirt(1) = Workbooks("INP.xls").ActiveSheet.Cells(222, 8)
'
'  Assign user input to force coefficients for wind and wave loads
'
cdj = Workbooks("INP.xls").ActiveSheet.Cells(171, 8)
cb = Workbooks("INP.xls").ActiveSheet.Cells(173, 8)
ds = Workbooks("INP.xls").ActiveSheet.Cells(175, 8)
lf = Workbooks("INP.xls").ActiveSheet.Cells(177, 8)
marineg = Workbooks("INP.xls").ActiveSheet.Cells(178, 8)
For i = 0 To 30
   mg(i) = Workbooks("INP.xls").ActiveSheet.Cells(571 + 2 * i, 8)
Next i
'
'  Assign user inputs to boat landing and appurtenance projected areas
'
boatl(1) = Workbooks("INP.xls").ActiveSheet.Cells(374, 8)
boatl(2) = Workbooks("INP.xls").ActiveSheet.Cells(376, 8)
For i = 0 To 10
   dequapp(i) = Workbooks("INP.xls").ActiveSheet.Cells(378 + 2 * i, 8)
   If nbay > 10 Then dequapp(i + 10) = Workbooks("INP.xls").ActiveSheet.Cells(403 + 2 * i, 8)
   If nbay > 20 Then dequapp(i + 20) = Workbooks("INP.xls").ActiveSheet.Cells(430 + 2 * i, 8)
Next i
'
'  Assign user inputs to material and soil properties
'
fy = Workbooks("INP.xls").ActiveSheet.Cells(129, 8)
e = Workbooks("INP.xls").ActiveSheet.Cells(131, 8)
kbuck = Workbooks("INP.xls").ActiveSheet.Cells(133, 8)
bres = Workbooks("INP.xls").ActiveSheet.Cells(135, 8)
stype = Workbooks("INP.xls").ActiveSheet.Cells(137, 8)
su1 = Workbooks("INP.xls").ActiveSheet.Cells(121, 8)
su2 = Workbooks("INP.xls").ActiveSheet.Cells(143, 8)
sphi = Workbooks("INP.xls").ActiveSheet.Cells(123, 8)
gammas = Workbooks("INP.xls").ActiveSheet.Cells(125, 8)
scour = Workbooks("INP.xls").ActiveSheet.Cells(127, 8)
If bres = 0 Then bres = 1
'
'  Assign user inputs to uncertainties and biases
'
wdbias = Workbooks("INP.xls").ActiveSheet.Cells(292, 8)
If wdbias = 0 Then wdbias = 1
wdcov = Workbooks("INP.xls").ActiveSheet.Cells(293, 8)
wjbias = Workbooks("INP.xls").ActiveSheet.Cells(295, 8)
If wjbias = 0 Then wjbias = 0.9
wjcov = Workbooks("INP.xls").ActiveSheet.Cells(296, 8)
btbias = Workbooks("INP.xls").ActiveSheet.Cells(298, 8)
If btbias = 0 Then btbias = 1
btcov = Workbooks("INP.xls").ActiveSheet.Cells(299, 8)
bcbias = Workbooks("INP.xls").ActiveSheet.Cells(298, 8)
If bcbias = 0 Then bcbias = 1
bccov = Workbooks("INP.xls").ActiveSheet.Cells(299, 8)
jtbias = Workbooks("INP.xls").ActiveSheet.Cells(314, 8)
If jtbias = 0 Then jtbias = 1
```

```
jtcov = Workbooks("INP.xls").ActiveSheet.Cells(315, 8)
jcbias = Workbooks("INP.xls").ActiveSheet.Cells(314, 8)
If jcbias = 0 Then jcbias = 1
jccov = Workbooks("INP.xls").ActiveSheet.Cells(315, 8)
cabias = Workbooks("INP.xls").ActiveSheet.Cells(302, 8)
If cabias = 0 Then cabias = 1
cacov = Workbooks("INP.xls").ActiveSheet.Cells(303, 8)
clbias = Workbooks("INP.xls").ActiveSheet.Cells(305, 8)
If clbias = 0 Then clbias = 1
clcov = Workbooks("INP.xls").ActiveSheet.Cells(306, 8)
sabias = Workbooks("INP.xls").ActiveSheet.Cells(308, 8)
If sabias = 0 Then sabias = 1
sacov = Workbooks("INP.xls").ActiveSheet.Cells(309, 8)
slbias = Workbooks("INP.xls").ActiveSheet.Cells(311, 8)
If slbias = 0 Then slbias = 1
slcov = Workbooks("INP.xls").ActiveSheet.Cells(312, 8)
Windows("ulslea.xls").Activate
'
'
'
End Sub
```

```
'
' Module 2
'
Sub reliability()
'
' This procedure computes means and standard deviations for component capacities
' and loads according to MVFOSM approximation methods. These means and standard
' deviations are used to explicitly solve for the reliability indices of each component.
'
' Created by: Merhdad Mortazavi
' Created on: 1/22/96
'
' Last Modified by:  James Stear
' Last Modified on:  6/1/97
'
'
For i = 1 To 2
'
' Begin by computing safety indices for deck and jacket bays
'
  For j = 0 To nbay
'
' Find coefficients of variation for deck and jacket bay capacities
' Note: Mcr COV is assumed to be 0.1, and Pcrl COV is assumed to be 0.1
'
    If j = 0 And bayh(0) <= 1.5 Then
      sigmacap(i, j) = (sigmaalpha(i, 1) ^ 2 * sumksq(i, 1) + (covload(i, 1) * 2 * legfh(i, 1)) ^ 2) ^ 0.5
    ElseIf j = 0 And bayh(0) > 1.5 Then
      sigmacap(i, j) = (((0.1 * dlmcr) * 2 * nleg / bayh(0) * Cos(3.14 / 2 * qdeck / nleg / fy / dla)) ^ 2 + ((0.1 * fy * dla) * (2 *
          dlmcr * qdeck / (bayh(0) * fy ^ 2 * dla ^ 2) * Sin(Pi * qdeck / nleg / 2 / fy / dla))) ^ 2 + 2 * (0.1 * dlmcr) * 2 * nleg /
          bayh(0) * (Cos(3.14 / 2 * qdeck / nleg / fy / dla)) * (0.1 * fy * dla) * (2 * dlmcr * qdeck / (bayh(0) * fy ^ 2 * dla ^ 2) *
          Sin(Pi * qdeck / nleg / 2 / fy / dla))) ^ 0.5
    ElseIf j = 1 And structuretype = "monopod_braced" Or structuretype = "monopod_guyed" Then
'
' COV of monopod braces is assuumed to be 40%
'
      sigmacap(i, j) = lbcap(i, j) * 0.4
    Else
      sigmacap(i, j) = (sigmaalpha(i, j) ^ 2 * sumksq(i, j) + (covload(i, j) * 2 * legfh(i, j)) ^ 2) ^ 0.5
    End If
    covcap(i, j) = sigmacap(i, j) / lbcap(i, j)
'
' Now compute beta
'
    If meanload(i, j) = 0 Then meanload(i, j) = 0.01
    meanmarg(i, j) = Application.Ln(Application.Max((lbcap(i, j) / meanload(i, j)), 0.001) * ((1 + covload(i, j) ^ 2) / (1 +
        covcap(i, j) ^ 2)) ^ 0.5)
    sigmamarg(i, j) = (Application.Ln(1 + covcap(i, j) ^ 2) + Application.Ln(1 + covload(i, j) ^ 2)) ^ 0.5
    If sigmamarg(i, j) = 0 Then
      beta(i, j) = 0
    Else
      beta(i, j) = meanmarg(i, j) / sigmamarg(i, j)
    End If
  Next j
'
' Now compute foundation safety indices
'
' stype = 1 refers to piles founded in sand
'
' First find coefficient of variation for horizontal capacity
'
  If stype = 1 Then
    covlfcap(i) = ((slcov * (pus(i) + spus(i))) ^ 2 + (covload(i, nbay) * 2 * legfh(i, nbay + 1)) ^ 2) ^ 0.5 / fcap(i)
    covafcap(i) = sacov
  Else
    covlfcap(i) = ((clcov * (puc(i) + spuc(i))) ^ 2 + (covload(i, nbay) * 2 * legfh(i, nbay + 1)) ^ 2) ^ 0.5 / fcap(i)
    covafcap(i) = cacov
  End If
  If meanload(i, nbay + 1) = 0 Then meanload(i, nbay + 1) = 0.01
```

```
    meanmarg(i, nbay + 1) = Application.Ln(Application.Max((fcap(i) / meanload(i, nbay + 1)), 0.001) * ((1 + covload(i, nbay) ^ 2)
        / (1 + covlfcap(i) ^ 2)) ^ 0.5)
    sigmamarg(i, nbay + 1) = (Application.Ln(1 + covlfcap(i) ^ 2) + Application.Ln(1 + covload(i, nbay) ^ 2)) ^ 0.5
'
'   Now compute foundation horizontal beta
'
    If sigmamarg(i, nbay + 1) = 0 Then
      beta(i, nbay + 1) = 0
    Else
      beta(i, nbay + 1) = meanmarg(i, nbay + 1) / sigmamarg(i, nbay + 1)
    End If
'
'   Now compute betas for axial tension and compression capacities
'
    If rsrc(i) = 0 Then
      meanmargacf(i) = 0
    Else
      meanmargacf(i) = Application.Ln(rsrc(i) * ((1 + covload(i, nbay) ^ 2) / (1 + covafcap(i) ^ 2)) ^ 0.5)
    End If
    If rsrt(i) = 0 Then
      meanmargatf(i) = 0
    Else
      meanmargatf(i) = Application.Ln(rsrt(i) * ((1 + covload(i, nbay) ^ 2) / (1 + covafcap(i) ^ 2)) ^ 0.5)
    End If
    sigmamargaf(i) = (Application.Ln(1 + covafcap(i) ^ 2) + Application.Ln(1 + covload(i, nbay) ^ 2)) ^ 0.5
    If sigmamargaf(i) = 0 Then
      betaacf(i) = 0
      betaatf(i) = 0
    Else
      betaacf(i) = meanmargacf(i) / sigmamargaf(i)
      betaatf(i) = meanmargatf(i) / sigmamargaf(i)
    End If
Next i

End Sub

'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Sub graphing()
'
'   Output
'
'   This portion of the program provides data to the charts on Sheet2, as well as printed
'   output for Sheet3.
'
anchoreoold = 0
anchorbsold = 0
anchorvold = 0

For i = 1 To 100
    Worksheets("Sheet2").Cells(8 + i, 2) = elev(i) - wdep
    Worksheets("Sheet2").Cells(8 + i, 3) = wvel(i)
    Worksheets("Sheet2").Cells(8 + i, 4) = cvel(i) / cb
    Worksheets("Sheet2").Cells(8 + i, 5) = wvel(i) + cvel(i) / cb
    Worksheets("Sheet2").Cells(8 + i, 6) = f(2, i)
    Worksheets("Sheet2").Cells(8 + i, 7) = cumf(2, i)
    Worksheets("Sheet2").Cells(8 + i, 8) = jcapbar(2, i)
    Worksheets("Sheet2").Cells(8 + i, 9) = lbcapbar(2, i)
    Worksheets("Sheet2").Cells(8 + i, 10) = ubcapbar(2, i)
    Worksheets("Sheet2").Cells(8 + i, 11) = f(1, i)
    Worksheets("Sheet2").Cells(8 + i, 12) = cumf(1, i)
    Worksheets("Sheet2").Cells(8 + i, 13) = jcapbar(1, i)
    Worksheets("Sheet2").Cells(8 + i, 14) = lbcapbar(1, i)
    Worksheets("Sheet2").Cells(8 + i, 15) = ubcapbar(1, i)
    anchoreo = Application.Max(cumf(2, i), jcapbar(2, i), lbcapbar(2, i), ubcapbar(2, i))
    If anchoreo > anchoreoold Then anchoreoold = anchoreo
    anchorbs = Application.Max(cumf(1, i), jcapbar(1, i), lbcapbar(1, i), ubcapbar(1, i))
    If anchorbs > anchorbsold Then anchorbsold = anchorbs
    anchorv = wvel(i) + cvel(i) / cb
    If anchorv > anchorvold Then anchorvold = anchorv
```

```
Next i
For i = 1 To 100
   Worksheets("Sheet2").Cells(8 + i, 18) = -wdep
   Worksheets("Sheet2").Cells(8 + i, 19) = anchoreoold * (i / 100)
   Worksheets("Sheet2").Cells(8 + i, 20) = anchorbsold * (i / 100)
   Worksheets("Sheet2").Cells(8 + i, 21) = anchorvold * (i / 100)
Next i
For i = 1 To 10
   Worksheets("Sheet2").Cells(108 + i, 2) = -wdep - i
   Worksheets("Sheet2").Cells(108 + i, 9) = fcap(2)
   Worksheets("Sheet2").Cells(108 + i, 10) = fcap(2)
   Worksheets("Sheet2").Cells(108 + i, 14) = fcap(1)
   Worksheets("Sheet2").Cells(108 + i, 15) = fcap(1)
Next i
For i = 1 To 2
  Worksheets("Sheet2").Cells(3, 17 + i) = rsrc(i)
  Worksheets("Sheet2").Cells(4, 17 + i) = rsrt(i)
Next i
For i = 1 To 2
   Worksheets("Sheet2").Cells(8, 25 + i) = beta(i, 0)
   For j = 1 To nbay
      Worksheets("Sheet2").Cells(11 + j, 25 + i) = beta(i, j)
      Worksheets("Sheet2").Cells(11 + j, 25) = j
   Next j
   Worksheets("Sheet2").Cells(11 + nbay + 2, 25 + i) = beta(i, nbay + 1)
   Worksheets("Sheet2").Cells(11 + nbay + 4, 25 + i) = betaacf(i)
   Worksheets("Sheet2").Cells(11 + nbay + 6, 25 + i) = betaatf(i)
Next i
Worksheets("Sheet2").Cells(8, 25) = "deck legs"
Worksheets("Sheet2").Cells(10, 25) = "jacket bay"
Worksheets("Sheet2").Cells(11 + nbay + 2, 25) = "found. lateral"
Worksheets("Sheet2").Cells(11 + nbay + 4, 25) = "found. axial comp."
Worksheets("Sheet2").Cells(11 + nbay + 6, 25) = "found. axial tens."
'
'  Echo of input information, data stored on Sheet3
'
Worksheets("Sheet3").Cells(10, 5) = "ULSLEA"
Worksheets("Sheet3").Cells(11, 5) = "Ultimate Limit State Limit Equilibrium Analysis"
Worksheets("Sheet3").Cells(15, 5) = "Input Parameters"
'
'  Environmental conditions for storm loads
'
Worksheets("Sheet3").Cells(59, 2) = "ENVIRONMENTAL CONDITIONS"
Worksheets("Sheet3").Cells(60, 1) = "water depth"
Worksheets("Sheet3").Cells(60, 2) = "surge"
Worksheets("Sheet3").Cells(60, 3) = "wind vel."
Worksheets("Sheet3").Cells(60, 4) = "wave H"
Worksheets("Sheet3").Cells(60, 5) = "wave T"
Worksheets("Sheet3").Cells(60, 6) = "current vel."
Worksheets("Sheet3").Cells(60, 7) = "current vel."
Worksheets("Sheet3").Cells(61, 1) = "(ft)"
Worksheets("Sheet3").Cells(61, 2) = "(ft)"
Worksheets("Sheet3").Cells(61, 3) = "(mph)"
Worksheets("Sheet3").Cells(61, 4) = "(ft)"
Worksheets("Sheet3").Cells(61, 5) = "(sec)"
Worksheets("Sheet3").Cells(61, 6) = "swl (fps)"
Worksheets("Sheet3").Cells(61, 7) = "mdl (fps)"

Worksheets("Sheet3").Cells(62, 1) = wdep
Worksheets("Sheet3").Cells(62, 2) = sdep
Worksheets("Sheet3").Cells(62, 3) = vrh
Worksheets("Sheet3").Cells(62, 4) = wavh
Worksheets("Sheet3").Cells(62, 5) = wavp
Worksheets("Sheet3").Cells(62, 6) = cswl
Worksheets("Sheet3").Cells(62, 7) = cmdl
'
'  Global parameters of structure
'
Worksheets("Sheet3").Cells(64, 2) = "GLOBAL PARAMETERS"
Worksheets("Sheet3").Cells(65, 1) = "# legs"
```

```
Worksheets("Sheet3").Cells(65, 2) = "# bays"
Worksheets("Sheet3").Cells(65, 3) = "# decks"
Worksheets("Sheet3").Cells(65, 4) = "deck weight"
Worksheets("Sheet3").Cells(65, 5) = "eo base"
Worksheets("Sheet3").Cells(65, 6) = "eo top"
Worksheets("Sheet3").Cells(65, 7) = "bs base"
Worksheets("Sheet3").Cells(65, 8) = "bs top"
Worksheets("Sheet3").Cells(65, 9) = "bs middle"
Worksheets("Sheet3").Cells(66, 4) = "(kips)"
Worksheets("Sheet3").Cells(66, 5) = "width (ft)"
Worksheets("Sheet3").Cells(66, 6) = "width (ft)"
Worksheets("Sheet3").Cells(66, 7) = "width (ft)"
Worksheets("Sheet3").Cells(66, 8) = "width (ft)"
Worksheets("Sheet3").Cells(66, 9) = "width (ft)"

Worksheets("Sheet3").Cells(67, 1) = nleg
Worksheets("Sheet3").Cells(67, 2) = nbay
Worksheets("Sheet3").Cells(67, 3) = ndeck
Worksheets("Sheet3").Cells(67, 4) = qdeck
Worksheets("Sheet3").Cells(67, 5) = bcw(1)
Worksheets("Sheet3").Cells(67, 6) = tcw(1)
Worksheets("Sheet3").Cells(67, 7) = bcw(2)
Worksheets("Sheet3").Cells(67, 8) = tcw(2)
Worksheets("Sheet3").Cells(67, 9) = msw
'
'  Deck areas and other deck information
'
Worksheets("Sheet3").Cells(69, 2) = "Platform Deck Areas"
Worksheets("Sheet3").Cells(70, 1) = "deck #"
Worksheets("Sheet3").Cells(70, 2) = "bottom elev."
Worksheets("Sheet3").Cells(70, 3) = "top elev."
Worksheets("Sheet3").Cells(70, 4) = "eo width"
Worksheets("Sheet3").Cells(70, 5) = "bs width"
Worksheets("Sheet3").Cells(70, 6) = "drag coef."
Worksheets("Sheet3").Cells(70, 7) = "shape coef."
Worksheets("Sheet3").Cells(71, 2) = "(ft)"
Worksheets("Sheet3").Cells(71, 3) = "(ft)"
Worksheets("Sheet3").Cells(71, 4) = "(ft)"
Worksheets("Sheet3").Cells(71, 5) = "(ft)"

For i = 1 To ndeck
   Worksheets("Sheet3").Cells(71 + i, 1) = i
   Worksheets("Sheet3").Cells(71 + i, 2) = uk(i)
   Worksheets("Sheet3").Cells(71 + i, 3) = ok(i)
   Worksheets("Sheet3").Cells(71 + i, 4) = deckw(2, i)
   Worksheets("Sheet3").Cells(71 + i, 5) = deckw(1, i)
   Worksheets("Sheet3").Cells(71 + i, 6) = cdd(i)
   Worksheets("Sheet3").Cells(71 + i, 7) = wsc(i)
Next i
'
'  Heights of bays in the structure
'
Worksheets("Sheet3").Cells(77, 2) = "Jacket Bays"
Worksheets("Sheet3").Cells(78, 1) = "bay #"
Worksheets("Sheet3").Cells(78, 2) = "bay height"
Worksheets("Sheet3").Cells(78, 3) = "eo # of diag."
Worksheets("Sheet3").Cells(78, 4) = "bs # of diag."
Worksheets("Sheet3").Cells(79, 2) = "(ft)"
Worksheets("Sheet3").Cells(79, 3) = "braces"
Worksheets("Sheet3").Cells(79, 4) = "braces"
For i = 1 To 2
   For j = 1 To nbay
      Worksheets("Sheet3").Cells(79 + j, 1) = j
      Worksheets("Sheet3").Cells(79 + j, 2) = bayh(j)
      Worksheets("Sheet3").Cells(79 + j, 2 + i) = ndb(i, j)
   Next j
Next i
'
'  Member parameters and other information
'
```

```
'  Deck legs and main diagonals
'
Worksheets("Sheet3").Cells(1, 13) = "LOCAL PARAMETERS"
Worksheets("Sheet3").Cells(2, 13) = "Deck Legs and End-On Vertical Diagonal Braces"
Worksheets("Sheet3").Cells(1, 26) = "LOCAL PARAMETERS"
Worksheets("Sheet3").Cells(2, 26) = "Broadside Vertical Diagonal Braces"
Worksheets("Sheet3").Cells(3, 10) = "deck legs"
Worksheets("Sheet3").Cells(3, 11) = "d (in)"
Worksheets("Sheet3").Cells(3, 12) = "t (in)"
Worksheets("Sheet3").Cells(4, 11) = dld
Worksheets("Sheet3").Cells(4, 12) = dlt
For i = 1 To 2
   sumdiag = 0
   For j = 1 To nbay
      Worksheets("Sheet3").Cells(5 + sumdiag, 10 + 14 * (i - 1)) = "jacket bay #"
      Worksheets("Sheet3").Cells(5 + sumdiag, 11 + 14 * (i - 1)) = j
      Worksheets("Sheet3").Cells(6 + sumdiag, 10 + 14 * (i - 1)) = "brace #"
      Worksheets("Sheet3").Cells(6 + sumdiag, 11 + 14 * (i - 1)) = "d"
      Worksheets("Sheet3").Cells(6 + sumdiag, 12 + 14 * (i - 1)) = "t"
      Worksheets("Sheet3").Cells(6 + sumdiag, 13 + 14 * (i - 1)) = "pos."
      Worksheets("Sheet3").Cells(6 + sumdiag, 14 + 14 * (i - 1)) = "type"
      Worksheets("Sheet3").Cells(6 + sumdiag, 15 + 14 * (i - 1)) = "conf."
      Worksheets("Sheet3").Cells(6 + sumdiag, 16 + 14 * (i - 1)) = "joint i"
      Worksheets("Sheet3").Cells(6 + sumdiag, 17 + 14 * (i - 1)) = "joint j"
      Worksheets("Sheet3").Cells(6 + sumdiag, 18 + 14 * (i - 1)) = "cond."
      Worksheets("Sheet3").Cells(6 + sumdiag, 19 + 14 * (i - 1)) = "dd"
      Worksheets("Sheet3").Cells(6 + sumdiag, 20 + 14 * (i - 1)) = "oos"
      Worksheets("Sheet3").Cells(6 + sumdiag, 21 + 14 * (i - 1)) = "pu"
      For k = 1 To ndb(i, j)
         Worksheets("Sheet3").Cells(6 + k + sumdiag, 10 + 14 * (i - 1)) = k
         Worksheets("Sheet3").Cells(6 + k + sumdiag, 11 + 14 * (i - 1)) = dbd(i, j, k)
         Worksheets("Sheet3").Cells(6 + k + sumdiag, 12 + 14 * (i - 1)) = dbt(i, j, k)
         If dbpos(i, j, k) = 1 Then
            Worksheets("Sheet3").Cells(6 + k + sumdiag, 13 + 14 * (i - 1)) = "L"
         ElseIf dbpos(i, j, k) = 2 Then
            Worksheets("Sheet3").Cells(6 + k + sumdiag, 13 + 14 * (i - 1)) = "M"
         Else
            Worksheets("Sheet3").Cells(6 + k + sumdiag, 13 + 14 * (i - 1)) = "R"
         End If
         If dbtype(i, j, k) = 1 Then
            Worksheets("Sheet3").Cells(6 + k + sumdiag, 14 + 14 * (i - 1)) = "T"
         Else
            Worksheets("Sheet3").Cells(6 + k + sumdiag, 14 + 14 * (i - 1)) = "C"
         End If
         If dbconf(i, j, k) = 1 Then
            Worksheets("Sheet3").Cells(6 + k + sumdiag, 15 + 14 * (i - 1)) = "S"
         ElseIf dbconf(i, j, k) = 2 Then
            Worksheets("Sheet3").Cells(6 + k + sumdiag, 15 + 14 * (i - 1)) = "K"
         Else
            Worksheets("Sheet3").Cells(6 + k + sumdiag, 15 + 14 * (i - 1)) = "X"
         End If
         Worksheets("Sheet3").Cells(6 + k + sumdiag, 16 + 14 * (i - 1)) = dbjointi(i, j, k)
         Worksheets("Sheet3").Cells(6 + k + sumdiag, 17 + 14 * (i - 1)) = dbjointj(i, j, k)
         If dbcond(i, j, k) = 1 Then
            Worksheets("Sheet3").Cells(6 + k + sumdiag, 18 + 14 * (i - 1)) = "I"
         ElseIf dbcond(i, j, k) = 2 Then
            Worksheets("Sheet3").Cells(6 + k + sumdiag, 18 + 14 * (i - 1)) = "D"
         Else
            Worksheets("Sheet3").Cells(6 + k + sumdiag, 18 + 14 * (i - 1)) = "R"
         End If
         Worksheets("Sheet3").Cells(6 + k + sumdiag, 19 + 14 * (i - 1)) = ddep(i, j, k)
         Worksheets("Sheet3").Cells(6 + k + sumdiag, 20 + 14 * (i - 1)) = oos(i, j, k)
         Worksheets("Sheet3") Cells(6 + k + sumdiag, 21 + 14 * (i - 1)) = Application.Round(dbpu(i, j, k), 0)
      Next k
      sumdiag = sumdiag + 2 + ndb(i, j)
   Next j
Next i
'
'  Horizontal braces
'
```

```
Worksheets("Sheet3").Cells(1, 40) = "LOCAL PARAMETERS"
Worksheets("Sheet3").Cells(2, 40) = "Horizontal Braces"
sumhorz = 0
For i = 1 To nbay + 1
   Worksheets("Sheet3").Cells(5 + sumhorz, 38) = "jacket bay #"
   Worksheets("Sheet3").Cells(5 + sumhorz, 39) = i
   If nhb(i) = 0 Then
      Worksheets("Sheet3").Cells(6 + sumhorz, 38) = "no braces"
   Else
      Worksheets("Sheet3").Cells(6 + sumhorz, 38) = "brace #"
      Worksheets("Sheet3").Cells(6 + sumhorz, 39) = "d (in)"
      Worksheets("Sheet3").Cells(6 + sumhorz, 40) = "t (in)"
      Worksheets("Sheet3").Cells(6 + sumhorz, 41) = "l (ft)"
      Worksheets("Sheet3").Cells(6 + sumhorz, 42) = "angle"
      For j = 1 To nhb(i)
         Worksheets("Sheet3").Cells(6 + j + sumhorz, 38) = j
         Worksheets("Sheet3").Cells(6 + j + sumhorz, 39) = hbd(i, j)
         Worksheets("Sheet3").Cells(6 + j + sumhorz, 40) = hbt(i, j)
         Worksheets("Sheet3").Cells(6 + j + sumhorz, 41) = hbl(i, j)
         Worksheets("Sheet3").Cells(6 + j + sumhorz, 42) = hbang(i, j)
      Next j
   End If
   sumhorz = sumhorz + 2 + nhb(i)
Next i
'
'  Tubular joints
'
Worksheets("Sheet3").Cells(1, 49) = "LOCAL PARAMETERS"
Worksheets("Sheet3").Cells(2, 49) = "Tubular Joints"
Worksheets("Sheet3").Cells(6, 47) = "joint #"
Worksheets("Sheet3").Cells(6, 48) = "type"
Worksheets("Sheet3").Cells(6, 49) = "grout"
Worksheets("Sheet3").Cells(6, 50) = "chord d (in)"
Worksheets("Sheet3").Cells(6, 51) = "chord t (in)"
Worksheets("Sheet3").Cells(6, 52) = "branch d (in)"
Worksheets("Sheet3").Cells(6, 53) = "gap (in)"
Worksheets("Sheet3").Cells(6, 54) = "angle"
Worksheets("Sheet3").Cells(6, 55) = "pu tens."
Worksheets("Sheet3").Cells(6, 56) = "pu comp."
For i = 1 To njoint
   Worksheets("Sheet3").Cells(6 + i, 47) = i
   If jtype(i) = 1 Then
      Worksheets("Sheet3").Cells(6 + i, 48) = "K"
   ElseIf jtype(i) = 2 Then
      Worksheets("Sheet3").Cells(6 + i, 48) = "Y"
   Else
      Worksheets("Sheet3").Cells(6 + i, 48) = "X"
   End If
   Worksheets("Sheet3").Cells(6 + i, 49) = jgrout(i)
   Worksheets("Sheet3").Cells(6 + i, 50) = jchd(i)
   Worksheets("Sheet3").Cells(6 + i, 51) = jcht(i)
   Worksheets("Sheet3").Cells(6 + i, 52) = jbrd(i)
   Worksheets("Sheet3").Cells(6 + i, 53) = jgap(i)
   Worksheets("Sheet3").Cells(6 + i, 54) = jang(i)
   Worksheets("Sheet3").Cells(6 + i, 55) = Application.Round(jput(i), 0)
   Worksheets("Sheet3").Cells(6 + i, 56) = Application.Round(jpuc(i), 0)
Next i
'
'  Foundation piles
'
Worksheets("Sheet3").Cells(118, 2) = "Foundation"
Worksheets("Sheet3").Cells(119, 1) = "main piles"
Worksheets("Sheet3").Cells(119, 2) = "d (in)"
Worksheets("Sheet3").Cells(119, 3) = "t (in)"
Worksheets("Sheet3").Cells(119, 4) = "l (ft)"
Worksheets("Sheet3").Cells(119, 5) = "plug"
Worksheets("Sheet3").Cells(121, 1) = "skirt piles"
Worksheets("Sheet3").Cells(121, 2) = "d (in)"
Worksheets("Sheet3").Cells(121, 3) = "t (in)"
Worksheets("Sheet3").Cells(121, 4) = "l (ft)"
```

```
Worksheets("Sheet3").Cells(121, 5) = "plug"

Worksheets("Sheet3").Cells(120, 2) = piled
Worksheets("Sheet3").Cells(120, 3) = pilet
Worksheets("Sheet3").Cells(120, 4) = pilel
Worksheets("Sheet3").Cells(122, 5) = plug
Worksheets("Sheet3").Cells(122, 2) = spiled
Worksheets("Sheet3").Cells(122, 3) = spilet
Worksheets("Sheet3").Cells(122, 4) = spilel
Worksheets("Sheet3").Cells(122, 5) = splug
'
'  Hydrodynamic force coefficients
'
Worksheets("Sheet3").Cells(124, 2) = "Force Coefficients"
Worksheets("Sheet3").Cells(125, 1) = "cd"
Worksheets("Sheet3").Cells(125, 2) = "cb"
Worksheets("Sheet3").Cells(125, 3) = "ds"
Worksheets("Sheet3").Cells(125, 4) = "lf"

Worksheets("Sheet3").Cells(126, 1) = cdj
Worksheets("Sheet3").Cells(126, 2) = cb
Worksheets("Sheet3").Cells(126, 3) = ds
Worksheets("Sheet3").Cells(126, 4) = lf
'
'  Boatlanding projected areas
'
Worksheets("Sheet3").Cells(128, 2) = "Boatlandings"
Worksheets("Sheet3").Cells(129, 1) = "eo boatl."
Worksheets("Sheet3").Cells(129, 2) = "bs boatl."
Worksheets("Sheet3").Cells(130, 1) = "(sqf)"
Worksheets("Sheet3").Cells(130, 2) = "(sqf)"
Worksheets("Sheet3").Cells(131, 1) = boatl(2)
Worksheets("Sheet3").Cells(131, 2) = boatl(1)
'
'  Member and soil properties
'
Worksheets("Sheet3").Cells(133, 3) = "Member Strength, Material and Soil Properties"
Worksheets("Sheet3").Cells(134, 1) = "fy"
Worksheets("Sheet3").Cells(134, 2) = "e"
Worksheets("Sheet3").Cells(134, 3) = "k"
Worksheets("Sheet3").Cells(134, 4) = "alpha"
Worksheets("Sheet3").Cells(134, 5) = "su1"
Worksheets("Sheet3").Cells(134, 6) = "su2"
Worksheets("Sheet3").Cells(134, 7) = "phi"
Worksheets("Sheet3").Cells(134, 8) = "gamma"
Worksheets("Sheet3").Cells(134, 9) = "scour"

Worksheets("Sheet3").Cells(135, 1) = "(ksi)"
Worksheets("Sheet3").Cells(135, 2) = "(ksi)"
Worksheets("Sheet3").Cells(135, 5) = "(ksf)"
Worksheets("Sheet3").Cells(135, 6) = "(ksf)"
Worksheets("Sheet3").Cells(135, 8) = "(kcf)"
Worksheets("Sheet3").Cells(135, 9) = "(ft)"

Worksheets("Sheet3").Cells(136, 1) = fy
Worksheets("Sheet3").Cells(136, 2) = e
Worksheets("Sheet3").Cells(136, 3) = kbuck
Worksheets("Sheet3").Cells(136, 4) = bres
Worksheets("Sheet3").Cells(136, 5) = su1
Worksheets("Sheet3").Cells(136, 6) = su2
Worksheets("Sheet3").Cells(136, 7) = sphi
Worksheets("Sheet3").Cells(136, 8) = gammas
Worksheets("Sheet3").Cells(136, 9) = scour
'
'  Uncertainties and biases
'
Worksheets("Sheet3").Cells(138, 2) = "Uncertainties and Biases"
Worksheets("Sheet3").Cells(141, 1) = "cov"
Worksheets("Sheet3").Cells(142, 1) = "bias"
```

```
Worksheets("Sheet3").Cells(139, 2) = "wave in"
Worksheets("Sheet3").Cells(139, 3) = "jacket"
Worksheets("Sheet3").Cells(139, 4) = "brace"
Worksheets("Sheet3").Cells(139, 5) = "joint"
Worksheets("Sheet3").Cells(139, 6) = "axial pile"
Worksheets("Sheet3").Cells(139, 7) = "lateral pile"
Worksheets("Sheet3").Cells(139, 8) = "axial pile"
Worksheets("Sheet3").Cells(139, 9) = "lateral pile"
Worksheets("Sheet3").Cells(140, 2) = "deck ldg."
Worksheets("Sheet3").Cells(140, 3) = "ldg."
Worksheets("Sheet3").Cells(140, 4) = "capacity"
Worksheets("Sheet3").Cells(140, 5) = "capacity"
Worksheets("Sheet3").Cells(140, 6) = "cap.(sand)"
Worksheets("Sheet3").Cells(140, 7) = "cap.(sand)"
Worksheets("Sheet3").Cells(140, 8) = "cap.(clay)"
Worksheets("Sheet3").Cells(140, 9) = "cap.(clay)"

Worksheets("Sheet3").Cells(142, 2) = wdbias
Worksheets("Sheet3").Cells(141, 2) = wdcov
Worksheets("Sheet3").Cells(142, 3) = wjbias
Worksheets("Sheet3").Cells(141, 3) = wjcov
Worksheets("Sheet3").Cells(142, 4) = bcbias
Worksheets("Sheet3").Cells(141, 4) = bccov
Worksheets("Sheet3").Cells(142, 5) = jcbias
Worksheets("Sheet3").Cells(141, 5) = jccov
Worksheets("Sheet3").Cells(142, 6) = sabias
Worksheets("Sheet3").Cells(141, 6) = sacov
Worksheets("Sheet3").Cells(142, 7) = slbias
Worksheets("Sheet3").Cells(141, 7) = slcov
Worksheets("Sheet3").Cells(142, 8) = cabias
Worksheets("Sheet3").Cells(141, 8) = cacov
Worksheets("Sheet3").Cells(142, 9) = clbias
Worksheets("Sheet3").Cells(141, 9) = clcov
'
'  Platform tonnage estimate.  This estimate does not include deck equipment or structure.
'
Worksheets("Sheet3").Cells(144, 2) = "Steel Tonnage"
Worksheets("Sheet3").Cells(145, 1) = "deck"
Worksheets("Sheet3").Cells(145, 2) = "jacket"
Worksheets("Sheet3").Cells(145, 3) = "piles"
Worksheets("Sheet3").Cells(145, 4) = "total"
Worksheets("Sheet3").Cells(146, 1) = "(kip)"
Worksheets("Sheet3").Cells(146, 2) = "(kip)"
Worksheets("Sheet3").Cells(146, 3) = "(kip)"
Worksheets("Sheet3").Cells(146, 4) = "(kip)"

Worksheets("Sheet3").Cells(147, 1) = Application.Round(decklegsw, 0)
Worksheets("Sheet3").Cells(147, 2) = Application.Round(jacketw, 0)
Worksheets("Sheet3").Cells(147, 3) = Application.Round(pilew, 0)
Worksheets("Sheet3").Cells(147, 4) = Application.Round(steelw, 0)

End Sub
```

```
'
'          Module 3
'


Dim fdjointi(2, 30, 30), fdjointj(2, 30, 30), SCFaxial(100), SCFbend(100), fatiguelife(2, 30, 30)
Dim connectionarea

Sub fatigue_analysis()
'
'  FATIGUE DAMAGE ALGORITHYM
'  Created by: James D. Stear
'  Created on: April 22, 1997
'
'  Last modified:  June 1, 1997
'
'  Published reference for code:
'
'
'  First, load exposure time, S-N curve parameters, stress response parameters, and
'  average and extreme wave height distributions.
'
'
'
'  Access Macro1 to get structure data.  Access Macro2 to calculate brace parameters
'  and jacket bay stiffnesses.
'
'
'  Now find fatigue damage constant.  Normally, this will be different for each bay,
'  depending on the value of "g" input.  However, the program current only allows a
'  single specification for "g".
'
    Texposure = DialogSheets("Dialog1").EditBoxes("TD").Text
    SNm = DialogSheets("Dialog1").EditBoxes("m").Text
    SNK = DialogSheets("Dialog1").EditBoxes("K").Text
    Stressg = DialogSheets("Dialog1").EditBoxes("g").Text
    StressR = DialogSheets("Dialog1").EditBoxes("R").Text
    WaveH0 = DialogSheets("Dialog1").EditBoxes("H0").Text
    WaveEta0 = DialogSheets("Dialog1").EditBoxes("Eta0").Text
    WaveN0 = DialogSheets("Dialog1").EditBoxes("N0").Text
    WaveH1 = DialogSheets("Dialog1").EditBoxes("H1").Text
    WaveEta1 = DialogSheets("Dialog1").EditBoxes("Eta1").Text
    WaveN1 = DialogSheets("Dialog1").EditBoxes("N1").Text
    Tspec = DialogSheets("Dialog1").EditBoxes("Tspec").Text


    Ynought = (WaveN0 / Tspec) * WaveH0 ^ (SNm * Stressg) * (Log(WaveN0)) ^ (-Stressg * SNm / WaveEta0) *
          Exp(Application.GammaLn(1 + Stressg * SNm / WaveEta0))
    Yone = (WaveN1 / Tspec) * WaveH1 ^ (SNm * Stressg) * (Log(WaveN1)) ^ (-Stressg * SNm / WaveEta1) *
          Exp(Application.GammaLn(1 + Stressg * SNm / WaveEta1))
    fconstant = (Texposure / SNK) * ((1 - StressR) / (wavh ^ Stressg)) ^ SNm * (Ynought + Yone)
'
'
'  Find stress concentration factors for all joint types.  These factors are based on
'  those suggested by API RP 2A LRFD (1993) Commentary Table F-1.1.
'
'
For i = 1 To njoint
    If structuretype = "monopod_braced" Or structuretype = "monopod_guyed" Then
        SCFaxial(i) = 1
        SCFbend(i) = 1
    Else
    If jtype(i) = 1 Then
        alphaSCF = 1
    ElseIf jtype(i) = 2 Then
        alphaSCF = 1.7
    Else
        alphaSCF = 2
    End If
    SCFaxial(i) = alphaSCF * 1.8 * (jchd(i) / 2 / jcht(i)) ^ 0.5 / jcht(i) * Sin(jang(i))
```

```
    SCFbend(i) = 1.8 * (jchd(i) / 2 / jcht(i)) ^ 0.5 / jcht(i) * Sin(jang(i))
    End If
'
'   Note: diagonal brace thickness is omitted here.  It is accounted for in the next loop,
'   when looping is performed over each brace.  Also, brace SCF's are calculated, and the
'   highest SCF (chord or brace) is used.
'
Next i
'
'   With the constant parameters specified, now it is possible to loop through the bays
'   of the structure, calculating the accumulated fatigue damage at each brace end.
'
'   Looping over frames, End-On and Broadside
For i = 1 To 2
'   Looping over bays in jacket
    For j = 1 To nbay

        If j = 1 Then
            baydelta = (shear(i) + meanload(i, j) - 2 * legfh(i, j)) / kbay(i, j)
        Else
            baydelta = (meanload(i, j) - 2 * legfh(i, j)) / kbay(i, j)
        End If
        If structuretype = "tripod" And i = 2 Then
            baydelta = baydelta * Sin(offangle)
        End If
'       Looping over diagonal braces in each bay
        For k = 1 To ndb(i, j)
        If structuretype = "monopod_braced" Or structuretype = "monopod_guyed" Then
            peakstressi = jtbias * (Abs((meanload(i, j) + shear(i)) / connectionarea) + Abs(dbl(i, 1, 1) ^ 2 * dbw(j, k) / 12 / (3 * Pi *
                    jcht(1) * (jchd(1) / 2) ^ 2)))
            peakstressj = peakstressi
        Else
            axialstressi = dbki(i, j, k) * Cos(theta(i, j, k)) * baydelta / dba(i, j, k) * Application.Max(SCFaxial(dbjointi(i, j, k)), 1 + 0.375
                    * (1 + (dbt(i, j, k) / jcht(dbjointi(i, j, k)) / dbd(i, j, k) * jchd(dbjointi(i, j, k)))) ^ 0.5 * SCFaxial(dbjointi(i, j, k))) * dbt(i,
                    j, k)
            axialstressj = dbki(i, j, k) * Cos(theta(i, j, k)) * baydelta / dba(i, j, k) * Application.Max(SCFaxial(dbjointj(i, j, k)), 1 + 0.375
                    * (1 + (dbt(i, j, k) / jcht(dbjointj(i, j, k)) / dbd(i, j, k) * jchd(dbjointj(i, j, k)))) ^ 0.5 * SCFaxial(dbjointj(i, j, k))) * dbt(i,
                    j, k)

            bendstressi = dbki(i, j, k) ^ 2 * dbw(j, k) / 12 * dbd(i, j, k) / 2 / dbi(i, j, k) * Application.Max(SCFbend(dbjointi(i, j, k)), 1 +
                    0.375 * (1 + (dbt(i, j, k) / jcht(dbjointi(i, j, k)) / dbd(i, j, k) * jchd(dbjointi(i, j, k)))) ^ 0.5 * SCFbend(dbjointi(i, j, k)))
                    * dbt(i, j, k)
            bendstressj = dbki(i, j, k) ^ 2 * dbw(j, k) / 12 * dbd(i, j, k) / 2 / dbi(i, j, k) * Application.Max(SCFbend(dbjointj(i, j, k)), 1 +
                    0.375 * (1 + (dbt(i, j, k) / jcht(dbjointj(i, j, k)) / dbd(i, j, k) * jchd(dbjointj(i, j, k)))) ^ 0.5 * SCFbend(dbjointj(i, j, k)))
                    * dbt(i, j, k)

            peakstressi = jtbias * (Abs(axialstressi) + Abs(bendstressi))
            peakstressj = jtbias * (Abs(axialstressj) + Abs(bendstressj))
        End If
            fdjointi(i, j, k) = fconstant * peakstressi ^ SNm
            fdjointj(i, j, k) = fconstant * peakstressj ^ SNm
            fatiguelife(i, j, k) = Application.Min(Texposure / fdjointi(i, j, k), Texposure / fdjointj(i, j, k), Texposure)
        Next k
    Next j
Next i

'
'
'   With the fatigue damage at each joint calculated, these values will now be placed on
'   Sheet5 where they may be viewed as tabular output.
'
'   Fatigue Assessment Parameters
'
Worksheets("Sheet5").Cells(2, 2) = "FATIGUE ASSESSMENT PARAMETERS"
Worksheets("Sheet5").Cells(3, 2) = "Design Wave Height (ft)"
Worksheets("Sheet5").Cells(4, 2) = "S-N K"
Worksheets("Sheet5").Cells(5, 2) = "S-N m"
Worksheets("Sheet5").Cells(6, 2) = "Stress g"
Worksheets("Sheet5").Cells(7, 2) = "Stress R"
Worksheets("Sheet5").Cells(8, 2) = "Exposure Period (years)"
```

```
Worksheets("Sheet5").Cells(3, 3) = wavh
Worksheets("Sheet5").Cells(4, 3) = SNK
Worksheets("Sheet5").Cells(5, 3) = SNm
Worksheets("Sheet5").Cells(6, 3) = Stressg
Worksheets("Sheet5").Cells(7, 3) = StressR
Worksheets("Sheet5").Cells(8, 3) = Texposure
Worksheets("Sheet5").Cells(10, 2) = "H0 (ft)"
Worksheets("Sheet5").Cells(11, 2) = "Eta0"
Worksheets("Sheet5").Cells(12, 2) = "N0 (cycles)"
Worksheets("Sheet5").Cells(14, 2) = "H1 (ft)"
Worksheets("Sheet5").Cells(15, 2) = "Eta1"
Worksheets("Sheet5").Cells(16, 2) = "N1 (cycles)"
Worksheets("Sheet5").Cells(17, 2) = "Spectrum Period (years)"
Worksheets("Sheet5").Cells(10, 3) = WaveH0
Worksheets("Sheet5").Cells(11, 3) = WaveEta0
Worksheets("Sheet5").Cells(12, 3) = WaveN0
Worksheets("Sheet5").Cells(14, 3) = WaveH1
Worksheets("Sheet5").Cells(15, 3) = WaveEta1
Worksheets("Sheet5").Cells(16, 3) = WaveN1
Worksheets("Sheet5").Cells(17, 3) = Tspec
'
'
'
'
Worksheets("Sheet5").Cells(1, 10) = "FATIGUE ASSESSMENT"
Worksheets("Sheet5").Cells(2, 10) = "Fatigue Damage Ratings for End-On Tubular Joints"
Worksheets("Sheet5").Cells(1, 19) = "FATIGUE ASSESSMENT"
Worksheets("Sheet5").Cells(2, 19) = "Fatigue Damage Ratings for Broadside Tubular Joints"
For i = 1 To 2
   sumdiag = 0
   For j = 1 To nbay
      Worksheets("Sheet5").Cells(5 + sumdiag, 10 + 9 * (i - 1)) = "jacket bay #"
      Worksheets("Sheet5").Cells(5 + sumdiag, 11 + 9 * (i - 1)) = j
      Worksheets("Sheet5").Cells(6 + sumdiag, 10 + 9 * (i - 1)) = "brace #"
      Worksheets("Sheet5").Cells(6 + sumdiag, 11 + 9 * (i - 1)) = "joint i"
      Worksheets("Sheet5").Cells(6 + sumdiag, 12 + 9 * (i - 1)) = "joint j"
      Worksheets("Sheet5").Cells(6 + sumdiag, 13 + 9 * (i - 1)) = "Ts (years)"
      For k = 1 To ndb(i, j)
         Worksheets("Sheet5").Cells(6 + k + sumdiag, 10 + 9 * (i - 1)) = k
         Worksheets("Sheet5").Cells(6 + k + sumdiag, 11 + 9 * (i - 1)) = Application.Round(fdjointi(i, j, k), 4)
         Worksheets("Sheet5").Cells(6 + k + sumdiag, 12 + 9 * (i - 1)) = Application.Round(fdjointj(i, j, k), 4)
         Worksheets("Sheet5").Cells(6 + k + sumdiag, 13 + 9 * (i - 1)) = Application.Round(fatiguelife(i, j, k), 0)
      Next k
      sumdiag = sumdiag + 2 + ndb(i, j)
   Next j
Next i

End Sub

'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Sub tripod_jacket_capacity() '  Start TRIPOD JACKET CAPACITY
For i = 1 To 2
   For j = 1 To nbay
      For k = 1 To ndb(i, j)
         If i = 1 Then
            kbay(i, j) = kbay(i, j) + dbki(i, j, k)
            sumksq(i, j) = sumksq(i, j) + dbki(i, j, k) ^ 2
         Else
            kbay(1, j) = kbay(1, j) + dbki(i, j, k) * (Cos(offangle)) ^ 2
            kbay(i, j) = kbay(i, j) + dbki(i, j, k) * (Cos(Pi / 2 - offangle)) ^ 2
            sumksq(1, j) = sumksq(1, j) + (dbki(i, j, k) * (Cos(offangle)) ^ 2) ^ 2
            sumksq(i, j) = sumksq(i, j) + (dbki(i, j, k) * (Cos(Pi / 2 - offange)) ^ 2) ^ 2
         End If
      Next k
   Next j
Next i

For i = 1 To 2
```

```
   For j = 1 To nbay
      mltfbrace = 1
      mltfjoint = 1
'
'  Find most-likely-to-fail member in bay.  For the case of broadside loading, it is
'  assumed that the MLTF member will be in the plane being loaded.
'
      For k = 1 To ndb(i, j)
         If jpu(i, j, k) * Cos(theta(i, j, k)) / dbki(i, j, k) <= jpu(i, j, mltfjoint) * Cos(theta(i, j, mltfjoint)) / dbki(i, j, mltfjoint) Then
               mltfjoint = k
         If dbpu(i, j, k) * Cos(theta(i, j, k)) / dbki(i, j, k) <= dbpu(i, j, mltfbrace) * Cos(theta(i, j, mltfbrace)) / dbki(i, j, mltfbrace)
               Then mltfbrace = k
      Next k
'
'  Now jacket bay shear capacity based on:
'     load in bay when first diagonal member fails (lower-bound bay capacity)
'     load in bay when all diagonal members are at their ultimate tensile or post-
'        buckling strengths (upper-bound bay capacity)
'     load in bay when first tubular joint fails (joint-based bay capacity)
'
'  Joint-based bay capacity:
'
         jcap(i, j) = jpu(i, j, mltfjoint) / (dbki(i, j, mltfjoint) * Sin(offangle) ^ (i - 1)) * kbay(i, j) * Cos(theta(i, j, mltfjoint))
'
'  Lowe-bound bay capacity:
'
         cap(i, j) = dbpu(i, j, mltfbrace) / (dbki(i, j, mltfbrace) * Sin(offangle) ^ (i - 1)) * kbay(i, j) * Cos(theta(i, j, mltfbrace))
'
'  Note: Upper-bound bay capacity is not calculated for tripods; it is set equal to lower-
'  bound capacity.
'
         capu(i, j) = cap(i, j)
'
'  On the next line, following the "/8" should be "/dbdelta(i,j,l)".  This has been disabled
'  due to the severe impact of dbdelta on the reliability.  This is being checked for
'  realism.
'
         sigmapu(i, j) = ((bccov * dbpu(i, j, mltfbrace)) ^ 2 + (dblx(i, j, mltfbrace) ^ 2 / 8) ^ 2 * (wjcov * dbw(j, mltfbrace)) ^ 2) ^
0.5
         sigmaalpha(i, j) = sigmapu(i, j) / (dbki(i, j, mltfbrace) * Sin(offangle) ^ (i - 1)) * Cos(theta(i, j, mltfbrace))

   Next j
Next i

End Sub '   End TRIPOD JACKET CAPACITY

'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Sub monopod_capacity() '   Start MONOPOD CAPACITY
'
'  Find vertical stiffness
'
pileheadkz = axialbias * e * pilea / pilel
spileheadkx = 12 * 18.2 * Gsoil * spiled / 2 * (1 - nu ^ 2) / (2 - nu) ^ 2
spileheadkz = axialbias * e * spilea / spilel

kvertical = nleg * legeffectivea * e / elevation(0)
kvertical = kvertical * pileheadkz / (kvertical + pileheadkz)

'
'  GUYED MONOPOD
'
If structuretype = "monopod_guyed" Then
'
'  Find capacities of attachment point, wire, and anchor
'
connectionarea = jchd(1)
connectioncaptension = fy * connectionarea
connectioncapshear = fy * connectionarea * Cos(theta(1, 1, 1)) / 2 / Sin(theta(1, 1, 1))
wirecap = fy * (dbd(1, 1, 1) / 2) ^ 2 * Pi
```

```
If stype = 1 Then
   anchoraxialcap = sqts * Cos(theta(1, 1, 1)) / Sin(theta(1, 1, 1))
   anchorhorizontalcap = pusbar
Else
   anchoraxialcap = sqtc * Cos(theta(1, 1, 1)) / Sin(theta(1, 1, 1))
   anchorhorizontalcap = pucbar
End If

kwire = e * (dbd(1, 1, 1) / 2) ^ 2 * Pi / bcw(1) * Cos(theta(1, 1, 1))
anchorkx = spileheadkx
anchorkz = spileheadkz / Sin(theta(1, 1, 1)) ^ 2 / Cos(theta(1, 1, 1))

If pretension < 3 Then
   kbay(1, 1) = kwire * anchorkz * anchorkx / (kwire * anchorkx + anchorkz * anchorkx + anchorkz * kwire)
Else
   kbay(1, 1) = 2 * kwire * anchorkz * anchorkx / (kwire * anchorkx + anchorkz * anchorkx + anchorkz * kwire)
End If

kbay(2, 1) = kbay(1, 1)

cap(1, 1) = Application.Min(connectioncaptension, connectioncapshear, wirecap, anchoraxialcap, anchorhorizontalcap) -
        pretension * Cos(theta(1, 1, 1))
cap(1, 1) = cap(1, 1) + (cap(1, 1) / kbay(1, 1) * 3 * e * piler ^ 2 * pilea / 144 / (bayh(1) + 10 * piled / 12) ^ 3)

cap(2, 1) = cap(1, 1)
capu(1, 1) = cap(1, 1)
capu(2, 1) = cap(2, 1)

jcap(1, 1) = 0
jcap(2, 1) = 0

kvertical = kvertical + 3 * kbay(1, 1) * (Sin(theta(1, 1, 1)) ^ 2) / (Cos(theta(1, 1, 1)) ^ 2)


Else
'
'  BRACED MONOPOD
'
'
'  Find capacities of attachment point and imbedded portion of brace
'
connectionarea = jcht(1) * (jchd(1) - jcht(1)) * Pi / 2
connectioncap = fy * connectionarea
For i = 1 To 2
   kbay(i, 1) = ((e * dba(i, 1, 1) * axialbias / dbl(i, 1, 1) / spilel) / (1 / dbl(i, 1, 1) + axialbias / spilel)) * (Cos(theta(i, 1, 1))) ^ 2
   If dbtype(i, 1, 1) = 1 Then
      If stype = 1 Then
         imbedcap = sqts
      Else
         imbedcap = sqtc
      End If
   Else
      If stype = 1 Then
         imbedcap = sqcs
      Else
         imbedcap = sqcc
      End If
   End If
   cap(i, 1) = Application.Min(imbedcap * Cos(theta(i, 1, 1)), dbpu(i, 1, 1) * Cos(theta(i, 1, 1)), connectioncap)
   cap(i, 1) = cap(i, 1) + (cap(i, 1) / kbay(i, 1) * 3 * e * piler ^ 2 * pilea / 144 / (bayh(1) + 10 * piled / 12) ^ 3)
   capu(i, 1) = cap(i, 1)
   jcap(i, 1) = 0
Next i
'
'  Vertical stiffness of braced monopod (for earthquake analysis)
'

kvertical = kvertical + 2 * kbay(1, 1) * (Sin(theta(1, 1, 1)) ^ 2) / (Cos(theta(1, 1, 1)) ^ 2)
```

```
End If
End Sub '  End MONOPOD CAPACITY
'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
'
'          Module 4
'
'  This module contains various subroutines which control OK, CANCEL, the pop-up title
'  screen, the assigning of custom menus, and the execution calls for the storm, fatigue
'  and earthquake analysis routines.
'
'
'  Calculation menu item STORM
'
Sub storm()
   Module4.structure_class
   Application.Sheets("Sheet1").Activate
   Module1.get_data
   Module5.geometry
   Module5.weight
   Module7.kinematics
   Module7.deck_forces
   If structuretype = "monopod_braced" Or structuretype = "monopod_guyed" Then
      If structuretype = "monopod_braced" Then
         Module5.main_diagonals
      End If
      Module5.foundation_capacity
      Module3.monopod_capacity
      Module5.deck_bay
   Else
      Module5.joint_capacity
      Module5.main_diagonals
      If structuretype = "tripod" Then
         Module3.tripod_jacket_capacity
      Else
         Module5.jacket_capacity
      End If
      Module5.deck_bay
      Module5.foundation_capacity
   End If
   Module7.projected_areas
   Module7.hydro_profile
   Module5.capacity_profile
   Module2.reliability
   Module2.graphing
   Application.Sheets("Sheet2").Calculate
   Beep
   End
End Sub
'
'  Calculation menu item EARTHQUAKE
'
Sub earthquake()

DialogSheets("Dialog4").Show

DialogSheets("Dialog5").Show

numconductors = DialogSheets("Dialog5").EditBoxes("numconductors").Text
diaconductors = DialogSheets("Dialog5").EditBoxes("diaconductors").Text
tconductors = DialogSheets("Dialog5").EditBoxes("tconductors").Text
equipmentperiod = DialogSheets("Dialog5").EditBoxes("equipmentperiod").Text

Application.Sheets("Sheet1").Activate

pga = DialogSheets("Dialog4").EditBoxes("PGA").Text
Gsoil = DialogSheets("Dialog4").EditBoxes("Gsoil").Text
nu = DialogSheets("Dialog4").EditBoxes("nu").Text
cm = DialogSheets("Dialog4").EditBoxes("cm").Text
axialkbias = DialogSheets("Dialog4").EditBoxes("axialbias").Text
horizkbias = DialogSheets("Dialog4").EditBoxes("horizontalbias").Text
eqcov = DialogSheets("Dialog4").EditBoxes("eqcov").Text

If DialogSheets("Dialog4").OptionButtons("SRSS").Value = xlOn Then
```

```
    modecom = 1
Else
    modecom = 2
End If

If DialogSheets("Dialog4").OptionButtons("A").Value = xlOn Then
    APIsoil = 1
ElseIf DialogSheets("Dialog4").OptionButtons("B").Value = xlOn Then
    APIsoil = 2
Else
    APIsoil = 3
End If

    Module4.structure_class
    Module1.get_data
    Module5.geometry
    Module5.weight
    If structuretype = "monopod_braced" Or structuretype = "monopod_guyed" Then
        If structuretype = "monopod_braced" Then
            Module5.main_diagonals
        End If
        Module5.foundation_capacity
        Module3.monopod_capacity
        Module5.deck_bay
    Else
        Module5.joint_capacity
        Module5.main_diagonals
        If structuretype = "tripod" Then
            Module3.tripod_jacket_capacity
        Else
            Module5.jacket_capacity
        End If
        Module5.deck_bay
        Module5.foundation_capacity
        Module5.foundation_stiffness
    End If
    Module6.masses
    Module6.eigen
    Module6.RSA
    Module5.capacity_profile
    Module2.reliability
    Module2.graphing
    Application.Sheets("Sheet2").Calculate
    Beep
    End
End Sub
'
'
'  Calculation menu item FATIGUE
'
Sub fatigue()

    DialogSheets("Dialog1").Show

    Application.Sheets("Sheet1").Activate

    Module4.structure_class
    Module1.get_data
    Module5.geometry
    Module7.kinematics
    Module7.deck_forces
    If structuretype = "monopod_braced" Or structuretype = "monopod_guyed" Then
        If structuretype = "monopod_braced" Then
            Module5.main_diagonals
        End If
        Module5.foundation_capacity
        Module3.monopod_capacity
        Module5.deck_bay
    Else
        Module5.joint_capacity
```

```
      Module5.main_diagonals
      If structuretype = "tripod" Then
         Module3.tripod_jacket_capacity
      Else
         Module5.jacket_capacity
      End If
      Module5.deck_bay
      Module5.foundation_capacity
   End If
   Module7.projected_areas
   Module7.hydro_profile
   Module3.fatigue_analysis

   Beep
   End

End Sub
'
' Cancel switches; these allow for clean termination of input and analysis options
'
Sub Dialog4_cancel1_Click()
   End
End Sub
'
Sub cancel1_Click()
   End
End Sub
'
Sub cancel_Click()
   End
End Sub
'
'  KILL macro.  This clears all public variables.  It should be used following an
'  interrupted execution.
'
Sub kill()
End
End Sub
'
'  Title screen, show at startup
'
Sub title()
   DialogSheets("Dialog2").Show
End Sub
'
'  Design options menu
'
Sub designops()
   DialogSheets("Dialog3").Show
End Sub
'
'  Bring up structure class menu, and assign class type
'
Sub structure_class()

If DialogSheets("Dialog6").OptionButtons("jacket").Value = xlOn Then
   structuretype = "jacket"
ElseIf DialogSheets("Dialog6").OptionButtons("multijacket").Value = xlOn Then
   structuretype = "multijacket"
ElseIf DialogSheets("Dialog6").OptionButtons("tripod").Value = xlOn Then
   structuretype = "tripod"
ElseIf DialogSheets("Dialog6").OptionButtons("braced_monopod").Value = xlOn Then
   structuretype = "monopod_braced"
Else
   structuretype = "monopod_guyed"
End If

End Sub
'
'  Bring up ULSLEA menu bar
```

```
'
Sub ULSLEA_Menu()
    MenuBars("ULSLEAMenu").Activate
End Sub
'
'   Restore Excel menu bar
'
Sub XLS_Menu()
    MenuBars(xlWorksheet).Activate
End Sub
'
'   Auto-open at startup
'
Sub Auto_Open()
    Application.ScreenUpdating = False
    Workbooks.Open "INP.XLS"
    Workbooks("ULSLEA.XLS").Activate
    Sheets("Sheet1").Select
    With ActiveWindow
        .DisplayFormulas = False
        .DisplayGridlines = False
        .DisplayHeadings = False
        .DisplayZeros = False
        .GridlineColorIndex = 0
        .DisplayOutline = False
    End With
    ActiveWorkbook.DisplayDrawingObjects = 1
    Sheets("Sheet2").Select
    With ActiveWindow
        .DisplayFormulas = False
        .DisplayGridlines = False
        .DisplayHeadings = False
        .DisplayZeros = False
        .GridlineColorIndex = 0
        .DisplayOutline = False
    End With
    ActiveWorkbook.DisplayDrawingObjects = 1
    Sheets("Sheet4").Select
    With ActiveWindow
        .DisplayFormulas = False
        .DisplayGridlines = False
        .DisplayHeadings = False
        .DisplayZeros = False
        .GridlineColorIndex = 0
        .DisplayOutline = False
    End With
    ActiveWorkbook.DisplayDrawingObjects = 1
    MenuBars("ULSLEAMenu").Activate
    Sheets("Sheet1").Select
    With Application
        .FixedDecimal = False
        .DisplayStatusBar = False
        .DisplayFormulaBar = False
        .TransitionMenuKey = "/"
        .IgnoreRemoteRequests = False
        .MoveAfterReturn = False
        .CommandUnderlines = 1
        .DisplayNoteIndicator = False
        .CellDragAndDrop = False
        .DisplayInfoWindow = False
        .EditDirectlyInCell = False
    End With
    With ActiveWindow
        .DisplayHorizontalScrollBar = True
        .DisplayVerticalScrollBar = True
    End With
    Module4.title
End Sub
'
'   The following are menu commands which call the macros contained within INP.XLS
```

```
'
'
'  File Menu
'
Sub open_input()
Application.Run ("INP.XLS!OPEN")
End Sub
'
Sub save_input()
Application.Run ("INP.XLS!SAVE")
End Sub
'
Sub print_input()
Application.Run ("INP.XLS!PRINT")
End Sub
'
Sub exit_ulslea()
Application.Run ("INP.XLS!EXIT")
End Sub
'
'  Input Menu
'
Sub get_structure_class()
DialogSheets("Dialog6").Show
End Sub

Sub get_environment()
Application.Run ("INP.XLS!Record1")
End Sub
'
Sub get_global()
Application.Run ("INP.XLS!Record2")
End Sub
'
Sub get_decklegs()
Application.Run ("INP.XLS!DECKLEGS")
End Sub
'
Sub get_horizontals()
Application.Run ("INP.XLS!RECORD9")
End Sub
'
Sub get_joints()
Application.Run ("INP.XLS!RECORD5")
End Sub
'
Sub get_foundation()
Application.Run ("INP.XLS!Record10")
End Sub
'
Sub get_force_coefs()
Application.Run ("INP.XLS!Record7")
End Sub
'
Sub get_boat()
Application.Run ("INP.XLS!Record8")
End Sub
'
Sub get_materials()
Application.Run ("INP.XLS!Record6")
End Sub
'
Sub get_biases()
Application.Run ("INP.XLS!REL")
End Sub
'
'  Output Menu
'
Sub show_endon()
Application.Run ("INP.XLS!ENDON")
```

```
End Sub
'
Sub show_broadside()
Application.Run ("INP.XLS!BROADSIDE")
End Sub
'
Sub show_foundation()
Application.Run ("INP.XLS!FOUNDATION")
End Sub
'
Sub show_kinematics()
Application.Run ("INP.XLS!KINEMATICS")
End Sub
'
Sub show_EO_reliability()
Application.Run ("INP.XLS!RELEO")
End Sub
'
Sub show_BS_reliability()
Application.Run ("INP.XLS!RELBS")
End Sub
'
Sub show_inputs()
Application.Sheets("Sheet3").Activate
Cells(1, 1).Show
End Sub
'
Sub show_fatigue()
Application.Sheets("Sheet5").Activate
Cells(1, 1).Show
End Sub
'
Sub show_modes()
Application.Sheets("Sheet5").Activate
Cells(115, 15).Show
End Sub
'
Sub show_deck_accel()
Application.Sheets("Sheet5").Activate
Cells(115, 1).Show
End Sub

****
```

```
'            Module 5
'
'  Created by: Merhdad Mortazavi
'  Created on: 1/22/96
'
'  Last Modified by:  James Stear
'  Last Modified on:  6/1/97
'
'
'
Public foundationkx, foundationktheta(2), kvertical, kthetabend(2), pileloadcomp(2)
Public pileloadtens(2), sandpilecaptens(2), sandpilecapcomp(2), claypilecaptens(2)
Public claypilecapcomp(2), Gsoil, nu, axialkbias, horizkbias, cornerskirts, designcd
Public legeffectivea, axialbias, pretension
'
'
'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'
Sub geometry() '  Begin GEOMETRY
'
'  Check to see if design options are used
'
If DialogSheets("Dialog3").CheckBoxes("prelim").Value = xlOn Then
   prelim = True
Else
   prelim = False
End If
'
'
'
If DialogSheets("Dialog3").CheckBoxes("designCD").Value = xlOn Then
   designcd = True
Else
   designcd = False
End If
'
'
'
If DialogSheets("Dialog3").CheckBoxes("pgrout").Value = xlOn Then
   pgrout = True
Else
   pgrout = False
End If
'
'
'
If DialogSheets("Dialog3").CheckBoxes("cornerskirts").Value = xlOn Then
   cornerskirts = True
Else
   cornerskirts = False
End If
'
'  Assign densities, and initialize structure weights
'
steelg = 0.42   '  steel density, in kips/cu ft
rho = 0.064     '  water density, in kips/cu ft
groutg = 0.13   '  grout density, in kips/cu ft
decklegsw = 0
jacketw = 0
pilew = 0
'
'  Define gravity constant g (ft/sec^2) and Pi
'
g = 32.174
Pi = 3.14159
'
'  Assign parameters for special configurations
'
If structuretype = "tripod" Then
   nleg = 3
```

```
    offangle = Application.Acos(bcw(1) / bcw(2) / 2)
Else
    offangle = Pi / 2
End If

If structuretype = "monopod_braced" Or structuretype = "monopod_guyed" Then
    nleg = 1
    nsk = 1
    axialbias = nskirt(2)
    nskirt(1) = 0
    nskirt(2) = 0
    dld = piled
    dlt = pilet
    Gsoil = jcht(1)
    nu = jbrd(1)
    pretension = jgap(1)
    For i = 1 To 2
        jld(i, 1) = piled
        jlt(i) = pilet
    Next i
End If
'
'   Now begin definition of structure geometry and member information
'
'
'   Find total structure height, and elevation increments for plotting
'
htotal = 0
For i = 1 To nbay
    htotal = bayh(i) + htotal
Next i

i = 0
elevation(0) = htotal + bayh(0)
Do
    elevation(i + 1) = elevation(i) - bayh(i)
    i = i + 1
Loop While i <= nbay

elevbar = elevation(0)
interval = elevation(0) / 100

For j = 1 To 100
    elev(j) = elevbar
    elevbar = elevbar - interval
Next j
'
'
'   Horizontal bay dimensions
'
For i = 1 To 2
    alpha(i) = Atn((bcw(i) - tcw(i)) / 2 / htotal)
Next i
'
'   End-on elevation, subject to broadside loads
'
i = 1
For j = 1 To nbay
    If nleg < 12 Then
        lh(i, 1) = tcw(i)
        lh(i, j + 1) = lh(i, j) + 2 * (bayh(j) * Tan(alpha(i)))
        lht(i, j) = lh(i, j)
    Else
        lh(i, 1) = tcw(i) / 2
        lh(i, j + 1) = lh(i, j) + (bayh(j) * Tan(alpha(i)))
        lht(i, j) = 2 * lh(i, j)
    End If
Next j
'
'   Broadside elevation, subject to end-on loads
```

```
'
i = 2
For j = 1 To nbay
   If nleg <= 4 Then
      lh(i, 1) = tcw(i)
      lh(i, j + 1) = lh(i, j) + 2 * (bayh(j) * Tan(alpha(i)))
      lht(i, j) = lh(i, j)
   ElseIf nleg = 6 Then
      lh(i, 1) = tcw(i) / 2
      lh(i, j + 1) = lh(i, j) + (bayh(j) * Tan(alpha(i)))
      lht(i, j) = 2 * lh(i, j)
   Else
      lh(i, 1) = (tcw(i) - msw) / 2
      lh(i, j + 1) = lh(i, j) + (bayh(j) * Tan(alpha(i)))
      lht(i, j) = 2 * lh(i, j) + msw
   End If
Next j
'
'  Now begin definition of main diagonals
'
'  Frames seen from end-on elevation, which resist broadside loads
'
i = 1
If nleg < 12 Then
   For j = 1 To nbay
      For k = 1 To ndb(i, j)
         If dbconf(i, j, k) = 1 Then
            dbl(i, j, k) = Sqr(((lh(i, j + 1) + lh(i, j)) / 2) ^ 2 + bayh(j) ^ 2)
         ElseIf dbconf(i, j, k) = 2 Then
            If dbtype(i, j, k) = 1 Then
               If dbpos(i, j, k) = 1 Then
                  dbl(i, j, k) = Sqr(((lh(i, j + 1)) / 2) ^ 2 + bayh(j) ^ 2)
               Else
                  dbl(i, j, k) = Sqr(((lh(i, j)) / 2) ^ 2 + bayh(j) ^ 2)
               End If
            Else
               If dbpos(i, j, k) = 1 Then
                  dbl(i, j, k) = Sqr(((lh(i, j)) / 2) ^ 2 + bayh(j) ^ 2)
               Else
                  dbl(i, j, k) = Sqr(((lh(i, j + 1)) / 2) ^ 2 + bayh(j) ^ 2)
               End If
            End If
         Else
            dbl(i, j, k) = Sqr(((lh(i, j + 1) + lh(i, j)) / 2) ^ 2 + bayh(j) ^ 2)
         End If
      Next k
   Next j
Else
   For j = 1 To nbay
      For k = 1 To ndb(i, j)
         If dbconf(i, j, k) = 1 Then
            If dbtype(i, j, k) = 1 Then
               If dbpos(i, j, k) = 1 Then
                  dbl(i, j, k) = Sqr(lh(i, j + 1) ^ 2 + bayh(j) ^ 2)
               Else
                  dbl(i, j, k) = Sqr(lh(i, j) ^ 2 + bayh(j) ^ 2)
               End If
            Else
               If dbpos(i, j, k) = 1 Then
                  dbl(i, j, k) = Sqr(lh(i, j) ^ 2 + bayh(j) ^ 2)
               Else
                  dbl(i, j, k) = Sqr(lh(i, j + 1) ^ 2 + bayh(j) ^ 2)
               End If
            End If
         ElseIf dbconf(i, j, k) = 2 Then
            If dbtype(i, j, k) = 1 Then
               If dbpos(i, j, k) = 1 Then
                  dbl(i, j, k) = Sqr((lh(i, j + 1) - lh(i, j) / 2) ^ 2 + bayh(j) ^ 2)
               Else
                  dbl(i, j, k) = Sqr(((lh(i, j)) / 2) ^ 2 + bayh(j) ^ 2)
```

```
            End If
          Else
            If dbpos(i, j, k) = 1 Then
              dbl(i, j, k) = Sqr(((lh(i, j)) / 2) ^ 2 + bayh(j) ^ 2)
            Else
              dbl(i, j, k) = Sqr((lh(i, j + 1) - lh(i, j) / 2) ^ 2 + bayh(j) ^ 2)
            End If
          End If
        Else
          If dbtype(i, j, k) = 1 Then
            If dbpos(i, j, k) = 1 Then
              dbl(i, j, k) = Sqr(lh(i, j + 1) ^ 2 + bayh(j) ^ 2)
            Else
              dbl(i, j, k) = Sqr(lh(i, j) ^ 2 + bayh(j) ^ 2)
            End If
          Else
            If dbpos(i, j, k) = 1 Then
              dbl(i, j, k) = Sqr(lh(i, j) ^ 2 + bayh(j) ^ 2)
            Else
              dbl(i, j, k) = Sqr(lh(i, j + 1) ^ 2 + bayh(j) ^ 2)
            End If
          End If
        End If
      Next k
    Next j
End If
'
'  Frames seen from broadside elevation, which resist end-on loads
'
i = 2
If nleg <= 4 Then
    For j = 1 To nbay
      For k = 1 To ndb(i, j)
        If dbconf(i, j, k) = 1 Then
          dbl(i, j, k) = Sqr(((lh(i, j + 1) + lh(i, j)) / 2) ^ 2 + bayh(j) ^ 2)
        ElseIf dbconf(i, j, k) = 2 Then
          If dbtype(i, j, k) = 1 Then
            If dbpos(i, j, k) = 1 Then
              dbl(i, j, k) = Sqr(((lh(i, j + 1)) / 2) ^ 2 + bayh(j) ^ 2)
            Else
              dbl(i, j, k) = Sqr(((lh(i, j)) / 2) ^ 2 + bayh(j) ^ 2)
            End If
          Else
            If dbpos(i, j, k) = 1 Then
              dbl(i, j, k) = Sqr(((lh(i, j)) / 2) ^ 2 + bayh(j) ^ 2)
            Else
              dbl(i, j, k) = Sqr(((lh(i, j + 1)) / 2) ^ 2 + bayh(j) ^ 2)
            End If
          End If
        Else
          dbl(i, j, k) = Sqr(((lh(i, j + 1) + lh(i, j)) / 2) ^ 2 + bayh(j) ^ 2)
        End If
      Next k
    Next j
ElseIf nleg = 6 Then
  For j = 1 To nbay
      For k = 1 To ndb(i, j)
        If dbconf(i, j, k) = 1 Then
          If dbtype(i, j, k) = 1 Then
            If dbpos(i, j, k) = 1 Then
              dbl(i, j, k) = Sqr(lh(i, j + 1) ^ 2 + bayh(j) ^ 2)
            Else
              dbl(i, j, k) = Sqr(lh(i, j) ^ 2 + bayh(j) ^ 2)
            End If
          Else
            If dbpos(i, j, k) = 1 Then
              dbl(i, j, k) = Sqr(lh(i, j) ^ 2 + bayh(j) ^ 2)
            Else
              dbl(i, j, k) = Sqr(lh(i, j + 1) ^ 2 + bayh(j) ^ 2)
            End If
```

```
            End If
        ElseIf dbconf(i, j, k) = 2 Then
            If dbtype(i, j, k) = 1 Then
                If dbpos(i, j, k) = 1 Then
                    dbl(i, j, k) = Sqr((lh(i, j + 1) - lh(i, j) / 2) ^ 2 + bayh(j) ^ 2)
                Else
                    dbl(i, j, k) = Sqr(((lh(i, j)) / 2) ^ 2 + bayh(j) ^ 2)
                End If
            Else
                If dbpos(i, j, k) = 1 Then
                    dbl(i, j, k) = Sqr(((lh(i, j)) / 2) ^ 2 + bayh(j) ^ 2)
                Else
                    dbl(i, j, k) = Sqr((lh(i, j + 1) - lh(i, j) / 2) ^ 2 + bayh(j) ^ 2)
                End If
            End If
        Else
            If dbtype(i, j, k) = 1 Then
                If dbpos(i, j, k) = 1 Then
                    dbl(i, j, k) = Sqr(lh(i, j + 1) ^ 2 + bayh(j) ^ 2)
                Else
                    dbl(i, j, k) = Sqr(lh(i, j) ^ 2 + bayh(j) ^ 2)
                End If
            Else
                If dbpos(i, j, k) = 1 Then
                    dbl(i, j, k) = Sqr(lh(i, j) ^ 2 + bayh(j) ^ 2)
                Else
                    dbl(i, j, k) = Sqr(lh(i, j + 1) ^ 2 + bayh(j) ^ 2)
                End If
            End If
        End If
    Next k
  Next j
Else
  For j = 1 To nbay
    For k = 1 To ndb(i, j)
      If dbconf(i, j, k) = 1 Then
        If dbtype(i, j, k) = 1 Then
          If dbpos(i, j, k) = 1 Then
            dbl(i, j, k) = Sqr(lh(i, j + 1) ^ 2 + bayh(j) ^ 2)
          ElseIf dbpos(i, j, k) = 2 Then
            dbl(i, j, k) = Sqr(msw ^ 2 + bayh(j) ^ 2)
          Else
            dbl(i, j, k) = Sqr(lh(i, j) ^ 2 + bayh(j) ^ 2)
          End If
        Else
          If dbpos(i, j, k) = 1 Then
            dbl(i, j, k) = Sqr(lh(i, j) ^ 2 + bayh(j) ^ 2)
          ElseIf dbpos(i, j, k) = 2 Then
            dbl(i, j, k) = Sqr(msw ^ 2 + bayh(j) ^ 2)
          Else
            dbl(i, j, k) = Sqr(lh(i, j + 1) ^ 2 + bayh(j) ^ 2)
          End If
        End If
      ElseIf dbconf(i, j, k) = 2 Then
        If dbtype(i, j, k) = 1 Then
          If dbpos(i, j, k) = 1 Then
            dbl(i, j, k) = Sqr((lh(i, j + 1) - lh(i, j) / 2) ^ 2 + bayh(j) ^ 2)
          ElseIf dbpos(i, j, k) = 2 Then
            dbl(i, j, k) = Sqr((msw / 2) ^ 2 + bayh(j) ^ 2)
          Else
            dbl(i, j, k) = Sqr(((lh(i, j)) / 2) ^ 2 + bayh(j) ^ 2)
          End If
        Else
          If dbpos(i, j, k) = 1 Then
            dbl(i, j, k) = Sqr(((lh(i, j)) / 2) ^ 2 + bayh(j) ^ 2)
          ElseIf dbpos(i, j, k) = 2 Then
            dbl(i, j, k) = Sqr((msw / 2) ^ 2 + bayh(j) ^ 2)
          Else
            dbl(i, j, k) = Sqr((lh(i, j + 1) - lh(i, j) / 2) ^ 2 + bayh(j) ^ 2)
          End If
```

```
            End If
        Else
            If dbtype(i, j, k) = 1 Then
                If dbpos(i, j, k) = 1 Then
                    dbl(i, j, k) = Sqr(lh(i, j + 1) ^ 2 + bayh(j) ^ 2)
                ElseIf dbpos(i, j, k) = 2 Then
                    dbl(i, j, k) = Sqr(msw ^ 2 + bayh(j) ^ 2)
                Else
                    dbl(i, j, k) = Sqr(lh(i, j) ^ 2 + bayh(j) ^ 2)
                End If
            Else
                If dbpos(i, j, k) = 1 Then
                    dbl(i, j, k) = Sqr(lh(i, j) ^ 2 + bayh(j) ^ 2)
                ElseIf dbpos(i, j, k) = 2 Then
                    dbl(i, j, k) = Sqr(msw ^ 2 + bayh(j) ^ 2)
                Else
                    dbl(i, j, k) = Sqr(lh(i, j + 1) ^ 2 + bayh(j) ^ 2)
                End If
            End If
        End If
    Next k
  Next j
End If
'
' Find angle each brace makes with the horizontal, as well as the length subject to
' buckling (dblx).
'
For i = 1 To 2
  For j = 1 To nbay
    For k = 1 To ndb(i, j)
      theta(i, j, k) = Application.Asin(bayh(j) / dbl(i, j, k))
      If dbconf(i, j, k) = 1 Or dbconf(i, j, k) = 2 Then
        dblx(i, j, k) = dbl(i, j, k)
      ElseIf i = 1 And nleg < 12 Then                    '...X-Braced Elements...
        dblx(i, j, k) = lh(i, j + 1) / 2 / Cos(theta(i, j, k))
      ElseIf i = 1 And nleg = 12 Then
        If dbpos(i, j, k) = 1 Then
          thetax(i, j, k) = Atn((bayh(j) / lh(i, j + 1)))
          dblx(i, j, k) = Sin(thetax(i, j, k)) * lh(i, j + 1) / Sin(3.14 - theta(i, j, k) - thetax(i, j, k))
        Else
          thetax(i, j, k) = Atn((bayh(j) / lh(i, j)))
          dblx(i, j, k) = Sin(thetax(i, j, k)) * lh(i, j + 1) / Sin(3.14 - theta(i, j, k) - thetax(i, j, k))
        End If
      ElseIf i = 2 And nleg <= 4 Then
        dblx(i, j, k) = lh(i, j + 1) / 2 / Cos(theta(i, j, k))
      ElseIf i = 2 And nleg = 6 Then
        If dbpos(i, j, k) = 1 Then
          thetax(i, j, k) = Atn((bayh(j) / lh(i, j + 1)))
          dblx(i, j, k) = Sin(thetax(i, j, k)) * lh(i, j + 1) / Sin(3.14 - theta(i, j, k) - thetax(i, j, k))
        Else
          thetax(i, j, k) = Atn((bayh(j) / lh(i, j)))
          dblx(i, j, k) = Sin(thetax(i, j, k)) * lh(i, j + 1) / Sin(3.14 - theta(i, j, k) - thetax(i, j, k))
        End If
      Else                                               '...i=2 and nleg >=8...
        If dbpos(i, j, k) = 1 Then
          thetax(i, j, k) = Atn((bayh(j) / lh(i, j + 1)))
          dblx(i, j, k) = Sin(thetax(i, j, k)) * lh(i, j + 1) / Sin(3.14 - theta(i, j, k) - thetax(i, j, k))
        ElseIf dbpos(i, j, k) = 2 Then
          dblx(i, j, k) = dbl(i, j, k) / 2
        Else
          thetax(i, j, k) = Atn((bayh(j) / lh(i, j)))
          dblx(i, j, k) = Sin(thetax(i, j, k)) * lh(i, j + 1) / Sin(3.14 - theta(i, j, k) - thetax(i, j, k))
        End If
      End If
    Next k
  Next j
Next i

End Sub ' End GEOMETRY
```

```
'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Sub brace_design()  ' Begin BRACE DESIGN
'
'  Preliminary design.  If this option is checked, braces are sized according to
'  kL/r ratios of 50 and D/t ratios of 40.
'
   For i = 1 To 2
      For j = 1 To nbay
         For k = 1 To ndb(i, j)
            dummy = kbuck * dbx(i, j, k) * 12 / 50 / 0.35
            dbd(i, j, k) = Application.Round(dummy, 0)
            dummy = dbd(i, j, k) / 40
            dbt(i, j, k) = Application.Round(dummy, 1)
         Next k
      Next j
   Next i

End Sub '  End BRACE DESIGN

'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Sub weight() ' Start WEIGHT
'
'  Total weight of horizontal braces
'
For j = 1 To nbay + 1
   For k = 1 To nhb(j)
      jacketw = jacketw + steelg * (Pi * (hbd(j, k) - hbt(j, k)) * hbt(j, k) * hbl(j, k)) / 144
   Next k
Next j
'
'  Add weight of brace to total jacket weight
'
For i = 1 To 2
   For j = 1 To nbay
      For k = 1 To ndb(i, j)
         jacketw = jacketw + steelg * (Pi * dbd(i, j, k) / 12 * dbt(i, j, k) / 12 * dbl(i, j, k))
      Next k
   Next j
Next i
'
'  Calculate jacket leg weight and add to total jacket weight
'
jacketw = jacketw + steelg * nleg * Pi * jld(1, 1) / 12 * jlt(1) / 12 * htotal
'
'  Find deck leg weight
'
decklegsw = steelg * dla / 144 * bayh(0) * nleg
'
'  Find pile weight and total steel weight
'
pilew = steelg * nleg * pilea / 144 * (pilel + htotal)
pilew = pilew + steelg * nsk * spilea / 144 * spilel
steelw = decklegsw + jacketw + pilew

End Sub ' End WEIGHT

'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Sub joint_capacity()   ' Start JOINT CAPACITY
'
'  Tubular joint capacities.  Tubular joint capacities are estimated from API (1993)
'  RP 2A Section E formulas.  Only axial load is considered.
'
For i = 1 To njoint
   If jgrout(i) = True Then
      jcht(i) = jcht(i) + pilet
   End If
   jbeta(i) = jbrd(i) / jchd(i)
```

```
    jgamma(i) = jchd(i) / (2 * jcht(i))
    If jbeta(i) <= 0.6 Then
       jqbeta(i) = 1
    Else
       jqbeta(i) = 0.3 / (jbeta(i) * (1 - 0.8333 * jbeta(i)))
    End If
    If jgamma(i) <= 20 Then
       jqg(i) = 1.8 - 0.1 * (jgap(i) / jcht(i))
    Else
       jqg(i) = 1.8 - 4 * (jgap(i) / jchd(i))
    End If
    If jqg(i) < 1 Then jqg(i) = 1
    jdummy(i) = fy * jcht(i) ^ 2 / Sin(jang(i) * 2 * 3.14 / 360)
    If jtype(i) = 1 Then
       jput(i) = jcbias * jdummy(i) * (3.4 + 19 * jbeta(i)) * jqg(i)
       jpuc(i) = jcbias * jdummy(i) * (3.4 + 19 * jbeta(i)) * jqg(i)
    Else
       jput(i) = jcbias * jdummy(i) * (3.4 + 19 * jbeta(i))
       If jtype(i) = 3 Then
          jpuc(i) = jcbias * jdummy(i) * (3.4 + 13 * jbeta(i)) * jqbeta(i)
       Else
          jpuc(i) = jcbias * jdummy(i) * (3.4 + 19 * jbeta(i))
       End If
    End If
Next i

End Sub ' End JOINT CAPACITY

'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Sub main_diagonals() ' Start MAIN DIAGONALS
'
'   Main diagonal cross-section properties and capacities
'
If prelim = True Then
   Module5.brace_design
End If
'
'
'
For i = 1 To 2
   For j = 1 To nbay
'
'   Cd scaling for local forces on braces
'
If (crest - (elevation(j) - bayh(j) / 2) < vcrest ^ 2 / g) And (designcd = False) And (crest - (elevation(j) - bayh(j) / 2) > 0) Then
       cdmem = cdj * (crest - (elevation(j) - bayh(j) / 2)) / (vcrest ^ 2 / g)
    Else
       cdmem = cdj
    End If
    l = 1
    l1 = 1
    cap(i, j) = 0
    capu(i, j) = 0
    jcap(i, j) = 0
    sumksq(i, j) = 0
       For k = 1 To ndb(i, j)
       dba(i, j, k) = (dbd(i, j, k) - dbt(i, j, k)) * dbt(i, j, k) * 3.14
       dbr(i, j, k) = 1 / 4 * (dbd(i, j, k) ^ 2 + (dbd(i, j, k) - 2 * dbt(i, j, k)) ^ 2) ^ 0.5
       dbi(i, j, k) = dbr(i, j, k) ^ 2 * dba(i, j, k)
       dbzp(i, j, k) = 1.3 * 3.14 / 32 * (dbd(i, j, k) ^ 4 - (dbd(i, j, k) - 2 * dbt(i, j, k)) ^ 4) / dbd(i, j, k)
       dblam(i, j, k) = (1 / 3.14) * (fy / e) ^ 0.5 * kbuck * (dbbx(i, j, k) * 12) / dbr(i, j, k)

       dbpe(i, j, k) = 3.14 ^ 2 * e * dba(i, j, k) / (kbuck * dbbx(i, j, k) * 12 / dbr(i, j, k)) ^ 2
       dbpy(i, j, k) = fy * dba(i, j, k)
       If dblam(i, j, k) < 2 ^ 0.5 Then
          dbpcr(i, j, k) = (1 - 0.25 * dblam(i, j, k) ^ 2)
       Else
          dbpcr(i, j, k) = 1 / (dblam(i, j, k) ^ 2)
       End If
```

```
        dbpcr(i, j, k) = dbpcr(i, j, k) * fy * dba(i, j, k)
        If dbd(i, j, k) / dbt(i, j, k) <= 60 Then
          dbpcrl(i, j, k) = dbpy(i, j, k)
        Else
          dbpcrl(i, j, k) = dbpy(i, j, k) * (1.64 - 0.23 * (dbd(i, j, k) / dbt(i, j, k)) ^ 0.25)
        End If
        dbmp(i, j, k) = dbzp(i, j, k) * fy / 12
        If fy * dbd(i, j, k) / dbt(i, j, k) < 1500 Then
          dbmcr(i, j, k) = 1
        ElseIf fy * dbd(i, j, k) / dbt(i, j, k) < 3000 Then
          dbmcr(i, j, k) = (1.13 - 2.58 * fy * dbd(i, j, k) / dbt(i, j, k) / e)
        Else
          dbmcr(i, j, k) = (0.94 - 0.76 * fy * dbd(i, j, k) / dbt(i, j, k) / e)
        End If
        dbmcr(i, j, k) = dbmcr(i, j, k) * dbmp(i, j, k)
'
' Main diagonal brace stiffness and capacities
'
        dbw(j, k) = wjbias * ((velocity(j) * Sin(theta(i, j, k))) ^ 2 * cdmem * (dbd(i, j, k) + 2 * mg(j)) / 12 / 1000) * lf
        dbeps0(i, j, k) = dblx(i, j, k) * 12 * (dbpcr(i, j, k) / e / dbi(i, j, k)) ^ 0.5
        dbdelta(i, j, k) = Abs(Cos(3.1416 / 2 * dbpcr(i, j, k) / dbpy(i, j, k)) * dbmcr(i, j, k) / ((1 / (1 + 2 * Sin(0.5 * dbeps0(i, j, k)) /
            Sin(dbeps0(i, j, k)))) * 1 / (dbeps0(i, j, k) ^ 2) * (1 / Cos(dbeps0(i, j, k) / 2) - 1) * 8 * dbpcr(i, j, k)))
        dbki(i, j, k) = e * dba(i, j, k) * Cos(theta(i, j, k)) ^ 2 / dbl(i, j, k)
        If dbtype(i, j, k) = 1 Then
'
' Main diagonals in tension, regardless of damage condition
'
          jpu(i, j, k) = Application.Min(jput(dbjointi(i, j, k)), jput(dbjointj(i, j, k)))
          dbpu(i, j, k) = dbpy(i, j, k)
          dbpuhu(i, j, k) = dbpy(i, j, k) * Cos(theta(i, j, k))
        ElseIf dbcond(i, j, k) = 1 Then
'
' Main diagonals, undamaged, in compression
'
          dbpu(i, j, k) = dbpcr(i, j, k)
'
' Newton-Raphson iteration to find dbpu for undamaged main diagonals
'
          Do
            Formula = ((1 / (1 + 2 * Sin(0.5 * dbeps0(i, j, k)) / Sin(dbeps0(i, j, k))) * 1 / dbeps0(i, j, k) ^ 2 * (1 / Cos((dbeps0(i, j,
                k)) / 2) - 1) * (dbw(j, k) * dblx(i, j, k) ^ 2 + 8 * dbpu(i, j, k) * dbdelta(i, j, k))) / dbmcr(i, j, k)) - Cos(3.1416 / 2 *
                dbpu(i, j, k) / dbpy(i, j, k))
            Formula1 = ((1 / (1 + 2 * Sin(0.5 * dbeps0(i, j, k)) / Sin(dbeps0(i, j, k))) * 1 / dbeps0(i, j, k) ^ 2 * (1 / Cos((dbeps0(i,
                 j, k)) / 2) - 1) * (8 * dbdelta(i, j, k))) / dbmcr(i, j, k)) + 3.1416 / 2 / dbpy(i, j, k) * Sin(3.1416 / 2 * dbpu(i, j, k) /
                 dbpy(i, j, k))
            dbputest = dbpu(i, j, k)
            dbpu(i, j, k) = dbpu(i, j, k) - Formula / Formula1
          Loop While ((Abs(dbputest - dbpu(i, j, k)) > 1) And (Abs(Formula) > 0.0001))
          dbpu(i, j, k) = bcbias * dbpu(i, j, k)
        ElseIf dbcond(i, j, k) = 2 Then
'
' Buckling strengths of damaged members using Loh's equations
'
          dbpcrld(i, j, k) = dbpcrl(i, j, k) * Application.Max(0.45, Exp(-0.08 * ddep(i, j, k) / dbt(i, j, k)))
          dbmcrd(i, j, k) = dbmcr(i, j, k) * Application.Max(0.55, Exp(-0.06 * ddep(i, j, k) / dbt(i, j, k)))
          dbpe(i, j, k) = dbpe(i, j, k) * Application.Max(0.55, Exp(-0.06 * ddep(i, j, k) / dbt(i, j, k)))
          dblamd(i, j, k) = (dbpcrld(i, j, k) / dbpe(i, j, k)) ^ 0.5
          If dblamd(i, j, k) < 2 ^ 0.5 Then
            dbpcrd0(i, j, k) = (1 - 0.25 * dblamd(i, j, k) ^ 2) * dbpcrld(i, j, k)
          Else
            dbpcrd0(i, j, k) = (1 / dblamd(i, j, k) ^ 2) * dbpcrld(i, j, k)
          End If
          dbpcrd(i, j, k) = dbpcrd0(i, j, k)
'
' Incremental solution of dbpcrd for damaged main diagonals
'
          Do
            Formula = 1 - dbpcrd(i, j, k) / dbpcrd0(i, j, k) - dbpcrd(i, j, k) * oos(i, j, k) / 12 / (1 - dbpcrd(i, j, k) / dbpe(i, j, k)) /
                dbmcrd(i, j, k)
            dbpcrd(i, j, k) = dbpcrd(i, j, k) - 1
```

```
        Loop While Formula <= 0
        dbpu(i, j, k) = dbpcrd(i, j, k)
'
' Incremental solution of dbpu for damaged main diagonals
'
        Do
          Formula = 1 - dbpu(i, j, k) / dbpcrd(i, j, k) - ((dbw(j, k) * dblx(i, j, k) ^ 2 / 24 / (1 - dbpu(i, j, k) / dbpe(i, j, k)) /
                dbmcrd(i, j, k)) ^ (2 - 3 * ddep(i, j, k) / dbd(i, j, k))) ^ 0.5
          dbpu(i, j, k) = dbpu(i, j, k) - 1
        Loop While Formula <= 0
        dbpu(i, j, k) = bcbias * dbpu(i, j, k)
      Else
'
' Buckling strengths of members using Parsanejad's equations
'
        dbalhpa(i, j, k) = Application.Acos(1 - 2 * ddep(i, j, k) / dbd(i, j, k))
        dbag(i, j, k) = dbd(i, j, k) ^ 2 / 4 * (3.14 - dbalpha(i, j, k) + 0.5 * Sin(2 * dbalpha(i, j, k)))
        dbatr(i, j, k) = dba(i, j, k) + dbag(i, j, k) / 7
        dbes(i, j, k) = dbd(i, j, k) / (2 * 3.14) * (Sin(dbalpha(i, j, k)) - dbalpha(i, j, k) * Cos(dbalpha(i, j, k)))
        dbeg(i, j, k) = (dbd(i, j, k) * Sin(dbalpha(i, j, k))) ^ 3 / 12 / dbag(i, j, k)
        dbetr(i, j, k) = (dba(i, j, k) * dbes(i, j, k) + dbag(i, j, k) / 7 * dbeg(i, j, k)) / dbatr(i, j, k)
        dbis(i, j, k) = dbd(i, j, k) ^ 3 * dbt(i, j, k) / 4 * ((3.14 - dbalpha(i, j, k)) / 2 - Sin(2 * dbalpha(i, j, k)) / 4 + dbalpha(i, j, k) *
              (Cos(dbalpha(i, j, k))) ^ 2 - (Sin(dbalpha(i, j, k)) - dbalpha(i, j, k) * Cos(dbalpha(i, j, k))) ^ 2 / 3.14)
        dbig(i, j, k) = dbd(i, j, k) ^ 4 / 64 * (3.14 - dbalpha(i, j, k) + Sin(4 * dbalpha(i, j, k)) / 4) - dbd(i, j, k) ^ 4 * (Sin(dbalpha(i,
              j, k))) ^ 6 / 144 / dbag(i, j, k)
        dbitr(i, j, k) = dbis(i, j, k) + dbig(i, j, k) / 7 + dba(i, j, k) * (dbetr(i, j, k) - dbes(i, j, k)) ^ 2 + dbag(i, j, k) / 7 * (dbeg(i, j, k)
              - dbetr(i, j, k)) ^ 2
        dbrtr(i, j, k) = (dbitr(i, j, k) / dbatr(i, j, k)) ^ 0.5
        dbztr(i, j, k) = dbitr(i, j, k) / (dbd(i, j, k) / 2 * Cos(dbalpha(i, j, k)) + dbes(i, j, k)) 'check
        dbastr(i, j, k) = dba(i, j, k) + 3.14 * dbd(i, j, k) ^ 2 / 4 / 7
        dblamp(i, j, k) = 1 / 3.14 * kbuck * dblx(i, j, k) * 12 / dbrtr(i, j, k) * (fy / e) ^ 0.5
        dbm(i, j, k) = dbatr(i, j, k) / dbastr(i, j, k)
        dbpu(i, j, k) = 1
'
' Newton-Raphson iteration to find buckling strength of grout-repaired member
'
        Do
          dbk(i, j, k) = dbatr(i, j, k) * (dbetr(i, j, k) + oos(i, j, k) + dbw(j, k) * 12 * dblx(i, j, k) ^ 2 / 24 / dbpu(i, j, k)) / dbztr(i, j,
                k)
          Formula = ((dbpu(i, j, k) / dbastr(i, j, k) / fy) ^ 2 - ((1 + dbk(i, j, k)) / (dblamp(i, j, k) ^ 2) + dbm(i, j, k)) * (dbpu(i, j, k)
                / dbastr(i, j, k) / fy) + dbm(i, j, k) / (dblamp(i, j, k) ^ 2))
          Formula1 = ((2 * dbpu(i, j, k) / (dbastr(i, j, k) * fy) ^ 2) - ((1 + dbk(i, j, k)) / (dblamp(i, j, k) ^ 2) + dbm(i, j, k)) * (1 /
                dbastr(i, j, k) / fy))
          dbputest = dbpu(i, j, k)
          dbpu(i, j, k) = dbpu(i, j, k) - Formula / Formula1
        Loop While ((Abs(dbputest - dbpu(i, j, k)) > 1) And (Abs(Formula) > 0.0001))
        dbpu(i, j, k) = bcbias * dbpu(i, j, k)
      End If
      If dbtype(i, j, k) = 2 Then jpu(i, j, k) = Application.Min(jpuc(dbjointi(i, j, k)), jpuc(dbjointj(i, j, k)))
    Next k
  Next j
Next i

End Sub '  End MAIN DIAGONALS

'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Sub jacket_capacity() '  Start JACKET CAPACITY
For i = 1 To 2
  For j = 1 To nbay
    mltfbrace = 1
    mltfjoint = 1
    For k = 1 To ndb(i, j)
      kbay(i, j) = kbay(i, j) + dbki(i, j, k)
      If jpu(i, j, k) * Cos(theta(i, j, k)) / dbki(i, j, k) <= jpu(i, j, mltfjoint) * Cos(theta(i, j, mltfjoint)) / dbki(i, j, mltfjoint) Then
            mltfjoint = k
      If dbpu(i, j, k) * Cos(theta(i, j, k)) / dbki(i, j, k) <= dbpu(i, j, mltfbrace) * Cos(theta(i, j, mltfbrace)) / dbki(i, j, mltfbrace)
            Then mltfbrace = k
    Next k
'
```

```
'  Now jacket bay shear capacity based on:
'     load in bay when first diagonal member fails (lower-bound bay capacity)
'     load in bay when all diagonal members are at their ultimate tensile or post-
'        buckling strengths (upper-bound bay capacity)
'     load in bay when first tubular joint fails (joint-based bay capacity)
'
      For k = 1 To ndb(i, j)
         jpuh(i, j, k) = jpu(i, j, mltfjoint) / dbki(i, j, mltfjoint) * dbki(i, j, k) * Cos(theta(i, j, mltfjoint))
         dbpuh(i, j, k) = dbpu(i, j, mltfbrace) / dbki(i, j, mltfbrace) * dbki(i, j, k) * Cos(theta(i, j, mltfbrace))
'  Joint-based bay capacity:
         jcap(i, j) = jcap(i, j) + jpuh(i, j, k)
'  Lowe-bound bay capacity:
         cap(i, j) = cap(i, j) + dbpuh(i, j, k)
         If dbtype(i, j, k) = 2 Then dbpuhu(i, j, k) = bres * dbpu(i, j, k) * Cos(theta(i, j, k))
'  Upper-bound bay capacity:
         capu(i, j) = capu(i, j) + dbpuhu(i, j, k)

'  Sum-of-squares for reliability calculation:
         sumksq(i, j) = sumksq(i, j) + dbki(i, j, k) ^ 2

      Next k
'
'  On the next line, following the "/8" should be "/dbdelta(i,j,l)". This has been disabled
'  due to the severe impact of dbdelta on the reliability. This is being checked for
'  realism.
'
         sigmapu(i, j) = ((bccov * dbpu(i, j, mltfbrace)) ^ 2 + (dblx(i, j, mltfbrace) ^ 2 / 8) ^ 2 * (wjcov * dbw(j, mltfbrace)) ^ 2) ^
               0.5
         sigmaalpha(i, j) = sigmapu(i, j) / dbki(i, j, mltfbrace) * Cos(theta(i, j, mltfbrace))
   Next j
Next i

End Sub '  End JACKET CAPACITY

'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Sub deck_bay()  ' Start DECK BAY
'
'  Capacity of deck bay.  Capacity is formulated based on simultaneous hinging of
'  the tops and bottoms of all deck legs.  This formulation assumes the bay is unbraced.
'
For i = 1 To 2
   dla = (dld - dlt) * dlt * 3.14
   dli = 3.14 / 64 * (dld ^ 4 - (dld - 2 * dlt) ^ 4)
   If structuretype = "monopod_braced" Or structuretype = "monopod_guyed" Then
      ibay1(i) = dli
   Else
      If pgrout = True Then
         ibay1(i) = 3.14 / 64 * (jld(i, 1) ^ 4 - (piled - 2 * pilet) ^ 4)
      Else
         ibay1(i) = 3.14 / 64 * (jld(i, 1) ^ 4 - (jld(i, 1) - 2 * jlt(i)) ^ 4)
      End If
   End If
   dlzp = 1.3 * 3.14 / 32 * (dld ^ 4 - (dld - 2 * dlt) ^ 4) / dld
   dlr = 1 / 4 * (dld ^ 2 + (dld - 2 * dlt) ^ 2) ^ 0.5
   dlmp = dlzp * fy / 12
   If fy * dld / dlt < 1500 Then
      dlmcr = 1
   ElseIf fy * dld / dlt < 3000 Then
      dlmcr = (1.13 - 2.58 * fy * dld / dlt / e)
   Else
      dlmcr = (0.94 - 0.76 * fy * dld / dlt / e)
   End If
   dlmcr = dlmcr * dlmp
   dlm = dlmcr * Cos(3.14 / 2 * qdeck / nleg / fy / dla)
   dlcs(i) = kbay(i, 1) / nleg
   If structuretype = "monopod_braced" Or structuretype = "monopod_guyed" Then
      dlcm(i) = ((bayh(1) + 10 * piled / 12) / (e * ibay1(i) / 144) - 3 * dlcs(i) * (bayh(1)) ^ 4 / (4 * dlcs(i) * (bayh(1)) ^ 3 * e * ibay1(i)
            / 144 + 12 * (e * ibay1(i) / 144) ^ 2)) ^ -1
      deckk(i) = nleg * ((bayh(0) / (3 * e * dli / 144) + 1 / dlcm(i)) ^ (-1)) / (bayh(0) ^ 2)
```

```
      dldelta(i) = dlm * bayh(0) * (bayh(0) / (3 * e * dli / 144) + 1 / dlcm(i))
      dlcap(i) = (nleg * dlm - qdeck * dldelta(i)) / bayh(0)
   Else
      dlcm(i) = (bayh(1) / (e * ibay1(i) / 144) - 3 * dlcs(i) * bayh(1) ^ 4 / (4 * dlcs(i) * bayh(1) ^ 3 * e * ibay1(i) / 144 + 12 * (e *
            ibay1(i) / 144) ^ 2)) ^ -1
      deckk(i) = nleg * ((bayh(0) / (6 * e * dli / 144) + 1 / dlcm(i)) ^ (-1)) / (bayh(0) ^ 2)
      dldelta(i) = dlm * bayh(0) * (bayh(0) / (6 * e * dli / 144) + 1 / dlcm(i))
      dlcap(i) = (2 * nleg * dlm - qdeck * dldelta(i)) / bayh(0)
   End If
Next i

End Sub '  End DECK BAY

'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Sub foundation_capacity()  '  Start FOUNDATION CAPACITY
'
'  Axial and lateral capacities of foundation piles
'
'  Cross-section properties of main piles
'
pilea = (piled - pilet) * pilet * Pi
pilepy = pilea * fy
pilezp = 1.3 * Pi / 32 * (piled ^ 4 - (piled - 2 * pilet) ^ 4) / piled
piler = 1 / 4 * (piled ^ 2 + (piled - 2 * pilet) ^ 2) ^ 0.5
pilemp = pilezp * fy / 12
If fy * piled / pilet < 1500 Then
   pilemcr = 1
ElseIf fy * piled / pilet < 3000 Then
   pilemcr = (1.13 - 2.58 * fy * piled / pilet / e)
Else
   pilemcr = (0.94 - 0.76 * fy * piled / pilet / e)
End If
pilemcr = pilemcr * pilemp
pilem = pilemcr * Cos(Pi / 2 * qdeck / nleg / fy / pilea)
If pgrout = True Then
   legeffectivea = pilea + (jld(1, 1) - jlt(1)) * jlt(1) * Pi
Else
   legeffectivea = pilea
End If
'
'  Cross-section properties of skirt piles
'
If skirt = True Then
   If structuretype = "tripod" Then
      nsk = nleg * nskirt(1)
      nskirt(2) = nskirt(1)
   ElseIf structuretype <> "monopod_braced" And structuretype <> "monopod_guyed" Then
      nsk = nskirt(1) + nskirt(2)
      If cornerskirts = True Then
         nskirt(1) = nsk
         nskirt(2) = nsk
      End If
   End If
   spilea = (spiled - spilet) * spilet * Pi
   spilepy = spilea * fy
   spilezp = 1.3 * Pi / 32 * (spiled ^ 4 - (spiled - 2 * spilet) ^ 4) / spiled
   spiler = 1 / 4 * (spiled ^ 2 + (spiled - 2 * spilet) ^ 2) ^ 0.5
   spilemp = spilezp * fy / 12
   If fy * spiled / spilet < 1500 Then
      spilemcr = 1
   ElseIf fy * spiled / spilet < 3000 Then
      spilemcr = (1.13 - 2.58 * fy * spiled / spilet / e)
   Else
      spilemcr = (0.94 - 0.76 * fy * spiled / spilet / e)
   End If
   spilemcr = spilemcr * spilemp
   spilem = spilemcr
End If
'
```

```
'  Find number of piles on each face
'
If nleg <= 3 Then
   n(1) = 1
   n(2) = 1
ElseIf nleg = 4 Then
   n(1) = 2
   n(2) = 2
ElseIf nleg = 6 Then
   n(1) = 3
   n(2) = 2
ElseIf nleg = 8 Then
   n(1) = 4
   n(2) = 2
Else
   n(1) = 4
   n(2) = 3
End If

'
If stype = 2 Then
'
'  Now determine axial capacity of main piles if founded in clay
'
   su = (su1 + su2) / 2
   If su < 0.5 Then
      ff = su
   ElseIf su > 1.5 Then
      ff = 0.5 * su
   Else
      ff = (1 - (su - 0.5) / 2) * su
   End If
   wp = 0
   If plug = True Then wp = gammas * Pi * ((piled - 2 * pilet) / 12) ^ 2 / 4
   wp = wp + 0.42 * Pi * ((piled / 12) ^ 2 - ((piled - 2 * pilet) / 12) ^ 2) / 4
   If plug = True Then
      dummy = Application.Min(9 * su2 * 3.14 * (piled / 12) ^ 2 / 4, ff * Pi * (piled - 2 * pilet) / 12 * pilel)
   Else
      dummy = 9 * su2 * Pi * (piled / 12) * (pilet / 12)
   End If
   qcc = cabias * (Application.Min(pilepy, dummy + (ff * Pi * piled / 12 - wp) * pilel))
   qtc = cabias * (Application.Min(pilepy, (ff * Pi * piled / 12 + wp) * pilel))
'
'  Axial capacity of skirt piles if founded in clay
'
   If skirt = True Then
      swp = 0
      If splug = True Then swp = gammas * Pi * ((spiled - 2 * spilet) / 12) ^ 2 / 4
      swp = swp + 0.42 * Pi * ((spiled / 12) ^ 2 - ((spiled - 2 * spilet) / 12) ^ 2) / 4
      If splug = True Then
         dummy = Application.Min(9 * su2 * Pi * (spiled / 12) ^ 2 / 4, ff * Pi * (spiled - 2 * spilet) / 12 * spilel)
      Else
         dummy = 9 * su2 * Pi * (spiled / 12) * (spilet / 12)
      End If
      sqcc = cabias * (Application.Min(spilepy, dummy + (ff * Pi * spiled / 12 - swp) * spilel))
      sqtc = cabias * (Application.Min(spilepy, (ff * Pi * spiled / 12 + swp) * spilel))
   Else
      sqcc = 0
      sqtc = 0
   End If
'
'  Lateral capacity of main piles if founded in clay
'
   a = 9 * su1 * piled / 12
   b = 9 * su2 * piled / 12
   seta = (b - a) / pilel
   psi = 1.5 * piled / 12 + scour
   pucbar = 0.5 * (-(27 * (piled / 12) ^ 2 * s u1) + ((27 * (piled / 12) ^ 2 * su1) ^ 2 + 144 * su1 * (piled / 12) * pilem) ^ 0.5)
'
'  Newton-Raphson iteration to find lateral capacity in clay
```

```
'
    Do
      If su1 = su2 Then
        c = pucbar / a
      Else
        c = 1 / seta * (-(a + seta * psi) + ((a + seta * psi) ^ 2 + 2 * seta * pucbar) ^ 0.5)
      End If
      cbar = 1 / ((a + seta * psi) ^ 2 + 2 * seta * pucbar) ^ 0.5
      Formula = pucbar * (c + psi) - 2 * pilem - (a + seta * psi) * c ^ 2 / 2 - seta / 2 * c ^ 3 / 3
      Formula1 = cbar * (pucbar - (a + seta * psi) * c - seta / 2 * c ^ 2) + c + seta
      pucbartest = pucbar
      pucbar = pucbar - Formula / Formula1
    Loop While ((Abs(pucbar - pucbartest) > 1) And (Abs(Formula) > 0.0001))
    For i = 1 To 2
      puc(i) = clbias * pucbar * nleg
    Next i
'
'   Lateral capacity of skirt piles if founded in clay
'
    If skirt = True Then
      a = 9 * su1 * spiled / 12
      b = 9 * su2 * spiled / 12
      seta = (b - a) / spilel
      psi = 1.5 * spiled / 12 + scour
      pucbar = 0.5 * (-(27 * (spiled / 12) ^ 2 * su1) + ((27 * (spiled / 12) ^ 2 * su1) ^ 2 + 144 * su1 * (spiled / 12) * spilem) ^ 0.5)
'
'     Newton-Raphson iteration to find lateral capacity in clay
'
      Do
        If su1 = su2 Then
          c = pucbar / a
        Else
          c = 1 / seta * (-(a + seta * psi) + ((a + seta * psi) ^ 2 + 2 * seta * pucbar) ^ 0.5)
        End If
        cbar = 1 / ((a + seta * psi) ^ 2 + 2 * seta * pucbar) ^ 0.5
        Formula = pucbar * (c + psi) - 2 * spilem - (a + seta * psi) * c ^ 2 / 2 - seta / 2 * c ^ 3 / 3
        Formula1 = cbar * (pucbar - (a + seta * psi) * c - seta / 2 * c ^ 2) + c + seta
        pucbartest = pucbar
        pucbar = pucbar - Formula / Formula1
      Loop While ((Abs(pucbar - pucbartest) > 1) And (Abs(Formula) > 0.0001))
      For i = 1 To 2
        spuc(i) = clbias * pucbar * nsk
      Next i
    Else
      For i = 1 To 2
        spuc(i) = 0
      Next i
    End If
  Else
'
'   Axial capacity of main piles if founded in sand
'
    If sphi < 20 Then
      nq = 8
    ElseIf sphi > 35 Then
      nq = 40
    ElseIf sphi > 30 Then
      nq = 40 - (35 - sphi) * 4
    ElseIf sphi > 25 Then
      nq = 20 - (30 - sphi) * 1.6
    Else
      nq = 12 - (25 - sphi) * 0.8
    End If
    qmax = 5 * nq
    If sphi < 20 Then
      ffmax = 1
    ElseIf sphi > 35 Then
      ffmax = 2
    ElseIf sphi > 25 Then
      ffmax = 2 - (35 - sphi) * 0.06
```

```
    Else
       ffmax = 1.4 - (25 - sphi) * 0.08
    End If
    plc = ffmax / (gammas * (Tan((sphi - 5) * Pi / 180)))
    plt = ffmax / (0.7 * gammas * Tan((sphi - 5) * Pi / 180))
    If pilel < plc Then
       fas = 0.5 * pilel ^ 2 * gammas * Tan((sphi - 5) * Pi / 180)
    Else
       fas = ffmax * (pilel - 0.5 * plc) * piled / 12 * Pi
    End If
    dummy = 0
    If plug = True Then dummy = Application.Min(qmax, nq * pilel * gammas) * Pi * (piled / 12) ^ 2 / 4
    qcs = sabias * (Application.Min(pilepy, fas + dummy) - wp * pilel)
    If pilel < plt Then
       dummy = 0.7 * 0.5 * pilel ^ 2 * gammas * Tan((sphi - 5) * 3.14 / 180)
    Else
       dummy = ffmax * (pilel - 0.5 * plt)
    End If
    qts = sabias * (Application.Min(pilepy, dummy * 3.14 * piled / 12 + wp * pilel))
'
'   Axial capacity of skirt piles if founded in sand
'
    If skirt = True Then
       If spilel < plc Then
          fas = 0.5 * spilel ^ 2 * gammas * Tan((sphi - 5) * 3.14 / 180)
       Else
          fas = ffmax * (spilel - 0.5 * plc) * spiled / 12 * 3.14
       End If
       dummy = 0
       If splug = True Then dummy = Application.Min(qmax, nq * spilel * gammas) * 3.14 * (spiled / 12) ^ 2 / 4
       sqcs = sabias * (Application.Min(spilepy, fas + dummy) - swp * spilel)
       If spilel < plt Then
          dummy = 0.7 * 0.5 * spilel ^ 2 * gammas * Tan((sphi - 5) * 3.14 / 180)
       Else
          dummy = ffmax * (spilel - 0.5 * plt)
       End If
       sqts = sabias * (Application.Min(spilepy, dummy * 3.14 * spiled / 12 + swp * spilel))
    Else
       sqcs = 0
       sqts = 0
    End If
'
'   Lateral capacity of main piles if founded in sand
'
    kp = (Tan((45 + sphi / 2) * 3.14 / 180)) ^ 2
    If scour = 0 Then
       pusbar = (2.382 * pilem ^ (2 / 3) * (gammas * piled / 12 * kp) ^ (1 / 3))
    Else
       pusbar = (2.382 * pilem ^ (2 / 3) * (gammas * piled / 12 * kp) ^ (1 / 3))
'
'   Newton-Raphson iteration to find lateral capacity
'
       Do
          Formula = pusbar - (2 * pilem / (scour + 0.544 * (pusbar / gammas / piled / 12 / kp)) ^ 0.5)
          Formula1 = 1 + pilem * 0.544 / (gammas * piled / 12 * kp) * (scour + 0.544 * (pusbar / gammas / piled / 12 / kp)) ^ -1.5
          pusbartest = pusbar
          pusbar = pusbar - Formula / Formula1
       Loop While ((Abs(pusbartest - pusbar) > 1) And (Abs(Formula) > 0.0001))
    End If
    For i = 1 To 2
       pus(i) = slbias * pusbar * nleg
    Next i
'
'   Lateral capacity of skirt piles if founded in sand
'
    If skirt = True Then
       If scour = 0 Then
          pusbar = (2.382 * spilem ^ (2 / 3) * (gammas * spiled / 12 * kp) ^ (1 / 3))
       Else
          pusbar = (2.382 * spilem ^ (2 / 3) * (gammas * spiled / 12 * kp) ^ (1 / 3))
```

```
'
'  Newton-Raphson iteration to find lateral capacity
'
        Do
          Formula = pusbar - (2 * spilem / (scour + 0.544 * (pusbar / gammas / spiled / 12 / kp)) ^ 0.5)
          Formula1 = 1 + spilem * 0.544 / (gammas * spiled / 12 * kp) * (scour + 0.544 * (pusbar / gammas / spiled / 12 / kp)) ^
                -1.5
          pusbartest = pusbar
          pusbar = pusbar - Formula / Formula1
        Loop While ((Abs(pusbartest - pusbar) > 1) And (Abs(Formula) > 0.0001))
      End If
      For i = 1 To 2
        spus(i) = slbias * pusbar * nsk
      Next i
    Else
      For i = 1 To 2
        spus(i) = 0
      Next i
    End If
End If
'
'  Now find pile axial capacities
'
For i = 1 To 2
If structuretype = "monopod_braced" Or structuretype = "monopod_guyed" Then
    If stype = 1 Then
      sandpilecapcomp(i) = Application.Max(qcs, 0.001)
      sandpilecaptens(i) = Application.Max(qts, 0.001)
    Else
      claypilecapcomp(i) = Application.Max(qcc, 0.001)
      claypilecaptens(i) = Application.Max(qtc, 0.001)
    End If
ElseIf structuretype = "tripod" Then
      If stype = 1 Then
        sandpilecapcomp(i) = Application.Max((qcs * n(i) + sqcs * nskirt(i)) / (n(i) + nskirt(i)), 0.001)
        sandpilecaptens(i) = Application.Max((qts * n(i) + sqts * nskirt(i)) / (n(i) + nskirt(i)), 0.001)
      Else
        claypilecapcomp(i) = Application.Max((qcc * n(i) + sqcc * nskirt(i)) / (n(i) + nskirt(i)), 0.001)
        claypilecaptens(i) = Application.Max((qtc * n(i) + sqtc * nskirt(i)) / (n(i) + nskirt(i)), 0.001)
      End If
Else
      If stype = 1 Then
        sandpilecapcomp(i) = Application.Max((qcs * n(i) + sqcs * nskirt(i) / 2) / (n(i) + nskirt(i) / 2), 0.001)
        sandpilecaptens(i) = Application.Max((qts * n(i) + sqts * nskirt(i) / 2) / (n(i) + nskirt(i) / 2), 0.001)
      Else
        claypilecapcomp(i) = Application.Max((qcc * n(i) + sqcc * nskirt(i) / 2) / (n(i) + nskirt(i) / 2), 0.001)
        claypilecaptens(i) = Application.Max((qtc * n(i) + sqtc * nskirt(i) / 2) / (n(i) + nskirt(i) / 2), 0.001)
      End If
End If
Next i

End Sub '  End FOUNDATION CAPACITY

'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Sub foundation_stiffness() '  Start FOUNDATION STIFFNESS
'
'  Pilehead stiffnesses.  All are in kip and ft.  Penzien springs are used.
'
pileheadkx = horizkbias * 12 * 18.2 * Gsoil * piled / 2 * (1 - nu ^ 2) / (2 - nu) ^ 2
pileheadkz = axialkbias * e * pilea / pilel
'
'  Find vertical stiffness
'
kvertical = nleg * legeffectivea * e / elevation(0)


If skirt = True Then
'
'  Skirt pilehead stiffnesses.  All are in kip and ft
```

```
'
   spileheadkx = horizkbias * 12 * 18.2 * Gsoil * spiled / 2 * (1 - nu ^ 2) / (2 - nu) ^ 2
   spileheadkz = axialkbias * e * spilea / spilel

End If
'
'  Determine foundation vertical, horizontal, and overturning stiffness.  It is assumed
'  that pile-structure connections are rigid.
'
foundationkx = nleg * pileheadkx + nsk * spileheadkx
If nleg = 3 Then
   kthetabend(1) = (2 * 3 * e * n(1) * (legeffectivea) * (bcw(1) / 2) ^ 2) / elevation(0)
   kthetabend(2) = (3 * e * legeffectivea * (2 * (bcw(2) * Sin(offangle) / 3) ^ 2 + (2 * bcw(2) * Sin(offangle) / 3) ^ 2)) /
elevation(0)
   foundationktheta(1) = 2 * pileheadkz * (bcw(1) / 2) ^ 2 + 2 * nskirt(1) * spileheadkz * (bcw(1) / 2) ^ 2
   foundationktheta(2) = 2 * pileheadkz * (bcw(2) * Sin(offangle) / 3) ^ 2 + 2 * nskirt(1) * spileheadkz * (bcw(2) * Sin(offangle) /
         3) ^ 2 + pileheadkz * (bcw(2) * 2 * Sin(offangle) / 3) ^ 2 + nskirt(2) * spileheadkz * (2 * bcw(2) * Sin(offangle) / 3) ^ 2
ElseIf nleg = 4 Or nleg = 6 Then
   If structuretype = "multijacket" Then
      kthetabend(1) = (12 * e * 2 * n(1) * (legeffectivea) * (bcw(1) / 2) ^ 2) / elevation(0)
      kthetabend(2) = (12 * e * 2 * n(2) * (legeffectivea) * (bcw(2) / 2) ^ 2) / elevation(0)
   Else
      kthetabend(1) = (3 * e * 2 * n(1) * (legeffectivea) * (bcw(1) / 2) ^ 2) / elevation(0)
      kthetabend(2) = (3 * e * 2 * n(2) * (legeffectivea) * (bcw(2) / 2) ^ 2) / elevation(0)
   End If
   foundationktheta(1) = 2 * n(1) * pileheadkz * (bcw(1) / 2) ^ 2 + nskirt(1) * spileheadkz * (bcw(1) / 2) ^ 2
   foundationktheta(2) = 2 * n(2) * pileheadkz * (bcw(2) / 2) ^ 2 + nskirt(2) * spileheadkz * (bcw(2) / 2) ^ 2
Else
   kthetabend(1) = (3 * e * 2 * n(1) * (legeffectivea) * (bcw(1) / 2) ^ 2) / elevation(0)
   kthetabend(2) = (3 * e * 2 * n(2) * (legeffectivea) * (bcw(2) / 2) ^ 2) / elevation(0) + (2 * 3 * e * n(2) * (legeffectivea / 144) *
         (msw / 2) ^ 2) / elevation(0)
   foundationktheta(1) = 2 * n(1) * pileheadkz * (bcw(1) / 2) ^ 2 + nskirt(1) * spileheadkz * (bcw(1) / 2) ^ 2
   foundationktheta(2) = 2 * n(2) * pileheadkz * ((bcw(2) / 2) ^ 2 + (msw / 2) ^ 2) + nskirt(2) * spileheadkz * (bcw(2) / 2) ^ 2
End If
'
'  Now find total rotational stiffness, assuming bending and foundation rotation are
'  series springs.
'
For i = 1 To 2
   foundationktheta(i) = foundationktheta(i) * kthetabend(i) / (foundationktheta(i) + kthetabend(i))
Next i
'
'  Find vertical stiffness
'
foundationkz = nleg * pileheadkz + nsk * spileheadkz
kvertical = kvertical * foundationkz / (kvertical + foundationkz)

End Sub '      End FOUNDATION STIFFNESS

'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Sub capacity_profile() '  Start CAPACITY PROFILE
'
'  Now determine platform horizontal loading capacity profiles, including reduction in
'  first jacket bay capacity due to deck leg end moments, and increase in jacket bay
'  capacities due to better forces.
'
For i = 1 To 2
   If bayh(0) <= 1.5 Then
      lbcap(i, 0) = cap(i, 1) + 2 * legfh(i, 1)
      ubcap(i, 0) = capu(i, 1) + 2 * legfh(i, 1)
   Else
      lbcap(i, 0) = dlcap(i)
      ubcap(i, 0) = dlcap(i)
   End If
   For j = 1 To nbay
      jcap(i, j) = jcap(i, j) + 2 * legfh(i, j)
      lbcap(i, j) = cap(i, j) + 2 * legfh(i, j)
      ubcap(i, j) = capu(i, j) + 2 * legfh(i, j)
      If j = 1 Then jcap(i, j) = jcap(i, j) - shear(i)
```

```
      If j = 1 Then lbcap(i, j) = lbcap(i, j) - shear(i)
      If j = 1 Then ubcap(i, j) = ubcap(i, j) - shear(i)
   Next j
   If stype = 1 Then
      fcap(i) = Application.Max(pus(i) + spus(i) + 2 * legfh(i, nbay + 1), 0.001)
      If pileloadcomp(i) = 0 Then
         rsrc(i) = 0
      Else
         rsrc(i) = sandpilecapcomp(i) / pileloadcomp(i)
      End If
      If pileloadtens(i) = 0 Then
         rsrt(i) = 0
      Else
         rsrt(i) = sandpilecaptens(i) / pileloadtens(i)
      End If
   Else
      fcap(i) = Application.Max(puc(i) + spuc(i) + 2 * legfh(i, nbay + 1), 0.001)
      If pileloadcomp(i) = 0 Then
         rsrc(i) = 0
      Else
         rsrc(i) = claypilecapcomp(i) / pileloadcomp(i)
      End If
      If pileloadtens(i) = 0 Then
         rsrt(i) = 0
      Else
         rsrt(i) = claypilecaptens(i) / pileloadtens(i)
      End If
   End If
Next i
'
'  Determine capacity profile plot lines
'
For i = 1 To 2
elevbar = elevation(0)
interval = elevation(0) / 100
k = 0
   For j = 1 To 100
      If elevbar > elevation(k + 1) Then
         jcapbar(i, j) = jcap(i, k)
         lbcapbar(i, j) = lbcap(i, k)
         ubcapbar(i, j) = ubcap(i, k)
      Else
         jcapbar(i, j) = jcap(i, k)
         lbcapbar(i, j) = lbcap(i, k + 1)
         ubcapbar(i, j) = ubcap(i, k + 1)
         k = k + 1
      End If
      elevbar = elevbar - interval
   Next j
Next i

End Sub '  End CAPACITY PROFILE'
```

```
'           Module 6
'
'  EARTHQUAKE ANALYSIS ROUTINES
'
'  This module contains the procedures necessary for determining earthquake forces on
'  platforms which are responding in the elastic or near-elastic range.
'
'  Created by: James Stear
'  Created on: 5/13/96
'
'  Last Modified by:  James Stear
'  Last Modified on:  6/1/97
'
'
'
Public sweep(30, 30), z(30, 30), coef(30, 30), coefnew(30, 30)
Public phitemp(30), phimode(2, 3, 30), period(2, 3), vmode, flex(30, 30), modemoment(2, 3, 30)
Public modegamma(2, 3), modeheight(2, 3), modemass(2, 3), modek(2, 3), modeshear(2, 3, 30)
Public eqshear(2, 30), eqlegf(2, 30), foundationshear(2), cm, eqcov, pga, APIsoil, modecom
Public numconductors, diaconductors, tconductors, equipmentaccel(3), equipmentperiod
'
'
'
Sub masses()
'
'  This subroutine determines the lumped horizontal and vertical masses for use in the
'  modal analysis procedure.  Added masses are determined assuming a Cm=0.9.
'
'  Created by: James Stear
'  Created on: 5/13/97
'
'  Last Modified by:  James Stear
'  Last Modified on:  6/1/97
'
'  First find horizontal brace masses and added masses.  Mass of horizontal braces
'  includes the mass of attached marine growth.
'
'
'
'
'
For i = 1 To 2
   elevcheck = elevation(0)
   For j = 1 To nbay + 1
      If elevcheck - bayh(j - 1) > wdep + sdep Then
         cmused = 0
      ElseIf elevcheck - bayh(j - 1) > 0.9 * (wdep + sdep) Then
         cmused = cm * ((0.1 * (wdep + sdep)) - (elevcheck - bayh(j - 1) - (wdep + sdep))) / (0.1 * (wdep + sdep))
      Else
         cmused = cm
      End If
      elevcheck = elevcheck - bayh(j - 1)
      For k = 1 To nhb(j)
         If i = 2 Then
            addedmasshb(j, i) = addedmasshb(j, i) + cmused * rho * Pi * ((hbd(j, k) / 2 + mg(j)) / 12) ^ 2 * hbl(j, k) *
                  (Cos(hbang(j, k) * Pi / 180))
            addedmasshb(j, i + 1) = addedmasshb(j, i + 1) + cmused * rho * Pi * ((hbd(j, k) / 2 + mg(j)) / 12) ^ 2 * hbl(j, k)
         Else
            masshb(j) = masshb(j) + steelg * Pi * (hbd(j, k) - hbt(j, k)) * hbt(j, k) * hbl(j, k) / 144 + rho * Pi * (hbd(j, k) + mg(j)) *
                  mg(j) * hbl(j, k) / 144
            addedmasshb(j, i) = addedmasshb(j, i) + cmused * rho * Pi * ((hbd(j, k) / 2 + mg(j)) / 12) ^ 2 * hbl(j, k) * (Sin(hbang(j,
                  k) * Pi / 180))
         End If
      Next k
   Next j
Next i
addedmasshb(nbay + 1, 1) = 0.5 * addedmasshb(nbay + 1, 1)
addedmasshb(nbay + 1, 2) = 0.5 * addedmasshb(nbay + 1, 2)
addedmasshb(nbay + 1, 3) = 0.5 * addedmasshb(nbay + 1, 3)
'
```

```
'
'  Now determine main diagonal brace mass, and associated added mass.
'
For i = 1 To 2
   elevcheck = htotal
   For j = 1 To nbay
      If elevcheck - bayh(j) > wdep + sdep Then
         cmused = 0
      ElseIf elevcheck - bayh(j) > 0.9 * (wdep + sdep) Then
         cmused = cm * ((0.1 * (wdep + sdep)) - (elevcheck - bayh(j) - (wdep + sdep))) / (0.1 * (wdep + sdep))
      Else
         cmused = cm
      End If
      If (elevcheck > wdep + sdep) And (elevcheck - bayh(j) < wdep + sdep) Then
         addlength = (bayh(j) - (elevcheck - (wdep + sdep))) / bayh(j)
      Else
         addlength = 1
      End If
      elevcheck = elevcheck - bayh(j)
      For k = 1 To ndb(i, j)
         If dbcond(i, j, k) = 3 Then
            masshb(j) = masshb(j) + groutg * Pi * ((dbd(i, j, k) - dbt(i, j, k)) / 2 / 12) ^ 2 * dbl(i, j, k)
         End If
         massdb(j) = massdb(j) + steelg * Pi * (dbd(i, j, k) - dbt(i, j, k)) * dbt(i, j, k) * dbl(i, j, k) / 144 + rho * Pi * (dbd(i, j, k) +
               mg(j)) * mg(j) * dbl(i, j, k) / 144
         addedmassdb(j, 3) = addedmassdb(j, 3) + cmused * addlength * rho * Pi * ((dbd(i, j, k) / 2 + mg(j)) / 12) ^ 2 * dbl(i, j, k)
               * (Cos(theta(i, j, k)))
         If i = 1 Then
            addedmassdb(j, 1) = addedmassdb(j, 1) + cmused * addlength * rho * Pi * ((dbd(1, j, k) / 2 + mg(j)) / 12) ^ 2 * dbl(1,
                  j, k) * (Sin(theta(1, j, k)))
            addedmassdb(j, 2) = addedmassdb(j, 2) + cmused * addlength * rho * Pi * ((dbd(1, j, k) / 2 + mg(j)) / 12) ^ 2 * dbl(1,
                  j, k)
         Else
            addedmassdb(j, 2) = addedmassdb(j, 2) + cmused * addlength * rho * Pi * ((dbd(2, j, k) / 2 + mg(j)) / 12) ^ 2 * dbl(2,
                     j, k) * (Sin(theta(2, j, k)) ^ 2 + (Cos(theta(2, j, k)) ^ 2) * Sin(offangle) ^ 2) ^ 0.5
            addedmassdb(j, 1) = addedmassdb(j, 1) + cmused * addlength * rho * Pi * ((dbd(2, j, k) / 2 + mg(j)) / 12) ^ 2 * dbl(2,
                     j, k) * (Sin(theta(2, j, k)) ^ 2 + (Cos(theta(2, j, k)) ^ 2) * Sin(Pi / 2 - offangle) ^ 2) ^ 0.5
         End If
      Next k
   Next j
Next i

If structuretype = "monopod_guyed" Then

   addedmassdb(1, 1) = cmused * addlength * rho * Pi * ((dbd(1, 1, 1) / 2 + mg(1)) / 12) ^ 2 * dbl(1, 1, 1) * (Sin(theta(1, 1, 1)))
                     + cmused * addlength * rho * Pi * ((dbd(2, 1, 1) / 2 + mg(1)) / 12) ^ 2 * dbl(2, 1, 1) * (Sin(theta(2, 1, 1)) ^ 2 +
                     (Cos(theta(2, 1, 1)) ^ 2) * Sin(Pi / 3) ^ 2) ^ 0.5
   addedmassdb(1, 2) = addedmassdb(1, 1)
   massdb(1) = steelg * Pi * (dbd(1, 1, 1) / 2) ^ 2 * dbl(1, 1, 1) / 144 + rho * Pi * (dbd(1, 1, 1) + mg(1)) * mg(1) * dbl(1, 1, 1) /
                     144

End If



'
'  Find jacket leg and pile mass and added mass.  Legs are assumed to be flooded.  This
'  includes the mass of any conductors, if present.  Conductors are assumed flooded.
'
If htotal > wdep + sdep Then
   cmused = 0
ElseIf htotal > 0.9 * (wdep + sdep) Then
   cmused = cm * ((0.1 * (wdep + sdep)) - (htotal - (wdep + sdep))) / (0.1 * (wdep + sdep))
Else
   cmused = cm
End If
If (elevation(0) > wdep + sdep) And (htotal < wdep + sdep) Then
   addlength = (bayh(0) - (elevation(0) - (wdep + sdep))) / bayh(i)
Else
   addlength = 1
```

```
End If
dumcd = numconductors * bayh(0) * (Pi * (diaconductors - tconductors) * tconductors * steelg / 144 + addlength * (cmused + 1)
        * rho * Pi * ((diaconductors / 2 + mg(i)) / 12) ^ 2)
decklegmass = decklegsw + addlength * cmused * rho * nleg * Pi * ((dld / 2 + mg(i)) / 12) ^ 2 * bayh(0) + dumcd
dumcd = 0
'
'
'
For i = 1 To nbay
   If elevcheck - bayh(i) > wdep + sdep Then
      cmused = 0
   ElseIf elevcheck - bayh(i) > 0.9 * (wdep + sdep) Then
      cmused = cm * ((0.1 * (wdep + sdep)) - (elevcheck - bayh(i) - (wdep + sdep))) / (0.1 * (wdep + sdep))
   Else
      cmused = cm
   End If
   If (elevcheck > wdep + sdep) And (elevcheck - bayh(i) < wdep + sdep) Then
      addlength = (bayh(i) - (elevcheck - (wdep + sdep))) / bayh(i)
   Else
      addlength = 1
   End If
   If pgrout = True Then
      pilelegmass(i) = steelg * nleg * Pi * ((jld(1, 1) - jlt(1)) * jlt(1) / 144 + (piled - pilet) * pilet / 144) * bayh(i) + groutg * nleg * Pi *
            (((jld(1, 1) - 2 * jlt(1)) / 2) ^ 2 - (piled / 2) ^ 2) * bayh(i) / 144
      pilelegaddedmass(i) = addlength * rho * nleg * Pi * (((jld(1, 1) + mg(i)) * mg(i) + (piled / 2) ^ 2) / 144) * bayh(i) + addlength
            * cmused * rho * nleg * Pi * ((jld(1, 1) / 2 + mg(i)) / 12) ^ 2 * bayh(i)
   Else
      If structuretype = "monopod_braced" Or structuretype = "monopod_guyed" Then
         pilelegmass(i) = steelg * nleg * Pi * ((jld(1, 1) - jlt(1)) * jlt(1) / 144) * bayh(i)
      Else
         pilelegmass(i) = steelg * nleg * Pi * ((jld(1, 1) - jlt(1)) * jlt(1) / 144 + (piled - pilet) * pilet / 144) * bayh(i)
      End If
      pilelegaddedmass(i) = addlength * rho * nleg * Pi * ((jld(1, 1) / 2 + mg(i)) / 12) ^ 2 * bayh(i) + addlength * cmused * rho *
            nleg * Pi * ((jld(1, 1) / 2 + mg(i)) / 12) ^ 2 * bayh(i)
   End If
      pilelegmass(i) = pilelegmass(i) + numconductors * bayh(i) * Pi * (diaconductors - tconductors) * tconductors * steelg / 144
      pilelegaddedmass(i) = pilelegaddedmass(i) + numconductors * bayh(i) * addlength * (cmused + 1) * rho * Pi *
            ((diaconductors / 2 + mg(i)) / 12) ^ 2
Next i
'
'
' Find skirt pile mass and added mass. It is assumed the skirt piles stand above the
' mudline a distance equal to the height of the bottom bay.
'
skirtmass = steelg * nsk * spilea / 144 * bayh(nbay)
skirtaddedmass = 2 * rho * Pi * nsk * (spiled / 2 + mg(nbay)) ^ 2 / 144 * bayh(nbay)
'
' Also find foundation mass. It is assumed an amount of soil (both added and internal)
' to a depth of 5 pile diameters rides with the bottom bay.
'
foundationmass = (steelg * nleg * pilea / 144 * 5 * piled / 12) + (gammas * nleg * Pi * (piled / 2) ^ 2 / 144 * 5 * piled / 12)
sfoundationmass = (steelg * nsk * spilea / 144 * 5 * spiled / 12) + (gammas * nsk * Pi * (spiled / 2) ^ 2 / 144 * 5 * spiled / 12)
'
' Now find lumped horizontal masses
'
For i = 1 To 2
   mass(i, 1) = qdeck + decklegmass / 2
   mass(i, 2) = decklegmass / 2
   For j = 1 To nbay
      mass(i, j + 1) = mass(i, j + 1) + masshb(j) + addedmasshb(j, i) + 0.5 * (massdb(j) + addedmassdb(j, i) + pilelegmass(j) +
            pilelegaddedmass(j))
      mass(i, j + 2) = mass(i, j + 2) + 0.5 * (massdb(j) + addedmassdb(j, i) + pilelegmass(j) + pilelegaddedmass(j))
      If i = 2 Then
         zmass = zmass + masshb(j) + addedmasshb(j, 3) + massdb(j) + addedmassdb(j, 3) + pilelegmass(j) + 0.5 *
            pilelegaddedmass(j)
      End If
   Next j
   mass(i, nbay + 1) = mass(i, nbay + 1) + 0.5 * skirtaddedmass + 0.5 * skirtmass
   mass(i, nbay + 2) = masshb(nbay + 1) + addedmasshb(nbay + 1, i) + mass(i, nbay + 2) + foundationmass + 0.5 *
         skirtaddedmass + 0.5 * skirtmass + sfoundationmass
```

```
Next i
'
'  Find total vertical mass
'
zmass = zmass + qdeck + decklegmass + 0.5 * skirtaddedmass + skirtmass + foundationmass + sfoundationmass +
        masshb(nbay + 1) + addedmasshb(nbay + 1, 3)
'
'  Temporary mass checking, change from weight to mass
'
For i = 1 To 2
   For j = 1 To nbay + 2
      mass(i, j) = mass(i, j) / g
   Next j
Next i
zmass = zmass / g
'
'
'
End Sub

'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Sub eigen()
'
'  This subroutine determines mode shapes and natural periods for the first three modes
'  of vibration for both the end-on and broadside directions.  The procedure makes use of
'  the fact that the mass matrix is diagonal with non-zero elements to reduce the
'  complexity of the problem to solving the standard eigenvalue problem:
'
'  A*phi=lambda*phi
'
'
'  Loop over both directions
'


For i = 1 To 2
'
'  Initialize matrix values to zero
'
   For j = 1 To nbay + 1
      For k = 1 To nbay + 1
         coef(j, k) = 0
         coefnew(j, k) = 0
         sweep(j, k) = 0
         z(j, k) = 0
         flex(j, k) = 0
      Next k
   Next j
'
'  First fill in diagonals
'
   bayleft = 1
   For j = 1 To nbay + 1
      If j = 1 Then
         flex(j, j) = 1 / deckk(i)
         For l = 1 To nbay
            flex(j, j) = flex(j, j) + 1 / kbay(i, l)
         Next l
      Else
         For l = bayleft To nbay
            flex(j, j) = flex(j, j) + 1 / kbay(i, l)
         Next l
         bayleft = bayleft + 1
      End If
   Next j
'
'  Now create remaining entries
'
   For j = 2 To nbay + 1
```

```
      For k = 1 To j - 1
         flex(k, j) = flex(j, j)
         flex(j, k) = flex(j, j)
      Next k
   Next j
'
'  Now combine with M to form coeficient matrix
'
   For k = 1 To nbay + 1
      For l = 1 To nbay + 1
         coef(k, l) = flex(k, l) * mass(i, l)
      Next l
   Next k
'
'  Now begin eigenvector iteration.  The algorythm used in this procedure is based upon
'  one developed by W. Weaver, jr. ("EIGIT3") and documented in VIBRATION PROBLEMS IN
'  ENGINEERING, 4th Edition, by S. Timoshenko, D. H. Young and W. Weaver, jr.
'
   vmode = 1
   dumb = 0
   Do While vmode < Application.Min(nbay + 2, 4)
      For j = 1 To nbay + 1
         phitemp(j) = 1
      Next j
      counter = 0
      Do While counter < nbay + 1
         For j = 1 To nbay + 1
            phimode(i, vmode, j) = 0    '  clears previous value
            For k = 1 To nbay + 1
               phimode(i, vmode, j) = phimode(i, vmode, j) + coef(j, k) * phitemp(k)
            Next k
         Next j
         eigenv = phimode(i, vmode, 1) / phitemp(1)
         norm = phimode(i, vmode, 1)
         counter = 0
         For j = 1 To nbay + 1
            phimode(i, vmode, j) = phimode(i, vmode, j) / norm
            If Abs(phitemp(j) - phimode(i, vmode, j)) < 0.0001 Then counter = counter + 1
            phitemp(j) = phimode(i, vmode, j)
         Next j
         dumb = dumb + 1
         If dumb > 10000 Then
            dumb = dumb
            End
         End If
      Loop
      period(i, vmode) = Abs(eigenv)
      For j = 1 To nbay + 1
         For k = 1 To nbay + 1
            If j = k Then
               sweep(j, k) = 1
            Else
               sweep(j, k) = 0
            End If
         Next k
      Next j
      If vmode = 1 Then
         sweep(nbay + 1, nbay + 1) = 0
         z(nbay + 1, 2) = phitemp(nbay + 1)
         cvalue = mass(i, nbay + 1) * phitemp(nbay + 1)
         For j = 2 To nbay + 1
            sweep(nbay + 1, nbay + 2 - j) = -1 * mass(i, nbay + 2 - j) * phitemp(nbay + 2 - j) / cvalue
            z(nbay + 2 - j, 2) = phitemp(nbay + 2 - j)
         Next j
      ElseIf vmode = 2 Then
         sweep(nbay, nbay) = 0
         z(nbay + 1, 1) = phitemp(nbay + 1)
         z(nbay, 1) = phitemp(nbay)
         cvalue = mass(i, nbay) * (z(nbay + 1, 2) * z(nbay, 1) - z(nbay + 1, 1) * z(nbay, 2))
         For j = 3 To nbay + 1
```

```
            z(nbay + 2 - j, 1) = phitemp(nbay + 2 - j)
            sweep(nbay, nbay + 2 - j) = -1 * mass(i, nbay + 2 - j) * (z(nbay + 1, 2) * z(nbay + 2 - j, 1) - z(nbay + 1, 1) * z(nbay + 2
                - j, 2)) / cvalue
         Next j
      End If
      For j = 1 To nbay + 1
         For k = 1 To nbay + 1
            For l = 1 To nbay + 1
               coefnew(j, k) = coefnew(j, k) + coef(j, l) * sweep(l, k)
            Next l
         Next k
      Next j
      For j = 1 To nbay + 1
         For k = 1 To nbay + 1
            coef(j, k) = coefnew(j, k)
            coefnew(j, k) = 0
         Next k
      Next j
      vmode = vmode + 1
   Loop
Next i
'
'  Find mode periods, and normalize mode shapes by largest value
'
Worksheets("Sheet5").Cells(105, 10) = "Periods and Mode Shapes"
Worksheets("Sheet5").Cells(107, 11) = "Broadside"
Worksheets("Sheet5").Cells(107, 15) = "End-On"
Worksheets("Sheet5").Cells(107, 19) = "Vertical"
Worksheets("Sheet5").Cells(109, 10) = "Mode"
Worksheets("Sheet5").Cells(111, 10) = "Period (sec)"
Worksheets("Sheet5").Cells(113, 10) = "Deck"
'
'
'
For i = 1 To 2
   For j = 1 To Application.Min(nbay + 1, 3)
      period(i, j) = 2 * Pi * (period(i, j)) ^ 0.5
      Worksheets("Sheet5").Cells(109, 10 + j + 4 * (i - 1)) = j
      Worksheets("Sheet5").Cells(111, 10 + j + 4 * (i - 1)) = Application.Round(period(i, j), 2)
      norm = 1
      For k = 1 To nbay + 1
         If k > 1 Then Worksheets("Sheet5").Cells(112 + k, 10) = k - 1
         If Abs(phimode(i, j, k)) > norm Then norm = Abs(phimode(i, j, k))
      Next k
      For k = 1 To nbay + 1
         phimode(i, j, k) = phimode(i, j, k) / norm
         Worksheets("Sheet5").Cells(112 + k, 10 + j + 4 * (i - 1)) = Application.Round(phimode(i, j, k), 3)
      Next k
   Next j
Next i

End Sub

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Function SA(t, soil)

If t < 0.04 Then
   SA = 1
ElseIf t >= 0.04 And t < 0.125 Then
   SA = 20 * t
Else
   SA = 2.5
End If
If t > 0.32 And soil = 1 Then
   SA = 0.8 / t
ElseIf t > 0.48 And soil = 2 Then
   SA = 1.2 / t
ElseIf t > 0.72 And soil = 3 Then
   SA = 1.8 / t
```

```
End If

End Function

'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Sub RSA()
'
'  First find mode participation factors
'
For i = 1 To 2
   For j = 1 To Application.Min(3, nbay + 1)
      Lhtemp = 0
      Mn = 0
      ltheta = 0
      elevcheck = elevation(0)
      For k = 1 To nbay + 1
         Mn = Mn + mass(i, k) * phimode(i, j, k) * phimode(i, j, k)
         Lhtemp = Lhtemp + mass(i, k) * phimode(i, j, k)
         ltheta = ltheta + mass(i, k) * phimode(i, j, k) * elevcheck
         elevcheck = elevcheck - bayh(k - 1)
      Next k
      modegamma(i, j) = Lhtemp / Mn
      modeheight(i, j) = ltheta / Lhtemp
      modemass(i, j) = Lhtemp ^ 2 / Mn
      modek(i, j) = 4 * Pi ^ 2 * modemass(i, j) / (period(i, j) ^ 2)
   Next j
Next i
'
'  Modify 1st mode periods to account for foundation effects
'
If foundationkx = 0 Then
   Tfoundation = 0
Else
   Tfoundation = 2 * Pi * (mass(1, nbay + 2) / foundationkx) ^ 0.5
End If
For i = 1 To 2
   foundationshear(i) = mass(i, nbay + 2) * SA(Tfoundation, APIsoil) * g * pga
   If foundationkx = 0 Or foundationktheta(i) = 0 Then
      period(i, 1) = period(i, 1)
   Else
      periodfixed = period(i, 1)
      Do While Abs(period(i, 1) - periodcheck) > 0.001
         periodcheck = period(i, 1)
         period(i, 1) = periodfixed * (1 + (modek(i, 1) / foundationkx) * (1 / (1 - (Tfoundation / period(i, 1)) ^ 2) + (foundationkx *
                modeheight(i, 1) ^ 2) / foundationktheta(i))) ^ 0.5
      Loop
   End If
   Worksheets("Sheet5").Cells(111, 11 + 4 * (i - 1)) = Application.Round(period(i, 1), 2)
Next i
'
'  Now find shear demands and moments at each bay specific to each mode
'
For i = 1 To 2
   For j = 1 To Application.Min(3, nbay + 1)
      level = 0
      For k = 1 To nbay + 1
         level = level + 1
         For l = 1 To level
            elevcheck = elevcheck + bayh(l - 1)
         Next l
         For l = 1 To level
            modeshear(i, j, k) = modeshear(i, j, k) + modegamma(i, j) * mass(i, l) * phimode(i, j, l) * SA(period(i, j), APIsoil) * g *
                  pga
            modemoment(i, j, k) = modemoment(i, j, k) + modegamma(i, j) * mass(i, l) * phimode(i, j, l) * SA(period(i, j), APIsoil)
                  * g * pga * elevcheck
            elevcheck = elevcheck - bayh(l - 1)
         Next l
      Next k
   Next j
```

```
Next i
'
' Find total shear demands on bays, as well as leg forces. Give option to use either SRSS or ABS
'
For i = 1 To 2
   For j = 1 To nbay + 1
      modesumshear = 0
      modesummoment = 0
      For k = 1 To Application.Min(3, nbay + 1)
         If modecom = 1 Then
            modesumshear = modesumshear + modeshear(i, k, j) ^ 2
            modesummoment = modesummoment + modemoment(i, k, j) ^ 2
         Else
            modesumshear = modesumshear + Abs(modeshear(i, k, j))
            modesummoment = modesummoment + Abs(modemoment(i, k, j))
         End If
      Next k
      If modecom = 1 Then
         modesumshear = modesumshear ^ 0.5
         modesummoment = modesummoment ^ 0.5
      End If

      If j = nbay + 1 Then
         meanload(i, j - 1) = modesumshear
         meanload(i, j) = (modesumshear ^ 2 + foundationshear(i) ^ 2) ^ 0.5
         legf(i, j) = modesummoment / bcw(i)
      Else
         meanload(i, j - 1) = modesumshear
         legf(i, j) = modesummoment / iht(i, j)
      End If
      legfh(i, j) = legf(i, j) * Sin(alpha(i))
      covload(i, j - 1) = eqcov
   Next j
'
' Find reduction in strength of top jacket bay due to bending of unbraced bay.
'
   If structuretype = "monopod_braced" Or structuretype = "monopod_guyed" Then
      dlmbar(i, 1) = Abs(Application.Min(meanload(i, 0) * bayh(0) * 1.5, dlm))
   Else
      dlmbar(i, 0) = Abs(Application.Min(meanload(i, 0) / nleg * bayh(0) * ((bayh(0) / (2 * e * dli / 144) + 1 / dlcm(i)) / (bayh(0) /
            (e * dli / 144) + 1 / dlcm(i))), dlm))
      dlmbar(i, 1) = Abs(Application.Min(dlmbar(i, 0) - meanload(i, 0) / nleg * bayh(0), dlm))
   End If
   If bayh(0) <= 1.5 Then
      shear(i) = 0
   Else
      If structuretype = "monopod_braced" Or structuretype = "monopod_guyed" Then
         shear(i) = dlmbar(i, 1) / (bayh(1) + 10 * piled / 12)
      Else
         shear(i) = dlmbar(i, 1) / bayh(1) * nleg
      End If
   End If
Next i
'
' Find vertical forces
'
Tvertical = 2 * PI / (kvertical / zmass) ^ 0.5
zforce = zmass * SA(Tvertical, APIsoil) * g * pga / 2
Worksheets("Sheet5").Cells(111, 19) = Application.Round(Tvertical, 2)
'
' Find acceleration for deck mounted equipment
'
If equipmentperiod / Tvertical > 1.25 Then
   If equipmentperiod / Tvertical > 2 Then
      equipmentaccel(3) = 1.4 * SA(equipmentperiod, APIsoil)
   Else
      equipmentaccel(3) = (3 - 1.6 * ((equipmentperiod / Tvertical - 1.25) / 0.75)) * SA(equipmentperiod, APIsoil)
   End If
Else
   If equipmentperiod / Tvertical < 0.2 Then
```

```
      equipmentaccel(3) = SA(Tvertical, APIsoil)
    ElseIf equipmentperiod / Tvertical < 1 Then
      equipmentaccel(3) = (1 + 7 * ((equipmentperiod / Tvertical - 0.2) / 0.8)) * SA(Tvertical, APIsoil)
    Else
      equipmentaccel(3) = (8 - 5 * ((equipmentperiod / Tvertical - 1) / 0.25)) * SA(Tvertical, APIsoil)
    End If
  End If
'
'
'
  For i = 1 To 2
    accelsum = 0
    accelprime = 0
    accel2prime = 0
    For j = 1 To Application.Min(3, nbay + 1)
      If equipmentperiod / period(i, j) > 1.25 Then
        If equipmentperiod / period(i, j) > 2 Then
          accelprime = accelprime + (1.4 * modegamma(i, j) * SA(equipmentperiod, APIsoil) * phimode(i, j, 1)) ^ 2
        Else
          accelprime = accelprime + ((3 - 1.6 * ((equipmentperiod / period(i, j) - 1.25) / 0.75)) * modegamma(i, j) *
              SA(equipmentperiod, APIsoil) * phimode(i, j, 1)) ^ 2
        End If
      Else
        If equipmentperiod / period(i, j) < 0.2 Then
          accel2prime = accel2prime + (modegamma(i, j) * SA(period(i, j), APIsoil) * phimode(i, j, 1)) ^ 2
        ElseIf equipmentperiod / period(i, j) < 1 Then
          accel2prime = accel2prime + ((1 + 7 * ((equipmentperiod / period(i, j) - 0.2) / 0.8)) * modegamma(i, j) * SA(period(i,
              j), APIsoil) * phimode(i, j, 1)) ^ 2
        Else
          accel2prime = accel2prime + ((8 - 5 * ((equipmentperiod / period(i, j) - 1) / 0.25)) * modegamma(i, j) * SA(period(i, j),
              APIsoil) * phimode(i, j, 1)) ^ 2
        End If
      End If
      accelsum = accelsum + (modegamma(i, j) * phimode(i, j, 1)) ^ 2
    Next j
    equipmentaccel(i) = (accelprime + accel2prime / accelsum) ^ 0.5
  Next i
  Worksheets("Sheet5").Cells(113, 2) = "BS Acceleration"
  Worksheets("Sheet5").Cells(114, 2) = "EO Acceleration"
  Worksheets("Sheet5").Cells(115, 2) = "Z Acceleration"
  Worksheets("Sheet5").Cells(113, 3) = Application.Round(equipmentaccel(1) * pga, 2)
  Worksheets("Sheet5").Cells(114, 3) = Application.Round(equipmentaccel(2) * pga, 2)
  Worksheets("Sheet5").Cells(115, 3) = Application.Round(equipmentaccel(3) * pga / 2, 2)
'
'  Find pile loads, SRSS is used to combine vertical forces
'
  For i = 1 To 2
'
'  Find forces due to global overturning for multi-jacket structures
'
    If structuretype = "multijacket" Then
      zforce = zforce + 2 * legf(i, nbay + 1) * bcw(i) / msw
    End If
    If structuretype = "tripod" Then
      pileloadcomp(i) = Application.Max((zforce + qdeck) / (nleg + nsk) + (1 / Sin(offangle)) ^ (i - 1) * legf(i, nbay + 1) / (nskirt(i)
          + n(i)), 0)
      pileloadtens(i) = Application.Max((zforce - qdeck) / (nleg + nsk) + (1 / Sin(offangle)) ^ (i - 1) * legf(i, nbay + 1) / (nskirt(i) +
          n(i)), 0)
    ElseIf structuretype = "monopod_braced" Or structuretype = "monopod_guyed" Then
      If structuretype = "monopod_guyed" Then
        qdeck = qdeck + 3 * pretension * Sin(theta(1, 1, 1)) ^ 2
      End If
      pileloadcomp(i) = Application.Max((zforce + qdeck) + legf(i, 2) + dlmbar(i, 1) / 3 / bcw(i), 0)
      pileloadtens(i) = Application.Max((zforce - qdeck) + legf(i, 2) + dlmbar(i, 1) / 3 / bcw(i), 0)
    Else
      pileloadcomp(i) = Application.Max((((zforce + qdeck) / (nleg + nskirt(1) + nskirt(2))) ^ 2 + (legf(i, nbay + 1) / (nskirt(i) / 2 +
          n(i))) ^ 2) ^ 0.5, 0.001)
      pileloadtens(i) = Application.Max((((zforce - qdeck) / (nleg + nskirt(1) + nskirt(2))) ^ 2 + (legf(i, nbay + 1) / (nskirt(i) / 2 +
          n(i))) ^ 2) ^ 0.5, 0.001)
    End If
```

```
Next i
'
'   Determine shear profiles, and zero-out water kinematics
'
For i = 1 To 2
   elevbar = elevation(0)
   interval = elevation(0) / 100
   k = 0
   For j = 1 To 100
      wvel(j) = 0
      cvel(j) = 0
      If elevbar > elevation(k + 1) Then
         cumf(i, j) = meanload(i, k)
      Else
         cumf(i, j) = meanload(i, k + 1)
         k = k + 1
      End If
      If k = nbay And elevbar <= 3 * interval Then
         cumf(i, j) = meanload(i, k + 1)
      End If
      elevbar = elevbar - interval
   Next j
Next i

End Sub

+++++++
```

Module 7:

```
'
'  This module contains routines relevant to the calculation of hydrodynamic forces on
'  a structure.
'
Sub kinematics() '  Start KINEMATICS
'
'  Determination water wave particle kinematics using Stokes' 5th-order
'  theory.  Program currently makes use of values supplied by Sheet4, a spreadsheet
'  developed by Darren Preston (1993 UCB NAOE).
'
'  Use Sheet4 to estimate wave number k and lambda from Stokes' 5th-order theory.
'
stk = 0
If wdep = 0 Then wdep = 1
If wavp = 0 Then wavp = 1
If wavh = 0 Then wavh = 1
Worksheets(4).Cells(8, 1) = wdep + sdep
Worksheets(4).Cells(8, 3) = wavp
Worksheets(4).Cells(8, 4) = wavh
Worksheets(4).Cells(11, 7).GoalSeek goal:=0, changingCell:=Worksheets(4).Cells(8, 6)
Worksheets(4).Cells(14, 7).GoalSeek goal:=0, changingCell:=Worksheets(4).Cells(14, 1)
stk = Worksheets(4).Cells(8, 6)
stlambda = Worksheets(4).Cells(14, 1)
stkd = stk * (wdep + sdep)
sts = (Application.Cosh(2 * stkd)) ^ (-1)
stc0 = (Application.Tanh(stkd)) ^ 0.5
stc2 = (stc0 * (2 + 7 * sts ^ 2)) / (4 * (1 - sts) ^ 2)
stc4 = (stc0 * (4 + 32 * sts - 116 * sts ^ 2 - 400 * sts ^ 3 - 71 * sts ^ 4 + 146 * sts ^ 5)) / (32 * (1 - sts) ^ 5)
stcosh = Application.Cosh(stkd)
stsinh = Application.Sinh(stkd)
c1 = (8 * stcosh ^ 4 - 8 * stcosh ^ 2 + 9) / (8 * stsinh ^ 4)
c2 = (3840 * stcosh ^ 12 - 4096 * stcosh ^ 10 - 2592 * stcosh ^ 8 - 1008 * stcosh ^ 6 + 5944 * stcosh ^ 4 - 1830 * stcosh ^ 2 +
      147) / (512 * stsinh ^ 10 * (6 * stcosh ^ 2 - 1))
a11 = 1 / stsinh
a13 = -(stcosh ^ 2 * (5 * stcosh ^ 2 + 1)) / (8 * stsinh ^ 5)
a15 = -(1184 * stcosh ^ 10 - 1440 * stcosh ^ 8 - 1992 * stcosh ^ 6 + 2641 * stcosh ^ 4 - 249 * stcosh ^ 2 + 18) / (1536 * stsinh ^
      11)
a22 = 3 / (8 * stsinh ^ 4)
a24 = (192 * stcosh ^ 8 - 424 * stcosh ^ 6 - 312 * stcosh ^ 4 + 480 * stcosh ^ 2 - 17) / (768 * stsinh ^ 10)
a33 = (13 - 4 * stcosh ^ 2) / (64 * stsinh ^ 7)
a35 = (512 * stcosh ^ 12 + 4224 * stcosh ^ 10 - 6800 * stcosh ^ 8 - 12808 * stcosh ^ 6 + 16704 * stcosh ^ 4 - 3154 * stcosh ^ 2
      + 107) / (4096 * stsinh ^ 13 * (6 * stcosh ^ 2 - 1))
a44 = (80 * stcosh ^ 6 - 816 * stcosh ^ 4 + 1338 * stcosh ^ 2 - 197) / (1536 * stsinh ^ 10 * (6 * stcosh ^ 2 - 1))
a55 = -(2880 * stcosh ^ 10 - 72480 * stcosh ^ 8 + 324000 * stcosh ^ 6 - 432000 * stcosh ^ 4 + 163470 * stcosh ^ 2 - 16245) /
      (61440 * stsinh ^ 11 * (6 * stcosh ^ 2 - 1) * (8 * stcosh ^ 4 - 11 * stcosh ^ 2 + 3))
b22 = (2 * stcosh ^ 2 + 1) * stcosh / (4 * stsinh ^ 3)
b24 = stcosh * (272 * stcosh ^ 8 - 504 * stcosh ^ 6 - 192 * stcosh ^ 4 + 322 * stcosh ^ 2 + 21) / (384 * stsinh ^ 9)
b33 = 3 * (8 * stcosh ^ 6 + 1) / (64 * stsinh ^ 6)
b35 = (88128 * stcosh ^ 14 - 208224 * stcosh ^ 12 + 70848 * stcosh ^ 10 + 54000 * stcosh ^ 8 - 21816 * stcosh ^ 6 + 6264 *
      stcosh ^ 4 - 54 * stcosh ^ 2 - 81) / (12288 * stsinh ^ 12 * (6 * stcosh ^ 2 - 1))
b44 = stcosh * (768 * stcosh ^ 10 - 448 * stcosh ^ 8 - 48 * stcosh ^ 6 + 48 * stcosh ^ 4 + 106 * stcosh ^ 2 - 21) / (384 * stsinh ^
      9 * (6 * stcosh ^ 2 - 1))
b55 = (192000 * stcosh ^ 16 - 262720 * stcosh ^ 14 + 83680 * stcosh ^ 12 + 20160 * stcosh ^ 10 - 7280 * stcosh ^ 8 + 7160 *
      stcosh ^ 6 - 1800 * stcosh ^ 4 - 1050 * stcosh ^ 2 + 225) / (12288 * stsinh ^ 10 * (6 * stcosh ^ 2 - 1) * (8 * stcosh ^ 4 -
      11 * stcosh ^ 2 + 3))
phi(1) = stlambda * a11 + stlambda ^ 3 * a13 + stlambda ^ 5 * a15
phi(2) = stlambda ^ 2 * a22 + stlambda ^ 4 * a24
phi(3) = stlambda ^ 3 * a33 + stlambda ^ 5 * a35
phi(4) = stlambda ^ 4 * a44
phi(5) = stlambda ^ 5 * a55
eta(1) = stlambda
eta(2) = stlambda ^ 2 * b22 + stlambda ^ 4 * b24
eta(3) = stlambda ^ 3 * b33 + stlambda ^ 5 * b35
eta(4) = stlambda ^ 4 * b44
eta(5) = stlambda ^ 5 * b55
celerity = (g * (wdep + sdep) * Application.Tanh(stkd) / stkd * (1 + stlambda ^ 2 * c1 + stlambda ^ 4 * c2)) ^ 0.5
dummy = 0
For i = 1 To 5
   dummy = dummy + eta(i)
```

```
Next i
crest = dummy / stk + wdep + sdep
For i = 1 To 100
   wvel(i) = 0 'initialize
Next i
wvcrest = 0
For i = 1 To 5
   wvcrest = wvcrest + i * Application.Cosh(i * crest * stk) * phi(i)
   For j = 1 To 100
      If elev(j) > crest Then
         wvel(j) = 0
      Else
         wvel(j) = wvel(j) + i * Application.Cosh(i * elev(j) * stk) * phi(i)
      End If
   Next j
Next i
wvcrest = celerity * wvcrest * ds
For j = 1 To 100
   wvel(j) = celerity * wvel(j) * ds
Next j
'
'  Now determine current velocity
'
For i = 1 To 100
   If elev(i) > crest Then
      cvel(i) = 0
   ElseIf cprof = 3 Then
      cvel(i) = cswl
   ElseIf cprof = 1 Then
      cvel(i) = cswl - ((cswl - cmdl) / crest) * (crest - elev(i))
   Else
      cvel(i) = cswl - ((cswl - cmdl) / crest ^ 2) * (crest - elev(i)) ^ 2
   End If
   cvel(i) = cvel(i) * cb
Next i
'
'  Compute total water particle velocities
'
cvcrest = cswl * cb
vcrest = wvcrest + cvcrest
For i = 1 To 100
   vel(i) = wvel(i) + cvel(i)
Next i
'
'  Find velocities at each bay
'
elevbar = elevation(0)
interval = elevation(0) / 100
k = 1
For j = 1 To 100
   velocity(k) = vel(j)
   If elevbar < elevation(k) Then k = k + 1
   elevbar = elevbar - interval
Next j

End Sub '  End KINEMATICS

'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Sub projected_areas() '  Start PROJECTED AREAS

For i = 1 To 2
'
'  Horizontal brace projected areas.
'
   For j = 1 To nbay + 1
      dummy = 0
      For k = 1 To nhb(j)
         If i = 2 Then
            dummy = dummy + (hbd(j, k) + 2 * mg(j)) / 12 * hbl(j, k) * (Cos(hbang(j, k) * Pi / 180)) ^ 3
```

```
        Else
          dummy = dummy + (hbd(j, k) + 2 * mg(j)) / 12 * hbl(j, k) * (Sin(hbang(j, k) * Pi / 180)) ^ 3
        End If
      Next k
      hbequa(i, j) = dummy
    Next j
'
'   Main diagonal brace unit projected areas.
'
    For j = 1 To nbay
      For k = 1 To ndb(i, j)
        If i = 1 Then
          dbdeque(2, j, k) = ((dbd(1, j, k) + 2 * mg(j)) / Sin(theta(1, j, k))) / 12
          dbdeque(1, j, k) = ((dbd(1, j, k) + 2 * mg(j)) * Sin(theta(1, j, k)) ^ 2) / 12
          dbdequb(1, j, k) = 0
          dbdequb(2, j, k) = 0
        Else
          dbdequb(1, j, k) = ((dbd(2, j, k) + 2 * mg(j)) / Sin(theta(2, j, k))) / 12 * (Sin(offangle) ^ 3 + (Cos(offangle) ^ 3) *
              Sin(theta(2, j, k)) ^ 3)
          dbdequb(2, j, k) = ((dbd(2, j, k) + 2 * mg(j)) / Sin(theta(2, j, k))) / 12 * (Sin(Pi / 2 - offangle) ^ 3 + (Cos(Pi / 2 -
              offangle) ^ 3) * Sin(theta(2, j, k)) ^ 3)
          dbdeque(2, j, k) = 0
          dbdeque(1, j, k) = 0
        End If
        dummye(1, j) = dummye(1, j) + dbdeque(1, j, k)
        dummye(2, j) = dummye(2, j) + dbdeque(2, j, k)
        dummyb(1, j) = dummyb(1, j) + dbdequb(1, j, k)
        dummyb(2, j) = dummyb(2, j) + dbdequb(2, j, k)
      Next k
      dbdequebar(1, j) = dummye(1, j)
      dbdequebar(2, j) = dummye(2, j)
      dbdequbbar(1, j) = dummyb(1, j)
      dbdequbbar(2, j) = dummyb(2, j)
    Next j

'
'   Deck leg unit projected areas.  Deck legs are accounted for as level(0) jacket legs.
'
    jld(i, 0) = dld
    dbdequebar(i, 0) = 0
    dbdequbbar(i, 0) = 0
Next i

If structuretype = "monopod_guyed" Then

    dbdeque(2, 1, 1) = 2 * ((dbd(2, 1, 1) + 2 * mg(1)) / Sin(theta(2, 1, 1))) / 12 * (Sin(Pi / 3) ^ 3 + (Cos(Pi / 3) ^ 3) * Sin(theta(2,
        1, 1)) ^ 3)
    dbdeque(1, 1, 1) = ((dbd(1, 1, 1) + 2 * mg(1)) * Sin(theta(1, 1, 1)) ^ 2) / 12

    dbdequb(2, 1, 1) = ((dbd(1, 1, 1) + 2 * mg(1)) * Sin(theta(1, 1, 1)) ^ 2) / 12
    dbdequb(1, 1, 1) = 2 * ((dbd(2, 1, 1) + 2 * mg(1)) / Sin(theta(2, 1, 1))) / 12 * (Sin(Pi / 3) ^ 3 + (Cos(Pi / 3) ^ 3) * Sin(theta(2,
        1, 1)) ^ 3)

End If


End Sub '  End PROJECTED AREAS

'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Sub deck_forces() '  Start DECK FORCES
'
'   Deck forces.  Deck forces are calculated according to API (1993) RP 2A LRFD Section 17.
'   Cd is scaled according to its proximity to the free surface unless design conditions
'   are specified.
'
crestbar = crest - wdep
For i = 1 To 2
    faerobar(i) = 0
    fhydrobar(i) = 0
```

```
    For j = 1 To ndeck
      deckh(j) = ok(j) - uk(j)
      decka(i, j) = deckh(j) * deckw(i, j)
      If crestbar > ok(j) Then
        If ((crestbar - ok(j) + deckh(j) / 2) < (vcrest ^ 2 / g)) And (designcd = False) Then
          cddeck = cdd(j) * (crestbar - ok(j) + deckh(j) / 2) / (vcrest ^ 2 / g)
        Else
          cddeck = cdd(j)
        End If
        fhydro(i, j) = (vcrest ^ 2 * cddeck * decka(i, j)) / 1000
        fhydroh(j) = uk(j) + deckh(j) / 2
        faero(i, j) = 0
        faeroh(j) = 0
      ElseIf crestbar < uk(j) Then
        fhydro(i, j) = 0
        fhydroh(j) = 0
        faero(i, j) = (0.00256 * vrh ^ 2 * wsc(j) * decka(i, j)) / 1000 'check units
        faeroh(j) = uk(j) + deckh(j) / 2
      Else
        If (((crestbar - uk(j)) / 2) < (vcrest ^ 2 / g)) And (designcd = False) Then
          cddeck = cdd(j) * ((crestbar - uk(j)) / 2) / (vcrest ^ 2 / g)
        Else
          cddeck = cdd(j)
        End If
        fhydro(i, j) = (vcrest ^ 2 * cddeck * (crestbar - uk(j)) * deckw(i, j)) / 1000
        fhydroh(j) = (uk(j) + crest - wdep) / 2
        faero(i, j) = (0.00256 * vrh ^ 2 * wsc(i) * (decka(i, j) - (crestbar - uk(j)) * deckw(i, j))) / 1000
        faeroh(j) = (ok(j) + crest - wdep) / 2
      End If
      faerobar(i) = faerobar(i) + faero(i, j)
      fhydrobar(i) = fhydrobar(i) + wdbias * fhydro(i, j)
    Next j
  Next i

End Sub '  End DECK FORCES

'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Sub hydro_profile() '  Start HYDRO PROFILE
'
'  Now determine storm shear profile for plotting of shear demands on bays
'
  i = 0
  elevation(0) = htotal + bayh(0)
  Do
    elevation(i + 1) = elevation(i) - bayh(i)
    i = i + 1
  Loop While i <= nbay
  For i = 1 To 2
  elevbar = elevation(0)
  interval = elevation(0) / 100
    dummy = 0
    For j = 1 To ndeck
      dummy = dummy + fhydro(i, j) + faero(i, j)
    Next j
    cumf(i, 0) = dummy * lf
    For l = 0 To nbay
      dequ(i, l) = dbdequebar(i, l) + dbdequbbar(i, l) + dequapp(l) + nleg * (jld(i, l) + 2 * mg(l)) / 12
    Next l
    k = 0
    For j = 1 To 100
      If (crest - elevbar < vcrest ^ 2 / g) And (designcd = False) Then
        cdmem = cdj * (crest - elevbar) / (vcrest ^ 2 / g)
      Else
        cdmem = cdj
      End If
      If elevbar > elevation(k + 1) Then
        f(i, j) = wjbias * cdmem * dequ(i, k) * interval * vel(j) ^ 2 / 1000 * lf
      Else
        If k = 0 Then
```

```
            f(i, j) = wjbias * cdmem * (dequ(i, k + 1) * interval + hbequa(i, k + 1) + boati(i)) * vel(j) ^ 2 / 1000 * lf
         Else
            f(i, j) = wjbias * cdmem * (dequ(i, k + 1) * interval + hbequa(i, k + 1)) * vel(j) ^ 2 / 1000 * lf
         End If
         k = k + 1
      End If
      elev(j) = elevbar
      elevbar = elevbar - interval
      If elevbar < 0 Then Exit For
      cumf(i, j) = cumf(i, j - 1) + f(i, j)
   Next j
   cumf(i, 100) = cumf(i, 99)
Next i
'
' Determine jacket leg forces from overturning moments, so that effective increase in
' jacket bay shear capacity from batter component can be evaluated.
'
For i = 1 To 2
   For j = 1 To nbay + 1
      dummy = 0
      For k = 1 To ndeck
         dummy = dummy + fhydro(i, k) * (fhydroh(k) + wdep - elevation(j)) + faero(i, k) * (faeroh(k) + wdep - elevation(j))
      Next k
      mbar(i, j) = dummy * lf
   Next j
   For j = 1 To 100
      For k = 1 To nbay + 1
         h(j, k) = elev(j) - elevation(k)
         If h(j, k) > 0 Then
            m(i, j, k) = f(i, j) * h(j, k)
         Else
            m(i, j, k) = 0
         End If
         mbar(i, k) = mbar(i, k) + m(i, j, k)
         If k > nbay Then lht(i, k) = bcw(i)
         legf(i, k) = mbar(i, k) / lht(i, k)
         legfh(i, k) = legf(i, k) * Sin(alpha(i))
      Next k
   Next j
Next i
'
' Find capacity reduction in top jacket bay due to deck bay action.
'
For i = 1 To 2
   If structuretype = "monopod_braced" Or structuretype = "monopod_guyed" Then
      dlmbar(i, 1) = Abs(Application.Min((faerobar(i) + fhydrobar(i)) * bayh(0) * 1.5, dlm))
   Else
      dlmbar(i, 0) = Abs(Application.Min((faerobar(i) + fhydrobar(i)) / nleg * bayh(0) * ((bayh(0) / (2 * e * dli / 144) + 1 / dlcm(i)) /
            (bayh(0) / (e * dli / 144) + 1 / dlcm(i))), dlm))
      dlmbar(i, 1) = Abs(Application.Min(dlmbar(i, 0) - (faerobar(i) + fhydrobar(i)) / nleg * bayh(0), dlm))
   End If
   If bayh(0) <= 1.5 Then
      shear(i) = 0
   Else
      If structuretype = "monopod_braced" Or structuretype = "monopod_guyed" Then
         shear(i) = dlmbar(i, 1) / (bayh(1) + 10 * piled / 12)
      Else
         shear(i) = dlmbar(i, 1) / bayh(1) * nleg
      End If
   End If
Next i
'
' Find mean hydrodynamic load for use in reliability calculation
'
For i = 1 To 2
elevbar = elevation(0)
interval = elevation(0) / 100
k = 0
   For j = 1 To 100
      If elevbar < elevation(k + 1) Then
```

```
            k = k + 1
        End If
        meanload(i, k) = cumf(i, j)
        elevbar = elevbar - interval
    Next j
    meanload(i, nbay + 1) = cumf(i, j - 1)
Next i
'
'   Find pile loads
'
For i = 1 To 2
'
'   Find forces due to global overturning
'
    If structuretype = "multijacket" Then
        zforce = 2 * mbar(i, nbay + 1) / msw
    Else
        zforce = 0
    End If
    If structuretype = "tripod" Then
        pileloadcomp(i) = Application.Max((zforce + qdeck) / (nleg + nsk) + (1 / Sin(offangle)) ^ (i - 1) * legf(i, nbay + 1) / (nskirt(i)
            + n(i)), 0)
        pileloadtens(i) = Application.Max((zforce - qdeck) / (nleg + nsk) + (1 / Sin(offangle)) ^ (i - 1) * legf(i, nbay + 1) / (nskirt(i) +
            n(i)), 0)
    ElseIf structuretype = "monopod_braced" Or structuretype = "monopod_guyed" Then
        If structuretype = "monopod_guyed" Then
            qdeck = qdeck + 3 * pretension * Sin(theta(1, 1, 1)) ^ 2
        End If
        pileloadcomp(i) = Application.Max((zforce + qdeck) + (mbar(i, 2) + dlmbar(i, 1) / 3) / bcw(i), 0)
        pileloadtens(i) = Application.Max((zforce - qdeck) + (mbar(i, 2) + dlmbar(i, 1) / 3) / bcw(i), 0)
    Else
        pileloadcomp(i) = Application.Max((zforce + qdeck) / (nleg + nsk) + legf(i, nbay + 1) / (nskirt(i) / 2 + n(i)), 0)
        pileloadtens(i) = Application.Max((zforce - qdeck) / (nleg + nsk) + legf(i, nbay + 1) / (nskirt(i) / 2 + n(i)), 0)
    End If
Next i
'
'   Find cov for use in reliability calculation
'
For i = 1 To 2
    dummy = 0
    For j = 1 To ndeck
        dummy = dummy + fhydro(i, j)
    Next j
    For j = 0 To nbay
        If dummy = 0 Then
            covload(i, j) = wjcov
        Else
            covload(i, j) = (wdcov ^ 2 + wjcov ^ 2) ^ 0.5
        End If
    Next j
Next i

End Sub '   End HYDRO PROFILE
```