223 C

TOPCAT UPDATING AND ENHANCEMENTS

Cnoidal Waves, Load Spatial Effects and Reliability Sensitivity

Screening Methodologies Project Phase IV

TOPCAT

Template Offshore Platform Capacity Assessment Tools

by

Zhaohui Jin

and

Professor R. G. Bea

Report to

Joint Industry Project Sponsors

Marine Technology and Management Group

Department of Civil and Environmental Engineering

University of California at Berkeley

June, 1998

REPORT TO JOINT INDUSTRY
TOPCAT PROJECT SPONSORS

TOPCAT UPDATING AND ENHANCEMENTS

Cnoidal Waves, Load Spatial Effects
and Reliability Sensitivity

by

Zhaohui Jin

James D. Stear

and Professor Robert G. Bea

Marine Technology & Management Group

Department of Civil & Environmental Engineering

University of California at Berkeley

Table of Contents

1.0 Introduction

This report documents the updating and enhancements of the TOPCAT program made in the Spring, 1998. Based on the research schedule determined in the last meeting with sponsors, three major tasks have been finished this semester.

The three tasks are:

- coding an analysis module handling shallow water wave kinematics into TOPCAT, by application of cnoidal wave theory;
- introducing the effects of spatially distributed load on large-scale structures, by consideration of the variation of horizontal velocity and wave surface elevation with respect to distance and time, etc., phase angles;
- building a model doing the comprehensive sensitivity study of reliability analysis, by application of first-order-second-moment reliability approximation.

The following chapters detail the formulation, coding, and discussion of these tasks. Chapter 2 talks about the theoretical basis, realization and validity of cnoidal wave module in TOPCAT. It also gives some calculation examples and extreme cases as calibration. Chapter 3 deals with a simplified model reflecting the effects of spatially distributed wave loading on large-scale offshore structures. The basic assumptions and approaches are discussed. A calculation example is also given. As a part of this effort, a new platform configuration - multi multi-leg platform is also studied. Chapter 4 documents the analysis model created to analyze the dependence of structure safety index on the user-defined input parameters such as biases and COV's. A case demonstration is also attached. Chapter 5 is a brief description about the basic approach being adopted to solve the problem of diagonal loading on offshore structures. Chapter 6 is conclusion and future project development plan. Besides these documents, the source codes of the program modules created and extensively modified during this phase are attached as an appendix.

2.0 Shallow Water Wave Kinematics: Application of Cnoidal Wave Theory in TOPCAT

The previous version TOPCAT has no calculation module dealing with shallow water kinematics, which is considerably different from that predicted by Stokes V deep water wave theory. For some platforms in the shallow water region, the calculated loading is not quite correct. And for some cases, for example, a water depth less than 30 feet, the current Stokes V module in TOPCAT just simply can not converge during iteration. It produces results which obviously are not reasonable such as very large free surface elevations.

This chapter summarizes the recent development of an new wave kinematics analysis module which has been added to TOPCAT. This module calculates shallow water wave kinematics based on the second order approximation of nonlinear cnoidal wave theory.

2.1 General

The present TOPCAT version uses Stokes V theory to calculate the wave kinematics. This is good for deep water, for some transitory range between deep and shallow water and even for some other water depths depending on the wave characteristics. However, this kind of low order Stokes finite amplitude wave theory becomes inadequate in very shallow water. This is because many coefficients of the higher order terms "blow up": they become excessive relative to the lowest order terms, so that they can not be neglected in the governing motion equations. The basic assumptions for Stokes wave theory are no longer satisfied. For high waves with longer wave lengths, nonlinear wave theories should be used in the shallow water range.

Nonlinear periodic wave theories suitable for shallow water have been developed since the last century. Among them, a fundamental one is cnoidal wave theory.

2.2 Cnoidal Wave Theory

The cnoidal wave theory was first developed by Korteweg and de Vries(1895). The wave profile is developed in terms of Jacobian elliptic function "cnu", and this is source of the name "cnoidal" came from. Like Stokes theory, the governing fluid motion equation in cnoidal theory can be solved to different order approximations. The solution of this theory has two special limit cases. The cnoidal wave theory spans the range from sinusoidal or Airy theory in deep water to solitary wave theory in shallow water.

The definition of the wave characteristics are shown in Figure 2.1.

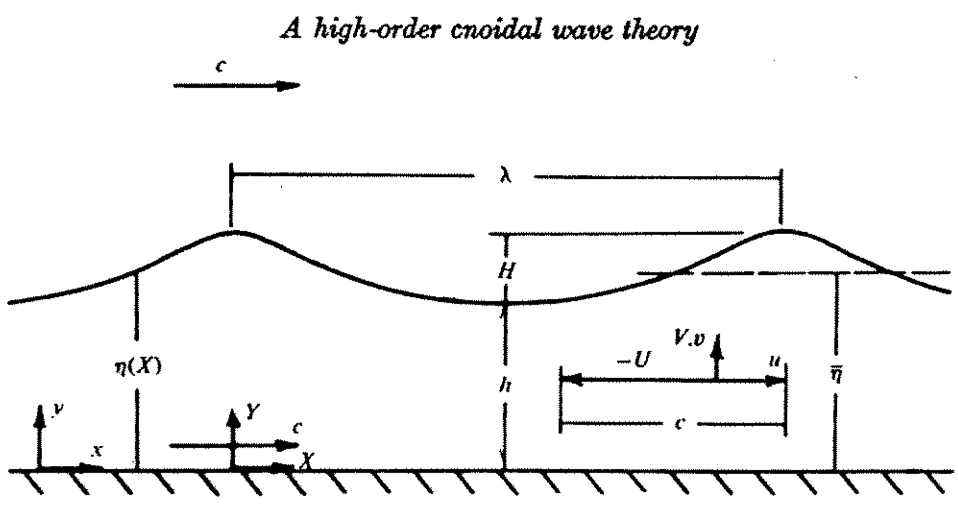

Figure 2.1 Wave characteristics definition for cnoidal waves

In Figure 2.1, η is the free surface elevation, λ is the wave length(or L). $\overline{\eta}$ is the still water level(or water depth d), H is the wave height, and h is the trough water depth. c is the celerity.

The cnoidal theory has been developed to higher order approximations. Besides Korteweg and de Vries's model, other presentations of the first order approximation were made by Kulegan and Patterson(1940), Keller(1948) and Laitone(1961). Laitone(1961) and Chappelear(1962) have developed respectively second and third order approximation to cnoidal wave theory. Fenton(1979) developed a cnoidal wave theory which is capable of extension to any desired order approximation. However, as to engineering application, Fenton commented: there is absolutely no need to extend the solution higher than 5th order approximation, and second order approximation will give results good enough (Fenton, 1979).

2.3 Formulation of Cnoidal Wave Theory in TOPCAT

Based on the theoretical results of the cnoidal wave theory, the second order approximation is used to calculate the shallow water wave kinematics in TOPCAT. There is an assumption here, the celerity, horizontal, particle velocity and wave period are affected by this assumption. The assumption depends on what kind of celerity definition we use. Stokes first definition (zero mean horizontal velocity) and Stokes second definition (zero mean horizontal momentum) have different calculation formulae. In TOPCAT, the Stokes first definition (zero mean horizontal velocity) of celerity was chosen because it is usually applied in engineering practice.

The Jacobian elliptic function modulus κ forms the fundamental parameter in terms of which the solution is expressed. κ has a value that ranges from zero to unity. And the shallow water waves usually have a κ approaching unity. One thing should be noted: "approaching" means very close to 1, saying .999. Near unity, κ is a very sensitive parameter for cnoidal waves. A small variation in κ, for example 0.999 to 0.9999 will results in a completely different wave profiles. We can see this in the cnoidal wave profiles demonstrated in Figure 2.3

Functions of κ used here are: $K(\kappa)$ is the complete elliptic integral of the first kind, $E(\kappa)$ is the complete elliptic integral of the second kind.

$$E(\kappa) = \int_0^{\pi/2} \sqrt{1-\kappa^2 \sin^2 x}\,dx$$

$$K(\kappa) = \int_0^{\pi/2} \frac{1}{\sqrt{1-\kappa^2 \sin^2 x}}\,dx$$

The variation with distance x along the wave propagating direction and the time t of the different variables of interest are realized through the Jacobian elliptic function argument q. $q = K(kx-\omega t)/\pi = K\theta/\pi$. γ is the ratio of $E(\kappa)/K(\kappa)$, and $\kappa'^2 = 1 - \kappa^2$. Also, the Ursell number $U=HL^2/d^3$ and its first approximation $U1=16k^2K^2/3$ reflects the relative magnitudes of the parameters H/L and d/L, which expresses the relative water depth and has an important influence on wave shape and kinematics in shallow water,

Once the environmental characteristics, such as wave period T , wave height H and water depth d are determined, the Jacobian elliptic function modulus κ is also determined if the wave is in shallow water and the cnoidal theory is applicable. An iteration procedure can solve the value of κ. The formulae used in the cnoidal wave calculation module are as follows:

Trough Depth, h $\quad \frac{h}{d} = 1 - \varepsilon h_1 - \varepsilon^2 h_2$

Surface elevation, η $$\frac{\eta}{d} = \varepsilon(cn^2q - h_1) - \varepsilon^2\left[\frac{3}{4}cn^2q(1 - cn^2q) + h_2\right]$$

Wave celerity, c $$\frac{c}{\sqrt{gd}} = 1 + \varepsilon c_1 + \varepsilon^2 c_2 + 0\left[\varepsilon^3\right]$$

Wave length, L $$\frac{L}{d} = \frac{4\kappa K}{\sqrt{3\varepsilon}}\left\{1 - \varepsilon l_1 + 0\left[\varepsilon^3\right]\right\}$$

Wave Period, T $$\frac{d}{gT^2} = \frac{3\varepsilon}{16k^2K^2}\left\{\left(\frac{1 + \varepsilon c_1 + \varepsilon^2 c_2}{1 - \varepsilon l_1}\right)^2 + \left[\varepsilon^3\right]\right\}$$

Horizontal particle velocity, u

$$\frac{u}{\sqrt{gd}} = \varepsilon\left(cn^2q - h_1\right) + \varepsilon^2\left\{\left(f_1 + f_2cn^2q - cn^4q\right) - \frac{3}{4\kappa^2}\left(\frac{s}{d}\right)^2\left[\kappa'^2 + 2\left(2\kappa^2 - 1\right)cn^2q - 3\kappa^2cn^4q\right]\right\} + 0\left[\varepsilon^3\right]$$

Vertical particle velocity, w

$$\frac{w}{\sqrt{gd}} = \frac{\varepsilon\sqrt{3\varepsilon}}{\kappa}\left(\frac{s}{d}\right)cnqdnqsnq\left\{1 + \varepsilon\left[f^3 - 2cn^2q - \left(\frac{s}{d}\right)^2\left(\frac{2\kappa^2 - 1}{2\kappa^2} - \frac{3}{2}cn^2q\right)\right] + 0\left[\varepsilon^2\right]\right\}$$

Horizontal particle acceleration, $\partial u/\partial t$

$$\frac{1}{g}\frac{\partial u}{\partial t} = \frac{\varepsilon\sqrt{3\varepsilon}}{\kappa}cnqdnqsnq\left\{1 + \varepsilon\left[f_4 - 2cn^2q - \left(\frac{s}{d}\right)^2\left(\frac{3\left(2\kappa^2 - 1\right)}{2\kappa^2} - \frac{9}{2}cn^2q\right)\right] + 0\left[\varepsilon^3\right]\right\}$$

Vertical particle acceleration, $\partial w/\partial t$

$$\frac{1}{g}\frac{\partial w}{\partial t} = \frac{3\varepsilon^2}{2\kappa^2}\left(\frac{s}{d}\right)\left(\kappa'^2 + 2\left(2\kappa^2 - 1\right)cn^2q - 3\kappa^2 cn^4 q\right) + 0\left[\varepsilon^3\right]$$

where

$$\varepsilon = \frac{H}{d}$$

$$q = \frac{K\theta}{\pi} = \frac{K}{\pi}\left(kx - \omega t\right)$$

$$h_1 = \left\{\gamma - \kappa'\right\} / \kappa^2$$

$$h_2 = \left\{\gamma\left(\kappa^2 - 2\right) + 2\kappa'^2\right\} / 4\kappa^4$$

$$c_1 = (2 - k^2 - 3\gamma) / 2\kappa^2$$

$$c_2 = \left\{-5\gamma(15\gamma + 19\gamma^2 - 38) - 18\kappa^4 - 88\kappa'^2\right\} / 120\kappa^4$$

$$l_1 = \left\{12\gamma + 5\kappa^2 - 10\right\} / 8 / \kappa^2$$

$$f_1 = \left\{-\gamma(6\gamma + 11\kappa^2 - 16) + \kappa'^2(9k^2 - 10)\right\} / 12\kappa^4$$

$$f_2 = \left\{2\gamma + 7\kappa^2 - 6\right\} / 4\kappa^4$$

The Jacobian elliptic function cnq and cnqdnqsnq can be expressed by a Fourier series in θ:

$$cn^2 q = \sum_{n=0}^{\infty} A_n \cos(n\theta)$$

$$cnqdnqsnq = \frac{\pi}{K}\sum_{n=1}^{\infty} nA_n \sin(n\theta)$$

where the Fourier coefficients A_n can be determined by:

$$A_n = \begin{cases} \dfrac{2\pi^2}{\kappa^2 K^2}\left(\dfrac{nr^n}{1 - r^{2n}}\right) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\, n \geq 1 \\ \dfrac{\gamma - \kappa'^2}{\kappa^2} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\, n = 0 \end{cases}$$

where $r = \exp[-\pi K(\kappa')/K(\kappa)]$

Figure 2.2 shows the behaviors of complete elliptic integrals and the Ursell number with respect to the Jacobian elliptic function modulus κ. Figure 2.3 demonstrates the free surface profiles of the cnoidal waves for different κ values.

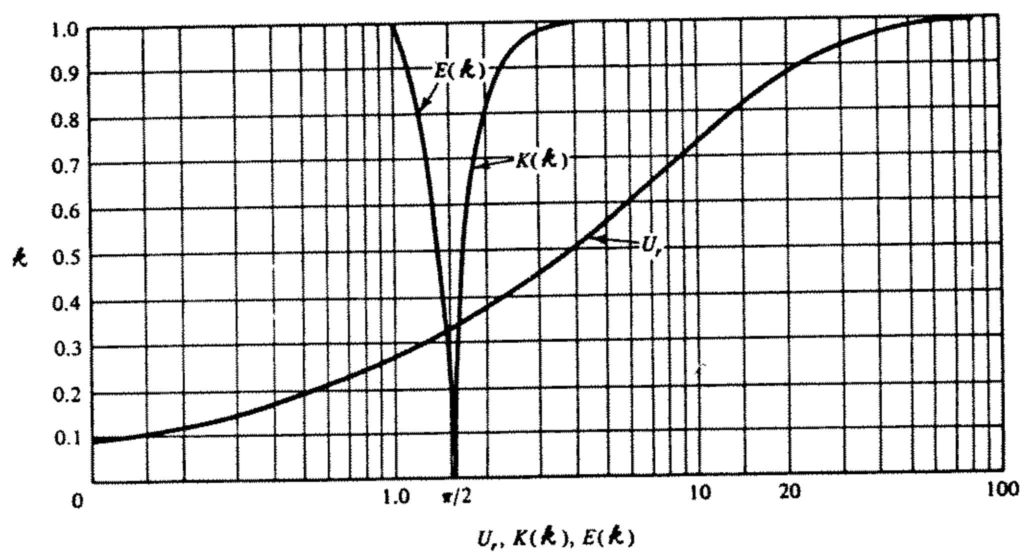

Figure 2.2 E(κ), K(κ) and U as functions of κ

2.4 Validity of the Cnoidal Wave Theory

It is important to know which of the various water wave theories to apply to particular problem, where the wave characteristics and water depth are specified. In order to address these problems, the validity of the various theories must be known. This validity is composed of two parts: the mathematical validity and the physical validity.

The mathematical validity is the validity of any given wave theory to satisfy the mathematically posed boundary value problem. For example, all the theories in use satisfy the bottom boundary condition exactly, however, the cnoidal and solitary wave theories only approximately satisfy the Laplace equation within the fluid. All of the theories only satisfy the dynamic free surface boundary approximately.

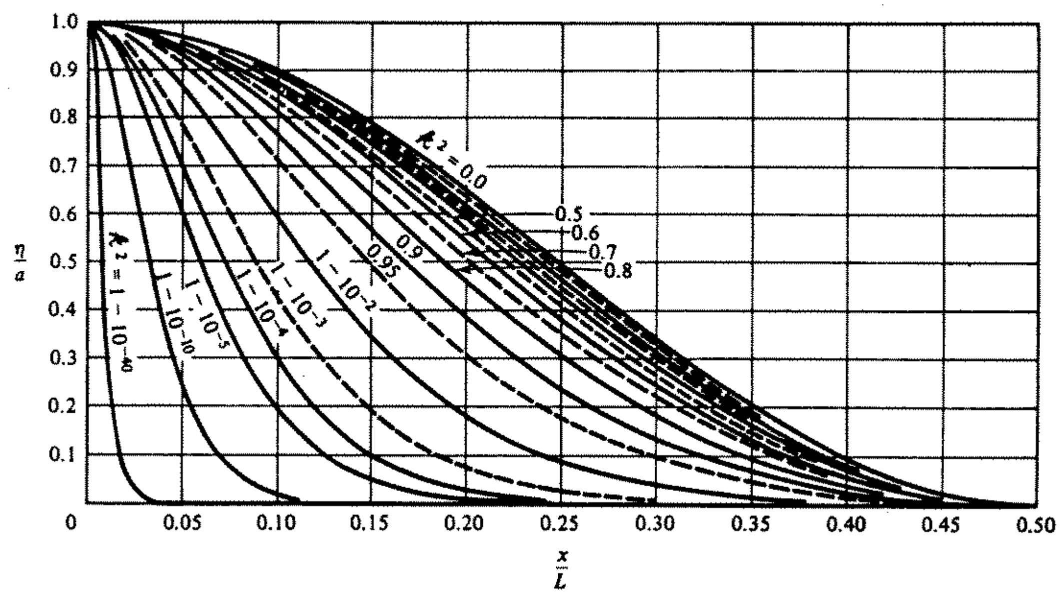

Figure 2.3 Free surface profile of the cnoidal waves

The second aspect, physical validity refers to how well the prediction of the various theories agrees with actual measurements. This part of the validity has been difficult to obtain due to the problem of wave tank design and measurement requirements in field.

Dean(1970) examined the analytical validity of many wave theories. Figure 2.4 shows the results of the comparison of the theories. We can see that cnoidal wave theory best fits the dynamic free surface boundary conditions in most range in shallow water.

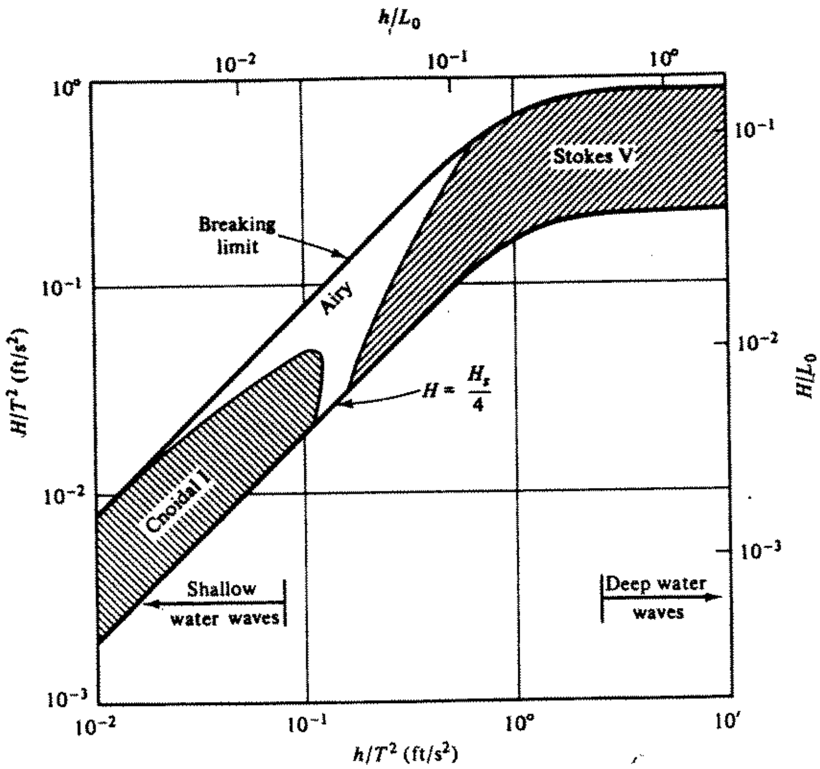

Figure 2.4 Comparison of wave theories to best fit free surface boundary condition

Mention shall also be made here of the Ursell number. It is very useful in assessing the relative importance of nonlinear shallow wave theories. Ursell number will take large values for high waves in shallow water. The application of the complete Kortewege-de Vries equation is appropriate when the Ursell number is of order unity or is moderately large. In general, following conclusions are valid.

Ursell number : $UR = \frac{HL^2}{h^3}$, where H is wave height; L is wave length; h is water depth.

If UR<<1 (at most 26), stokes theory applicable. (intermediate and deep water)

If UR<<1, no permanent wave form.

If UR = order of 1, or moderately large, cnoidal wave theory applicable (shallow water).

The arguments above form the basis for a check-up mechanism in TOPCAT to automatically chose which theory, Stokes V or cnoidal, shall be used for a specific problem. Meanwhile, for the sake of comparison, there are options left for users to judge which one is better. Users can force the execution of Stokes wave theory by choose the options in input menu.

2.5 Calculation Examples

A Visual Basic module was developed to perform the calculation of shallow water wave kinematics using cnoidal wave theory. The algorithm applied the formulae described before. The main subroutine is concise.

Several examples were calculated to check the accuracy of the module. With an input of wave height H = 8ft, wave period T = 15sec, water depth d = 25ft. The cnoidal module gave the following results: Jacobian elliptic function modulus $\kappa = 0.99743$; horizontal velocity at sea bottom u(0) = 5.22ft/sec; trough water depth h = 23.18ft; horizontal velocity at wave crest u(H+h) = 8.62ft/sec. The above results duplicate the results Sarpkaya presented(Sarpkaya, 1980). The accuracy requirement is satisfied.

Other examples are parametric study of the two limit cases of cnoidal wave theory. The value of κ changes between 0 and 1. When κ approaches 0, cnoidal wave theory will converge to Airy wave theory. The wave height and wave period is small relative to water depth. As κ approaches 1, cnoidal wave theory converges to solitary wave theory. The wave period becomes infinite, and the wave is skewed thoroughly above the still water level. It shall be emphasized that the elliptic integral of the first kind $K(\kappa)$ approaches infinity as κ approaches 1. This asymptotic procedure is not very fast, for example, K(0.999999) = 7.9228, K(0.9999999999) = 44.9815. We can see an interesting phenomenon that for a κ value very very close to unity, the K value is far from infinity. This means that care should be taken with respect to the convergence speed of the K function. This has been addressed numerically by the cnoidal module. However, the present algorithm is not very efficient, the calculation speed is not fast. Effort is being given to speed up the execution speed. Both limit cases have been realized by the cnoidal wave calculation module. Given very small wave height and relatively small wave period, the module will yields a κ very close to 0. And the wave profile is identical with that of Airy theory. Given that a very large wave period is input, the module will get a κ very close to 1. The wave profile is the same as that of solitary wave theory. This is another proof of the validity of the cnoidal module. For example, with an input of H = 5ft, T = 300000sec, d = 25ft, the cnoidal module gives a value of κ, κ = 0.999999999, the horizontal velocities at sea bottom, at the middle of depth and wave crest are respectively: u(0) =4.721ft/sec, u(H/2+h/2)=5.026ft/sec and u(H+h)=5.945ft/sec; while the exact solitary formulae give a κ=1 and the horizontal velocities: u(0)=4.82ft/sec, u(H/2+h/2)=5.13ft/sec and u(H+h)=6.05ft/sec. The results are quite comparable.

3.0 Effects of Spatially Distributed Wave Loading

This document addresses a simplified analysis model that can reflects the spatially distributed wave and current loads on offshore structures examined by TOPCAT.

3.1 General

The previous version TOPCAT has a module calculating the environment loads on offshore structures, including wave and current loads. The wave kinematics is evaluated using Stokes V theory. All the structure elements are modeled as equivalent vertical cylinders that are assumed to be put at the wave crest. For the inclined members, the effective vertical projected area is determined by multiplying the product of member length and diameter by the cube of the cosine of its angle with the horizontal. It is considered to be conservative. Considering the phase difference between the maximum drag and inertia force components and the relatively small dimensions of a typical jacket-type platform with respect to wave lengths and heights in an extreme condition, at the time the drag forces acting on the platform reach a maximum value the inertia forces are relatively small; hence, only the drag force component of Morison equation is estimated. Thus the maximum force acting on the portions of structure below the wave crest is evaluated by:

$$F = \frac{1}{2} C_d \rho A u |u|$$

Where, u is the total water particle velocity profile under the wave crest, which combines the wave and current effects, the current blockage and directional spreading are also recognized in u.

The current simplified model is demonstrated in Figure 3.1.

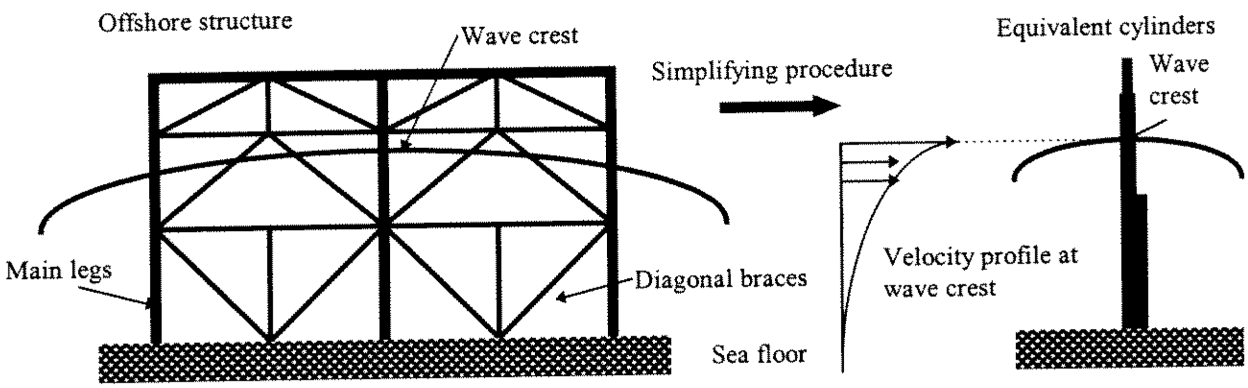

Figure 3.1 Simplified load model in TOPCAT current version: transforming offshore structures to equivalent vertical cylinders at wave crest

However, some offshore structures have relatively large dimensions. The algorithm described above may cause excessively conservative results. The reason is obvious. For a large offshore structure, only part of it is at the wave crest, while the rest parts of the structure are located at the positions where the surface elevation is smaller than that of wave crest. As a result, the projected area is smaller and the water particle velocities may be much smaller than those at the wave crest.

To reflect this kind of wave and current load distribution with respect to the spatial variation along the wave hitting the structure, a simplified model has been developed to modify the wave and current kinematics calculation in TOPCAT. The spatial effects are counted in by calculating the change of elevation and phases in wave motion equations.

3.2 Basic approach of the new simplified model

The basic approach in this module is illustrated in Figure 3.2. Instead of "collapsing" all the structure elements to the effective cylinders at the wave crest, the new model takes the structure as several groups of cylinders along the wave propagating direction. The number of the groups is determined by the structure geometric characteristics. This is an input left to the users for multi multi-leg jacket structures. For standard platforms, this is determined automatically. It is assumed that the maximum force occurs when the center line of the structure is at the wave crest. The typical storm wave length in deep water is > 1000ft, and the typical storm wave length in shallow water is >300ft. Considering these facts, an offshore structure is unlikely to span over more than one wave length. So the new model puts the structure on one design wave, the structure center line is at the wave crest, as shown in Figure 3.2. The locations of groups of effective cylinders are determined automatically according to structure geometry.

For example, given the structure in Figure 3.2, there are three groups of cylinders. The locations coincide with the locations of the main structure legs. The wave crest is at the center legs. The inclined braced between main legs are transformed to effective vertical cylinders to the nearest group, using the existing area-projecting mechanism in TOPCAT. Then we have three groups of cylinders spanning over one wave, the real wave and current loads have a spatial distribution on these cylinders.

This spatial distribution of loads is recognized by the variation in phase angle. The phase angle for a water wave is given by $\theta = kx - \omega t$. To simplify the problem, we can assume that wave crest occurs at $x=0$ and $t=0$. So the distance from the structure center line, x, reflects the wave profile variation along a wave length. It controls the wave and current kinematics and the water surface elevation as well.

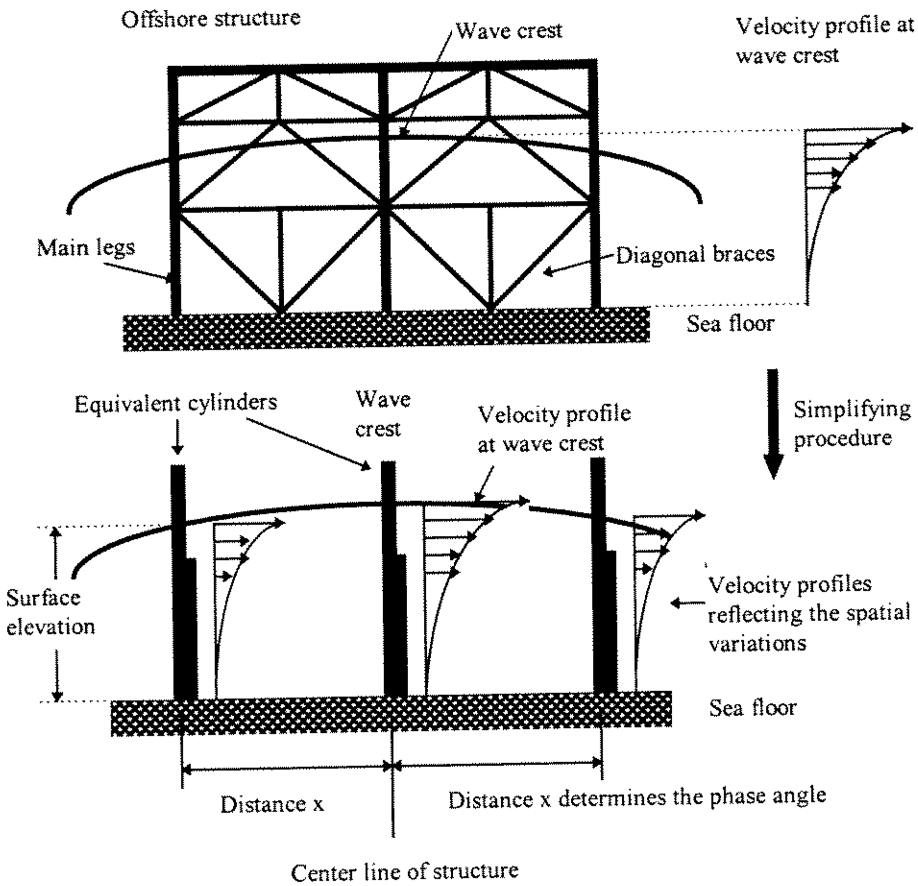

Figure 3.2 The modified simplifying procedure: reflecting the spatial distribution of loads by considering variation of phase angle and water surface elevation

TOPCAT has two modules calculating deep water and shallow water kinematics, using Stokes V and Cnoidal wave theories respectively. Modifications are made to these modules to the calculation of the variation of surface elevation and horizontal velocities with respect to variation in phases.

For Stokes V theory:

Surface elevation, η: $k\eta = \sum_{n=1}^{5} \eta_n' \cos(n\theta)$

Horizontal particle velocity, u: $\frac{u}{c} = \sum_{n=1}^{5} n\phi'_n \cosh(nks)\sin(n\theta)$

For Cnoidal wave theory(second order Approximation):

Surface elevation, η: $\frac{\eta}{d} = \varepsilon(cn^2q - h_1) - \varepsilon^2\left[\frac{3}{4}cn^2q(1 - cn^2q) + h_2\right]$

Horizontal particle velocity, u:

$$\frac{u}{\sqrt{gd}} = \varepsilon\left(cn^2q - h_1\right) + \varepsilon^2\left\{\left(f_1 + f_2cn^2q - cn^4q\right) - \frac{3}{4\kappa^2}\left(\frac{s}{d}\right)^2\left[\kappa'^2 + 2\left(2\kappa^2 - 1\right)cn^2q - 3\kappa^2cn^4q\right]\right\} + 0\left[\varepsilon^3\right]$$

Where, $q = \frac{K\theta}{\pi} = \frac{K}{\pi}\left(kx - \omega t\right)$.

Once the distance from center line is known, the phases(θ or q) can be determined. Substitutes the phase value into the formulae above, the surface elevation and the particle velocity profile can be figured out. Then, applying Morison equation to calculate the total force:

$$F_{total} = \sum_{n=1}^{N} \frac{1}{2} C_d \rho A'_n u'_n \left|u'_n\right|$$

Where, N is the number of groups of effective cylinders, A' is the modified projected area, u' is the modified horizontal velocity, considering the spatial variation of wave profile along wave length.

Or, we can also express the formulae is another way:

$$F_{spatial-distributed} = F_{crest} \bullet C_\theta \bullet C_\eta$$

Where, F_{crest} is the force obtained from the previous formulation. C_θ and C_η are the modification factor due to phase correction and surface elevation correction. C_θ is a function of change of phase angle at the location of the main platform legs. It reflects the change in the water particle velocities. It is evaluated numerically by a subroutine added to the program. C_η is a function of the change of free surface elevation and the equivalent projected cylinder areas. It is also evaluated numerically by an algorithm coded in the load spatial effect module.

3.3 Case Study

A case study of the load spatial effect module was conducted. The example platform is SP62. This is a 8-leg regular platform. The main inputs are as followings:

Environmental:	water depth	d = 340 ft
	wave height	H = 80 ft
	wave period	T = 13.5 s
	surge height	s =3 ft
Structural:	8-leg; 6 jacket bays; 2 deck bays	
BS:	bottom width	bw1 = 122 ft
	top width	tw1 = 51 ft
EO:	bottom width	bw2 = 202 ft
	top width	tw2 = 131 ft
	middle width	mw = 45 ft

The analysis results produced by the load spatial effect module is summarized in Table 3.1.

It can be seen that the mean load on platform SP62 is notable. The load on each jacket bay decreases by a percentage of 10-20%. For the EO case, the storm wave length is determined by Stokes V module, which is around 977 feet. Then the phase angle difference in EO direction is around 60 degree. At the center legs, the phase angles are 30 degree ahead or behind the wave crest. The surface elevations above SWL(still water depth) at center legs(wave crest) and side legs are 45.7 feet and 36.2 feet respectively, while the horizontal velocities are 21.1 ft/s and 18.0 ft/s. From these results, we can conclude that the change in velocity is the major contribution to the load spatial effect. The variation in surface elevation is not a big deal in changing the loading pattern.

Table 3.1 Load Spatial Effect for SP62 platform

Loading spatial effect: comparison with results without considering spatial effect						
Jacket Bay	Broad Side Loading(kips)			End On Loading(kips)		
Bay #	Without Effect	With Effect	Ratio	Without Effect	With Effect	Ratio
1	3344	2772	1.2061	3247	2757	1.1778
2	4529	3870	1.1702	4493	3904	1.1509
3	5421	4685	1.1571	5506	4839	1.1379
4	6021	5250	1.147	6203	5488	1.1303
5	6473	5671	1.1415	6752	6003	1.1248
6	6922	6093	1.1361	7271	6492	1.1201

Considering the fact that SP62 is not a very large structure, this decrease is not very small. For larger structures, such as multi multi-leg jacket platforms, more obvious load spatial effects is expected. However, one thing should be mentioned. The above arguments are based on the assumption that the loading on the platform is drag dominated and the inertial component is negligible. For some special large structures, such as multi multi-leg jacket platforms, which may span over an entire wave length, this assumption may not be true. The inertial component may not be neglected. This special case is still being checked.

4.0 Sensitivity Analysis of Reliability

The structure reliability analysis is the "heart" of TOPCAT. The objectives of this analysis are to identify the potential failure modes and weak-links of the structure and to estimate bounds on the probability of system failure by taking into account the biases and uncertainties associated with loadings and capacities. This is demonstrated by the values of safety indices for the structure components of the platforms: deck bay, jacket bay and pile foundation. The statistical properties of safety indices are derived considering the uncertainties associated with environmental conditions, structure conditions, kinematics, structure component capacity estimation and force calculation procedures. The basic approach of the reliability analysis has been documented in the ULSLEA reports(Mortazavi, 1996 and Stear, 1996, 1997)

4.1 General

Besides the evaluation of safety indices in a reliability analysis, it is often of interest to know the sensitivity of the failure probability or the safety indices to the variations of statistical parameters of the random variables involved in the problem. The parameters, denoted by **p,** may include parameters in the distributions of the basic variables and deterministic parameters in the limit state functions. As for our case, the reliability sensitivity study in TOPCAT only deal with the dependence of safety index β on the user-defined statistical parameters **p**. The parameters in the limit state functions are taken as fixed and are not considered. In the current TOPCAT version, **p** is a vector consisting of biases and coefficients of variation(COV) in loading and capacity estimation. The major parameters are:

Bias: wjbias - bias of wave force on jacket

wdbias - bias of wave force on deck

eqbias - bias of earthquake loading

bcbias - bias of estimation of ultimate capacity of main diagonal braces

mcrbias - bias of critical moment of deck portal

pcrlbias - bias of local buckling axial loading of deck portal

jtbias - bias of estimation of joint capacity(tension)

jcbias - bias of estimation joint capacity(compression)

qabias - bias of axial capacity of pile foundation

pubias - bias of lateral capacity of pile foundation

COV: wjcov - COV wave force on jacket

wdcov - COV of wave force on deck

eqcov - COV of earthquake loading

bccov - COV of estimation of ultimate capacity of main diagonal braces

mcrcov - COV of critical moment of deck portal

pcrlcov - COV of local buckling axial loading of deck portal

jtcov - COV of estimation of joint capacity(tension)

jtcov - COV of estimation of joint capacity(compression)

qacov - COV of axial capacity of pile foundation

pucov - COV of lateral capacity of pile foundation

Above, biases take care of the calculation method uncertainties(TYPE II) while COV takes care of the natural uncertainties (TYPE I).

To do a reliability and sensitivity analysis, the basic random variables **Z** are transformed into independent and standardized normal variables **U**:

$$\mathbf{Z}(\mathbf{p}) = \mathbf{T}^{-1}(\mathbf{U},\mathbf{p})$$

T is the transformation from z-space to u-space. By this way, the possible dependence of the distribution of **Z** on the elements in **p** is explicitly expressed. The failure set in the u-space is F(**p**):

$$F(\mathbf{p}) = \{\mathbf{u} | g(\mathbf{u}, \mathbf{p}) \le 0\}$$

This is the standard reliability analysis approach that TOPCAT follows. Now we are interested in the dependence of β on the parameters **p**. The reliability index, expressed as a function of **p** is $\beta_R(\mathbf{p})$:

$$\beta_R(\mathbf{p}) = -\Phi^{-1}[P(F(\mathbf{p})]$$

The reliability index is computed at $\mathbf{p}=\mathbf{p}_0$, $\mathbf{p}_0$ is the parameter set used in the reliability analysis. The sensitivity of reliability index with respect to changes in parameter values is expressed by the partial derivatives: $\partial\beta_R(\mathbf{p}_0)/\partial p_i$. This is called the sensitivity vector. These partial derivatives can be computed by numerical differentiation, as we will do in some of the sensitivity vector calculations in TOPCAT. But there are more efficient methods which do not require a repeated computation of the reliability index. Following is one of them due to (Hohenbicher,1984). It includes a useful set of asymptotic results for the partial derivatives. The asymptotic results have been found to provide very good approximations.

First, Breitung (1984) showed the asymptotic results, which in terms of the safety index and β_R are,

$$\beta_R(\mathbf{p}_0) \sim \beta(\mathbf{p}_0), \qquad \text{as } \beta(\mathbf{p}_0) \to \infty$$

$\beta(\mathbf{p}_0)$ is the first-order reliability index. The result are asymptotic in the sense that the product of β and the main curvatures of the failure surface at the working points is fixed as β approaches infinity. In the same asymptotic sense Hohenbichler (1984) has shown:

$$\frac{\partial}{\partial \mathbf{p}_i}\beta_R(\mathbf{p}_0) \sim \frac{\partial}{\partial \mathbf{p}_i}\beta(\mathbf{p}_0), \qquad \text{as} \qquad \beta(\mathbf{p}_0) \to \infty$$

We know, the first-order safety index is:

$$\beta(\mathbf{p}_0) = \left|\mathbf{u}^*\right| = (\mathbf{u}^{*T}\mathbf{u}^*)^{1/2}$$

where $\mathbf{u}^* = -\beta(\mathbf{p}_0)\dfrac{\nabla g(\mathbf{u}^*, \mathbf{p}_0)}{\left|\nabla g(\mathbf{u}^*, \mathbf{p}_0\right|} = \lambda \nabla g(\mathbf{u}^*, \mathbf{p}_0)$.

So, for a distribution parameter p_i, the component in the sensitivity vector with respect to it follows:

$$\frac{\partial}{\partial \mathbf{p}_i}\beta(\mathbf{p}_0) = \frac{1}{\beta}\mathbf{u}^{*T}\frac{\partial}{\partial \mathbf{p}_i}\mathbf{u}^* = \frac{1}{\beta}\mathbf{u}^{*T}\frac{\partial}{\partial \mathbf{p}_i}\mathbf{T}(\mathbf{z}^*, \mathbf{p}_0)$$

where $\mathbf{z} = \mathbf{T}^{-1}(\mathbf{u}^*, \mathbf{p}_0)$.

Unfortunately, the transformation **T** in our case is extremely complicated, which is very difficult to express in a close form formulation. We divided the reliability sensitivity analysis into several levels. At each level, the transformation is relatively simple so that most components in the sensitivity vector can be handled analytically. Following this approach, with some assistance of numerical analysis, the sensitivity vectors for each structure component are derived with respect to the user-defined biases and COV's.

4.2 Structure Component Reliability Sensitivity Analysis

4.2.1 Sensitivity Vector Formulation

The reliability analysis in TOPCAT was based on a first order second moment (FOSM) approach. The platform is divided into structure components such as deck bay, jacket bay and foundation. Each component is evaluated for its safety index separately. Using the FOSM formulation for capacity R and loading S, and assuming log-normally distributed R and S, the safety margin and safety index are defined for each component:

$$M = \ln R - \ln S$$

$$\beta = \frac{\mu_M}{\sigma_M}$$

where, $\mu_M = \ln\left(\frac{\mu_R}{\mu_S}\sqrt{\frac{1+V_S^2}{1+V_R^2}}\right)$ and $\sigma_M^2 = \ln\left(1+V_R^2\right) + \ln\left(1+V_S^2\right) - 2\ln\left(1+\rho_{RS}V_R V_S\right)$

β is a function of the sets of parameters p_i such as biases and COV's described before. To derive the sensitivity vectors, the partial derivatives of β with respect to p_i, chain rule is applied:

$$\frac{\partial\beta}{\partial p_i} = \sum_{j=1}^{n} \frac{\partial\beta}{\partial F_j}\frac{\partial F_j}{\partial p_i}$$

where F_j is the interim function connecting β to p_i.

To facilitate the evaluation process, a hierarchical system is introduced to evaluate the partial derivatives level by level. The top level is safety index β, while the bottom level is the uncertainty parameters. At each level, the asymptotic approach mention before is used to calculate the partial derivative components in the sensitivity vector which can be expressed in a closed form transformation **T**. For those components which have no close form **T**, such as wave loading, which includes numerical integration, the partial derivatives are evaluated numerically. Figure 4.1 shows such a hierarchical system for a platform structure component.

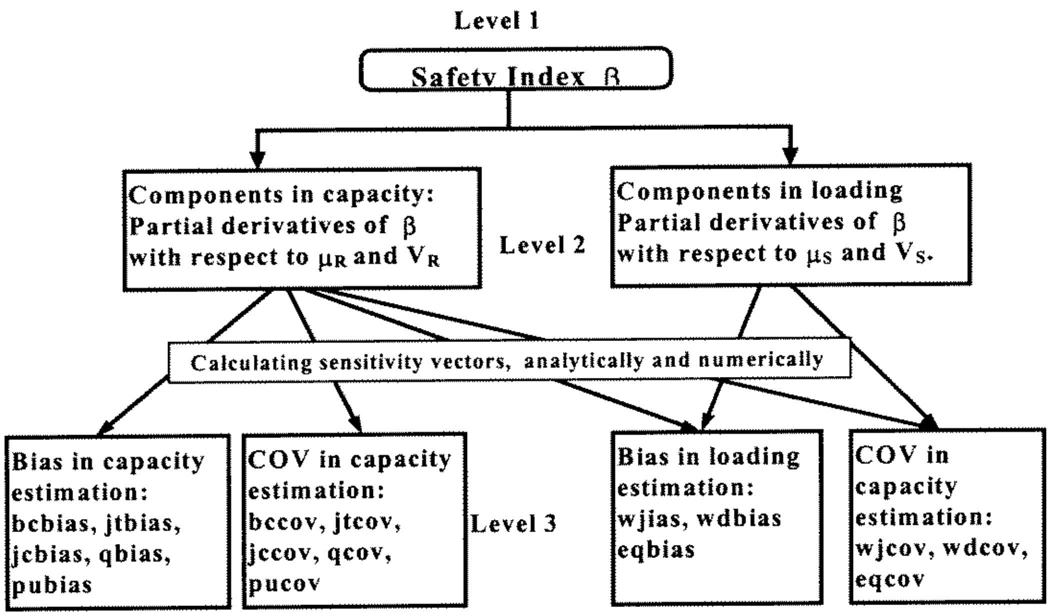

Figure 4.1 Hierarchical system to evaluate partial derivatives in sensitivity vectors

Based on this approach, the sensitivity vector can be express as:

$$\vec{V}_{sensititvity} = \vec{V}_{\beta}^{T} \cdot \left\lfloor \vec{V}_{\mu_R} \quad \vec{V}_{\mu_S} \quad \vec{V}_{V_R} \quad \vec{V}_{V_S} \right\rfloor$$

where, $\vec{V}_{\beta}$ is the first level sensitivity analysis for a structure component. The evaluation at the first leve2 is almost the same for all structure components:

$$\bar{V}_\beta = \begin{bmatrix} \frac{\partial \beta}{\partial \mu_R} \\ \frac{\partial \beta}{\partial \mu_S} \\ \frac{\partial \beta}{\partial V_R} \\ \frac{\partial \beta}{\partial V_S} \end{bmatrix} = \begin{bmatrix} \frac{1}{\sigma_M \mu_R} \\ -\frac{1}{\sigma_M \mu_S} \\ \frac{V_R}{1+V_R^2} \cdot \frac{-\mu_M - \sigma_M^2}{\sigma_M^3} \\ \frac{V_S}{1+V_S^2} \cdot \frac{\sigma_M^2 - \mu_M}{\sigma_M^3} \end{bmatrix}$$

For level 3, the evaluation procedure is quite complicated and depends largely on the different formulations of capacities and loadings of different components. Following is a discussion about the derivation of sensitivity vector of loading and capacity for each structure component.

4.2.2 Sensitivity Vector of Loading

The sensitivity of β to wave loading uncertainties is straightforward and almost the same for every structure components. But it's evaluation process needs numerical methods. As stated in the ULSLEA reports, the variation in loading comes from the variation in wave force. The wind force uncertainty has little contribution and is neglected. The wave loading is formulated as :

$$S_h = K_d K_u H^2$$

Where, S_h is the total integrated hydrodynamic drag force acting on a surface piercing vertical cylinder. K_u is a numerical integration that integrates the velocities along the cylinder and is a function of wave steepness and the wave theory used to estimate the velocities. K_d is a force coefficient and a function of mass density of water ρ, diameter of the cylinder D, and drag coefficient C_d. The mean forces acting on the elements are integrated and the shear force at each component level is calculated. These integrated shear forces define the means of the load variables S_D for deck, S_{Ji} for each jacket bay, and the base shear S_F for the foundation bay. The coefficient of variation of the wave load is given as:

$$V_s = \sqrt{V_{K_d}^2 + V_{K_u}^2 + (2V_H)^2}$$

The sensitivity vector for loading can be easily derived:

$$\vec{V}_{u_s} = \begin{bmatrix} \frac{\partial \mu_s}{\partial \mu_{k_d}} \\ \frac{\partial \mu_s}{\partial \mu_{k_u}} \\ \frac{\partial \mu_s}{\partial \mu_H} \\ \frac{\partial \mu_s}{\partial b_s} \end{bmatrix} = \begin{bmatrix} \mu_{k_H} \mu_H^2 \\ \mu_{k_d} \mu_H^2 \\ \mu_{k_d} \mu_{k_u} \mu \\ \mu_s \end{bmatrix} \quad \text{and} \quad \vec{V}_{V_s} = \begin{bmatrix} \frac{\partial V_s}{\partial V_{k_d}} \\ \frac{\partial V_s}{\partial V_{k_u}} \\ \frac{\partial V_s}{\partial V_H} \end{bmatrix} = \begin{bmatrix} \frac{V_{k_d}}{V_s} \\ \frac{V_{k_u}}{V_s} \\ \frac{4V_H}{V_s} \end{bmatrix}$$

Where, μ and **V** are mean value and COV of the random variables.

However, the function K_u and K_d are not close-form. They are numerical integration process in TOPCAT. The loading uncertainty parameters such as wjcov and wdcov are deeply embedded in the loading estimation procedure and in the reliability analysis module. So the sensitivity vectors have to be numerically evaluated also. Using the chain rule, we can get the sensitivity vectors of loading with respect to wave force COV as follows. The partial derivatives in the equations are calculated numerically.

$$\vec{V}_{u_s} = \begin{bmatrix} \frac{\partial \mu_s}{\partial Wdbias} \\ \frac{\partial \mu_s}{\partial Wjbias} \end{bmatrix} \quad \text{and} \quad \vec{V}_{V_s} = \begin{bmatrix} \frac{\partial V_s}{\partial Wjcov} \\ \frac{\partial V_s}{\partial Wd\,cov} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \text{or} \begin{bmatrix} \frac{Wjcov}{V_s} \\ \frac{Wd\,cov}{V_s} \end{bmatrix}$$

4.2.3 Jacket Bay Reliability Sensitivity Analysis

Shear capacity in a given jacket bay is assumed to be reached when the vertical diagonal braces or their joints are no longer capable of resisting the lateral load acting on the jacket bay. Tensile and compressive capacity of the diagonal braces, the associated joint capacities, and the batter component of axial forces in the legs due to overturning moment are included to estimate the jacket bay shear capacity. The reliability analysis is taken for only the low bound capacity of the jacket bay.

To estimate the lateral capacity of a given jacket bay, it is assumed that interconnecting horizontal brace elements are rigid. Thus, the lower-bound capacity of the n^{th} jacket bay R_{Jn}, which is associated with the first member failure in that bay, can be given as:

$$R_{Jn} = \sum_i \overline{\alpha}_n K_i + F_L$$

where F_L is the sum of batter components of axial pile and leg forces in the given bay and $\overline{\alpha}_n = \frac{P_{u,MLTF}}{K_{MLTF}}$ is the lateral drift of the n^{th} jacket bay at the onset of first member failure. K_i are deterministic factors accounting for geometry and relative member stiffness ($\overline{\alpha}\, K_i$ = horizontal shear force of brace element i at the onset of first brace or joint failure within the given bay). Assuming that there is no correlation between the capacity of the MLTF member and lateral shear in the jacket legs, the variance of the lower-bound capacity of the n^{th} jacket bay can be given as:

$$\sigma^2_{RJn} = \left(\sum_i K_i\right)^2 \sigma^2_{\overline{\alpha}} + B^2_{F_L} \sigma^2_{F_L}$$

where $\sigma_{\bar{\alpha}} = \frac{\sigma_{Pu,MLTF}}{K_{MLTF}}$, B_{FL} denotes the bias associated with the batter component of axial leg forces F_L. And

$$P_u = \frac{M_u}{8\Delta_0\left(\frac{1}{1+2\frac{\sin 0.5\varepsilon}{\sin\varepsilon}}\right)\frac{1}{\varepsilon^2}\left(\frac{1}{\cos\frac{\varepsilon}{2}}-1\right)} - \frac{wl^2}{8\Delta_0}$$

Thus the variance of the compression capacity of a brace can be given by

$$\sigma_{Pu}^2 = \sigma_{Pcr}^2 + \left(\frac{l^2}{8\Delta_0}\right)^2 \sigma_w^2$$

Based on this capacity formulation, the sensitivity analysis is conducted. The mean capacity and the capacity COV are functions of biases and COV's defined by users. The sensitivity vectors can be expressed as:

$$\vec{V}_{\mu_{R_{J_n}}} = \begin{bmatrix} \frac{\partial \mu_{R_{J_n}}}{\partial \mu_{\bar{\alpha}_n}} \\ \frac{\partial \mu_{R_{J_n}}}{\partial \mu_{\beta_{F_L}}} \end{bmatrix} = \begin{bmatrix} \sum_i \mu_{k_i} \\ 1 \end{bmatrix} \quad \text{and} \quad \vec{V}_{V_{R_{J_n}}} = \begin{bmatrix} \frac{\partial V_{R_{J_n}}}{\partial \sigma_{\bar{\alpha}_n}} \\ \frac{\partial V_{R_{J_n}}}{\partial \sigma_{F_L}} \\ \frac{\partial V_{R_{J_n}}}{\partial \beta_{F_L}} \end{bmatrix} = \begin{bmatrix} \frac{(\sum_i \mu_{k_i})^2 \cdot \sigma_{\bar{\alpha}_n}}{\sigma_{R_{J_n}} \cdot \mu_{R_{J_n}}} \\ \frac{B_{F_L}^2 \cdot \sigma_{F_L}}{\sigma_{R_{J_n}} \cdot \mu_{R_{J_n}}} \\ \frac{B_{F_L} \cdot \sigma_{F_L}^2}{\sigma_{R_{J_n}} \cdot \mu_{R_{J_n}}} \end{bmatrix}$$

Keep the sensitivity analysis going to the lower levels, we get more sensitivity vectors:

$$\vec{V}_{\bar{\mu}_{\alpha_n}} = \begin{bmatrix} \frac{\partial \mu_{\bar{\alpha}_n}}{\partial bcbias} \\ \frac{\partial \mu_{\bar{\alpha}_n}}{\partial Wjbias} \end{bmatrix} = \begin{bmatrix} \frac{P_{u,MLTF}}{bcbias \cdot K_{MLTF}} \\ \frac{1}{K_{MLTF}} \cdot \left(-\frac{\mu_W \cdot l^2}{8\Delta_0 \cdot Wjbias}\right)_{MLTF} \end{bmatrix}$$

$$\vec{V}_{\sigma_{F_L}} = \begin{bmatrix} \frac{\partial \sigma_{F_L}}{\partial Wjcov} \\ \frac{\partial \sigma_{F_L}}{\partial Wdcov} \\ \frac{\partial \sigma_{F_L}}{\partial wjbias} \\ \frac{\partial \sigma_{F_L}}{\partial wdbias} \end{bmatrix} = \begin{bmatrix} \mu_{F_L} \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} \text{or} \mu_{F_L} \cdot \begin{bmatrix} \frac{Wjcov}{V_s} \\ \frac{Wdcov}{V_s} \end{bmatrix} \\ cov\,load \cdot \frac{\partial \mu_{F_L}}{\partial wjbias} \\ cov\,load \cdot \frac{\partial \mu_{F_L}}{\partial wdbias} \end{bmatrix}$$

$$\vec{V}_{\sigma_{\bar{\alpha}_s}} = \begin{bmatrix} \frac{\partial \sigma_{\bar{\alpha}_s}}{\partial bc\,cov} \\ \frac{\partial \sigma_{\bar{\alpha}_s}}{\partial wjcov} \\ \frac{\partial \sigma_{\bar{\alpha}_s}}{\partial wjbias} \\ \frac{\partial \sigma_{\bar{\alpha}_s}}{\partial bcbias} \end{bmatrix} = \begin{bmatrix} \frac{P_{u,MLTF}^{\;2} \cdot bc\,cov}{\sigma_{P_{u,MLTF}} \cdot K_{MLTF}} \\ \frac{(\frac{1^2}{8\Delta_0})^2 \cdot \mu_w^{\;2} \cdot wjcov}{K_{MLTF} \cdot \sigma_{P_{u,MLTF}}} \\ \frac{(\frac{1^2}{8\Delta_0})^2 \cdot (wjcov)^2 \cdot \mu_w \cdot wjbias}{\sigma_{P_{u,MLTF}} \cdot K_{MLTF}} \\ \frac{bc\,cov \cdot P_{u,MLTF}}{K_{MLTF} \cdot bcbias} \end{bmatrix}$$

Note that the batter component of leg forces F_L is included in the jacket capacity, but it depends on the wave loading, so it's mean value and standard deviation are functions of the wave loading biases and COV's. i.e. $\mu_{F_L} = F(wdbias, wjbias)$. F is a numerical integration. So, again, we encounter a non-close-form transformation, the partial derivatives in the sensitivity vector $\vec{V}_{\mu_{F_L}} = \begin{bmatrix} \frac{\partial \mu_{F_L}}{\partial wdbias} \\ \frac{\partial \mu_{F_L}}{\partial wjbias} \end{bmatrix}$ need to be numerically evaluated.

4.2.4 Deck Bay Reliability Sensitivity Analysis

As to the deck capacity, two kinds of decks are formulated: unbraced deck portal and braced deck bay. For the braced deck bay, the capacity formulation is just the same as the jacket bay. So the sensitivity analysis follows the same as that of jacket bay described above.

For an unbraced deck portal, a mechanism in the deck leg bay would form when plastic hinges are developed at the top and bottom of all of the deck legs. Using this failure mode as a virtual displacement, virtual work principle can be utilized to estimate the deck leg shear resistance R_d:

$$R_d = \frac{1}{L_d}(2n\, M_u - Q\Delta)$$

where $\Delta = M_U L_d \left(\frac{L_d}{6EI} + \frac{1}{C_r} \right)$

$$\frac{M_u}{M_{cr}} - \cos\left(\frac{\pi}{2} \frac{Q/n}{P_{crl}} \right) = 0$$

The moment capacity of the legs M_{cr} and the local buckling capacity P_{crl} are treated as random variables. Assuming perfect correlation between M_{cr} and P_{crl}, the variance of deck legs capacity can be given as:

$$\sigma_{Rd}^2 = \sigma_{Mcr}^2 \left(\frac{\partial R_d}{\partial M_{cr}} \right)^2 + \sigma_{Pcrl}^2 \left(\frac{\partial R_d}{\partial P_{crl}} \right)^2 + 2\sigma_{Mcr}\sigma_{Pcrl} \left(\frac{\partial R_d}{\partial M_{cr}} \right) \left(\frac{\partial R_d}{\partial P_{crl}} \right)$$

where $\frac{\partial R_d}{\partial M_{cr}}$ and $\frac{\partial R_d}{\partial P_{crl}}$ are the partial derivatives of the deck legs shear capacity R_d with respect to critical moment and buckling capacities M_{cr} and P_{crl}, evaluated at the mean values μ_{Mcr} and μ_{Pcrl}. Based on these formulation, the sensitivity vectors can be derived as:

$$\vec{V}_\beta = \begin{bmatrix} \dfrac{\partial\beta}{\partial\mu_R} \\ \dfrac{\partial\beta}{\partial\mu_S} \\ \dfrac{\partial\beta}{\partial\sigma_R} \\ \dfrac{\partial\beta}{\partial V_S} \end{bmatrix} \qquad \text{where } \frac{\partial\beta}{\partial\sigma_R} = \frac{V_R}{1+V_R^2}\cdot\frac{-\mu_M-\sigma_M^2}{\sigma_M^2}\cdot\frac{1}{\mu_{R_d}}$$

As stated before, in the first level sensitivity vectors, the components in load $\vec{V}_{\mu_s}$ and $\vec{V}_{V_s}$ are numerically evaluated.

The capacity of unbraced deck portal is mainly determined by deck section geometry and structure members' characteristics. The mean value capacity μ_{R_d} is assumed unbiased in the previous TOPCAT version. And the COV's of Mcr and P_{crl} are assumed to be fixed at 10%. In the new version, these fixed values are changed into user-defined parameters. The input menu for biases and uncertainties are modified. Also, in the input menu, the capacity bias and COV for caisson are added. The standard deviation σ_{R_d} is assumed to be determined by variations in moment capacity M_{cr} and local buckling capacity P_{crl}. The sensitivity vector of σ_{R_d} is evaluated by:

$$\vec{V}_{\mu_{R_d}} = \begin{bmatrix} \dfrac{\partial\mu_{R_d}}{\partial M_{cr}\text{bias}} \\ \dfrac{\partial\mu_{R_d}}{\partial P_{crl}\text{bias}} \end{bmatrix} = \begin{bmatrix} \dfrac{2n}{L_d}\cdot\dfrac{\partial M_u}{\partial M_{cr}\text{bias}} - \dfrac{Q}{L_d}\cdot\dfrac{\partial\Delta}{\partial M_{cr}\text{bias}} \\ \dfrac{2n}{L_d}\cdot\dfrac{\partial M_u}{\partial P_{crl}\text{bias}} - \dfrac{Q}{L_d}\cdot\dfrac{\partial\Delta}{\partial P_{crl}\text{bias}} \end{bmatrix}$$

$$\vec{V}_{\sigma_{R_d}} = \begin{bmatrix} \dfrac{\partial \sigma_{R_d}}{\partial M_{cr} \text{cov}} \\ \dfrac{\partial \sigma_{R_d}}{\partial P_{crl} \text{cov}} \\ \dfrac{\partial \sigma_{R_d}}{\partial M_{cr} \text{bias}} \\ \dfrac{\partial \sigma_{R_d}}{\partial P_{crl} \text{bias}} \end{bmatrix} = \begin{bmatrix} \dfrac{\mu_{R_d}}{\sigma_{R_d}} \left[(\dfrac{\partial R_d}{\partial M_{cr}})^2 \cdot \sigma_{M_{cr}} + (\dfrac{\partial R_d}{\partial M_{cr}})(\dfrac{\partial R_d}{\partial P_{crl}}) \sigma_{P_{crl}} \right] \\ \dfrac{\mu_{R_d}}{\sigma_{R_d}} \left[(\dfrac{\partial R_d}{\partial P_{crl}})^2 \cdot \sigma_{P_{crl}} + (\dfrac{\partial R_d}{\partial M_{cr}})(\dfrac{\partial R_d}{\partial P_{crl}}) \sigma_{M_{cr}} \right] \\ -V_R \cdot \dfrac{\partial \mu_{R_d}}{\partial M_{cr} \text{bias}} \\ -V_R \cdot \dfrac{\partial \mu_{R_d}}{\partial P_{crl} \text{bias}} \end{bmatrix}$$

where, $\dfrac{\partial M_u}{\partial M_{cr} \text{bias}} = \dfrac{M_u}{M_{cr} \text{bias}}$

$$\frac{\partial M_u}{\partial P_{crl} \text{bias}} = \frac{M_{cr} \text{Sin}(\frac{\pi}{2} \frac{Q_n}{P_{crl}}) \cdot \frac{\pi}{2} \cdot \frac{Q_n}{p_{crl}}}{P_{crl} \text{bias}^2}$$

$$\frac{\partial \Delta}{\partial M_{cr} \text{bias}} = \frac{1}{L_d} (\frac{L_d}{6EI} + \frac{1}{C_r}) \cdot \frac{\partial M_u}{\partial M_{crl} \text{bias}}$$

$$\frac{\partial \Delta}{\partial P_{crl} \text{bias}} = \frac{1}{L_d} (\frac{L_d}{6EI} + \frac{1}{C_r}) \cdot \frac{\partial M_u}{\partial P_{crl} \text{bias}}$$

And the partial derivatives $\dfrac{\partial R_d}{\partial M_{cr}}$ and $\dfrac{\partial R_d}{\partial P_{crl}}$ are numerically evaluated.

4.2.4 Foundation Reliability Sensitivity Analysis

Two basic types of failure mode in the foundation are considered: lateral and axial. The lateral failure mode of the piles is similar to that of the deck legs. In addition to moment resistance of the piles, the lateral support provided by foundation soils and the batter shear component of the piles are considered. The lateral and axial capacity equations for piles in sand and clay are given in the ULSLEA report. These formulations are used to calculate the best estimate capacities. Considering the uncertainties in soil and pile material properties, the uncertainties associated with foundation capacities can also be estimated. However, due to lack of data regarding modeling uncertainties, the total uncertainties associated with axial and lateral pile capacities are used in the current TOPCAT, which implicitly include the uncertainties associated with soil and pile parameters and capacity modeling. Obviously, it is a raw reliability model. The uncertainty associated with the batter component of the pile force is added to the total capacity uncertainty for vertically driven piles.

Based on the capacity and loading formulation of pile foundation. The sensitivity vectors are derived. For a laterally loaded pile,

$$\mu_R = \text{pubias} \cdot \mu'_{P_U} + F_{L,\text{foundation}}$$

where, $F_{L,\text{foundation}}$ is the batter force component in the foundation.

The sensitivity at different levels are as follows:

$$\vec{V}_{\mu_R} = \begin{bmatrix} \dfrac{\partial \mu_R}{\partial \text{pubias}} \\ \dfrac{\partial \mu_R}{\partial \text{wdbias}} \\ \dfrac{\partial \mu_R}{\partial \text{wjbias}} \end{bmatrix} = \begin{bmatrix} \dfrac{\mu_R - \mu_{F_L}}{\text{pubias}} \\ \dfrac{\partial \mu_{F_L}}{\partial \text{wdbias}} \\ \dfrac{\partial \mu_{F_L}}{\partial \text{wjbias}} \end{bmatrix}$$

The later two terms is $\vec{V}_{\mu_R}$ are evaluated numerically in the same way as the of jacket bay.

The loading sensitivity vector $\vec{V}_{\mu_S} = \begin{bmatrix} \dfrac{\partial \mu_S}{\partial \text{wdbias}} \\ \dfrac{\partial \mu_S}{\partial \text{wdjias}} \end{bmatrix}$ and $\vec{V}_{V_S} = \begin{bmatrix} \dfrac{\partial V_S}{\partial \text{wdbias}} \\ \dfrac{\partial V_S}{\partial \text{wdjias}} \end{bmatrix}$ is numerically evaluated.

At last the capacity COV sensitivity vector is:

$$\bar{V}_{V_R} = \begin{bmatrix} \frac{\partial V_R}{\partial pubias} \\ \frac{\partial V_R}{\partial wdbias} \\ \frac{\partial V_R}{\partial wjbias} \\ \frac{\partial V_R}{\partial pu\,cov} \\ \frac{\partial V_R}{\partial wd\,cov} \\ \frac{\partial V_R}{\partial wj\,cov} \end{bmatrix} = \begin{bmatrix} \frac{\partial \mu_R}{\partial pubias}\left(\frac{pu\,cov \cdot \mu_{pu}}{\mu_R \sigma_R} - \frac{\sigma_R}{\mu_R^2}\right) \\ \frac{\partial \mu_{F_L}}{\partial wdbias}\left(\frac{\mu_{F_L} \cdot cov\,load}{\sigma_R \mu_R} - \frac{\sigma_R}{\mu_R^2}\right) \\ \frac{\partial \mu_{F_L}}{\partial wjbias}\left(\frac{\mu_{F_L} \cdot cov\,load}{\sigma_R \mu_R} - \frac{\sigma_R}{\mu_R^2}\right) \\ \frac{pu\,cov \cdot \mu_{P_u}^2}{\mu_R \cdot \sigma_R} \\ \frac{\sigma_{F_L} \cdot \mu_{F_L}}{\mu_R^2 \cdot V_R} \cdot \frac{\partial V_S}{\partial Wdbias} \\ \frac{\sigma_{F_L} \cdot \mu_{F_L}}{\mu_R^2 \cdot V_R} \cdot \frac{\partial V_S}{\partial Wjbias} \end{bmatrix}$$

where, $\frac{\partial \mu_{F_L}}{\partial wdbias}, \frac{\partial \mu_{F_L}}{\partial wdjias}$ is evaluated in the same way as described before.

For the axially loaded pile (both tension and compression), the sensitivity analysis procedure just follows the same approach as that of laterally loaded pile.

4.3 Case Study

An example platform is analyzed to verify the reliability sensitivity module. Again, SP62 is taken as such an example. The total sensitivity vectors of safety index with respect to biases and COV's are summarized in Table 4.1.

From the results, we can conclude that the safety index is the most sensitive to biases of capacity and loading, especially to the strength bias and jacket wave force bias. The components of the sensitivity vectors with respect to these biases are close to 1, which is a relatively large value, means that the magnitude in change of β is almost equal to the change in bias. For COV's, β is less sensitive.

Table 4.1 Sensitivity vectors of safety index for SP62

Reliability Sensitivity Vectors of Safety Indexes with respect to Biases and COV's								
Jacket Bay								
Bay #	Broad Side	Bcbias	Wdbias	Wjbias	Bccov	Wdcov	Wjcov	
1		0.8831	-0.0278	-0.9417	-0.3207	0.0511	0.0511	
2		0.8292	-0.0194	-0.9392	-0.1545	0.1132	0.1132	
3		0.8046	-0.0162	-0.9393	-0.1628	0.0362	0.0362	
4		0.7502	-0.0142	-0.9218	-0.2297	0.0461	0.0461	
5		0.7076	-0.0129	-0.9032	-0.1125	0.053	0.053	
6		0.7472	-0.0126	-0.9307	-0.3504	-0.0581	-0.0581	
Bay #	End On	Bcbias	Wdbias	Wjbias	Bccov	Wdcov	Wjcov	
1		0.8735	-0.0234	-0.9586	-0.4354	0.049	0.049	
2		0.8747	-0.0165	-0.973	-0.1865	0.1269	0.1269	
3		0.8456	-0.0132	-0.9718	-0.2996	0.1204	0.1204	
4		0.8198	-0.0115	-0.9592	-0.1579	0.1243	0.1243	
5		0.7832	-0.0104	-0.9481	-0.281	0.1101	0.1101	
6		0.746	-0.0096	-0.941	-0.4273	0.0856	0.0856	
Deck Bay								
Deck Portal	Dmcrbias	Dpcrlbias	Wdbias	Wjbias	Dmcrcov	Dpcrlcov	Wdcov	Wjcov
Broad Side	0.9795	0.0789	-0.0832	-0.7734	-0.2821	-0.0289	-0.1633	-0.1633
End On	0.9801	0.0789	-0.0703	-0.8169	-0.2874	-0.0295	-0.1789	-0.1789
Foundation								
Broad Side	Pubias	Wdbias	Wjbias	Pucov	Wdcov	Wjcov		
Lateral	0.6674	-0.0094	-0.7574	-0.1858	0	-0.0585		
Axial(COM)	0.8698	-0.0149	-0.7409	-0.5128	0.1526	0.1526		
Axial(TEN)	0.8698	-0.0232	-1.151	-0.6366	0.0494	0.0494		
End On	Pubias	Wdbias	Wjbias	Pucov	Wdcov	Wjcov		
Lateral	0.6538	-0.0083	-0.8483	-0.2313	0	-0.0279		
Axial(COM)	0.8698	-0.0115	-0.7422	-0.529	0.139	0.139		
Axial(TEN)	0.8698	-0.0185	-1.1951	-0.6623	0.028	0.028		

These results have some important implication for engineering practice. These sensitivity vectors provide users very useful information. They help users identity which input parameters are more important than others, lead users' attention to the determination of values of these input parameters, thus help avoid wrong input.

These vectors can even help understand the change tendency of safety index. For example, increasing strength bias can be taken as equal to increasing structure strength, and increasing loading bias can be equivalent to increasing storm condition. By this way, these vectors can tell users how much the structure should be strengthened to get a required safety level. Or, they can tell users how much more loading can be imposed on the structure before it is endangered.

5.0 Diagonal Loading on Offshore Structures

Present TOPCAT is mainly a 2-D analysis tool. It only calculates BS and EO loading and capacity characteristics. This is a study of two idealized cases, which obviously are not realistic. However, offshore structures in field subject to loads from different directions. One effort suggested by sponsors is the extend the 2-D TOPCAT to a 3-D analysis tool. The first step to take in this research direction is the study of diagonal loading on offshore structures.

This problem has be formulated by MTMG in UC Berkeley. Following is a brief description of the basic approach to the problem. The analysis process can be divided into two major step:

- Decomposition and superposition at the global level;
- Detailed loading and structure analysis at local level, no superposition and decomposition are allowed at this level;

The decomposition and superposition procedures at global level are demonstrated in Figure 5.1. It is assumed the diagonal loading pass through the geometry center of the platform, thus no global torque is introduced. This assumption also require that the spatial effects is not considered, this is because that torque will be created if spatial effects are considered. Of course, all these effects can be counted in. But at this time, we only consider the idealized no-torsion situation. The detailed study of torsion will be left for the next phase.

In Figure 5.1, the diagonal force is decomposed into BS and EO components, which are resisted by BS and EO frames respectively. Then, at the global level, the analysis of BS and EO capacities can follow exact the same approach applied in the current TOPCAT, but with two exceptions. One is that the batter components of leg forces will be completely different. The other is the change of the load patterns for pile foundation.

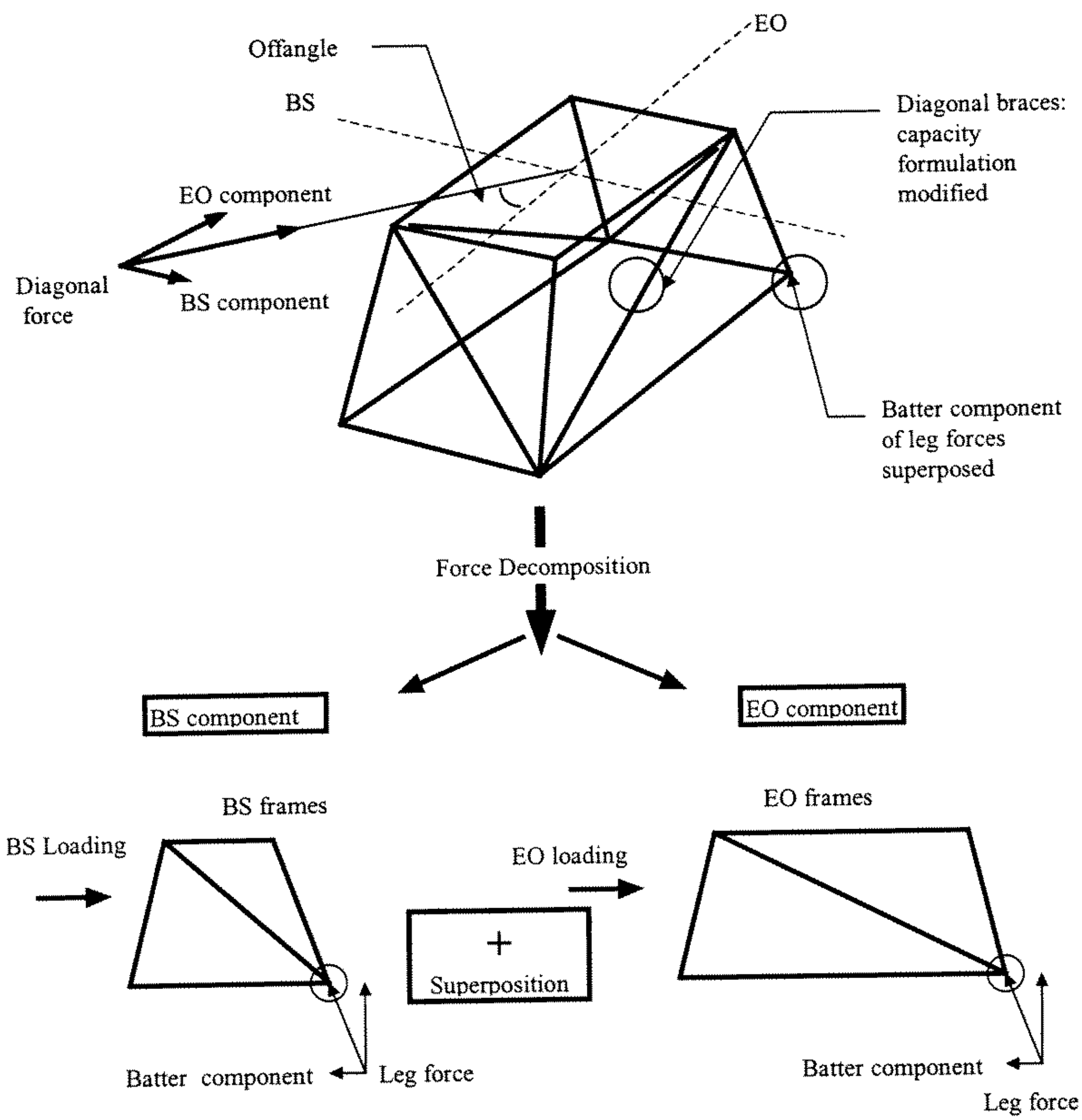

Figure 5.1 Diagonal loading: decomposition and superposition

Both problems can be solved by superposition of leg forces obtained from BS and EO analysis. The change in batter forces is considered to be the most important contribution to the change of the loading and capacities caused by the diagonal offangle. After the new BS and EO capacities are determined, they are compared with the BS and EO components of loading. As the platform usually has different BS and EO capacity, the offangle determines which frame, BS or EO, will have the first diagonal brace failure. Then this lower capacity is the structure's capacity resisting the diagonal loading.

As to the local level, Figure 5.2 is a cartoon showing the main concerns in the loading and capacity formulation.

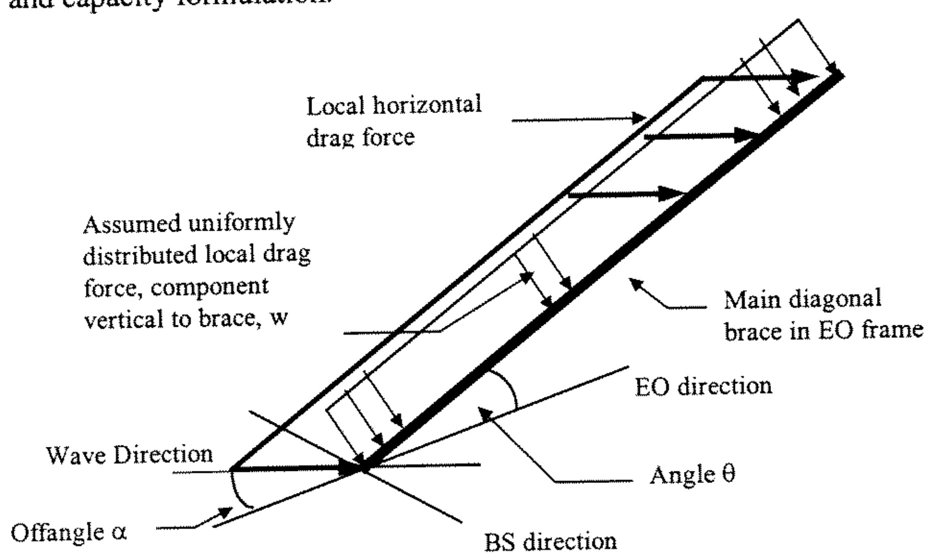

Figure 5.2 Diagonal wave force and diagonal brace capacity at local level

The effects of the diagonal offangle at local level are mainly reflected in the equivalent cylinder diameter of the braces and the ultimate capacity of braces. The first effect will also have its influence at the global level. The total diagonal loading largely depends on the value of offangle α. For example, for a diagonal brace in EO frame. The variation of the equivalent diameter D_{equ} can be formulated as:

α=0 degree, EO loading: $D_{equ} = D \cdot Sin^2\theta$

α=90 degree, BS loading: $D_{equ} = D / Sin^2\theta$

any α, diagonal loading: $D_{equ} = D(sin^3 \alpha + Cos^3\alpha \cdot Sin^3\theta) / Sin\theta$

As to the ultimate capacity P_u, we already know that it is formulated as:

$$P_u = \frac{M_u}{8\Delta_0 \left(\frac{1}{1 + 2\frac{sin\,0.5\varepsilon}{sin\,\varepsilon}} \right) \frac{1}{\varepsilon^2} \left(\frac{1}{cos\frac{\varepsilon}{2}} - 1 \right)} - \frac{w l^2}{8\Delta_0}$$

The uniformly distributed vertical loading, w, in the formula is a function of the offangle too. So the capacity P_u is also a function of the offangle. This is the second order effect on the total structure capacity formulation of the diagonal wave loading.

Above are the basic approaches being applied in the development of the diagonal loading analysis module. The program coding is still under way.

6.0 Conclusion and Future Developments

Following the schedule set in last project meeting, MTMG in UC Berkeley finished three tasks in the TOPCAT project in Spring, 1998. These tasks are:

- shallow water wave kinematics: cnoidal wave theory;
- spatially distributed wave loading on large structures;
- reliability sensitivity analysis;

This report summarizes the theory basis, problem formulation and problem solving strategies developed during working on these tasks. The program modules performing the analysis have been developed and integrated into TOPCAT. Case-study results show good agreements with expectation. More extensive verification of the validity of these modules is expected to be performed by new students in MTMG group. The users participation and help in this verification will be highly appreciated.

This report also documents the basic approach of the diagonal loading analysis. This approach is a part of the effort to extend the 2-D TOPCAT model to a more advanced 3-D analysis tool. This will lead to the global torsion analysis and "optimal" platform layouts. This is the main objective of the next phase of TOPCAT project. Besides this effort, another thrust will be put on the development of an simplified module that analyzes the platform deck element. This includes both the truss deck in Gulf of Mexico and the box girder deck of platforms in North Sea.

Meanwhile, based on the work finished this semester, some small modifications are expected. These may include the possibility of introducing the inertial force term to make the load spatial effects more reasonable; the upgrading of iteration algorithm in cnoidal wave kinematics module to improve the calculation efficiency; the updating of formulation of multi 4-leg jacket platforms so that it can handle multi multi-leg jacket platform; integration of a small module that can do the analysis of long-term reliability.

References

American Petroleum Institute(1993), *Recommended Practice for Planning Designing and Constructing Fixed Offshore Platforms, API RP 2A -WSD*, 20th edition.

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

Bea, R. G. (1997), *CE205 B Class Note*, University of California at Berkeley.

Dean, R. G., Dalrymple R. A. (1984), *Water Wave Mechanics for Engineers and Scientists,* Prentice-Hall, New York.

Fenton, J. D. (1979), *A high-order cnoidal wave theory*, Journal of Fluid Mechanics, voil. 94, part 1, pp. 129-161.

Laitone, E. V.(1960), *The second approximation to cnoidal and solitary waves*, Journal of Fluid Mechanics, vol. 9, pp430-444.

Mortazavi, M., and Bea, R. G.(1996), *Screen Methodologies for use in platform assessments and requalifications*, final project report, Marine Technology and Management Group, Dept. of Civil Engineering, University of California, Berkeley, Jan., 1996.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., Flannery, B. P.(1988), *Numerical Recipes in C: The Art of Scientific Computing, Second Edition*, Cambridge University Press.

Sarpkaya, T(1981), *Mechanics of Wave Forces on Offshore Structures*, New York, Van Nostrand Reihold Co.

U.S. Department of Commerce, National Bureau of Standards(1964), *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables.*

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., Mortazavi, M., and Bea, R. G.(1996), *ULSLEA, Manual of Operation*, Marine Technology and Management Group, Dept. of Civil Engineering, University of California, Berkeley, Jan., 1996.

Microsoft Corporation(1994), *Visual Basic User's Guide, Microsoft Excel, Automating, Customizing and Programming in Microsoft Excel with the Microsoft Visual Basic Programming System, Application Edition.*

Madsen, H. O., Krenk, S., Lind, N. C., *Methods of Structural Safety*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1986

Neufeville, R., Stafford, J. H., Systems *Analysis for Engineers and Managers*. McGraw-Hill Book Company, 1971.

Cruz, J. B., *Benchmark Papers in Electrical Engineering and Computer Science, System Sensitivity Analysis*, Dowden, Hutchingon and Ross Inc., Stroudsburg, Pennsylvania, 1973.

Appendix

Modules Created or Extensively Modified

```
'
' This Module contains one procedure, Macro3.
'
' Last Modified by:  Zhaohui Jin
'          on:  May 22.1998

Dim senvb(2, 30, 4)

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:   5/22/97
'
' Last Modified by:   Zhaohui Jin
'          on:  5/22/98
' The algorithm calculating the sensitivity vector of reliability analysis
' is integrated in this sub and other modules: module 3, module 4, module 5
' and module 7
'
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 deckbaybraces = True 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 deckbaybraces = False Then
        If nleg < 5 Then
          '$$$$$$$
          prdpmcr1 = 2 * nleg / bayh(0) * Cos(Pi / 2 * qdeck / nleg / fy / dla(1))
          prdppcr1 = 2 * dlmcr(1) * qdeck / (bayh(0) * fy ^ 2 * dla(1) ^ 2) * Sin(Pi * qdeck / nleg / 2 / fy / dla(1))
          sigmacap(i, j) = dmcrcov * dlmcr(1) * prdpmcr1 + dpcrlcov * (fy * dla(1)) * prdppcr1
          psmcov = dlmcr(1) * prdpmcr1
          pspcov = fy * dla(1) * dpcrlbias * prdppcr1
          'psmbias = dmcrcov * p
          ' 0.1 should be changed to a cov in input: dmcrcov, dpcrlcov
          'sigmacap(I, j) = (((0.1 * dlmcr(1)) * 2 * nleg / bayh(0) * Cos(PI / 2 * qdeck / nleg / fy / dla(1))) ^ 2 + ((0.1 * fy * dla(1)) * (2 * dlmcr(1) * qdeck / (bay
h(0) * fy ^ 2 * dla(1) ^ 2) * Sin(PI * qdeck / nleg / 2 / fy / dla(1)))) ^ 2 + 2 * (0.1 * dlmcr(1)) * 2 * nleg / bayh(0) * (Cos(PI / 2 * qdeck / nleg / fy / dla(1))) * (0.1
* fy * dla(1)) * (2 * dlmcr(1) * qdeck / (bayh(0) * fy ^ 2 * dla(1) ^ 2) * Sin(PI * qdeck / nleg / 2 / fy / dla(1)))) ^ 0.5
        Else
          prdpmcr1 = 2 * 4 / bayh(0) * Cos(Pi / 2 * qdeck / nleg / fy / dla(1))
          prdppcr1 = 2 * dlmcr(1) * qdeck / (bayh(0) * fy ^ 2 * dla(1) ^ 2) * Sin(Pi * qdeck / nleg / 2 / fy / dla(1))
'          sigmacap(i, j) = (((0.1 * dlmcr(1)) * 2 * 4 / bayh(0) * Cos(Pi / 2 * qdeck / nleg / fy / dla(1))) ^ 2 + ((0.1 * fy * dla(1)) * (2 * dlmcr(1) * qdeck / (bayh(0
) * fy ^ 2 * dla(1) ^ 2) * Sin(Pi * qdeck / nleg / 2 / fy / dla(1)))) ^ 2 + 2 * (0.1 * dlmcr(1)) * 2 * 4 / bayh(0) * (Cos(Pi / 2 * qdeck / nleg / fy / dla(1))) * (0.1 * fy *
dla(1)) * (2 * dlmcr(1) * qdeck / (bayh(0) * fy ^ 2 * dla(1) ^ 2) * Sin(Pi * qdeck / nleg / 2 / fy / dla(1)))) ^ 0.5
          prdpmcr2 = 2 * (nleg - 4) / bayh(0) * Cos(Pi / 2 * qdeck / nleg / fy / dla(2))
          prdppcr2 = 2 * dlmcr(2) * qdeck / (bayh(0) * fy ^ 2 * dla(2) ^ 2) * Sin(Pi * qdeck / nleg / 2 / fy / dla(2))
          sigmacap(i, j) = (prdpmcr1 * dmcrcov * dlmcr(1) + prdppcr1 * dpcrlcov * fy * dla(1)) + (prdpmcr2 * dmcrcov * dlmcr(2) + prdppcr2 * dpcrlcov * fy * d
la(2))
'          sigmacap(i, j) = sigmacap(i, j) + (((0.1 * dlmcr(2)) * 2 * (nleg - 4) / bayh(0) * Cos(Pi / 2 * qdeck / nleg / fy / dla(2))) ^ 2 + ((0.1 * fy * dla(2)) * (2 * dlm
cr(2) * qdeck / (bayh(0) * fy ^ 2 * dla(2) ^ 2) * Sin(Pi * qdeck / nleg / 2 / fy / dla(2)))) ^ 2 + 2 * (0.1 * dlmcr(2)) * 2 * (nleg - 4) / bayh(0) * (Cos(Pi / 2 * qdeck /
nleg / fy / dla(2))) * (0.1 * fy * dla(2)) * (2 * dlmcr(2) * qdeck / (bayh(0) * fy ^ 2 * dla(2) ^ 2) * Sin(Pi * qdeck / nleg / 2 / fy / dla(2)))) ^ 0.5
          psmcov = prdpmcr1 * dlmcr(1) + prdpmcr2 * dlmcr(2)
          pspcov = prdppcr1 * fy * dla(1) + prdppcr2 * fy * dla(2)
        End If
      ElseIf j = 1 And pltype = 7 Or pltype = 8 Then
'
' COV of caisson braces or guywires is assumed to be 30%
' The COV and bias of caisson capacity are added to the uncertainty input dialogsheet
        sigmacap(i, j) = lbcap(i, j) * caissoncapcov
      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)
        ' $$$$$$$
        ' sensitivity vector w.r.t beta, 1 - meancap, 2 - meanload,
        ' 3 - COV capacity, 4 - COV load,
        senvb(i, j, 1) = 1 / lbcap(i, j) / sigmamarg(i, j)
        senvb(i, j, 2) = -1 / meanload(i, j) / sigmamarg(i, j)
        senvb(i, j, 3) = covcap(i, j) / (1 + covcap(i, j) ^ 2) * (-meanmarg(i, j) - sigmamarg(i, j) ^ 2) / sigmamarg(i, j) ^ 3
        senvb(i, j, 4) = covload(i, j) / (1 + covload(i, j) ^ 2) * (sigmamarg(i, j) ^ 2 - meanmarg(i, j)) / sigmamarg(i, j) ^ 3
'sens vector of COVload
        If (covload(i, j) - wjcov) < 0.00001 Then
            pvswdcov = 0
            pvswjcov = 1
        Else
            pvswdcov = wdcov / covload(i, j)
            pvswjcov = wjcov / covload(i, j)
        End If

        If (j > 0 And pltype < 7) Or (j = 0 And deckbaybraces = "True") Then 'jackt bay or braced deck bay sensitivity

            'sens vector of meancap : pmrpmalpha=sumki, pmrpmlegf=1
            'sens vector of meanload: ploadwdbias(I), ploadwjbias(I,j)

            'sens vector of COVcap
            pvrsigmaalpha = sumksq(i, j) * sigmaalpha(i, j) / sigmacap(i, j) / lbcap(i, j)
            pvrsigmalegfh = covload(i, j) * legfh(i, j) / sigmacap(i, j) / lbcap(i, j)


            ' sens vector of sigmaalpha
            psabccov = dbpu(i, j, mltf#(i, j)) ^ 2 * bccov / sigmapu(i, j) / dbki(i, j, mltf#(i, j))
            psawjcov = (dblx(i, j, mltf#(i, j)) / 8) ^ 2 * dbw(j, mltf#(i, j)) ^ 2 * wjcov / dbki(i, j, mltf#(i, j)) / sigmapu(i, j)
            psabcbias = bccov * dbpu(i, j, mltf#(i, j)) / bcbias / dbki(i, j, mltf#(i, j))
            psawjbias = (dblx(i, j, mltf#(i, j)) / 8) ^ 2 * dbw(j, mltf#(i, j)) ^ 2 * wjcov ^ 2 / dbki(i, j, mltf#(i, j)) / sigmapu(i, j) / wjbias

            ' sens vector of sigmalegfh
            pslwdbias = covload(i, j) * plfhwdbias(i, j)
            pslwjbias = covload(i, j) * plfhwjbias(i, j)
            If (covload(i, j) - wjcov) < 0.00001 Then
                pslwdcov = 0
                pslwjcov = legfh(i, j)
            Else
                pslwdcov = legfh(i, j) * wdcov / covload(i, j)
                pslwjcov = legfh(i, j) * wjcov / covload(i, j)
            End If

            'total sens vector
            'sens vector w.r.t bcbias
            rbsensv(i, j, 1) = senvb(i, j, 1) * sumksq(i, j) ^ 0.5 * mltfalpha(i, j) / bcbias + senvb(i, j, 3) * pvrsigmaalpha * psabcbias
            'sens vector wrt wdbias
            rbsensv(i, j, 2) = senvb(i, j, 1) * plfhwdbias(i, j) + senvb(i, j, 2) * ploadwdbias(i) + senvb(i, j, 3) * pvrsigmalegfh * pslwdbias
            'sens vector wrt wjbias
            rbsensv(i, j, 3) = senvb(i, j, 1) * plfhwjbias(i, j) + senvb(i, j, 2) * ploadwjbias(i, j) + senvb(i, j, 3) * (pvrsigmalegfh * pslwjbias + pvrsigmaalpha * psawjb
ias)

            'sens vector wrt bccov
            rbsensv(i, j, 4) = senvb(i, j, 3) * pvrsigmaalpha * psabccov
            'sens vector wrt wdcov
            rbsensv(i, j, 5) = senvb(i, j, 3) * pvrsigmalegfh * pslwdcov + senvb(i, j, 4) * pvswdcov
            'sens vector wrt wjcov
            rbsensv(i, j, 6) = senvb(i, j, 3) * (pvrsigmalegfh * pslwjcov + pvrsigmaalpha * psawjcov) + senvb(i, j, 4) * pvswjcov
```

```
      ElseIf j = 0 And deckbaybraces = Flase Then '  deck portal sensitivity
        'wrt dmcrbias
        rbsensv(i, 0, 1) = senvb(i, j, 1) * prddmcrbias(i) + senvb(i, j, 3) * (-1 * covcap(i, 0) / lbcap(i, 0)) * prddmcrbias(i)
        'wrt dpcrlbias
        rbsensv(i, 0, 2) = senvb(i, j, 1) * prddpcrlbias(i) + senvb(i, j, 3) * (-1 * covcap(i, 0) / lbcap(i, 0)) * prddpcrlbias(i)
        'wrt wdbias
        rbsensv(i, 0, 3) = senvb(i, j, 2) * ploadwdbias(i)
        'wrt wjbias
        rbsensv(i, 0, 4) = senvb(i, j, 2) * ploadwjbias(i, 0)

        'wrt dmcrcov
        rbsensv(i, 0, 5) = senvb(i, j, 3) / lbcap(i, 0) * psmcov
        'wrt dpcrlcov
        rbsensv(i, 0, 6) = senvb(i, j, 3) / lbcap(i, 0) * pspcov
        'wrt wdcov
        rbsensv(i, 0, 7) = senvb(i, j, 4) * pvswdcov
        'wrt wjcov
        rbsensv(i, 0, 8) = senvb(i, j, 4) * pvswjcov
      Else  'caisson jacket bay sensitivity
        'caisson jacket cap bias
        rbsensv(i, j, 1) = senvb(i, j, 1) * lbcap(i, 1) / caissoncapbias
        'wdbias
        rbsensv(i, j, 2) = senvb(i, j, 2) * ploadwdbias(i)
        rbsensv(i, j, 3) = senvb(i, j, 2) * ploadwjbias(i, j)

        'caisson cap cov
        rbsensv(i, j, 4) = senvb(i, j, 3)
        rbsensv(i, j, 5) = senvb(i, j, 4) * pvswdcov
        rbsensv(i, j, 6) = senvb(i, j, 4) * pvswjcov

      End If

    End If
  Next j
'
' Now compute foundation safety indices
'
' First find coefficient of variation for horizontal capacity
'
    covlfcap(i) = ((pucov * pilehorzcap) ^ 2 + (covload(i, nbay) * 2 * legfh(i, nbay + 1)) ^ 2) ^ 0.5 / fcap(i)
    covafcap(i) = qcov

  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
    For j = 1 To 6
    rbsensv(i, nbay + 1, j) = 0
    Next j
  Else
    beta(i, nbay + 1) = meanmarg(i, nbay + 1) / sigmamarg(i, nbay + 1)
   ' sensitivity vector w.r.t beta, 1 - meancap, 2 - meanload,
      ' 3 - COV capacity, 4 - COV load,
    senvb(i, nbay + 1, 1) = 1 / fcap(i) / sigmamarg(i, nbay + 1)
    senvb(i, nbay + 1, 2) = -1 / meanload(i, nbay + 1) / sigmamarg(i, nbay + 1)
    senvb(i, nbay + 1, 3) = covlfcap(i) / (1 + covlfcap(i) ^ 2) * (-1 * meanmarg(i, nbay + 1) - sigmamarg(i, nbay + 1) ^ 2) / sigmamarg(i, nbay + 1) ^ 3
    senvb(i, nbay + 1, 4) = covload(i, nbay) / (1 + covload(i, nbay) ^ 2) * (sigmamarg(i, nbay + 1) ^ 2 - meanmarg(i, nbay + 1)) / sigmamarg(i, nbay + 1) ^ 3

    If verticalface = True Then
      batter = 1
    Else
      batter = 2
    End If

    plfcappubias = pilehorzcap / pubias
    plfcapwdbias = batter * plfhwdbias(i, nbay + 1)
    plfcapwjbias = batter * plfhwjbias(i, nbay + 1)
```

```
'sigmapilehorzcapu = pilehorzcap * pucov
sigmalfh = batter * legfh(i, nbay + 1) * covload(i, nbay)

If (covload(i, j) - wjcov) < 0.00001 Then
   pslwdcov = 0
   pslwjcov = legfh(i, nbay + 1)
Else
   pslwdcov = legfh(i, nbay + 1) * wdcov / covload(i, nbay)
   pslwjcov = legfh(i, nbay + 1) * wjcov / covload(i, nbay)
End If

pvrpucov = (pilehorzcap) ^ 2 * pucov / fcap(i) ^ 2 / covlfcap(i)
pvrwdcov = sigmalfh * pslwdcov / fcap(i) ^ 2 / covlfcap(i)
pvrwjcov = sigmalfh * pslwjcov / fcap(i) ^ 2 / covlfcap(i)

pvrpubias = 1 / fcap(i) ^ 2 / covlfcap(i) * pucov * pilehorzcap ^ 2 / pubias - 1 / fcap(i) * covlfcap(i) * plfcappubias
pvrwdbias = 1 / (fcap(i) ^ 2 * covlfcap(i)) * sigmalfh * plfhwdbias(i, nbay + 1) - 1 / fcap(i) * covlfcap(i) * plfcapwdbias
pvrwjbias = 1 / (fcap(i) ^ 2 * covlfcap(i)) * sigmalfh * plfhwjbias(i, nbay + 1) - 1 / fcap(i) * covlfcap(i) * plfcapwjbias

'total sense vector
rbsensv(i, nbay + 1, 1) = senvb(i, nbay + 1, 1) * plfcappubias + senvb(i, nbay + 1, 3) * pvrpubias
rbsensv(i, nbay + 1, 2) = senvb(i, nbay + 1, 1) * plfcapwdbias + senvb(i, nbay + 1, 3) * pvrwdbias + senvb(i, nbay + 1, 2) * ploadwdbias(i)
rbsensv(i, nbay + 1, 3) = senvb(i, nbay + 1, 1) * plfcapwjbias + senvb(i, nbay + 1, 3) * pvrwjbias + senvb(i, nbay + 1, 2) * ploadwjbias(i, nbay + 1)

rbsensv(i, nbay + 1, 4) = senvb(i, nbay + 1, 3) * pvrpucov
rbsensv(i, nbay + 1, 5) = senvb(i, nbay + 1, 3) * pvrwdcov + senvb(i, nbay + 1, 2) * pslwdcov / legfh(i, nbay + 1)
rbsensv(i, nbay + 1, 6) = senvb(i, nbay + 1, 3) * pvrwjcov + senvb(i, nbay + 1, 2) * pslwjcov / legfh(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

'sensitivity : foundation axial
' wrt rsr,  compression--29,  tension---30
senvb(i, 29, 1) = 1 / rsrc(i) / sigmamargaf(i)
senvb(i, 30, 1) = 1 / rsrt(i) / sigmamargaf(i)
' wrt capacity COV
senvb(i, 29, 2) = covafcap(i) / (1 + covafcap(i) ^ 2) * (-1 * meanmargacf(i) - sigmamargaf(i) ^ 2) / sigmamargaf(i) ^ 3
senvb(i, 30, 2) = covafcap(i) / (1 + covafcap(i) ^ 2) * (-1 * meanmargatf(i) - sigmamargaf(i) ^ 2) / sigmamargaf(i) ^ 3
'wrt load COV
senvb(i, 29, 3) = covload(i, nbay) / (1 + covload(i, nbay) ^ 2) * (-1 * meanmargacf(i) + sigmamargaf(i) ^ 2) / sigmamargaf(i) ^ 3
senvb(i, 30, 3) = covload(i, nbay) / (1 + covload(i, nbay) ^ 2) * (-1 * meanmargatf(i) + sigmamargaf(i) ^ 2) / sigmamargaf(i) ^ 3

If (covload(i, nbay) - wjcov) < 0.00001 Then
   pvswdcov = 0
   pvswjcov = 1
Else
   pvswdcov = wdcov / covload(i, nbay)
   pvswjcov = wjcov / covload(i, nbay)
End If

'total sense vector:  wrt qbias
```

```
  rbsensv(i, 29, 1) = senvb(i, 29, 1) / pileloadcomp(i) * avpilecapcomp(i) / qbias
  rbsensv(i, 30, 1) = senvb(i, 30, 1) / pileloadtens(i) * avpilecaptens(i) / qbias
  'wrt wdbias
  rbsensv(i, 29, 2) = senvb(i, 29, 1) * (-1 * rsrc(i)) / pileloadcomp(i) * pileacwdbias(i)
  rbsensv(i, 30, 2) = senvb(i, 30, 1) * (-1 * rsrt(i)) / pileloadtens(i) * pileatwdbias(i)
  'wrt wjbias
  rbsensv(i, 29, 3) = senvb(i, 29, 1) * (-1 * rsrc(i)) / pileloadcomp(i) * pileacwjbias(i)
  rbsensv(i, 30, 3) = senvb(i, 30, 1) * (-1 * rsrt(i)) / pileloadtens(i) * pileatwjbias(i)

  'wrt qcov
  rbsensv(i, 29, 4) = senvb(i, 29, 2)
  rbsensv(i, 30, 4) = senvb(i, 30, 2)
  rbsensv(i, 29, 5) = senvb(i, 29, 3) * pvswdcov
  rbsensv(i, 30, 5) = senvb(i, 30, 3) * pvswdcov
  rbsensv(i, 29, 6) = senvb(i, 29, 3) * pvswjcov
  rbsensv(i, 30, 6) = senvb(i, 30, 3) * pvswjcov

Next i

End Sub

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

Sub tableout()

'
'  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) = "Printed Data"

'
'  Global parameters of structure
'
Worksheets("Sheet3").Cells(1, 1) = "GLOBAL PARAMETERS OF PLATFORM"
Worksheets("Sheet3").Cells(3, 1) = "Session Name"
Worksheets("Sheet3").Cells(5, 1) = "Platform Type"
Worksheets("Sheet3").Cells(6, 1) = "Number of Decks"

Worksheets("Sheet3").Cells(3, 4) = Worksheets("Sheet2").Cells(2, 13)
If pltype = 1 Then
   Worksheets("Sheet3").Cells(5, 4) = "4-leg Jacket"
ElseIf pltype = 2 Then
   Worksheets("Sheet3").Cells(5, 4) = "6-leg Jacket"
ElseIf pltype = 3 Then
   Worksheets("Sheet3").Cells(5, 4) = "8-leg Jacket"
ElseIf pltype = 4 Then
   Worksheets("Sheet3").Cells(5, 4) = "12-leg Jacket"
ElseIf pltype = 5 Then
   Worksheets("Sheet3").Cells(5, 4) = "Tripod Jacket"
ElseIf pltype = 6 Then
   Worksheets("Sheet3").Cells(5, 4) = "Multi 4-leg Jacket"
ElseIf pltype = 7 Then
   Worksheets("Sheet3").Cells(5, 4) = "Caisson (Braced)"
Else
   Worksheets("Sheet3").Cells(5, 4) = "Caisson (Guyed)"
End If

Worksheets("Sheet3").Cells(6, 4) = ndeck

If pltype > 6 Then

Worksheets("Sheet3").Cells(7, 1) = "Water Depth (ft)"
Worksheets("Sheet3").Cells(7, 4) = wdep

Worksheets("Sheet3").Cells(9, 1) = "PLATFORM GEOMETRY"
Worksheets("Sheet3").Cells(11, 1) = "Distance between caisson and pile (ft)"
Worksheets("Sheet3").Cells(12, 1) = "Distance from mudline to support point (ft)"
Worksheets("Sheet3").Cells(13, 1) = "Distance from support point to deck (ft)"

Worksheets("Sheet3").Cells(11, 5) = bcw(1)
```

```
Worksheets("Sheet3").Cells(12, 5) = bayh(1)
Worksheets("Sheet3").Cells(13, 5) = bayh(0)

If pltype = 7 Then

   Worksheets("Sheet3").Cells(15, 1) = "BRACE SUPPORT"
   Worksheets("Sheet3").Cells(17, 1) = "D (in)"
   Worksheets("Sheet3").Cells(18, 1) = "t (in)"
   Worksheets("Sheet3").Cells(19, 1) = "Pu (kips)"

   Worksheets("Sheet3").Cells(21, 1) = "Connection Area (in^2)"
   Worksheets("Sheet3").Cells(22, 1) = "Connection Capacity (kips)"
   Worksheets("Sheet3").Cells(23, 1) = "Connection SCF"

   Worksheets("Sheet3").Cells(17, 3) = dbd(1, 1, 1)
   Worksheets("Sheet3").Cells(18, 3) = dbt(1, 1, 1)
   Worksheets("Sheet3").Cells(19, 3) = Application.Round(dbpu(1, 1, 1), 0)

   If dbtype(1, 1, 1) = 2 Then
      Worksheets("Sheet3").Cells(19, 5) = "(compression)"
   Else
      Worksheets("Sheet3").Cells(19, 5) = "(tension)"
   End If

   Worksheets("Sheet3").Cells(21, 4) = jchd(1)
   Worksheets("Sheet3").Cells(22, 4) = Application.Round(jchd(1) * jyield(1), 0)

   bump = 3

 Else

   Worksheets("Sheet3").Cells(15, 1) = "WIRE SUPPORT"
   Worksheets("Sheet3").Cells(17, 1) = "D (in)"
   Worksheets("Sheet3").Cells(18, 1) = "Pre-Tension (kips)"
   Worksheets("Sheet3").Cells(19, 1) = "Pu (kips)"

   Worksheets("Sheet3").Cells(21, 1) = "Connection Area (in^2)"
   Worksheets("Sheet3").Cells(22, 1) = "Connection Capacity (kips)"
   Worksheets("Sheet3").Cells(23, 1) = "Connection SCF"

   Worksheets("Sheet3").Cells(17, 4) = dbd(1, 1, 1)
   Worksheets("Sheet3").Cells(18, 4) = pretension
   Worksheets("Sheet3").Cells(19, 4) = Application.Round(dbfyspecific(1, 1, 1) * (dbd(1, 1, 1) / 2) ^ 2 * Pi - pretension, 0)

   Worksheets("Sheet3").Cells(21, 4) = jchd(1)
   Worksheets("Sheet3").Cells(22, 4) = Application.Round(jchd(1) * jyield(1), 0)

   bump = 3

 End If

 Else

 Worksheets("Sheet3").Cells(7, 1) = "Number of Jacket Bays"
 Worksheets("Sheet3").Cells(8, 1) = "Water Depth (ft)"

 Worksheets("Sheet3").Cells(7, 4) = nbay
 Worksheets("Sheet3").Cells(8, 4) = wdep

 Worksheets("Sheet3").Cells(10, 1) = "SPECIAL CHARACTERISTICS"
 bump = 0
 If deckbaybraces = True Then
    Worksheets("Sheet3").Cells(12, 1) = "    Bracing in Deck Bay"
    bump = bump + 1
 End If
 If verticalface = True Then
    If pltype = 5 Then
       verttext = "Leg"
    Else
       verttext = "Face"
    End If
    Worksheets("Sheet3").Cells(12 + bump, 1) = "    Vertical " & verttext
```

```
  bump = bump + 1
End If
If pgrout = True Then
  Worksheets("Sheet3").Cells(12 + bump, 1) = "   Pile-Leg Annulus is Grouted"
  bump = bump + 1
End If
If skirt = True Then
  Worksheets("Sheet3").Cells(12 + bump, 1) = "   Skirt Piles"
  bump = bump + 1
End If
If bump = 0 Then
  Worksheets("Sheet3").Cells(12, 1) = "   None"
  bump = 2
Else
  bump = bump + 1
End If

Worksheets("Sheet3").Cells(12 + bump, 1) = "PLATFORM GEOMETRY"
Worksheets("Sheet3").Cells(14 + bump, 1) = "Broadside Frames Top Width (ft)"
Worksheets("Sheet3").Cells(15 + bump, 1) = "Broadside Frames Base Width (ft)"
Worksheets("Sheet3").Cells(16 + bump, 1) = "End-On Frames Top Width (ft)"
Worksheets("Sheet3").Cells(17 + bump, 1) = "End-On Frames Base Width (ft)"

Worksheets("Sheet3").Cells(14 + bump, 5) = tcw(1)
Worksheets("Sheet3").Cells(15 + bump, 5) = bcw(1)
Worksheets("Sheet3").Cells(16 + bump, 5) = tcw(2)
Worksheets("Sheet3").Cells(17 + bump, 5) = bcw(2)

If pltype > 1 And pltype < 5 Then
  Worksheets("Sheet3").Cells(18 + bump, 1) = "End-On Frames Center Section Width (ft)"
  Worksheets("Sheet3").Cells(18 + bump, 5) = msw
  bump = bump + 1
End If

'
' Structural Layout
'
Worksheets("Sheet3").Cells(19 + bump, 1) = "STRUCTURAL LAYOUT"
Worksheets("Sheet3").Cells(22 + bump, 3) = "Height (ft)"
Worksheets("Sheet3").Cells(21 + bump, 4) = "Diagonals"
Worksheets("Sheet3").Cells(21 + bump, 5) = "Diagonals"
Worksheets("Sheet3").Cells(21 + bump, 6) = "Horizontals"
Worksheets("Sheet3").Cells(21 + bump, 7) = "Corner"
Worksheets("Sheet3").Cells(21 + bump, 8) = "Corner"
Worksheets("Sheet3").Cells(22 + bump, 7) = "Legs D (in)"
Worksheets("Sheet3").Cells(22 + bump, 8) = "Legs t (in)"
If pltype > 1 And pltype < 5 Then
  Worksheets("Sheet3").Cells(21 + bump, 9) = "Center"
  Worksheets("Sheet3").Cells(21 + bump, 10) = "Center"
  Worksheets("Sheet3").Cells(22 + bump, 9) = "Legs D (in)"
  Worksheets("Sheet3").Cells(22 + bump, 10) = "Legs t (in)"
  Worksheets("Sheet3").Cells(21 + bump, 11) = "Appurt."
  Worksheets("Sheet3").Cells(21 + bump, 12) = "Marine"
  Worksheets("Sheet3").Cells(22 + bump, 11) = "Sum D (ft)"
  Worksheets("Sheet3").Cells(22 + bump, 12) = "Growth (in)"
Else
  Worksheets("Sheet3").Cells(21 + bump, 9) = "Appurt."
  Worksheets("Sheet3").Cells(21 + bump, 10) = "Marine"
  Worksheets("Sheet3").Cells(22 + bump, 9) = "Sum D (ft)"
  Worksheets("Sheet3").Cells(22 + bump, 10) = "Growth (in)"
End If
Worksheets("Sheet3").Cells(22 + bump, 4) = "BS Frames"
Worksheets("Sheet3").Cells(22 + bump, 5) = "EO Frames"
Worksheets("Sheet3").Cells(22 + bump, 6) = "Bay Floor"

For i = 1 To nbay + 1
  If i = 1 Then
    Worksheets("Sheet3").Cells(22 + bump + i, 1) = "Deck Bay"
    Worksheets("Sheet3").Cells(22 + bump + i, 7) = dld(1)
    Worksheets("Sheet3").Cells(22 + bump + i, 8) = dlt(1)
    If deckbaybraces = True Then
      Worksheets("Sheet3").Cells(22 + bump + i, 4) = ndb(1, nbay + 1)
```

```
        Worksheets("Sheet3").Cells(22 + bump + i, 5) = ndb(2, nbay + 1)
      End If
    Else
      Worksheets("Sheet3").Cells(22 + bump + i, 1) = "Jacket Bay " & i - 1
      Worksheets("Sheet3").Cells(22 + bump + i, 4) = ndb(1, i - 1)
      Worksheets("Sheet3").Cells(22 + bump + i, 5) = ndb(2, i - 1)
      Worksheets("Sheet3").Cells(22 + bump + i, 7) = jld(1, i - 1)
      Worksheets("Sheet3").Cells(22 + bump + i, 8) = jlt(1, i - 1)
    End If
    Worksheets("Sheet3").Cells(22 + bump + i, 3) = bayh(i - 1)
    Worksheets("Sheet3").Cells(22 + bump + i, 6) = nhb(i)
    If pltype > 1 And pltype < 5 Then
      If i = 1 Then
        Worksheets("Sheet3").Cells(22 + bump + i, 9) = dld(2)
        Worksheets("Sheet3").Cells(22 + bump + i, 10) = dlt(2)
        Worksheets("Sheet3").Cells(22 + bump + i, 11) = dequapp(i - 1)
        Worksheets("Sheet3").Cells(22 + bump + i, 12) = mg(i - 1)
      Else
        Worksheets("Sheet3").Cells(22 + bump + i, 9) = jld(2, i - 1)
        Worksheets("Sheet3").Cells(22 + bump + i, 10) = jlt(2, i - 1)
        Worksheets("Sheet3").Cells(22 + bump + i, 11) = dequapp(i - 1)
        Worksheets("Sheet3").Cells(22 + bump + i, 12) = mg(i - 1)
      End If
    Else
      Worksheets("Sheet3").Cells(22 + bump + i, 9) = dequapp(i - 1)
      Worksheets("Sheet3").Cells(22 + bump + i, 10) = mg(i - 1)
    End If
Next i

bump = bump + nbay + 3

End If '  ends large structural input block

Worksheets("Sheet3").Cells(22 + bump, 1) = "PLATFORM DECKS"
Worksheets("Sheet3").Cells(24 + bump, 3) = "Bottom"
Worksheets("Sheet3").Cells(24 + bump, 4) = "Top"
Worksheets("Sheet3").Cells(24 + bump, 5) = "Broadside"
Worksheets("Sheet3").Cells(24 + bump, 6) = "End-On"
Worksheets("Sheet3").Cells(24 + bump, 7) = "Weight"
Worksheets("Sheet3").Cells(25 + bump, 3) = "Elev. (ft)"
Worksheets("Sheet3").Cells(25 + bump, 4) = "Elev. (ft)"
Worksheets("Sheet3").Cells(25 + bump, 5) = "Width (ft)"
Worksheets("Sheet3").Cells(25 + bump, 6) = "Width (ft)"
Worksheets("Sheet3").Cells(25 + bump, 7) = "(kips)"

For i = 1 To ndeck
  Worksheets("Sheet3").Cells(25 + bump + i, 1) = "Deck " & i
  Worksheets("Sheet3").Cells(25 + bump + i, 3) = uk(i)
  Worksheets("Sheet3").Cells(25 + bump + i, 4) = ok(i)
  Worksheets("Sheet3").Cells(25 + bump + i, 5) = deckw(1, i)
  Worksheets("Sheet3").Cells(25 + bump + i, 6) = deckw(2, i)
  Worksheets("Sheet3").Cells(25 + bump + i, 7) = Workbooks(2).Sheets(1).Cells(821 + i * 7, 8)
Next i

bump = bump + ndeck + 2
'
'  Boatlanding projected areas
'
Worksheets("Sheet3").Cells(25 + bump, 1) = "BOAT-LANDINGS"
Worksheets("Sheet3").Cells(27 + bump, 1) = "Projected Area, End-On (ft^2)"
Worksheets("Sheet3").Cells(28 + bump, 1) = "Projected Area, Broadside (ft^2)"
Worksheets("Sheet3").Cells(29 + bump, 1) = "Total Weight (Kips)"
Worksheets("Sheet3").Cells(27 + bump, 5) = boatl(2)
Worksheets("Sheet3").Cells(28 + bump, 5) = boatl(1)
Worksheets("Sheet3").Cells(29 + bump, 5) = boatlw

bump = bump + 1
'
'  Conductors
'
Worksheets("Sheet3").Cells(30 + bump, 1) = "CONDUCTORS"
Worksheets("Sheet3").Cells(32 + bump, 1) = "Total Number"
```

```
Worksheets("Sheet3").Cells(33 + bump, 1) = "D (in)"
Worksheets("Sheet3").Cells(34 + bump, 1) = "Penetration (ft)"
Worksheets("Sheet3").Cells(35 + bump, 1) = "Fixity Above Mudline (ft)"
Worksheets("Sheet3").Cells(36 + bump, 1) = "Weight (kips / ft)"
Worksheets("Sheet3").Cells(37 + bump, 1) = "Plastic Moment (kip-ft)"
Worksheets("Sheet3").Cells(38 + bump, 1) = "Moment of Inertia (ft^4)"
Worksheets("Sheet3").Cells(39 + bump, 1) = "Group Strength Reduction (%)"
Worksheets("Sheet3").Cells(40 + bump, 1) = "Group Stiffness Reduction (%)"

Worksheets("Sheet3").Cells(32 + bump, 4) = numconductors
Worksheets("Sheet3").Cells(33 + bump, 4) = diaconductors
Worksheets("Sheet3").Cells(34 + bump, 4) = penconductors
Worksheets("Sheet3").Cells(35 + bump, 4) = conductorfix
Worksheets("Sheet3").Cells(36 + bump, 4) = wconductors * 1000
Worksheets("Sheet3").Cells(37 + bump, 4) = mpconductors
Worksheets("Sheet3").Cells(38 + bump, 4) = Iconductors
Worksheets("Sheet3").Cells(39 + bump, 4) = groupstr
Worksheets("Sheet3").Cells(40 + bump, 4) = groupstiff

bump = bump + 12
'
'  Platform tonnage estimate.
'
Worksheets("Sheet3").Cells(30 + bump, 1) = "PLATFORM TONNAGE ESTIMATE"
Worksheets("Sheet3").Cells(32 + bump, 1) = "Deck Section"
Worksheets("Sheet3").Cells(33 + bump, 1) = "Jacket"
Worksheets("Sheet3").Cells(34 + bump, 1) = "Piles"
Worksheets("Sheet3").Cells(36 + bump, 1) = "TOTAL"


Worksheets("Sheet3").Cells(32 + bump, 3) = Application.Round(decklegsw + qdeck, 0) & " kips"
Worksheets("Sheet3").Cells(33 + bump, 3) = Application.Round(jacketw, 0) & " kips"
Worksheets("Sheet3").Cells(34 + bump, 3) = Application.Round(pilew, 0) & " kips"
Worksheets("Sheet3").Cells(36 + bump, 3) = Application.Round(steelw + qdeck, 0) & " kips"
'
'  Global Material Parameters
'
Worksheets("Sheet3").Cells(38 + bump, 1) = "GLOBAL MATERIAL PARAMETERS"
Worksheets("Sheet3").Cells(40 + bump, 1) = "Steel Yield Stress (ksi)"
Worksheets("Sheet3").Cells(41 + bump, 1) = "Elastic Modulus (ksi)"
Worksheets("Sheet3").Cells(42 + bump, 1) = "Brace Effective Length Factor, k"
Worksheets("Sheet3").Cells(43 + bump, 1) = "Brace Post-Buckling Strength Factor"


Worksheets("Sheet3").Cells(40 + bump, 5) = fy
Worksheets("Sheet3").Cells(41 + bump, 5) = e
Worksheets("Sheet3").Cells(42 + bump, 5) = kbuck
Worksheets("Sheet3").Cells(43 + bump, 5) = bres

'
'  Biases and Uncertainties
'

Worksheets("Sheet3").Cells(45 + bump, 1) = "BIASES AND UNCERTAINTIES"
Worksheets("Sheet3").Cells(47 + bump, 1) = "Structural:"
Worksheets("Sheet3").Cells(49 + bump, 4) = "Bias"
Worksheets("Sheet3").Cells(49 + bump, 5) = "COV"

Worksheets("Sheet3").Cells(50 + bump, 1) = "Main Diagonal Strength"
Worksheets("Sheet3").Cells(51 + bump, 1) = "Tubular Joint Strength"
Worksheets("Sheet3").Cells(52 + bump, 1) = "Pile Axial Capacity"
Worksheets("Sheet3").Cells(53 + bump, 1) = "Pile Lateral Capacity"
Worksheets("Sheet3").Cells(54 + bump, 1) = "Pile Axial Stiffness"
Worksheets("Sheet3").Cells(55 + bump, 1) = "Pile Lateral Stiffness"

Worksheets("Sheet3").Cells(50 + bump, 4) = bcbias
Worksheets("Sheet3").Cells(51 + bump, 4) = jtbias
Worksheets("Sheet3").Cells(52 + bump, 4) = qbias
Worksheets("Sheet3").Cells(53 + bump, 4) = pubias
Worksheets("Sheet3").Cells(54 + bump, 4) = axialkbias
Worksheets("Sheet3").Cells(55 + bump, 4) = horizkbias
```

```
Worksheets("Sheet3").Cells(50 + bump, 5) = bccov
Worksheets("Sheet3").Cells(51 + bump, 5) = jtcov
Worksheets("Sheet3").Cells(52 + bump, 5) = qcov
Worksheets("Sheet3").Cells(53 + bump, 5) = pucov
Worksheets("Sheet3").Cells(54 + bump, 5) = axialkcov
Worksheets("Sheet3").Cells(55 + bump, 5) = horizkcov

Worksheets("Sheet3").Cells(57 + bump, 1) = "Load:"
Worksheets("Sheet3").Cells(59 + bump, 4) = "Bias"
Worksheets("Sheet3").Cells(59 + bump, 5) = "COV"

Worksheets("Sheet3").Cells(60 + bump, 1) = "Wave-in-Deck Force"
Worksheets("Sheet3").Cells(61 + bump, 1) = "Wave Force on Jacket"
Worksheets("Sheet3").Cells(62 + bump, 1) = "Earthquake Spectral Acceleration"

Worksheets("Sheet3").Cells(60 + bump, 4) = wdbias
Worksheets("Sheet3").Cells(61 + bump, 4) = wjbias
Worksheets("Sheet3").Cells(62 + bump, 4) = eqbias

Worksheets("Sheet3").Cells(60 + bump, 5) = wdcov
Worksheets("Sheet3").Cells(61 + bump, 5) = wjcov
Worksheets("Sheet3").Cells(62 + bump, 5) = eqcov

If pltype < 7 Then
'
' Member parameters and other information
'
' Main diagonals
'
Worksheets("Sheet6").Cells(1, 4) = "LOCAL PARAMETERS"
Worksheets("Sheet6").Cells(2, 4) = "Broadside Main Diagonals"
Worksheets("Sheet7").Cells(1, 4) = "LOCAL PARAMETERS"
Worksheets("Sheet7").Cells(2, 4) = "End-On Main Diagonals"

If deckbaybraces = True Then nbay = nbay + 1

For i = 1 To 2
  If i = 1 Then
    whichsheet = "Sheet6"
  Else
    whichsheet = "Sheet7"
  End If
  sumdiag = 0
  For j = 1 To nbay
    If deckbaybraces = True And j = nbay Then
      Worksheets(whichsheet).Cells(5 + sumdiag, 1) = "Deck Bay"
    Else
      Worksheets(whichsheet).Cells(5 + sumdiag, 1) = "Jacket Bay " & j
    End If
    Worksheets(whichsheet).Cells(6 + sumdiag, 1) = "Brace #"
    Worksheets(whichsheet).Cells(6 + sumdiag, 2) = "D (in)"
    Worksheets(whichsheet).Cells(6 + sumdiag, 3) = "t (in)"
    Worksheets(whichsheet).Cells(5 + sumdiag, 4) = "Bay"
    Worksheets(whichsheet).Cells(5 + sumdiag, 5) = "Load"
    Worksheets(whichsheet).Cells(5 + sumdiag, 6) = "Bracing"
    Worksheets(whichsheet).Cells(5 + sumdiag, 7) = "Joint i"
    Worksheets(whichsheet).Cells(5 + sumdiag, 8) = "Joint j"
    Worksheets(whichsheet).Cells(5 + sumdiag, 10) = "Dent"
    Worksheets(whichsheet).Cells(5 + sumdiag, 11) = "Out-of-"
    Worksheets(whichsheet).Cells(5 + sumdiag, 12) = "Pu"
    Worksheets(whichsheet).Cells(6 + sumdiag, 4) = "Position"
    Worksheets(whichsheet).Cells(6 + sumdiag, 5) = "Type"
    Worksheets(whichsheet).Cells(6 + sumdiag, 6) = "Pattern"
    Worksheets(whichsheet).Cells(6 + sumdiag, 7) = "Type #"
    Worksheets(whichsheet).Cells(6 + sumdiag, 8) = "Type #"
    Worksheets(whichsheet).Cells(6 + sumdiag, 9) = "Condition"
    Worksheets(whichsheet).Cells(6 + sumdiag, 10) = "Depth (in)"
    Worksheets(whichsheet).Cells(6 + sumdiag, 11) = "Straight (in)"
    Worksheets(whichsheet).Cells(6 + sumdiag, 12) = "(kips)"
    For k = 1 To ndb(i, j)
      Worksheets(whichsheet).Cells(6 + k + sumdiag, 1) = k
      Worksheets(whichsheet).Cells(6 + k + sumdiag, 2) = dbd(i, j, k)
```

```
            Worksheets(whichsheet).Cells(6 + k + sumdiag, 3) = dbt(i, j, k)
            If dbpos(i, j, k) = 1 Then
               Worksheets(whichsheet).Cells(6 + k + sumdiag, 4) = "left"
            ElseIf dbpos(i, j, k) = 2 Then
               Worksheets(whichsheet).Cells(6 + k + sumdiag, 4) = "center"
            Else
               Worksheets(whichsheet).Cells(6 + k + sumdiag, 4) = "right"
            End If
            If dbtype(i, j, k) = 1 Then
               Worksheets(whichsheet).Cells(6 + k + sumdiag, 5) = "tens."
            Else
               Worksheets(whichsheet).Cells(6 + k + sumdiag, 5) = "comp."
            End If
            If dbconf(i, j, k) = 1 Then
               Worksheets(whichsheet).Cells(6 + k + sumdiag, 6) = "S"
            ElseIf dbconf(i, j, k) = 2 Then
               Worksheets(whichsheet).Cells(6 + k + sumdiag, 6) = "K (V)"
            ElseIf dbconf(i, j, k) = 3 Then
               Worksheets(whichsheet).Cells(6 + k + sumdiag, 6) = "X"
            Else
               Worksheets(whichsheet).Cells(6 + k + sumdiag, 6) = "K (A)"
            End If
            Worksheets(whichsheet).Cells(6 + k + sumdiag, 7) = dbjointi(i, j, k)
            Worksheets(whichsheet).Cells(6 + k + sumdiag, 8) = dbjointj(i, j, k)
            If dbcond(i, j, k) = 1 Then
               Worksheets(whichsheet).Cells(6 + k + sumdiag, 9) = "intact"
            ElseIf dbcond(i, j, k) = 2 Then
               Worksheets(whichsheet).Cells(6 + k + sumdiag, 9) = "damaged"
            Else
               Worksheets(whichsheet).Cells(6 + k + sumdiag, 9) = "grouted"
            End If
            Worksheets(whichsheet).Cells(6 + k + sumdiag, 10) = ddep(i, j, k)
            Worksheets(whichsheet).Cells(6 + k + sumdiag, 11) = oos(i, j, k)
            Worksheets(whichsheet).Cells(6 + k + sumdiag, 12) = Application.Round(dbpu(i, j, k), 0)
         Next k
         sumdiag = sumdiag + 4 + ndb(i, j)
      Next j
   Next i

If deckbaybraces = True Then nbay = nbay - 1
'
'  Horizontal braces
'
Worksheets("Sheet8").Cells(1, 3) = "LOCAL PARAMETERS"
Worksheets("Sheet8").Cells(2, 3) = "Horizontal Braces"
sumhorz = 0
For i = 1 To nbay + 1
   Worksheets("Sheet8").Cells(5 + sumhorz, 2) = "Horizontal Frame " & i
   If nhb(i) = 0 Then
      Worksheets("Sheet8").Cells(6 + sumhorz, 1) = "No Braces"
   Else
      Worksheets("Sheet8").Cells(6 + sumhorz, 1) = "Brace #"
      Worksheets("Sheet8").Cells(6 + sumhorz, 2) = "D (in)"
      Worksheets("Sheet8").Cells(6 + sumhorz, 3) = "t (in)"
      Worksheets("Sheet8").Cells(6 + sumhorz, 4) = "L (ft)"
      Worksheets("Sheet8").Cells(6 + sumhorz, 5) = "Angle w/BS"
      For j = 1 To nhb(i)
         Worksheets("Sheet8").Cells(6 + j + sumhorz, 1) = j
         Worksheets("Sheet8").Cells(6 + j + sumhorz, 2) = hbd(i, j)
         Worksheets("Sheet8").Cells(6 + j + sumhorz, 3) = hbt(i, j)
         Worksheets("Sheet8").Cells(6 + j + sumhorz, 4) = hbl(i, j)
         Worksheets("Sheet8").Cells(6 + j + sumhorz, 5) = hbang(i, j)
      Next j
   End If
   sumhorz = sumhorz + 4 + nhb(i)
Next i
'
'  Tubular joints
'
Worksheets("Sheet9").Cells(1, 3) = "LOCAL PARAMETERS"
Worksheets("Sheet9").Cells(2, 3) = "Tubular Joints"
Worksheets("Sheet9").Cells(6, 1) = "Joint #"
```

```
Worksheets("Sheet9").Cells(6, 2) = "Type"
Worksheets("Sheet9").Cells(6, 3) = "Grouted?"
Worksheets("Sheet9").Cells(5, 4) = "Chord"
Worksheets("Sheet9").Cells(5, 5) = "Chord"
Worksheets("Sheet9").Cells(5, 6) = "Branch"
Worksheets("Sheet9").Cells(5, 7) = "Gap in K"
Worksheets("Sheet9").Cells(5, 8) = "Angle"
Worksheets("Sheet9").Cells(5, 9) = "+Pu"
Worksheets("Sheet9").Cells(5, 10) = "-Pu"
Worksheets("Sheet9").Cells(6, 4) = "D (in)"
Worksheets("Sheet9").Cells(6, 5) = "t (in)"
Worksheets("Sheet9").Cells(6, 6) = "D (in)"
Worksheets("Sheet9").Cells(6, 7) = "(in)"
Worksheets("Sheet9").Cells(6, 8) = "(degrees)"
Worksheets("Sheet9").Cells(6, 9) = "(kips)"
Worksheets("Sheet9").Cells(6, 10) = "(kips)"
For i = 1 To njoint
  Worksheets("Sheet9").Cells(6 + i, 1) = i
  If jtype(i) = 1 Then
    Worksheets("Sheet9").Cells(6 + i, 2) = "K"
  ElseIf jtype(i) = 2 Then
    Worksheets("Sheet9").Cells(6 + i, 2) = "Y"
  Else
    Worksheets("Sheet9").Cells(6 + i, 2) = "X"
  End If
  If jgrout(i) = True Then
    Worksheets("Sheet9").Cells(6 + i, 3) = "yes"
  Else
    Worksheets("Sheet9").Cells(6 + i, 3) = "no"
  End If
  Worksheets("Sheet9").Cells(6 + i, 4) = jchd(i)
  Worksheets("Sheet9").Cells(6 + i, 5) = jcht(i)
  Worksheets("Sheet9").Cells(6 + i, 6) = jbrd(i)
  Worksheets("Sheet9").Cells(6 + i, 7) = jgap(i)
  Worksheets("Sheet9").Cells(6 + i, 8) = jang(i)
  Worksheets("Sheet9").Cells(6 + i, 9) = Application.Round(jput(i), 0)
  Worksheets("Sheet9").Cells(6 + i, 10) = Application.Round(jpuc(i), 0)
Next i

End If '  end local parameters block
'
' Soil properties
'
Worksheets("Sheet13").Cells(1, 1) = "FOUNDATION"

Worksheets("Sheet13").Cells(3, 1) = "SOIL PROPERTIES"

Worksheets("Sheet13").Cells(5, 1) = "Number of Soil Layers: " & nsoillayer

bump2 = 0

For i = 1 To nsoillayer
bump2 = bump2 + 5 * (i - 1)
If stype(i) = 1 Then
  Worksheets("Sheet13").Cells(7 + bump2, 1) = "Layer " & i & ": Sand"
  Worksheets("Sheet13").Cells(9 + bump2, 1) = "    Effective Angle of Internal Friction (degrees)"
  Worksheets("Sheet13").Cells(10 + bump2, 1) = "    Submerged Unit Weight of Soil (lbs / ft^3)"
  Worksheets("Sheet13").Cells(9 + bump2, 6) = sphi(i)
  Worksheets("Sheet13").Cells(10 + bump2, 6) = Application.Round(gammas(i) * 1000, 0)
Else
  Worksheets("Sheet13").Cells(7 + bump2, 1) = "Layer " & i & ": Clay"
  Worksheets("Sheet13").Cells(9 + bump2, 1) = "    Undrained Shear Strength, Mudline (kip / ft^2)"
  Worksheets("Sheet13").Cells(10 + bump2, 1) = "    Undrained Shear Strength, Pile Tips (kip / ft^2)"
  Worksheets("Sheet13").Cells(11 + bump2, 1) = "    Submerged Unit Weight of Soil (lbs / ft^3)"
  Worksheets("Sheet13").Cells(9 + bump2, 6) = su1(i)
  Worksheets("Sheet13").Cells(10 + bump2, 6) = su2(i)
  Worksheets("Sheet13").Cells(11 + bump2, 6) = Application.Round(gammas(i) * 1000, 0)
  bump2 = bump2 + 1
End If

Next i
```

```
bump2 = bump2 + 2

Worksheets("Sheet13").Cells(10 + bump2, 1) = "Scour Depth (ft)"
Worksheets("Sheet13").Cells(10 + bump2, 6) = scour
'
' Foundation piles
'
Worksheets("Sheet13").Cells(12 + bump2, 1) = "PILES"
If pltype > 6 Then
  Worksheets("Sheet13").Cells(14 + bump2, 1) = "Caisson"
Else
  Worksheets("Sheet13").Cells(14 + bump2, 1) = "Main Piles: Corner"
End If
Worksheets("Sheet13").Cells(16 + bump2, 1) = "    D (in)"
Worksheets("Sheet13").Cells(17 + bump2, 1) = "    t (in)"
Worksheets("Sheet13").Cells(18 + bump2, 1) = "    L (ft)"
Worksheets("Sheet13").Cells(19 + bump2, 1) = "    Plugged?"

Worksheets("Sheet13").Cells(16 + bump2, 3) = piled(1)
Worksheets("Sheet13").Cells(17 + bump2, 3) = pilet(1)
Worksheets("Sheet13").Cells(18 + bump2, 3) = pilel(1)

Worksheets("Sheet13").Cells(16 + bump2, 5) = "Lateral Capacity (kips)"
Worksheets("Sheet13").Cells(17 + bump2, 5) = "Axial Capacity, Tension (kips)"
Worksheets("Sheet13").Cells(18 + bump2, 5) = "Axial Capacity, Compression (kips)"
Worksheets("Sheet13").Cells(16 + bump2, 9) = Application.Round(pilecaplat(1), 0)
Worksheets("Sheet13").Cells(17 + bump2, 9) = Application.Round(pilecaptens(1), 0)
Worksheets("Sheet13").Cells(18 + bump2, 9) = Application.Round(pilecapcomp(1), 0)

Worksheets("Sheet13").Cells(19 + bump2, 5) = "Lateral Pilehead Stiffness (kips / in)"
Worksheets("Sheet13").Cells(20 + bump2, 5) = "Axial Pilehead Stiffness (kips / in)"
Worksheets("Sheet13").Cells(19 + bump2, 9) = Application.Round(pileheadkx(1) / 12, 0)
Worksheets("Sheet13").Cells(20 + bump2, 9) = Application.Round(pileheadkz(1) / 12, 0)

If plug(1) = True Then
  Worksheets("Sheet13").Cells(19 + bump2, 3) = "yes"
Else
  Worksheets("Sheet13").Cells(19 + bump2, 3) = "no"
End If

If (pltype > 1 And pltype < 5) Or pltype > 6 Then

If pltype > 6 Then
  Worksheets("Sheet13").Cells(21 + bump2, 1) = "Pile"
Else
  Worksheets("Sheet13").Cells(21 + bump2, 1) = "Main Piles: Center"
End If
Worksheets("Sheet13").Cells(23 + bump2, 1) = "    D (in)"
Worksheets("Sheet13").Cells(24 + bump2, 1) = "    t (in)"
Worksheets("Sheet13").Cells(25 + bump2, 1) = "    L (ft)"
Worksheets("Sheet13").Cells(26 + bump2, 1) = "    Plugged?"

Worksheets("Sheet13").Cells(23 + bump2, 3) = piled(2)
Worksheets("Sheet13").Cells(24 + bump2, 3) = pilet(2)
Worksheets("Sheet13").Cells(25 + bump2, 3) = pilel(2)

Worksheets("Sheet13").Cells(23 + bump2, 5) = "Lateral Capacity (kips)"
Worksheets("Sheet13").Cells(24 + bump2, 5) = "Axial Capacity, Tension (kips)"
Worksheets("Sheet13").Cells(25 + bump2, 5) = "Axial Capacity, Compression (kips)"
Worksheets("Sheet13").Cells(23 + bump2, 9) = Application.Round(pilecaplat(2), 0)
Worksheets("Sheet13").Cells(24 + bump2, 9) = Application.Round(pilecaptens(2), 0)
Worksheets("Sheet13").Cells(25 + bump2, 9) = Application.Round(pilecapcomp(2), 0)

Worksheets("Sheet13").Cells(26 + bump2, 5) = "Lateral Pilehead Stiffness (kips / in)"
Worksheets("Sheet13").Cells(27 + bump2, 5) = "Axial Pilehead Stiffness (kips / in)"
Worksheets("Sheet13").Cells(26 + bump2, 9) = Application.Round(pileheadkx(2) / 12, 0)
Worksheets("Sheet13").Cells(27 + bump2, 9) = Application.Round(pileheadkz(2) / 12, 0)


If plug(2) = True Then
  Worksheets("Sheet13").Cells(26 + bump2, 3) = "yes"
Else
```

```
   Worksheets("Sheet13").Cells(26 + bump2, 3) = "no"
End If

bump2 = bump2 + 7

End If

If skirt = True Then

Worksheets("Sheet13").Cells(21 + bump2, 1) = "Skirt Piles"
Worksheets("Sheet13").Cells(23 + bump2, 1) = "    D (in)"
Worksheets("Sheet13").Cells(24 + bump2, 1) = "    t (in)"
Worksheets("Sheet13").Cells(25 + bump2, 1) = "    L (ft)"
Worksheets("Sheet13").Cells(26 + bump2, 1) = "    Plugged?"

Worksheets("Sheet13").Cells(23 + bump2, 3) = piled(3)
Worksheets("Sheet13").Cells(24 + bump2, 3) = pilet(3)
Worksheets("Sheet13").Cells(25 + bump2, 3) = pilel(3)

Worksheets("Sheet13").Cells(23 + bump2, 5) = "Lateral Capacity (kips)"
Worksheets("Sheet13").Cells(24 + bump2, 5) = "Axial Capacity, Tension (kips)"
Worksheets("Sheet13").Cells(25 + bump2, 5) = "Axial Capacity, Compression (kips)"
Worksheets("Sheet13").Cells(23 + bump2, 9) = Application.Round(pilecaplat(3), 0)
Worksheets("Sheet13").Cells(24 + bump2, 9) = Application.Round(pilecaptens(3), 0)
Worksheets("Sheet13").Cells(25 + bump2, 9) = Application.Round(pilecapcomp(3), 0)

Worksheets("Sheet13").Cells(26 + bump2, 5) = "Lateral Pilehead Stiffness (kips / in)"
Worksheets("Sheet13").Cells(27 + bump2, 5) = "Axial Pilehead Stiffness (kips / in)"
Worksheets("Sheet13").Cells(26 + bump2, 9) = Application.Round(pileheadkx(3) / 12, 0)
Worksheets("Sheet13").Cells(27 + bump2, 9) = Application.Round(pileheadkz(3) / 12, 0)

   If plug(3) = True Then
      Worksheets("Sheet13").Cells(26 + bump2, 3) = "yes"
   Else
      Worksheets("Sheet13").Cells(26 + bump2, 3) = "no"
   End If

bump2 = bump2 + 7

End If

bump2 = bump2 + 1
Worksheets("Sheet13").Cells(21 + bump2, 1) = "NOTE: Pile Self-Weight Has Been Deducted From Axial Capacities"
bump2 = bump2 + 2

If skirt = True Then

   Worksheets("Sheet13").Cells(21 + bump2, 1) = "Numbers of Skirt Piles"
   Worksheets("Sheet13").Cells(22 + bump2, 1) = "    At Corners"
   Worksheets("Sheet13").Cells(23 + bump2, 1) = "    In End-On Frames"
   Worksheets("Sheet13").Cells(24 + bump2, 1) = "    In Broadside Frames"

   Worksheets("Sheet13").Cells(22 + bump2, 4) = nskirtcorner
   Worksheets("Sheet13").Cells(23 + bump2, 4) = nskirteo
   Worksheets("Sheet13").Cells(24 + bump2, 4) = nskirtbs

   bump2 = bump2 + 5

End If

'
' Include conductor data
'

If diaconductors > 0 Then

Worksheets("Sheet13").Cells(21 + bump2, 1) = "Conductors"
Worksheets("Sheet13").Cells(23 + bump2, 1) = "    Number"
Worksheets("Sheet13").Cells(24 + bump2, 1) = "    D (in)"
Worksheets("Sheet13").Cells(25 + bump2, 1) = "    w (lbs/ft)"
Worksheets("Sheet13").Cells(26 + bump2, 1) = "    L (ft)"
```

```
Worksheets("Sheet13").Cells(23 + bump2, 3) = numconductors
Worksheets("Sheet13").Cells(24 + bump2, 3) = diaconductors
Worksheets("Sheet13").Cells(25 + bump2, 3) = wconductors * 1000
Worksheets("Sheet13").Cells(26 + bump2, 3) = penconductors

Worksheets("Sheet13").Cells(23 + bump2, 5) = "Lateral Capacity (kips)"
Worksheets("Sheet13").Cells(24 + bump2, 5) = "Group Strength Factor"
Worksheets("Sheet13").Cells(25 + bump2, 5) = "Total Conductor Capacity (kips)"
Worksheets("Sheet13").Cells(23 + bump2, 9) = Application.Round(conductorcap, 0)
Worksheets("Sheet13").Cells(24 + bump2, 9) = groupstr
Worksheets("Sheet13").Cells(25 + bump2, 9) = Application.Round(numconductors * conductorcap * groupstr, 0)

Worksheets("Sheet13").Cells(26 + bump2, 5) = "Lateral Stiffness (kips / in)"
Worksheets("Sheet13").Cells(27 + bump2, 5) = "Group Stiffness Factor"
Worksheets("Sheet13").Cells(26 + bump2, 9) = Application.Round(conductorstiff / 12, 0)
Worksheets("Sheet13").Cells(27 + bump2, 9) = groupstiff

bump2 = bump2 + 8

  If includeconductors = True Then
    Worksheets("Sheet13").Cells(21 + bump2, 1) = "NOTE: Conductor Strength And Stiffness Have Been Included In Analysis"
    Worksheets("Sheet13").Cells(22 + bump2, 1) = "     Conductor Framing Should Be Checked to Ensure Sufficient Strength Exists to Transfer Load"
    bump2 = bump2 + 3
  Else
    Worksheets("Sheet13").Cells(21 + bump2, 1) = "NOTE: Conductor Strength And Stiffness Have Not Been Included In Analysis"
    bump2 = bump2 + 2
  End If

End If

'
' Include mat and mudline element data
'
If pltype < 7 Then
  Worksheets("Sheet13").Cells(21 + bump2, 1) = "Mud Mats and Mudline Braces"
  Worksheets("Sheet13").Cells(23 + bump2, 1) = "Contact Areas (ft^2)"
  Worksheets("Sheet13").Cells(25 + bump2, 1) = "    Mudline Braces (total area)"
  Worksheets("Sheet13").Cells(26 + bump2, 1) = "    Corner Mud Mats (each mat)"
  Worksheets("Sheet13").Cells(25 + bump2, 5) = Application.Round(mudlinebraceA, 0)
  Worksheets("Sheet13").Cells(26 + bump2, 5) = cornermat
  If nleg > 4 Then
    Worksheets("Sheet13").Cells(27 + bump2, 1) = "    Center Mud Mats (each mat)"
    Worksheets("Sheet13").Cells(27 + bump2, 5) = centermat
    bump2 = bump2 + 1
  End If
  Worksheets("Sheet13").Cells(29 + bump2, 3) = "Bearing"
  Worksheets("Sheet13").Cells(29 + bump2, 4) = "Sliding"
  Worksheets("Sheet13").Cells(30 + bump2, 1) = "Strength Factor"
  Worksheets("Sheet13").Cells(31 + bump2, 1) = "Stiffness Factor"
  Worksheets("Sheet13").Cells(30 + bump2, 3) = bearingstr
  Worksheets("Sheet13").Cells(30 + bump2, 4) = slidingstr
  Worksheets("Sheet13").Cells(31 + bump2, 3) = bearingstiff
  Worksheets("Sheet13").Cells(31 + bump2, 4) = slidingstiff

  Worksheets("Sheet13").Cells(33 + bump2, 1) = "Contact Surface Loads (kip / ft^2)"
  Worksheets("Sheet13").Cells(35 + bump2, 1) = "    Max Bearing"
  Worksheets("Sheet13").Cells(36 + bump2, 1) = "    Max Sliding"
  Worksheets("Sheet13").Cells(35 + bump2, 4) = Application.Round(bearingstress, 2)
  Worksheets("Sheet13").Cells(36 + bump2, 4) = Application.Round(slidingstress, 2)

  bump2 = bump2 + 17
  If mudmats = True Then
    Worksheets("Sheet13").Cells(21 + bump2, 1) = "Mats And Mudline Braces, And Elements Supporting These Components,"
    Worksheets("Sheet13").Cells(22 + bump2, 1) = "Should Be Checked To Ensure The Required Surface Loads Can Be Carried."
    bump2 = bump2 + 3
  End If

End If

Worksheets("Sheet3").Cells(1, 10) = Worksheets("Sheet2").Cells(2, 13)

If pltype > 6 Then
```

```
    Worksheets("Sheet12").Cells(1, 10) = Worksheets("Sheet2").Cells(2, 13)
    Worksheets("Sheet14").Cells(1, 10) = Worksheets("Sheet2").Cells(2, 13)
    Worksheets("Sheet13").Cells(1, 10) = Worksheets("Sheet2").Cells(2, 13)
Else
    For i = 5 To 14
        Worksheets("Sheet" & i).Cells(1, 10) = Worksheets("Sheet2").Cells(2, 13)
    Next i
End If

' Reliability Sensetivity Analysis Output

Worksheets("Sheet15").Cells(1, 1) = "Reliability Sensitiviy Vectors of Safety Indexes with respect to Biases and COV's"
Worksheets("Sheet15").Cells(3, 1) = "Jacket Bay"
Worksheets("Sheet15").Cells(3, 3) = "Broad Side"
Worksheets("Sheet15").Cells(3, 9) = "End On"

' jacket bay
Worksheets("Sheet15").Cells(4, 1) = "Bay #"

For j = 1 To nbay

    Worksheets("Sheet15").Cells(4 + j, 1) = j
    For i = 1 To 2      '1 for Bs and 2 for Eo

        Worksheets("Sheet15").Cells(4, 3 + (i - 1) * 6) = "Bcbias"
        Worksheets("Sheet15").Cells(4, 4 + (i - 1) * 6) = "Wdbias"
        Worksheets("Sheet15").Cells(4, 5 + (i - 1) * 6) = "Wjbias"
        Worksheets("Sheet15").Cells(4, 6 + (i - 1) * 6) = "Bccov"
        Worksheets("Sheet15").Cells(4, 7 + (i - 1) * 6) = "Wdcov"
        Worksheets("Sheet15").Cells(4, 8 + (i - 1) * 6) = "Wjcov"

        If j = 1 And pltype = 7 Or pltype = 8 Then 'exception of caisson
            Worksheets("Sheet15").Cells(4, 3 + (i - 1) * 6) = "Caissoncapbias"
            Worksheets("Sheet15").Cells(4, 6 + (i - 1) * 6) = "Caissoncapcov"
        End If

        For k = 1 To 6
            Worksheets("Sheet15").Cells(4 + j, k + 2 + (i - 1) * 6) = Application.Round(rbsensv(i, j, k) * 10000, 0) / 10000
            'Worksheets("Sheet15").Cells(3 + j, 4 + (i - 1) * 6) = Application.Round(rbsensv(i, j, K) * 10000, 0) / 10000

        Next k
    Next i
Next j

'Deck
Worksheets("Sheet15").Cells(6 + nbay, 1) = "Deck Bay"
For i = 1 To 2
    If deckbaybraces = False Then
        Worksheets("Sheet15").Cells(7 + nbay, 1) = "Deck Portal"
        Worksheets("Sheet15").Cells(7 + nbay, 3 + (i - 1) * 8) = "Dmcrbias"
        Worksheets("Sheet15").Cells(7 + nbay, 4 + (i - 1) * 8) = "Dpcrlbias"
        Worksheets("Sheet15").Cells(7 + nbay, 5 + (i - 1) * 8) = "Wdbias"
        Worksheets("Sheet15").Cells(7 + nbay, 6 + (i - 1) * 8) = "Wjbias"
        Worksheets("Sheet15").Cells(7 + nbay, 7 + (i - 1) * 8) = "Dmcrcov"
        Worksheets("Sheet15").Cells(7 + nbay, 8 + (i - 1) * 8) = "Dpcrlcov"
        Worksheets("Sheet15").Cells(7 + nbay, 9 + (i - 1) * 8) = "Wdcov"
        Worksheets("Sheet15").Cells(7 + nbay, 10 + (i - 1) * 8) = "Wjcov"

        Worksheets("Sheet15").Cells(6 + nbay, 3) = "Broad Side"
        Worksheets("Sheet15").Cells(6 + nbay, 11) = "End On"
        For k = 1 To 8
            Worksheets("Sheet15").Cells(8 + nbay, k + 2 + (i - 1) * 8) = Application.Round(rbsensv(i, 0, k) * 10000, 0) / 10000
        Next k
    Else
        Worksheets("Sheet15").Cells(7 + nbay, 1) = "Braced Deck Bay"
        Worksheets("Sheet15").Cells(7 + nbay, 3 + (i - 1) * 6) = "Bcbias"
        Worksheets("Sheet15").Cells(7 + nbay, 4 + (i - 1) * 6) = "Wdbias"
        Worksheets("Sheet15").Cells(7 + nbay, 5 + (i - 1) * 6) = "Wjbias"
        Worksheets("Sheet15").Cells(7 + nbay, 6 + (i - 1) * 6) = "Bccov"
        Worksheets("Sheet15").Cells(7 + nbay, 7 + (i - 1) * 6) = "Wdcov"
        Worksheets("Sheet15").Cells(7 + nbay, 8 + (i - 1) * 6) = "Wjcov"
        Worksheets("Sheet15").Cells(6 + nbay, 3) = "Broad Side"
```

```
        Worksheets("Sheet15").Cells(6 + nbay, 9) = "End On"
        For k = 1 To 6
            Worksheets("Sheet15").Cells(8 + nbay, k + 2 + (i - 1) * 6) = Application.Round(rbsensv(i, 0, k) * 10000, 0) / 10000
        Next k
    End If
Next i

'foundation
Worksheets("Sheet15").Cells(10 + nbay, 1) = "Foundation"
Worksheets("Sheet15").Cells(10 + nbay, 3) = "Broad Side"
Worksheets("Sheet15").Cells(10 + nbay, 9) = "End On"
Worksheets("Sheet15").Cells(11 + nbay, 1) = "Lateral"
'lateral
For i = 1 To 2
    Worksheets("Sheet15").Cells(11 + nbay, 3 + (i - 1) * 6) = "Pubias"
    Worksheets("Sheet15").Cells(11 + nbay, 4 + (i - 1) * 6) = "Wdbias"
    Worksheets("Sheet15").Cells(11 + nbay, 5 + (i - 1) * 6) = "Wjbias"
    Worksheets("Sheet15").Cells(11 + nbay, 6 + (i - 1) * 6) = "Pucov"
    Worksheets("Sheet15").Cells(11 + nbay, 7 + (i - 1) * 6) = "Wdcov"
    Worksheets("Sheet15").Cells(11 + nbay, 8 + (i - 1) * 6) = "Wjcov"
    For k = 1 To 6
        Worksheets("Sheet15").Cells(12 + nbay, k + 2 + (i - 1) * 6) = Application.Round(rbsensv(i, nbay + 1, k) * 10000, 0) / 10000
    Next k
Next i

'axial
Worksheets("Sheet15").Cells(13 + nbay, 1) = "Axial"
Worksheets("Sheet15").Cells(14 + nbay, 1) = "Compression"
Worksheets("Sheet15").Cells(15 + nbay, 1) = "Tension"
For i = 1 To 2
    Worksheets("Sheet15").Cells(13 + nbay, 3 + (i - 1) * 6) = "Qbias"
    Worksheets("Sheet15").Cells(13 + nbay, 4 + (i - 1) * 6) = "Wdbias"
    Worksheets("Sheet15").Cells(13 + nbay, 5 + (i - 1) * 6) = "Wjbias"
    Worksheets("Sheet15").Cells(13 + nbay, 6 + (i - 1) * 6) = "Qcov"
    Worksheets("Sheet15").Cells(13 + nbay, 7 + (i - 1) * 6) = "Wdcov"
    Worksheets("Sheet15").Cells(13 + nbay, 8 + (i - 1) * 6) = "Wjcov"
    For k = 1 To 6
        Worksheets("Sheet15").Cells(14 + nbay, k + 2 + (i - 1) * 6) = Application.Round(rbsensv(i, 29, k) * 10000, 0) / 10000
        Worksheets("Sheet15").Cells(15 + nbay, k + 2 + (i - 1) * 6) = Application.Round(rbsensv(i, 30, k) * 10000, 0) / 10000
    Next k
Next i

'spatial effect in loading output

If (pltype = 1 Or pltype = 5 Or pltype = 7 Or pltype = 8) And mtpltype <> 1 Then
    MsgBox "No need to consider loading spatial effcet for small structures"
ElseIf spatial = True Then
    Worksheets("Sheet16").Cells(1, 1) = "Loading spatial effect: comparison with results without considering spatial effect "
    Worksheets("Sheet16").Cells(3, 1) = "Jacket Bay"
    Worksheets("Sheet16").Cells(3, 3) = "Broad Side Loading(kips)"
    Worksheets("Sheet16").Cells(3, 7) = "End On Loading(kips)"

' jacket bay
    Worksheets("Sheet16").Cells(4, 1) = "Bay #"


    For j = 1 To nbay

        Worksheets("Sheet16").Cells(4 + j, 1) = j
        For i = 1 To 2      '1 for Bs and 2 for Eo

            Worksheets("Sheet16").Cells(4, 3 + (i - 1) * 4) = "Without Effect"
            Worksheets("Sheet16").Cells(4, 4 + (i - 1) * 4) = "With Effect"
            Worksheets("Sheet16").Cells(4, 5 + (i - 1) * 4) = "Ratio"
            Worksheets("Sheet16").Cells(4 + j, 3 + (i - 1) * 4) = Application.Round(meanload(i, j), 0)
            Worksheets("Sheet16").Cells(4 + j, 4 + (i - 1) * 4) = Application.Round(spmeanload(i, j), 0)
            Worksheets("Sheet16").Cells(4 + j, 5 + (i - 1) * 4) = Application.Round(10000 * meanload(i, j) / spmeanload(i, j), 0) / 10000
        Next i
    Next j
End If
```

```
' MODULE 5
'
' Created by: Merhdad Mortazavi
' Created on: 1/22/96
'
'
' Last Modified by:  James Stear
' Last Modified on:  12/28/97
'
' Last Modified by:  Zhaohui Jin
'          on:  June 8, 1998
' This module contains the following macros:
'
' geometry
' brace design
' weight
' joint capacity
' main diagonals
' jacket capacity
' deck bay
' foundation capacity
' foundation stiffness
' capacity profile
'
'
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, jointsoff, dla(2), dli(2), dlzp(2), dlr(2)
Public dlmp(2), dlcm(2, 2), dlmcr(2), dlm(2), pileheadkx(3), pileheadkz(3), lehtemp(2)
Public mudmats, mudlinebraceA, Fmoment(2)
'
'
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'
Sub geometry() '  Begin GEOMETRY
'
' Check to see if design options are used
'
If DialogSheets("OPTIONS").CheckBoxes("prelim").Value = xlOn Then
  prelim = True
Else
  prelim = False
End If
'
'
If DialogSheets("OPTIONS").CheckBoxes("includeconductors").Value = xlOn Then
  includeconductors = True
Else
  includeconductors = False
End If
'
'
If DialogSheets("OPTIONS").CheckBoxes("designCD").Value = xlOn Then
  designcd = True
Else
  designcd = False
End If
'
'
If DialogSheets("OPTIONS").CheckBoxes("includemudmat").Value = xlOn Then
  mudmats = True
Else
  mudmats = False
End If
'
'
'If DialogSheets("OPTIONS").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
'
'
'
If DialogSheets("OPTIONS").CheckBoxes("jointsoff").Value = xlOn Then
   jointsoff = True
Else
   jointsoff = False
End If
'
'  Assign densities, and initialize structure weights
'
steelg = 0.49   '  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
mudlinebraceA = 1
extrapilecap(1) = 0
extrapilecap(2) = 0
'
'  Define gravity constant g (ft/sec^2) and Pi
'
g = 32.174
Pi = 3.14159
'
'  Assign parameters for special configurations
'
If pltype = 5 Then
   nleg = 3
   offangle = Application.Acos(bcw(1) / bcw(2) / 2)
Else
   offangle = Pi / 2
End If

If pltype = 7 Or pltype = 8 Then
   nleg = 1
   dld(1) = piled(1)
   dlt(1) = pilet(1)
   pretension = jgap(1)
   For i = 1 To 2
      jld(i, 1) = piled(1)
      jlt(i, 1) = pilet(1)
   Next i
   tcw(1) = bcw(1)
   tcw(2) = bcw(1)
   bcw(2) = bcw(1)

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
   If i = 2 And verticalface = True Then
      alpha(i) = Atn((bcw(i) - tcw(i)) / htotal)
   Else
      alpha(i) = Atn((bcw(i) - tcw(i)) / 2 / htotal)
   End If
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 verticalface = True Then
      If nleg <= 4 Then
         lh(i, 1) = tcw(i)
         lh(i, j + 1) = lh(i, j) + (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) is for left side only; right side is equal to tcw(i)/2
         lh(i, j + 1) = lh(i, j) + (bayh(j) * Tan(alpha(i)))
         lht(i, j) = lh(i, j) + lh(i, 1)
      Else
         lh(i, 1) = (tcw(i) - msw) / 2
         '  lh(i,j+1) is for left side only; right side is equal to lh(i,1)
         lh(i, j + 1) = lh(i, j) + (bayh(j) * Tan(alpha(i)))
         lht(i, j) = lh(i, j) + msw + lh(i, 1)
      End If
   Else
      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
   End If
```

```
Next j
'
' Now begin definition of main diagonals
'
' Frames seen from end-on elevation, which resist broadside loads
'
' Correct for presence of deck bay bracing
'
If deckbaybraces = True Then nbay = nbay + 1

i = 1
If nleg < 12 Then
   For j = 1 To nbay
      If deckbaybraces = True And j = nbay Then
         bayh(nbay) = bayh(0)
         lehtemp(i) = lh(i, j)
         lh(i, j) = lh(i, 1)
         lh(i, j + 1) = lh(i, 1)
      End If
'
' For all broadside frame braces, in platforms with less than 12 legs
'
      For k = 1 To ndb(i, j)
         If dbconf(i, j, k) = 1 Or dbconf(i, j, k) = 3 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
            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
      Next k
   Next j
Else
'
' The left and right sections of the broadside frames of 12 leg platforms
'
   For j = 1 To nbay
      If deckbaybraces = True And j = nbay Then
         bayh(nbay) = bayh(0)
         lehtemp(i) = lh(i, j)
         lh(i, j) = lh(i, 1)
         lh(i, j + 1) = lh(i, 1)
      End If
      For k = 1 To ndb(i, j)
         If dbconf(i, j, k) = 1 Or dbconf(i, j, k) = 3 Then
            If dbtype(i, j, k) = 1 Then   ' tension
               If dbpos(i, j, k) = 1 Then ' left
                  dbl(i, j, k) = Sqr(lh(i, j + 1) ^ 2 + bayh(j) ^ 2)
               Else                  ' right
                  dbl(i, j, k) = Sqr(lh(i, j) ^ 2 + bayh(j) ^ 2)
               End If
            Else                     ' compression
               If dbpos(i, j, k) = 1 Then ' left
                  dbl(i, j, k) = Sqr(lh(i, j) ^ 2 + bayh(j) ^ 2)
               Else                  ' right
                  dbl(i, j, k) = Sqr(lh(i, j + 1) ^ 2 + bayh(j) ^ 2)
               End If
            End If
         ElseIf dbconf(i, j, k) = 2 Then
            dbl(i, j, k) = Sqr((lh(i, j) / 2) ^ 2 + bayh(j) ^ 2)
         Else
            If dbtype(i, j, k) = 1 Then      ' tension
               If dbpos(i, j, k) = 1 Then    ' left
                  dbl(i, j, k) = Sqr((lh(i, j + 1) - lh(i, j) / 2) ^ 2 + bayh(j) ^ 2)
               Else                      ' right
                  dbl(i, j, k) = Sqr(((lh(i, j)) / 2) ^ 2 + bayh(j) ^ 2)
               End If
            Else                         ' compression
               If dbpos(i, j, k) = 1 Then    ' left
                  dbl(i, j, k) = Sqr(((lh(i, j)) / 2) ^ 2 + bayh(j) ^ 2)
               Else                      ' right
                  dbl(i, j, k) = Sqr((lh(i, j + 1) - lh(i, j) / 2) ^ 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
      If deckbaybraces = True And j = nbay Then
        bayh(nbay) = bayh(0)
        lehtemp(i) = lh(i, j)
        lh(i, j) = lh(i, 1)
        lh(i, j + 1) = lh(i, 1)
      End If
      For k = 1 To ndb(i, j)
        If dbconf(i, j, k) = 1 Or dbconf(i, j, k) = 3 Then      ' Single, X
          If verticalface = True Then
            If dbtype(i, j, k) = 1 Then    ' tension
              dbl(i, j, k) = Sqr((lh(i, j + 1)) ^ 2 + bayh(j) ^ 2)
            Else                           ' compression
              dbl(i, j, k) = Sqr((lh(i, j)) ^ 2 + bayh(j) ^ 2)
            End If
          Else
            dbl(i, j, k) = Sqr(((lh(i, j) + lh(i, j + 1)) / 2) ^ 2 + bayh(j) ^ 2)
          End If
        ElseIf dbconf(i, j, k) = 2 Then                    ' K
          dbl(i, j, k) = Sqr(((lh(i, j)) / 2) ^ 2 + bayh(j) ^ 2)
        Else                                               ' A
          If dbtype(i, j, k) = 1 Then        ' tension
            If verticalface = True 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 + 1)) / 2) ^ 2 + bayh(j) ^ 2)
            End If
          Else                               ' compression
            If verticalface = True 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
        End If
      Next k
    Next j
ElseIf nleg = 6 Then
    For j = 1 To nbay
      If deckbaybraces = True And j = nbay Then
        bayh(nbay) = bayh(0)
        lehtemp(i) = lh(i, j)
        lh(i, j) = lh(i, 1)
        lh(i, j + 1) = lh(i, 1)
      End If
      For k = 1 To ndb(i, j)
'
' 6 legs, single or X-braced
'
        If dbconf(i, j, k) = 1 Or dbconf(i, j, k) = 3 Then
          If dbtype(i, j, k) = 1 Then        ' tension
            If dbpos(i, j, k) = 1 Then     ' left
              dbl(i, j, k) = Sqr(lh(i, j + 1) ^ 2 + bayh(j) ^ 2)
            Else                           ' right
              If verticalface = True Then
                dbl(i, j, k) = Sqr((tcw(i) / 2) ^ 2 + bayh(j) ^ 2)
              Else
                dbl(i, j, k) = Sqr(lh(i, j) ^ 2 + bayh(j) ^ 2)
              End If
            End If
          Else                               ' compression
```

```
          If dbpos(i, j, k) = 1 Then     ' left
            dbl(i, j, k) = Sqr(lh(i, j) ^ 2 + bayh(j) ^ 2)
          Else                          ' right
            If verticalface = True Then
              dbl(i, j, k) = Sqr((tcw(i) / 2) ^ 2 + bayh(j) ^ 2)
            Else
              dbl(i, j, k) = Sqr(lh(i, j + 1) ^ 2 + bayh(j) ^ 2)
            End If
          End If
        End If
'
' 6 legs, A-braced
'
      ElseIf dbconf(i, j, k) = 4 Then
        If dbtype(i, j, k) = 1 Then      ' tension
          If dbpos(i, j, k) = 1 Then     ' left
            dbl(i, j, k) = Sqr((lh(i, j + 1) - lh(i, j) / 2) ^ 2 + bayh(j) ^ 2)
          Else                          ' right
            If verticalface = True Then
              dbl(i, j, k) = Sqr(((tcw(i)) / 4) ^ 2 + bayh(j) ^ 2)
            Else
              dbl(i, j, k) = Sqr(((lh(i, j)) / 2) ^ 2 + bayh(j) ^ 2)
            End If
          End If
        Else                            ' compression
          If dbpos(i, j, k) = 1 Then     ' left
            dbl(i, j, k) = Sqr(((lh(i, j)) / 2) ^ 2 + bayh(j) ^ 2)
          Else                          ' right
            If verticalface = True Then
              dbl(i, j, k) = Sqr(((tcw(i)) / 4) ^ 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
        End If
      Else
'
' 6 legs, K-braced
'
        If verticalface = True And dbpos(i, j, k) = 3 Then
          dbl(i, j, k) = Sqr(((tcw(i)) / 4) ^ 2 + bayh(j) ^ 2)
        Else
          dbl(i, j, k) = Sqr(((lh(i, j)) / 2) ^ 2 + bayh(j) ^ 2)
        End If
      End If
    Next k
  Next j
Else
'
' 8 and 12 legs
'
  For j = 1 To nbay
    If deckbaybraces = True And j = nbay Then
      bayh(nbay) = bayh(0)
      lehtemp(i) = lh(i, j)
      lh(i, j) = lh(i, 1)
      lh(i, j + 1) = lh(i, 1)
    End If
    For k = 1 To ndb(i, j)
'
' 8, 12 leg single or X-braced
'
      If dbconf(i, j, k) = 1 Or dbconf(i, j, k) = 3 Then
        If dbtype(i, j, k) = 1 Then      ' tension
          If dbpos(i, j, k) = 1 Then     ' left
            dbl(i, j, k) = Sqr(lh(i, j + 1) ^ 2 + bayh(j) ^ 2)
          ElseIf dbpos(i, j, k) = 2 Then ' center
            dbl(i, j, k) = Sqr(msw ^ 2 + bayh(j) ^ 2)
          Else                          ' right
            If verticalface = True Then
              dbl(i, j, k) = Sqr(lh(i, 1) ^ 2 + bayh(j) ^ 2)
            Else
```

```
                dbl(i, j, k) = Sqr(lh(i, j) ^ 2 + bayh(j) ^ 2)
              End If
            End If
          Else                          ' compression
            If dbpos(i, j, k) = 1 Then     ' left
              dbl(i, j, k) = Sqr(lh(i, j) ^ 2 + bayh(j) ^ 2)
            ElseIf dbpos(i, j, k) = 2 Then '  center
              dbl(i, j, k) = Sqr(msw ^ 2 + bayh(j) ^ 2)
            Else                        ' right
              If verticalface = True Then
                dbl(i, j, k) = Sqr(lh(i, 1) ^ 2 + bayh(j) ^ 2)
              Else
                dbl(i, j, k) = Sqr(lh(i, j + 1) ^ 2 + bayh(j) ^ 2)
              End If
            End If
          End If
'
' 8, 12 leg A-braced
'
        ElseIf dbconf(i, j, k) = 4 Then
          If dbtype(i, j, k) = 1 Then          ' tension
            If dbpos(i, j, k) = 1 Then         ' left
              dbl(i, j, k) = Sqr((lh(i, j + 1) - lh(i, j) / 2) ^ 2 + bayh(j) ^ 2)
            ElseIf dbpos(i, j, k) = 2 Then     ' center
              dbl(i, j, k) = Sqr((msw / 2) ^ 2 + bayh(j) ^ 2)
            Else                        ' right
              If verticalface = True Then
                dbl(i, j, k) = Sqr(((lh(i, 1)) / 2) ^ 2 + bayh(j) ^ 2)
              Else
                dbl(i, j, k) = Sqr(((lh(i, j)) / 2) ^ 2 + bayh(j) ^ 2)
              End If
            End If
          Else                          ' compression
            If dbpos(i, j, k) = 1 Then         ' left
              dbl(i, j, k) = Sqr(((lh(i, j)) / 2) ^ 2 + bayh(j) ^ 2)
            ElseIf dbpos(i, j, k) = 2 Then     ' center
              dbl(i, j, k) = Sqr((msw / 2) ^ 2 + bayh(j) ^ 2)
            Else                        ' right
              If verticalface = True Then
                dbl(i, j, k) = Sqr(((lh(i, 1)) / 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
          End If
'
' 8 or 12 leg, K-braced
'
        Else
          If dbpos(i, j, k) = 1 Then         ' left
            dbl(i, j, k) = Sqr((lh(i, j) / 2) ^ 2 + bayh(j) ^ 2)
          ElseIf dbpos(i, j, k) = 2 Then     ' center
            dbl(i, j, k) = Sqr((msw / 2) ^ 2 + bayh(j) ^ 2)
          Else                        ' right
            If verticalface = True Then
              dbl(i, j, k) = Sqr((lh(i, 1) / 2) ^ 2 + bayh(j) ^ 2)
            Else
              dbl(i, j, k) = Sqr((lh(i, j) / 2) ^ 2 + bayh(j) ^ 2)
            End If
          End If
        End If
      Next k
    Next j
  End If

'
' Reset bay floor lengths
'
  If deckbaybraces = True Then
    lh(1, nbay) = lehtemp(1)
    lh(2, nbay) = lehtemp(2)
  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
    If deckbaybraces = True And j = nbay Then
      bayh(nbay) = bayh(0)
      lehtemp(i) = lh(i, j)
      lh(i, j) = lh(i, 1)
      lh(i, j + 1) = lh(i, 1)
    End If
    For k = 1 To ndb(i, j)

      theta(i, j, k) = Application.Asin(bayh(j) / dbl(i, j, k))
'
' All braces except X: unsupported length is the same as physical length
'
      If dbconf(i, j, k) = 1 Or dbconf(i, j, k) = 2 Or dbconf(i, j, k) = 4 Then
        dblx(i, j, k) = dbl(i, j, k)
'
' X-braces. X-braces are assumed to buckle only in-plane, with the center of the
' X considered to be a support point.
'
' i = 1, nleg < 12
'
      ElseIf i = 1 And nleg < 12 Then
        dblx(i, j, k) = lh(i, j + 1) / 2 / Cos(theta(i, j, k))
'
' i = 1, nleg = 12
'
      ElseIf i = 1 And nleg = 12 Then
        If (dbpos(i, j, k) = 1 And dbtype(i, j, k) = 1) Or (dbpos(i, j, k) = 3 And dbtype(i, j, k) = 2) 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(Pi - theta(i, j, k) - thetax(i, j, k))
          dblx(i, j, k) = dbl(i, j, k) - ((lh(i, j) * bayh(j) / (lh(i, j) + lh(i, j + 1))) ^ 2 + (lh(i, j) * lh(i, j + 1) / (lh(i, j) + lh(i, j + 1))) ^ 2) ^ 0.5
        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(Pi - theta(i, j, k) - thetax(i, j, k))
          dblx(i, j, k) = ((bayh(j) - lh(i, j) * bayh(j) / (lh(i, j) + lh(i, j + 1))) ^ 2 + (lh(i, j) * lh(i, j + 1) / (lh(i, j) + lh(i, j + 1))) ^ 2) ^ 0.5
        End If
'
' i = 2, nleg <= 4
'
      ElseIf i = 2 And nleg <= 4 Then
        If verticalface = True Then
          If dbtype(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(Pi - theta(i, j, k) - thetax(i, j, k))
            dblx(i, j, k) = dbl(i, j, k) - ((lh(i, j) * bayh(j) / (lh(i, j) + lh(i, j + 1))) ^ 2 + (lh(i, j) * lh(i, j + 1) / (lh(i, j) + lh(i, j + 1))) ^ 2) ^ 0.5
          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(Pi - theta(i, j, k) - thetax(i, j, k))
            dblx(i, j, k) = ((bayh(j) - lh(i, j) * bayh(j) / (lh(i, j) + lh(i, j + 1))) ^ 2 + (lh(i, j) * lh(i, j + 1) / (lh(i, j) + lh(i, j + 1))) ^ 2) ^ 0.5
          End If
        Else
          dblx(i, j, k) = lh(i, j + 1) / 2 / Cos(theta(i, j, k))
        End If
'
' i = 2, nleg = 6
'
      ElseIf i = 2 And nleg = 6 Then
        If verticalface = True And dbpos(i, j, k) = 3 Then
          dblx(i, j, k) = tcw(i) / 4 / Cos(theta(i, j, k))
        Else
          If (dbpos(i, j, k) = 1 And dbtype(i, j, k) = 1) Or (dbpos(i, j, k) = 3 And dbtype(i, j, k) = 2) 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(Pi - theta(i, j, k) - thetax(i, j, k))
            dblx(i, j, k) = dbl(i, j, k) - ((lh(i, j) * bayh(j) / (lh(i, j) + lh(i, j + 1))) ^ 2 + (lh(i, j) * lh(i, j + 1) / (lh(i, j) + lh(i, j + 1))) ^ 2) ^ 0.5
          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(Pi - theta(i, j, k) - thetax(i, j, k))
                dblx(i, j, k) = ((bayh(j) - lh(i, j) * bayh(j) / (lh(i, j) + lh(i, j + 1))) ^ 2 + (lh(i, j) * lh(i, j + 1) / (lh(i, j) + lh(i, j + 1))) ^ 2) ^ 0.5
              End If
            End If
'
' i = 2, and nleg >= 8
'
        Else
          If verticalface = True And dbpos(i, j, k) = 3 Then
            dblx(i, j, k) = lh(i, 1) / 2 / Cos(theta(i, j, k))
          Else
            If (dbpos(i, j, k) = 1 And dbtype(i, j, k) = 1) Or (dbpos(i, j, k) = 3 And dbtype(i, j, k) = 2) 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(Pi - theta(i, j, k) - thetax(i, j, k))
              dblx(i, j, k) = dbl(i, j, k) - ((lh(i, j) * bayh(j) / (lh(i, j) + lh(i, j + 1))) ^ 2 + (lh(i, j) * lh(i, j + 1) / (lh(i, j) + lh(i, j + 1))) ^ 2) ^ 0.5
            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(Pi - theta(i, j, k) - thetax(i, j, k))
              dblx(i, j, k) = ((bayh(j) - lh(i, j) * bayh(j) / (lh(i, j) + lh(i, j + 1))) ^ 2 + (lh(i, j) * lh(i, j + 1) / (lh(i, j) + lh(i, j + 1))) ^ 2) ^ 0.5
            End If
          End If
        End If
      Next k
    Next j
  Next i

If deckbaybraces = True Then
  lh(1, nbay) = lehtemp(1)
  lh(2, nbay) = lehtemp(2)
  nbay = nbay - 1
End If

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.
'
If deckbaybraces = True Then nbay = nbay + 1

  For i = 1 To 2
    For j = 1 To nbay
      For k = 1 To ndb(i, j)
        dummy = kbuck * dblx(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

If deckbaybraces = True Then nbay = nbay - 1

End Sub '  End BRACE DESIGN

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

Sub weight() ' Start WEIGHT
'
' Total weight of horizontal braces.  Also find total projected area of mudline 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
    If j = nbay + 1 Then mudlinebraceA = mudlinebraceA + hbd(j, k) * hbl(j, k) / 12
  Next k
Next j
```

```
'
' Add weight of braces and jacket legs to total jacket weight
'
If deckbaybraces = True Then nbay = nbay + 1

For i = 1 To 2
  For j = 1 To nbay
    If i = 1 And nleg > 4 Then
      jacketw = jacketw + steelg * 4 * Pi * (jld(i, j) - jlt(i, j)) * jlt(i, j) * bayh(j) / 144
    ElseIf i = 2 And nleg > 4 Then
      jacketw = jacketw + steelg * (nleg - 4) * Pi * (jld(i, j) - jlt(i, j)) * jlt(i, j) * bayh(j) / 144
    ElseIf i = 1 And nleg <= 4 Then
      jacketw = jacketw + steelg * nleg * Pi * (jld(i, j) - jlt(i, j)) * jlt(i, j) * bayh(j) / 144
    End If
    For k = 1 To ndb(i, j)
      jacketw = jacketw + steelg * (Pi * (dbd(i, j, k) - dbt(i, j, k)) * dbt(i, j, k) * dbl(i, j, k)) / 144
    Next k
  Next j
Next i
'
' Find deck leg weight
'
If deckbaybraces = True Then
  nbay = nbay - 1
Else
  If nleg > 4 Then
    decklegsw = steelg * dla(1) / 144 * bayh(0) * 4 + steelg * dla(2) / 144 * bayh(0) * (nleg - 4)
  Else
    decklegsw = steelg * dla(1) / 144 * bayh(0) * nleg
  End If
End If
'
' Find pile weight and total steel weight
'
If nleg > 4 Or pltype > 6 Then
  For i = 1 To 2
    If i = 1 Then nanchor = 1
    If i = 2 And pltype = 7 Then nanchor = 2
    If i = 2 And pltype = 8 Then nanchor = 3
    pilea = (piled(i) - pilet(i)) * pilet(i) * Pi
  If pltype > 6 Then
    pilew = pilew + nanchor * steelg * pilea / 144 * (pilel(i))
  Else
    pilew = pilew + steelg * (4 + (nleg - 8) * (i - 1)) * pilea / 144 * (pilel(i) + htotal)
  End If
  Next i
Else
  pilea = (piled(1) - pilet(1)) * pilet(1) * Pi
  pilew = steelg * nleg * pilea / 144 * (pilel(1) + htotal)
End If

If skirt = True Then
  pilea = (piled(3) - pilet(3)) * pilet(3) * Pi
  pilew = pilew + steelg * nskirt * pilea / 144 * pilel(3)
End If

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(1)
  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) = jyield(i) * 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

If deckbaybraces = True Then
  nbay = nbay + 1
  bayh(nbay) = bayh(0)
  elevation(nbay) = elevation(0)
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
'
'  Now begin capacity calculations
'
    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) * (dbfyspecific(i, j, k) / e) ^ 0.5 * dbkspecific(i, j, k) * (dblx(i, j, k) * 12) / dbr(i, j, k)
```

```
        dbpe(i, j, k) = 3.14 ^ 2 * e * dba(i, j, k) / (dbkspecific(i, j, k) * dblx(i, j, k) * 12 / dbr(i, j, k)) ^ 2
        dbpy(i, j, k) = dbfyspecific(i, j, k) * 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) * dbfyspecific(i, j, k) * 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) * dbfyspecific(i, j, k) / 12
        If dbfyspecific(i, j, k) * dbd(i, j, k) / dbt(i, j, k) < 1500 Then
          dbmcr(i, j, k) = 1
        ElseIf dbfyspecific(i, j, k) * dbd(i, j, k) / dbt(i, j, k) < 3000 Then
          dbmcr(i, j, k) = (1.13 - 2.58 * dbfyspecific(i, j, k) * dbd(i, j, k) / dbt(i, j, k) / e)
        Else
          dbmcr(i, j, k) = (0.94 - 0.76 * dbfyspecific(i, j, k) * 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 + (velocity(j + 1) * Sin(theta(i, j, k))) ^ 2) / 2 * cdmem * (dbd(i, j, k) + 2 * mg(j)) / 12 / 1000) *
If
        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 / (dbe
ps0(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 - (S
in(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 * dbkspecific(i, j, k) * dblx(i, j, k) * 12 / dbrtr(i, j, k) * (dbfyspecific(i, j, k) / 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

If deckbaybraces = True Then nbay = nbay - 1

End Sub '  End MAIN DIAGONALS

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

Sub jacket_capacity() '  Start JACKET CAPACITY

If deckbaybraces = True Then nbay = nbay + 1

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
      mltf#(i, j) = 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)
'
'$$$$$
      mltfalpha(i, j) = dbpu(i, j, mltfbrace) / dbki(i, j, mltfbrace) * Cos(theta(i, j, mltfbrace))
      For k = 1 To ndb(i, j)
         jpuh(i, j, k) = jpu(i, j, mltfjoint) / dbki(i, j, mltfjoint) * dbki(i, j, k) * Cos(theta(i, j, mltfjoint))
         dbpuh(i, j, k) = mltfalpha(i, j) * dbki(i, j, k)
'$$$$$
' 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)

      Next k
      sumksq(i, j) = sumksq(i, j) ^ 2
'
' 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

If deckbaybraces = True Then
   nbay = nbay - 1
   deckk(1) = kbay(1, nbay + 1)
   deckk(2) = kbay(2, nbay + 1)
End If

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.
'
'
' First determine deck leg properties
' For structures with more than 4 legs, i=1 is for corners, i=2 is for centers
'
Dim pp1(2), pp2(2), dta(2)

If pltype > 4 Or pltype = 1 Then
   k = 1
Else
   k = 2 ' K num of deck leg types
End If
For i = 1 To k
   dla(i) = (dld(i) - dlt(i)) * dlt(i) * Pi
   dli(i) = Pi / 64 * (dld(i) ^ 4 - (dld(i) - 2 * dlt(i)) ^ 4)
   dlzp(i) = 1.3 * Pi / 32 * (dld(i) ^ 4 - (dld(i) - 2 * dlt(i)) ^ 4) / dld(i)
   dlr(i) = 1 / 4 * (dld(i) ^ 2 + (dld(i) - 2 * dlt(i)) ^ 2) ^ 0.5
   dlmp(i) = dlzp(i) * fy / 12
   If fy * dld(i) / dlt(i) < 1500 Then
      dlmcr(i) = 1
   ElseIf fy * dld(i) / dlt(i) < 3000 Then
```

```
      dlmcr(i) = (1.13 - 2.58 * fy * dld(i) / dlt(i) / e)
    Else
      dlmcr(i) = (0.94 - 0.76 * fy * dld(i) / dlt(i) / e)
    End If
'$$$$$$
    dlmcr(i) = dmcrbias * dlmcr(i) * dlmp(i)        'dmcrbias
    dlm(i) = dlmcr(i) * Cos(Pi / 2 * qdeck / nleg / fy / dla(i) / dpcrlbias) 'dpcrlbias
    pp1(i) = dlm(i) / dmcrbias   'Rd wrt dmcrbias
    pp2(i) = dlmcr(i) * Sin(Pi / 2 * qdeck / nleg / fy / dla(i) / dpcrlbias) * Pi / 2 * qdeck / nleg / fy / dla(i) / (dpcrlbias ^ 2) 'Rd wrt dpcrlbias

'
'   Effective rotational inertia of jacket leg / deck leg attachment point
'
    If pltype = 7 Or pltype = 8 Then   '  Caissons
      ibay1(i) = dli(i)
    Else
      If pgrout = True Then  '  All others
        ibay1(i) = Pi / 64 * (jld(i, 1) ^ 4 - (piled(i) - 2 * pilet(i)) ^ 4)
      Else
        ibay1(i) = Pi / 64 * (jld(i, 1) ^ 4 - (jld(i, 1) - 2 * jlt(i, 1)) ^ 4)
      End If
    End If
  Next i

  For i = 1 To 2  '  broadside, end-on
'
'   Effective lateral stiffness of 1st jacket bay
'
    dlcs(i) = kbay(i, 1) / nleg
'
'   Now find effective rotational stiffness of springs at base of deck legs
'
    For j = 1 To k  '  k = 2 only if there are more than 4 legs
      If pltype = 7 Or pltype = 8 Then   '  caissons
        dlcm(i, j) = ((bayh(1) + 10 * piled(j) / 12) / (e * ibay1(j) / 144) - 3 * dlcs(i) * (bayh(1)) ^ 4 / (4 * dlcs(i) * (bayh(1)) ^ 3 * e * ibay1(j) / 144 + 12 * (e * iba
y1(j) / 144) ^ 2)) ^ -1
      Else   '  all others
        dlcm(i, j) = (bayh(1) / (e * ibay1(j) / 144) - 3 * dlcs(i) * bayh(1) ^ 4 / (4 * dlcs(i) * bayh(1) ^ 3 * e * ibay1(j) / 144 + 12 * (e * ibay1(j) / 144) ^ 2)) ^ -1
      End If
    Next j
  Next i
'
'   Now compute deck bay stiffness, deck bay drift, deck bay capacity
'

  If nleg < 3 Then
    For i = 1 To 2  '  broadside, end-on
      deckk(i) = ((bayh(0) / (3 * e * dli(1) / 144) + 1 / dlcm(i, 1)) ^ (-1)) / (bayh(0) ^ 2)
      dummy = bayh(0) * (bayh(0) / (3 * e * dli(1) / 144) + 1 / dlcm(i, 1))
      dldelta(i) = dlm(1) * dummy
      dlcap(i) = (nleg * dlm(1) - qdeck * dldelta(i)) / bayh(0)
      '$$$$$$
      prddmcrbias(i) = 2 * nleg / bayh(0) * pp1(1) - qdeck / bayh(0) * dummy * pp1(1)
      prddpcrlbias(i) = 2 * nleg / bayh(0) * pp2(1) - qdeck / bayh(0) * dummy * pp2(1)
    Next i
  ElseIf nleg = 3 Or nleg = 4 Then
    For i = 1 To 2  '  broadside, end-on
      deckk(i) = nleg * ((bayh(0) / (6 * e * dli(1) / 144) + 1 / dlcm(i, 1)) ^ (-1)) / (bayh(0) ^ 2)
      dummy = bayh(0) * (bayh(0) / (6 * e * dli(1) / 144) + 1 / dlcm(i, 1))
      dldelta(i) = dlm(1) * dummy
      dlcap(i) = (2 * nleg * dlm(1) - qdeck * dldelta(i)) / bayh(0)
      prddmcrbias(i) = 2 * nleg / bayh(0) * pp1(1) - qdeck / bayh(0) * dummy * pp1(1)
      prddpcrlbias(i) = 2 * nleg / bayh(0) * pp2(1) - qdeck / bayh(0) * dummy * pp2(1)
    Next i
  Else
    For i = 1 To 2  '  broadside, end-on
      deckk(i) = 4 * ((bayh(0) / (6 * e * dli(1) / 144) + 1 / dlcm(i, 1)) ^ (-1)) / (bayh(0) ^ 2) + (nleg - 4) * ((bayh(0) / (6 * e * dli(2) / 144) + 1 / dlcm(i, 2)) ^ (-1)) /
  (bayh(0) ^ 2)
      dta(1) = dlm(1) * bayh(0) * (bayh(0) / (6 * e * dli(1) / 144) + 1 / dlcm(i, 1))
      dta(2) = dlm(2) * bayh(0) * (bayh(0) / (6 * e * dli(2) / 144) + 1 / dlcm(i, 2))
      If dta(1) <= dta(2) Then
        k = 2
```

```
    Else
      k = 1
    End If
    dldelta(i) = dta(k)
      'dldelta(i) = Application.Max(dlm(1) * bayh(0) * (bayh(0) / (6 * e * dli(1) / 144) + 1 / dlcm(i, 1)), dlm(2) * bayh(0) * (bayh(0) / (6 * e * dli(2) / 144) + 1 / 
dlcm(i, 2)))
    dlcap(i) = (2 * 4 * dlm(1) + 2 * (nleg - 4) * dlm(2) - qdeck * dldelta(i)) / bayh(0)
    prddmcrbias(i) = (8 * pp1(1) + 2 * (nleg - 4) * pp1(2) - qdeck * dta(k) / dlm(k) * pp1(k)) / bayh(0)
    prddpcrlbias(i) = (8 * pp2(1) + 2 * (nleg - 4) * pp2(2) - qdeck * dta(k) / dlm(k) * pp2(k)) / bayh(0)
  Next i
End If

End Sub '  End DECK BAY

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

Sub pile_capacity()  '  Start PILE CAPACITY

If nleg > 4 And skirt = True Then
  npiletype = 3
ElseIf skirt = True Then
  npiletype = 2
ElseIf nleg > 4 Then
  npiletype = 2
ElseIf pltype > 6 Then
  npiletype = 2
Else
  npiletype = 1
End If

'
'  First find unit soil resistances for lateral and vertical directions
'

maxpilepenetration = 0
maxpilediameter = 0
sumvert = 0

For i = 1 To npiletype
  If pilel(i) > maxpilepenetration Then maxpilepenetration = Application.Round(pilel(i), 0)
  If piled(i) / 12 > maxpilediameter Then maxpilediameter = piled(i) / 12
Next i

i = 1
j = 1
soilpressure = 0

Do

soilpressure = soilpressure + gammas(j)

If stype(j) = 2 Then   '  clay
'    horizontal
  If i < maxpilediameter * 1.5 Then
    suhorz(i) = 0
  ElseIf j = 1 Then
    suhorz(i) = 9 * (su1(j) + suslope(j) * i)
  Else
    suhorz(i) = 9 * (su1(j) + suslope(j) * (i - layerdepth(j - 1)))
  End If
'    vertical
  su = (su1(j) + su2(j)) / 2
  If su < 0.5 Then
    suvert(i) = su
  ElseIf su > 1.5 Then
    suvert(i) = 0.5 * su
  Else
    suvert(i) = (1 - (su - 0.5) / 2) * su
  End If
Else           '  sand
'    horizontal
  suhorz(i) = 3 * soilpressure * (Tan((45 + sphi(j) / 2) * Pi / 180)) ^ 2
'    vertical
```

```
   If sphi(j) < 20 Then
      ffmax = 1
   ElseIf sphi(j) > 35 Then
      ffmax = 2
   Else
      ffmax = 2 - (35 - sphi(j)) * 0.06
   End If
   suvert(i) = Application.Min(ffmax, 0.8 * soilpressure * Tan((sphi(j) - 5) * Pi / 180))
End If

i = i + 1

If i > layerdepth(j) Then j = j + 1

Loop While i <= maxpilepenetration


'
'  Axial and lateral capacities of foundation piles
'
'  Cross-section properties of main piles
'

For j = 1 To npiletype

   i = j
   If npiletype = 2 And skirt = True And j = 2 Then
      pilecaplat(j) = 0
      pilecapcomp(j) = 0
      pilecaptens(j) = 0
      i = j + 1
   End If
   pilea = (piled(i) - pilet(i)) * pilet(i) * Pi
   pilepy = pilea * pileyield(i)
   pilezp = 1.3 * Pi / 32 * (piled(i) ^ 4 - (piled(i) - 2 * pilet(i)) ^ 4) / piled(i)
   piler = 1 / 4 * (piled(i) ^ 2 + (piled(i) - 2 * pilet(i)) ^ 2) ^ 0.5
   pilemp = pilezp * pileyield(i) / 12
   If pileyield(i) * piled(i) / pilet(i) < 1500 Then
      pilemcr = 1
   ElseIf pileyield(i) * piled(i) / pilet(i) < 3000 Then
      pilemcr = (1.13 - 2.58 * pileyield(i) * piled(i) / pilet(i) / e)
   Else
      pilemcr = (0.94 - 0.76 * pileyield(i) * piled(i) / pilet(i) / e)
   End If
   pilemcr = pilemcr * pilemp
   pilem = pilemcr * Cos(Pi / 2 * qdeck / (nleg + nskirt) / pileyield(i) / pilea)
   If pgrout = True Then
      legeffectivea = pilea + (jld(1, 1) - jlt(1, 1)) * jlt(1, 1) * Pi
   Else
      legeffectivea = pilea
   End If


'
'  First calculate pile axial capacity
'
   suverttotal = 0
   wp = 0
   ii = 1
   k = 1

   Do

      If plug(i) = True Then wp = wp + gammas(ii) * Pi * ((piled(i) / 2 - pilet(i)) / 12) ^ 2
      suverttotal = suverttotal + suvert(k)
      If k > layerdepth(ii) Then ii = ii + 1
      k = k + 1

   Loop While k <= pilel(i)

   wp = wp + pilel(i) * steelg * Pi * ((piled(i) / 2 / 12) ^ 2 - ((piled(i) / 2 - pilet(i)) / 12) ^ 2)
```

```
  If stype(ii) = 2 Then
    qplug = suhorz(k - 1) * Pi * (piled(i) / 2 / 12) ^ 2
  Else
    If sphi(ii) < 20 Then
      nq = 8
    ElseIf sphi(ii) > 35 Then
      nq = 40
    ElseIf sphi(ii) > 30 Then
      nq = 40 - (35 - sphi(ii)) * 4
    ElseIf sphi(ii) > 25 Then
      nq = 20 - (30 - sphi(ii)) * 1.6
    Else
      nq = 12 - (25 - sphi(ii)) * 0.8
    End If
    qplug = (Application.Min(nq * soilpressure, 5 * nq)) * Pi * (piled(i) / 12) ^ 2 / 4
  End If

  If (qplug > suverttotal * Pi * (piled(i) - 2 * pilet(i)) / 12) Then qplug = 0

  pilecapcomp(i) = qbias * (Application.Min(qplug + suverttotal * Pi * piled(i) / 12, pilepy) - wp)
  pilecaptens(i) = qbias * (Application.Min(suverttotal * Pi * piled(i) / 12, pilepy) + wp)
'
' Now calculate pile lateral capacity
'

k = 1
pileshear = 0
pasttip = False

Do

  pileshear = pileshear + suhorz(k) * piled(i) / 12
  kk = 1
  pilerot = 2 * pilem / k
  Do
    pilerot = pilerot + suhorz(kk) * piled(i) / 12 * (k - kk) / (k + scour)
    kk = kk + 1
  Loop While kk <= k
  k = k + 1
  If k > pilel(i) Then
    pasttip = True
  End If
Loop While ((pasttip = False) And ((pilerot - pileshear) > 0))

If pasttip = True Then
  pilecaplat(i) = pileshear * pubias
Else
  pilecaplat(i) = pilerot * pubias
End If

'
' OLD CODE |
'        V
'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
'  If plug(i) = True Then
'    dummy = Application.Min(9 * su2 * Pi * (piled(i) / 12) ^ 2 / 4, ff * Pi * (piled(i) - 2 * pilet(i)) / 12 * pilel(i))
'  Else
'    dummy = 9 * su2 * Pi * (piled(i) / 12) * (pilet(i) / 12)
'  End If
'  pilecapcomp(i) = qbias * (Application.Min(pilepy, dummy + (ff * Pi * piled(i) / 12 - wp) * pilel(i)))
'  pilecaptens(i) = qbias * (Application.Min(pilepy, (ff * Pi * piled(i) / 12 + wp) * pilel(i)))
```

```
'
'
' Lateral capacity of main piles if founded in clay
'
' a = 9 * su1 * piled(i) / 12
' b = 9 * su2 * piled(i) / 12
' seta = (b - a) / pilel(i)
' psi = 1.5 * piled(i) / 12 + scour
' pucbar = 0.5 * (-(27 * (piled(i) / 12) ^ 2 * su1) + ((27 * (piled(i) / 12) ^ 2 * su1) ^ 2 + 144 * su1 * (piled(i) / 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))
' pilecaplat(i) = pubias * pucbar
'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(i) < plc Then
'   fas = 0.5 * pilel(i) ^ 2 * gammas * Tan((sphi - 5) * Pi / 180)
' Else
'   fas = ffmax * (pilel(i) - 0.5 * plc) * piled(i) / 12 * Pi
' End If
' dummy = 0
' If plug(i) = True Then dummy = Application.Min(qmax, nq * pilel(i) * gammas) * Pi * (piled(i) / 12) ^ 2 / 4
' pilecapcomp(i) = qbias * (Application.Min(pilepy, fas + dummy) - wp * pilel(i))
' If pilel(i) < plt Then
'   dummy = 0.7 * 0.5 * pilel(i) ^ 2 * gammas * Tan((sphi - 5) * Pi / 180)
' Else
'   dummy = ffmax * (pilel(i) - 0.5 * plt)
' End If
' pilecaptens(i) = qbias * (Application.Min(pilepy, dummy * Pi * piled(i) / 12 + wp * pilel(i)))
'
' Lateral capacity of main piles if founded in sand
'
' kp = (Tan((45 + sphi / 2) * Pi / 180)) ^ 2
' If scour = 0 Then
'   pusbar = (2.382 * pilem ^ (2 / 3) * (gammas * piled(i) / 12 * kp) ^ (1 / 3))
```

```
'  Else
'     pusbar = (2.382 * pilem ^ (2 / 3) * (gammas * piled(i) / 12 * kp) ^ (1 / 3))
'
'  Newton-Raphson iteration to find lateral capacity in sand
'
'     Do
'        Formula = pusbar - (2 * pilem / (scour + 0.544 * (pusbar / gammas / piled(i) / 12 / kp)) ^ 0.5)
'        Formula1 = 1 + pilem * 0.544 / (gammas * piled(i) / 12 * kp) * (scour + 0.544 * (pusbar / gammas / piled(i) / 12 / kp)) ^ -1.5
'        pusbartest = pusbar
'        pusbar = pusbar - Formula / Formula1
'     Loop While ((Abs(pusbartest - pusbar) > 1) And (Abs(Formula) > 0.0001))
'  End If
'  pilecaplat(i) = pubias * pusbar
'End If

Next j

'
' Find conductor lateral strength
'

If diaconductors > 0 Then

   k = 1
   pileshear = 0
   pasttip = False

   Do

      pileshear = pileshear + suhorz(k) * diaconductors / 12
      kk = 1
      pilerot = 2 * mpconductors / k
      Do
         pilerot = pilerot + suhorz(kk) * diaconductors / 12 * (k - kk) / (k + conductorfix + scour)
         kk = kk + 1
      Loop While kk <= k
      k = k + 1
      If k > penconductors Then
         pasttip = True
      End If
   Loop While ((pasttip = False) And ((pilerot - pileshear) > 0))

   If pasttip = True Then
      conductorcap = pileshear * pubias
   Else
      conductorcap = pilerot * pubias
   End If

End If

End Sub '  End PILE CAPACITY

'###############################################################################################

Sub foundation_capacity()  '  Start FOUNDATION CAPACITY

'
' Number of different types of piles: horizontal
'

If skirt = True Then
   npileh(3) = nskirteo + nskirtbs + nskirtcorner
Else
   npileh(3) = 0
End If
If nleg > 4 Then
   npileh(1) = 4
   npileh(2) = nleg - 4
Else
   npileh(1) = nleg
   npileh(2) = 0
End If
```

```
'
' Find horizontal capacity
'
pilehorzcap = 0
For i = 1 To 3
   pilehorzcap = pilehorzcap + npileh(i) * pilecaplat(i)
Next i

If includeconductors = True Then
   pilehorzcap = pilehorzcap + numconductors * conductorcap * groupstr
End If

'
' Now find pile axial capacities
'
'
' Find number of center piles (pile type 2) on each face
'
If nleg = 6 Then
   npilebs = 2
   npileeo = 0
ElseIf nleg = 8 Then
   npilebs = 4
   npileeo = 0
ElseIf nleg = 12 Then
   npilebs = 4
   npileeo = 2
Else
   npileeo = 0
   npilebs = 0
End If

If nleg > 4 Then
   avpilecapcomp(1) = Application.Max((pilecapcomp(1) * 4 + pilecapcomp(2) * npilebs + pilecapcomp(3) * (nskirtcorner + nskirtbs)) / (4 + npilebs + nskirtcorn
er + nskirtbs), 0.001)
   avpilecaptens(1) = Application.Max((pilecaptens(1) * 4 + pilecaptens(2) * npilebs + pilecaptens(3) * (nskirtcorner + nskirtbs)) / (4 + npilebs + nskirtcorner + n
skirtbs), 0.001)
   avpilecapcomp(2) = Application.Max((pilecapcomp(1) * 4 + pilecapcomp(2) * npileeo + pilecapcomp(3) * (nskirtcorner + nskirteo)) / (4 + npileeo + nskirtcor
ner + nskirteo), 0.001)
   avpilecaptens(2) = Application.Max((pilecaptens(1) * 4 + pilecaptens(2) * npileeo + pilecaptens(3) * (nskirtcorner + nskirteo)) / (4 + npileeo + nskirtcorner + n
skirteo), 0.001)
   Fmoment(1) = bcw(1) / 2 * (2 * (pilecapcomp(1) + pilecaptens(1)) + (npilebs / 2) * (pilecapcomp(2) + pilecaptens(2)) + (nskirtcorner + nskirtbs) / 2 * (pilecap
comp(3) + pilecaptens(3)))
   Fmoment(2) = bcw(2) / 2 * (2 * (pilecapcomp(1) + pilecaptens(1)) + (npileeo / 2) * (pilecapcomp(2) + pilecaptens(2)) + (nskirtcorner + nskirteo) / 2 * (pilecap
comp(3) + pilecaptens(3)))
Else
   avpilecapcomp(1) = Application.Max((pilecapcomp(1) * nleg + pilecapcomp(2) * npilebs + pilecapcomp(3) * (nskirtcorner + nskirtbs)) / (nleg + npilebs + nski
rtcorner + nskirtbs), 0.001)
   avpilecaptens(1) = Application.Max((pilecaptens(1) * nleg + pilecaptens(2) * npilebs + pilecaptens(3) * (nskirtcorner + nskirtbs)) / (nleg + npilebs + nskirtcorn
er + nskirtbs), 0.001)
   avpilecapcomp(2) = Application.Max((pilecapcomp(1) * nleg + pilecapcomp(2) * npileeo + pilecapcomp(3) * (nskirtcorner + nskirteo)) / (nleg + npileeo + nsk
irtcorner + nskirteo), 0.001)
   avpilecaptens(2) = Application.Max((pilecaptens(1) * nleg + pilecaptens(2) * npileeo + pilecaptens(3) * (nskirtcorner + nskirteo)) / (nleg + npileeo + nskirtcorn
er + nskirteo), 0.001)
   Fmoment(1) = bcw(1) / 2 * (nleg / 2 * (pilecapcomp(1) + pilecaptens(1)) + (npilebs / 2) * (pilecapcomp(2) + pilecaptens(2)) + (nskirtcorner + nskirtbs) / 2 * (pi
lecapcomp(3) + pilecaptens(3)))
   Fmoment(2) = bcw(2) / 2 * (nleg / 2 * (pilecapcomp(1) + pilecaptens(1)) + (npileeo / 2) * (pilecapcomp(2) + pilecaptens(2)) + (nskirtcorner + nskirteo) / 2 * (p
ilecapcomp(3) + pilecaptens(3)))
End If

End Sub

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

Sub foundation_stiffness() ' Start FOUNDATION STIFFNESS
'
' Pilehead stiffnesses. All are in kip and ft.
'
' The lateral stiffnesses are determined by assuming the pile is a fixed-fixed beam
' with fixity at the mudline and a depth of 10D below the mudline.
'
' The vertical stiffnesses assume a base stiffness of EA/L
```

```
'
' No accounting for jacket bending or vertical stiffness are taken here. This will
' be changed in future versions.
'
For i = 1 To 3
  If pilel(i) = 0 Then
    pileheadkx(i) = 0
    pileheadkz(i) = 0
  Else
    pileheadkx(i) = horizkbias * 12 * e * Pi * pilet(i) * (piled(i) / 2) ^ 3 / 144 / (10 * piled(i) / 12) ^ 3
    pileheadkz(i) = axialkbias * e * Pi * ((piled(i) / 2) ^ 2 - (piled(i) / 2 - pilet(i)) ^ 2) / pilel(i)
  End If
Next i
'
' Find conductor stiffness
'
If diaconductors > 0 Then conductorstiff = 12 * Iconductors * e * 144 / (10 * diaconductors / 12 + conductorfix) ^ 3

If includeconductors = True Then
  foundationkx = numconductors * groupstiff * conductorstiff
Else
  foundationkx = 0
End If

'
' Determine foundation vertical, horizontal, and overturning stiffness. It is assumed
' that pile-structure connections are rigid.
'
If nleg > 4 Then
  foundationkx = foundationkx + 4 * pileheadkx(1) + (nleg - 4) * pileheadkx(2) + nskirt * pileheadkx(3)
Else
  foundationkx = foundationkx + nleg * pileheadkx(1) + nskirt * pileheadkx(3)
End If

' 1 = broadside response, 2 = end-on response

If nleg = 3 Then
  foundationktheta(1) = 2 * pileheadkz(1) * (bcw(1) / 2) ^ 2 + 2 * pileheadkz(3) * (bcw(1) / 2) ^ 2
  foundationktheta(2) = 2 * pileheadkz(1) * (bcw(2) * Sin(offangle) / 3) ^ 2 + 2 * pileheadkz(3) * (bcw(2) * Sin(offangle) / 3) ^ 2 + pileheadkz(1) * (bcw(2) * 2
* Sin(offangle) / 3) ^ 2 + pileheadkz(3) * (2 * bcw(2) * Sin(offangle) / 3) ^ 2
Else
  foundationktheta(1) = (4 * pileheadkz(1) + npilebs * pileheadkz(2) + (nskirtbs + nskirtcorner) * pileheadkz(3)) * (bcw(1) / 2) ^ 2
  foundationktheta(2) = (4 * pileheadkz(1) + npileeo * pileheadkz(2) + (nskirteo + nskirtcorner) * pileheadkz(3)) * (bcw(2) / 2) ^ 2 + npilebs * pileheadkz(2) * (
msw / 2) ^ 2
End If
'
' Find vertical stiffness
'
If nleg > 4 Then
  kvertical = 4 * pileheadkz(1) + (nleg - 4) * pileheadkz(2) + nskirt * pileheadkz(3)
Else
  kvertical = nleg * pileheadkz(1) + nskirt * pileheadkz(3)
End If


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 batter forces.
'
If verticalface = True Then
  batter = 1
Else
  batter = 2
End If

For i = 1 To 2
```

```
  If deckbaybraces = True Then
    lbcap(i, 0) = cap(i, nbay + 1)
    ubcap(i, 0) = capu(i, nbay + 1)
  Else
    lbcap(i, 0) = dlcap(i)
    ubcap(i, 0) = dlcap(i)
  End If
  For j = 1 To nbay
    If jointsoff = False Then jcap(i, j) = jcap(i, j) + batter * legfh(i, j)
    lbcap(i, j) = cap(i, j) + batter * legfh(i, j)
    ubcap(i, j) = capu(i, j) + batter * legfh(i, j)
    If j = 1 And jointsoff = False 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
'
' Foundation
'
  fcap(i) = Application.Max(pilehorzcap + batter * legfh(i, nbay + 1), 0.001)
  If pileloadcomp(i) = 0 Then
    rsrc(i) = 0
  Else
    rsrc(i) = (avpilecapcomp(i) + extrapilecap(i)) / pileloadcomp(i)
  End If
  If pileloadtens(i) = 0 Then
    rsrt(i) = 0
  Else
    rsrt(i) = (avpilecaptens(i) + extrapilecap(i)) / pileloadtens(i)
  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
```

```
'
' This module contains routines relevant to the calculation of hydrodynamic forces on
' a structure.
'
' Last modified by : Zhaohui Jin
'         on : May 22, 1998
' Modification made for the calculation of sensitivity analysis and spatial loading effect

Public wavetheory

Dim mbar1(2, 30), mbar2(2, 30)

Sub kinematics() ' Start KINEMATICS

judge = 0
If (pltype = 2 Or pltype = 3 Or pltype = 4) And mtpltype <> 1 Then
  judge = 1
Else If mtpltype = 1 Then judge = 2
End If

If DialogSheets("OPTIONS").CheckBoxes("forcestokes").Value = xlOn Then
  forcestokes = True
Else
  forcestokes = False
End If

'$$$$$  load spatial effect consideration
'If DialogSheets("OPTIONS").CheckBoxes("Spatial").Value = xlOn Then
  'spatial = True
'Else
  'spatial = False
'End If

stk = 0
If wdep = 0 Then wdep = 1
If wavp = 0 Then wavp = 1
If wavh = 0 Then wavh = 1

If (((wdep + sdep) / wavp ^ 2 > 0.1) And (wavh / wavp ^ 2 > 0.05)) Or forcestokes = True Then
  shallow = False
  wavetheory = "Stokes"
Else
  shallow = True
  wavetheory = "Cnoidal"
End If

'
' Find water kinematics due to wave using either stokes or cnoidal theory
'

If shallow = True Then
  Module8.cnoidal
Else
  Module8.stokes
End If

'
' 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)
   If spatial = True And judge = 1 Then
      For j = 1 To 2    'BS or EO
         For k = 1 To 5     'center or side
            spvel(j, k, i) = spwvel(j, k, i) + cvel(i)
         Next k
      Next j
   End If
   '$$$$$$$$ judge=2
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

velocity(nbay + 1) = vel(99)

End Sub '  End KINEMATICS

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

Sub projected_areas() '  Start PROJECTED AREAS
' $$$$$$
'judge = 0
'If (pltype = 2 Or pltype = 3 Or pltype = 4) And mtpltype <> 1 Then
'   judge = 1
'Else If mtpltype = 1 Then judge = 2
'End If

'initialization
For i = 1 To 2
   For j = 1 To 30
      For k = 1 To 3
         hbarea(i, j, k) = 0
         dbarea(i, j, k) = 0
      Next k
   Next j
Next i

For i = 1 To 2
'
' Horizontal brace projected areas.
'
   For j = 1 To nbay + 1
      dummy = 0
      dummy1 = 0 'total vertical horizontal area
      dummy2 = 0 'total diagonal horizontal area
      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
            If spatial = True And mtpltype = 0 Then
               If hbang(j, k) > 0 And hbang(j, k) < 90 Then dummy2 = dummy2 + (hbd(j, k) + 2 * mg(j)) / 12 * hbl(j, k) * (Cos(hbang(j, k) * Pi / 180)) ^ 3
               dummy1 = dummy - dummy2
               If pltype = 2 Then  '6-leg
                  hbarea(2, j, 1) = dummy1 / 3 + dummy2 / 2  'equivalent cylinder at center
                  hbarea(2, j, 2) = dummy1 / 3 + dummy2 / 4  'equ cylinder at side
               End If
               If pltype = 3 Or pltype = 4 Then  '8 or 12 leg
                  hbarea(2, j, 1) = dummy1 / 4 + dummy2 / 3  'one equivalent cylinder at center
```

```
            hbarea(2, j, 2) = dummy1 / 4 + dummy2 / 6   'one equ cylinder at side
          End If

        'ElseIf spatial = True And mtpltype = 1 Then

        End If

      Else  'broad side loading,
        dummy = dummy + (hbd(j, k) + 2 * mg(j)) / 12 * hbl(j, k) * (Sin(hbang(j, k) * Pi / 180)) ^ 3
        If spatial = True And mtpltype = 0 Then
          If hbang(j, k) > 0 And hbang(j, k) < 90 Then dummy2 = dummy2 + (hbd(j, k) + 2 * mg(j)) / 12 * hbl(j, k) * (Sin(hbang(j, k) * Pi / 180)) ^ 3
          dummy1 = dummy - dummy2
          If pltype = 2 Or pltype = 3 Then '6-leg or 8 leg
            hbarea(1, j, 1) = dummy1 / 2 + dummy2 / 2  'one equivalent cylinder
            'hbarea(2, j, 2) = dummy1 / 3 + dummy2 / 4  'equ cylinder at side
          End If
          If pltype = 3 Then  '12 leg
            hbarea(1, j, 1) = dummy1 / 3 + dummy2 / 2  'equivalent cylinder at center
            hbarea(1, j, 2) = dummy1 / 3 + dummy2 / 4   'equ cylinder at side
          End If

        'ElseIf spatial = True And mtpltype=1 Then

        End If
      End If
    Next k
    hbequa(i, j) = dummy

   'If mtpltype = 1 Then hbequa(i, j) = hbequa(i, j) * jacnum(1) * jacnum(2)
  Next j
'
' Main diagonal brace unit projected areas.
'
  If deckbaybraces = True Then nbay = nbay + 1

  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
        'If spatial = True And judge = 1 Then

        'End If

      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)
    If spatial = True And judge = 1 Then
      If pltype = 2 Then
        dbarea(1, j, 1) = dbdequebar(1, j) / 2 + dbdequbbar(1, j) / 2
        dbarea(2, j, 1) = dbdequebar(2, j) / 3 + dbdequbbar(2, j) / 2
        dbarea(2, j, 2) = dbdequebar(2, j) / 3 + dbdequbbar(2, j) / 4
      End If
      If pltype = 3 Then
        dbarea(1, j, 1) = dbdequebar(1, j) / 2 + dbdequbbar(1, j) / 2
        dbarea(2, j, 1) = dbdequebar(2, j) / 4 + dbdequbbar(2, j) / 3
        dbarea(2, j, 2) = dbdequebar(2, j) / 4 + dbdequbbar(2, j) / 6
      End If
```

```
        If pltype = 4 Then
           dbarea(1, j, 1) = dbdequebar(1, j) / 2 + dbdequbbar(1, j) / 3
           dbarea(1, j, 2) = dbdequebar(1, j) / 4 + dbdequbbar(1, j) / 3
           dbarea(2, j, 1) = dbdequebar(2, j) / 4 + dbdequbbar(2, j) / 3
           dbarea(2, j, 2) = dbdequebar(2, j) / 4 + dbdequbbar(2, j) / 6
        End If
        '

      End If

   Next j

   If deckbaybraces = True Then nbay = nbay - 1
Next i

For i = 1 To 2
'
' Deck leg unit projected areas. Deck legs are accounted for as level(0) jacket legs.
'
If nleg > 4 Then
   jld(i, 0) = (4 * dld(1) + (nleg - 4) * dld(2)) / nleg
Else
   jld(i, 0) = dld(1)
End If

If deckbaybraces = True Then
   dbdequebar(i, 0) = dbdequebar(i, nbay + 1)
   dbdequbbar(i, 0) = dbdequbbar(i, nbay + 1)

   dbarea(i, 0, 1) = dbarea(i, nbay + 1, 1)
   dbarea(i, 0, 2) = dbarea(i, nbay + 1, 2)

   Else
   dbdequebar(i, 0) = 0
   dbdequbbar(i, 0) = 0

   dbarea(i, 0, 1) = 0
   dbarea(i, 0, 2) = 0

   ' for multi multi-leg platform, some modification here needed!
End If


Next i

If pltype = 8 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 (designed = 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 (designed = 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
   If mtpltype = 1 Then
      faerobar(i) = faerobar(i) * jacnum(1) * jacnum(2)
      fhydrobar(i) = fhydrobar(i) * jacnum(1) * jacnum(2)
   End If
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
'
judge = 0
If (pltype = 2 Or pltype = 3 Or pltype = 4) And mtpltype <> 1 Then
   judge = 1
Else If mtpltype = 1 Then judge = 2
End If


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
'$$$$$$$$$$$$
   cumdfhydro = 0
   For j = 1 To ndeck
      dummy = dummy + wdbias * fhydro(i, j) + faero(i, j)
      cumdfhydro = cumdfhydro + wdbias * fhydro(i, j)
   Next j
   cumf(i, 0) = dummy * lf

   If mtpltype = 1 Then cumf(i, 0) = cumf(i, 0) * jacnum(1) * jacnum(2)

   cumdfhydro = cumdfhydro * lf   'deck force from only hydro
```

```
ploadwdbias(i) = cumdfhydro / wdbias
For j = 0 To nbay  'including deck
  dequ(i, j) = dbdequebar(i, j) + dbdequbbar(i, j) + dequapp(j)
  'If spatial = True And judge = 1 Then
  '  spdequ(i, j, 1) = dbarea(i, j, 1)  'add appetance area at center later
  '  spdequ(i, j, 2) = dbarea(i, j, 2)
  'End If

  'judge=2

  If nleg > 4 Then
    dequ(i, j) = dequ(i, j) + 4 * (jld(1, j) + 2 * mg(j)) / 12 + (nleg - 4) * (jld(2, j) + 2 * mg(j)) / 12
    If spatial = True And judge = 1 Then
      If i = 1 Then
        If pltype = 2 Then
          spdequ(i, j, 1) = dbarea(i, j, 1) + dequapp(j) / 2 + 2 * (jld(1, j) + 2 * mg(j)) / 12 + (jld(2, j) + 2 * mg(j)) / 12
        ElseIf pltype = 3 Then
          spdequ(i, j, 1) = dbarea(i, j, 1) + dequapp(j) / 2 + 2 * (jld(2, j) + 2 * mg(j)) / 12 + 2 * (jld(1, j) + 2 * mg(j)) / 12
         ' spdequ(i, j, 2) = dbarea(i, j, 2) + 2 * (jld(1, j) + 2 * mg(j)) / 12 + 2 * (jld(2, j) + 2 * mg(j)) / 12
        Else   '1 - center, 2 - side
          spdequ(i, j, 1) = dbarea(i, j, 1) + dequapp(j) + 4 * (jld(2, j) + 2 * mg(j)) / 12
          spdequ(i, j, 2) = dbarea(i, j, 2) + 2 * (jld(1, j) + 2 * mg(j)) / 12 + 2 * (jld(2, j) + 2 * mg(j)) / 12
        End If
      Else
        If pltype = 3 Then
          spdequ(i, j, 1) = dbarea(i, j, 1) + dequapp(j) / 2 + 2 * (jld(2, j) + 2 * mg(j)) / 12
          spdequ(i, j, 2) = dbarea(i, j, 2) + 2 * (jld(1, j) + 2 * mg(j)) / 12
        ElseIf pltype = 4 Then
          spdequ(i, j, 1) = dbarea(i, j, 1) + dequapp(j) / 2 + 3 * (jld(2, j) + 2 * mg(j)) / 12
          spdequ(i, j, 2) = dbarea(i, j, 2) + 2 * (jld(1, j) + 2 * mg(j)) / 12 + (jld(2, j) + 2 * mg(j)) / 12
        Else
          spdequ(i, j, 1) = dbarea(i, j, 1) + dequapp(j) + 2 * (jld(2, j) + 2 * mg(j)) / 12
          spdequ(i, j, 2) = dbarea(i, j, 2) + 2 * (jld(1, j) + 2 * mg(j)) / 12
        End If
      End If
    End If
  Else
    dequ(i, j) = dequ(i, j) + nleg * (jld(1, j) + 2 * mg(j)) / 12
  End If

  If mlpltype = 1 Then

    spdequ(i, j, 1) = dequ(i, j)
    '$$$$$$ no spatial effect
    dequ(i, j) = dequ(i, j) * jacnum(1) * jacnum(2)

  End If

Next j
k = 0
DepthCheck = False
For j = 1 To 100
  If (crest - elevbar < vcrest ^ 2 / g) And (designed = 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
    If spatial = True And judge = 1 Then
      If i = 1 Then
        If pltype = 2 Or pltype = 3 Then
          spf(i, j) = wjbias * cdmem * interval * (spdequ(i, k, 1) * spvel(i, 1, j) ^ 2) * 2 / 1000 * lf   '2 side cylinders
        Else  'BS loading for 12-leg, velocity 1 for side, 2 for center  !!!!!!
          spf(i, j) = wjbias * cdmem * interval * (spdequ(i, k, 1) * spvel(i, 2, j) ^ 2 + 2 * spdequ(i, k, 2) * spvel(i, 1, j) ^ 2) / 1000 * lf '2 side cylinders + 1 ce
nter
        End If
      Else
        If pltype = 3 Or pltype = 4 Then
          spf(i, j) = wjbias * cdmem * interval * (spdequ(i, k, 1) * spvel(i, 1, j) ^ 2 + spdequ(i, k, 2) * spvel(i, 2, j) ^ 2) * 2 / 1000 * lf '2 side cylinders+2 cent
er
```

```
            Else
               spf(i, j) = wjbias * cdmem * interval * (spdequ(i, k, 1) * spvel(i, 1, j) ^ 2 + 2 * spdequ(i, k, 2) * spvel(i, 2, j) ^ 2) / 1000 * lf '2 side cylinders + 1 ce
nter
            End If
          End If
        ElseIf mtpltype = 1 Then

          If cylnum(i) * 2 - jacnum(i) = 0 Then
             center = 2
          Else
             center = 1
          End If

          nn = cylnum(1) * cylnum(2) / cylnum(i)

          For zz = 2 To cylnum(i)
             spf(i, j) = spf(i, j) + 2 * wjbias * cdmem * interval * (nn * spdequ(i, k, 1) * spvel(i, zz, j) ^ 2) / 1000 * lf
          Next zz
          spf(i, j) = spf(i, j) + center * wjbias * cdmem * interval * (nn * spdequ(i, k, 1) * spvel(i, 1, j) ^ 2) / 1000 * lf
        End If

      Else
        'f(i, j) = wjbias * cdmem * (dequ(i, k + 1) * interval + hbequa(i, k + 1) + boatl(i)) * vel(j) ^ 2 / 1000 * lf
        f(i, j) = wjbias * cdmem * (dequ(i, k + 1) * interval + hbequa(i, k + 1)) * vel(j) ^ 2 / 1000 * lf

        If mtpltype = 1 Then f(i, j) = wjbias * cdmem * (dequ(i, k + 1) * interval + hbequa(i, k + 1) * jacnum(1) * jacnum(2)) * vel(j) ^ 2 / 1000 * lf

        If spatial = True And judge = 1 Then
          If i = 1 Then
            If pltype = 2 Or pltype = 3 Then
              spf(i, j) = wjbias * cdmem * ((interval * spdequ(i, k + 1, 1) + hbarea(i, k + 1, 1)) * spvel(i, 1, j) ^ 2) * 2 / 1000 * lf '2 side cylinders
            Else  'BS loading for 12-leg, velocity 1 for side, 2 for center  !!!!!!
              spf(i, j) = wjbias * cdmem * ((interval * spdequ(i, k + 1, 1) + hbarea(i, k + 1, 1)) * spvel(i, 2, j) ^ 2 + 2 * (interval * spdequ(i, k + 1, 2) + hbarea(i,
k + 1, 2)) * spvel(i, 1, j) ^ 2) / 1000 * lf '2 side cylinders + 1 center
            End If
          Else
            If pltype = 3 Or pltype = 4 Then
              spf(i, j) = wjbias * cdmem * ((interval * spdequ(i, k + 1, 1) + hbarea(i, k + 1, 1)) * spvel(i, 1, j) ^ 2 + (interval * spdequ(i, k + 1, 2) + hbarea(i, k +
1, 2)) * spvel(i, 2, j) ^ 2) * 2 / 1000 * lf '2 side cylinders+2 center
            Else
              spf(i, j) = wjbias * cdmem * ((interval * spdequ(i, k + 1, 1) + hbarea(i, k + 1, 1)) * spvel(i, 1, j) ^ 2 + 2 * (interval * spdequ(i, k + 1, 2) + hbarea(i,
k + 1, 2)) * spvel(i, 2, j) ^ 2) / 1000 * lf '2 side cylinders + 1 center
            End If
          End If
        'ElseIf spatial = True And judge = 2 Then
        ElseIf mtpltype = 1 Then
          If cylnum(i) * 2 - jacnum(i) = 0 Then
            center = 2
          Else
            center = 1
          End If

          nn = cylnum(1) * cylnum(2) / cylnum(i)
          For zz = 2 To cylnum(i)
            spf(i, j) = spf(i, j) + 2 * wjbias * cdmem * nn * (interval * spdequ(i, k + 1, 1) + hbequa(i, k + 1)) * spvel(i, zz, j) ^ 2 / 1000 * lf
          Next zz
          spf(i, j) = spf(i, j) + center * wjbias * cdmem * nn * (interval * spdequ(i, k + 1, 1) + hbequa(i, k + 1)) * spvel(i, 1, j) ^ 2 / 1000 * lf

        End If

        k = k + 1
      End If
'
'  Add in forces on boatlanding, assuming landings are at MWL
'
      If elevbar <= wdep And DepthCheck = False Then
        f(i, j) = f(i, j) + wjbias * cdmem * boatl(i) * vel(j) ^ 2 / 1000 * lf
        'boatlanding still at crest for spatial load effect
        If spatial = True Then spf(i, j) = spf(i, j) + wjbias * cdmem * boatl(i) * vel(j) ^ 2 / 1000 * lf
        DepthCheck = True
      End If
      elev(j) = elevbar
```

```
      elevbar = elevbar - interval
      If elevbar < 0 Then Exit For
      cumf(i, j) = cumf(i, j - 1) + f(i, j)
      If spatial = True Then spcumf(i, j) = spcumf(i, j - 1) + spf(i, j)
    Next j
    cumf(i, 100) = cumf(i, 99)
    If spatial = True Then spcumf(i, 100) = spcumf(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
      dummy1 = 0
      For k = 1 To ndeck
        dummy = dummy + wdbias * fhydro(i, k) * (fhydroh(k) + wdep - elevation(j)) + faero(i, k) * (faeroh(k) + wdep - elevation(j))
        '$$$$$$
        dummy1 = dummy1 + wdbias * fhydro(i, k) * (fhydroh(k) + wdep - elevation(j))
      Next k
      mbar(i, j) = dummy * lf
      mbar2(i, j) = dummy1 * lf 'only moment from hydro
      ' sense vector component of leg force wrt wdbias
      plfhwdbias(i, j) = mbar2(i, j) / wdbias
      mbar1(i, j) = mbar(i, j)  'moment from deck hydro and aero
    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)  'overturn moment from both deck and jacke hydro plus aero
        If k > nbay Then lht(i, k) = bcw(i)
        legf(i, k) = mbar(i, k) / lht(i, k)     'lht(I,K) correct?
        legfh(i, k) = legf(i, k) * Sin(alpha(i))
      Next k
    Next j
    For j = 1 To nbay + 1
      If j > nbay Then lht(i, j) = bcw(i)
      plfhwdbias(i, j) = plfhwdbias(i, j) / lht(i, j) * Sin(alpha(i))
      plfhwjbias(i, j) = (mbar(i, j) - mbar1(i, j)) / lht(i, j) * Sin(alpha(i)) / wjbias
    Next j
Next i
'
' Find capacity reduction in top jacket bay due to deck bay action.
'
For i = 1 To 2
    If pltype = 7 Or pltype = 8 Then
      dlmbar(i, 1) = Abs(Application.Min((faerobar(i) + fhydrobar(i)) * bayh(0) * 1.5, dlm(1)))
    ElseIf deckbaybraces = False Then
      If nleg > 4 Then
        dlmbar(i, 0) = Abs(Application.Min(Application.Min((faerobar(i) + fhydrobar(i)) / nleg * bayh(0) * ((bayh(0) / (2 * e * dli(1) / 144) + 1 / dlcm(i, 1)) / (ba
yh(0) / (e * dli(1) / 144) + 1 / dlcm(i, 1))), dlm(1)), Application.Min((faerobar(i) + fhydrobar(i)) / nleg * bayh(0) * ((bayh(0) / (2 * e * dli(2) / 144) + 1 / dlcm(i,
2)) / (bayh(0) / (e * dli(2) / 144) + 1 / dlcm(i, 2))), dlm(2))))
        dlmbar(i, 1) = Abs(Application.Min(dlmbar(i, 0) - (faerobar(i) + fhydrobar(i)) / nleg * bayh(0), dlm))
      Else
        dlmbar(i, 0) = Abs(Application.Min((faerobar(i) + fhydrobar(i)) / nleg * bayh(0) * ((bayh(0) / (2 * e * dli(1) / 144) + 1 / dlcm(i, 1)) / (bayh(0) / (e * dli(1)
/ 144) + 1 / dlcm(i, 1))), dlm(1)))
        dlmbar(i, 1) = Abs(Application.Min(dlmbar(i, 0) - (faerobar(i) + fhydrobar(i)) / nleg * bayh(0), dlm(1)))
      End If
    End If

    If deckbaybraces = True Then
      shear(i) = 0
    Else
      If pltype = 7 Or pltype = 8 Then
```

```
          shear(i) = dlmbar(i, 1) / (bayh(1) + 30)
        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)

        If spatial = True Then spmeanload(i, k) = spcumf(i, j) 'meanload considering spatial effect

        '$$$$$  sens vector of meanload w.r.t. wjbias
        ploadwjbias(i, k) = (meanload(i, k) - cumf(i, 0)) / wjbias
        elevbar = elevbar - interval
      Next j
      meanload(i, nbay + 1) = cumf(i, j - 1)  ' last j=101    for foundation load
      If spatial = True Then spmeanload(i, nbay + 1) = spcumf(i, j - 1)
      ploadwjbias(i, nbay + 1) = (meanload(i, nbay + 1) - cumf(i, 0)) / wjbias
    Next i
    '
    ' Find pile loads
    '
    For i = 1 To 2  '   broadside, end-on load
    '
    ' Find forces due to global overturning
    '
      If pltype = 6 Then  '   Multi Jackets
        zforce = 2 * mbar(i, nbay + 1) / msw
      Else
        zforce = 0
      End If
      plfwdbias = mbar2(i, nbay + 1) / bcw(i) / wdbias 'axial force partial derivatives
      plfwjbias = (mbar(i, nbay + 1) - mbar1(i, nbay + 1)) / bcw(i) / wjbias

      If pltype = 5 Then
        pileloadcomp(i) = Application.Max((zforce + qdeck) / (nleg + nskirt) + (1 / Sin(offangle)) ^ (i - 1) * legf(i, nbay + 1) / (nskirtbs / 2 + nskirtcorner / 3 + 1), 0)
        pileacwdbias(i) = Application.Max((1 / Sin(offangle)) ^ (i - 1) * plfwdbias / (nskirtbs / 2 + nskirtcorner / 3 + 1), 0)
        pileacwjbias(i) = Application.Max((1 / Sin(offangle)) ^ (i - 1) * plfwjbias / (nskirtbs / 2 + nskirtcorner / 3 + 1), 0)

        pileloadtens(i) = Application.Max((zforce - qdeck) / (nleg + nskirt) + (1 / Sin(offangle)) ^ (i - 1) * legf(i, nbay + 1) / (nskirteo / 2 + nskirtcorner / 3 + 1), 0)
        pileatwdbias(i) = Application.Max((1 / Sin(offangle)) ^ (i - 1) * plfwdbias / (nskirteo / 2 + nskirtcorner / 3 + 1), 0)
        pileatwjbias(i) = Application.Max((1 / Sin(offangle)) ^ (i - 1) * plfwjbias / (nskirteo / 2 + nskirtcorner / 3 + 1), 0)
      ElseIf pltype = 7 Or pltype = 8 Then

        If Abs(dlmbar(i, 1) - dlm(1)) < 0.01 Then
          pmomwdbias = mbar2(i, 2) / wdbias        'moment wrt wdbias
          pmomwjbias = (mbar(i, 2) - mbar1(i, 2)) / wjbias
        Else
          pmomwdbias = mbar2(i, 2) / wdbias + fhydrobar(i) * bayh(0) * 1.5 / wdbias / 3
          pmomwjbias = (mbar(i, 2) - mbar1(i, 2)) / wjbias
        End If

        If pltype = 8 Then
          qdeck = qdeck + 3 * pretension * Sin(theta(1, 1, 1)) ^ 2
          pileloadcomp(i) = Application.Max((zforce + qdeck) + (mbar(i, 2) + dlmbar(i, 1) / 3) / bcw(i), 0)
          pileloadtens(i) = Application.Max((zforce) + (mbar(i, 2) + dlmbar(i, 1) / 3) / bcw(i), 0)

        Else
          If dbtype(1, 1, 1) = 1 Then
            pileloadcomp(i) = Application.Max((zforce + qdeck) + (mbar(i, 2) + dlmbar(i, 1) / 3) / bcw(i), 0)
            pileloadtens(i) = Application.Max((zforce) + (mbar(i, 2) + dlmbar(i, 1) / 3) / bcw(i), 0)
          Else
```

```
      pileloadcomp(i) = Application.Max((zforce) + (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)
    End If
  End If

  pileacwdbias(i) = Application.Max(pmomwdbias / bcw(i), 0)
  pileacwjbias(i) = Application.Max(pmomwjbias / bcw(i), 0)
  pileatwdbias(i) = pileacwdbias
  pileatwjbias(i) = pileacwjbias      'partial derivatives all the same for all the cases


 Else
   If i = 2 Then
     pileloadcomp(i) = Application.Max((zforce + qdeck) / (nleg + nskirt) + legf(i, nbay + 1) / ((nskirteo + nskirtcorner) / 2 + 2 + npileeo / 2), 0)
     pileacwdbias(i) = Application.Max(plfwdbias / ((nskirteo + nskirtcorner) / 2 + 2 + npileeo / 2), 0)
     pileacwjbias(i) = Application.Max(plfwjbias / ((nskirteo + nskirtcorner) / 2 + 2 + npileeo / 2), 0)

     pileloadtens(i) = Application.Max((zforce - qdeck) / (nleg + nskirt) + legf(i, nbay + 1) / ((nskirteo + nskirtcorner) / 2 + 2 + npileeo / 2), 0)
     pileatwdbias(i) = Application.Max(plfwdbias / ((nskirteo + nskirtcorner) / 2 + 2 + npileeo / 2), 0)
     pileatwjbias(i) = Application.Max(plfwjbias / ((nskirteo + nskirtcorner) / 2 + 2 + npileeo / 2), 0)

   Else
     pileloadcomp(i) = Application.Max((zforce + qdeck) / (nleg + nskirt) + legf(i, nbay + 1) / ((nskirtbs + nskirtcorner) / 2 + 2 + npilebs / 2), 0)
     pileacwdbias(i) = Application.Max(plfwdbias / ((nskirtbs + nskirtcorner) / 2 + 2 + npilebs / 2), 0)
     pileacwjbias(i) = Application.Max(plfwjbias / ((nskirtbs + nskirtcorner) / 2 + 2 + npilebs / 2), 0)

     pileloadtens(i) = Application.Max((zforce - qdeck) / (nleg + nskirt) + legf(i, nbay + 1) / ((nskirtbs + nskirtcorner) / 2 + 2 + npilebs / 2), 0)
     pileatwdbias(i) = Application.Max(plfwdbias / ((nskirtbs + nskirtcorner) / 2 + 2 + npilebs / 2), 0)
     pileatwjbias(i) = Application.Max(plfwjbias / ((nskirtbs + nskirtcorner) / 2 + 2 + npilebs / 2), 0)

   End If
 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 + wdbias * 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

Sub stormtable()
'
' Tabular Output for Storm Parameters
'
Worksheets("Sheet5").Cells(1, 2) = "SURGE, WIND, WAVE AND CURRENT"

Worksheets("Sheet5").Cells(3, 2) = "Surge / Tide Level (ft)"
Worksheets("Sheet5").Cells(5, 2) = "Wind Velocity, 30 ft Elevation (mph)"
Worksheets("Sheet5").Cells(7, 2) = "Wave Height (ft)"
Worksheets("Sheet5").Cells(7, 8) = "Wave Kinematic Theory Used: " & wavetheory
Worksheets("Sheet5").Cells(8, 2) = "Wave Period (sec)"
Worksheets("Sheet5").Cells(10, 2) = "Current Velocity, SWL (fps)"
Worksheets("Sheet5").Cells(11, 2) = "Current Velocity, Mudline (fps)"
Worksheets("Sheet5").Cells(12, 2) = "Current Velocity Profile"
Worksheets("Sheet5").Cells(3, 6) = sdep
Worksheets("Sheet5").Cells(5, 6) = vrh
Worksheets("Sheet5").Cells(7, 6) = wavh
Worksheets("Sheet5").Cells(8, 6) = wavp
Worksheets("Sheet5").Cells(10, 6) = cswl
Worksheets("Sheet5").Cells(11, 6) = cmdl
```

```
If cmdl = 0 And cswl = 0 Then
   Worksheets("Sheet5").Cells(12, 6) = "No Current"
ElseIf cprof = 1 Then
   Worksheets("Sheet5").Cells(12, 6) = "Linear"
ElseIf cprof = 2 Then
   Worksheets("Sheet5").Cells(12, 6) = "Quadratic"
ElseIf cprof = 3 Then
   Worksheets("Sheet5").Cells(12, 6) = "Constant"
End If

Worksheets("Sheet5").Cells(14, 2) = "LOAD FACTORS AND FORCE COEFFICIENTS"

Worksheets("Sheet5").Cells(16, 2) = "Global Load Factor"
Worksheets("Sheet5").Cells(16, 6) = lf

Worksheets("Sheet5").Cells(18, 2) = "Water Kinematics:"
Worksheets("Sheet5").Cells(19, 2) = "    Current Blockage, Cb (EO)"
Worksheets("Sheet5").Cells(20, 2) = "    Current Blockage, Cb (BS)"
Worksheets("Sheet5").Cells(21, 2) = "    Directional Spreading, wkf"
Worksheets("Sheet5").Cells(19, 6) = cbeo
Worksheets("Sheet5").Cells(20, 6) = cbbs
Worksheets("Sheet5").Cells(21, 6) = ds


Worksheets("Sheet5").Cells(23, 2) = "Hydrodynamic Drag Coefficients, Cd:"
Worksheets("Sheet5").Cells(24, 2) = "    All Members and Appurtenances"
Worksheets("Sheet5").Cells(24, 6) = cdj

For i = 1 To ndeck
   Worksheets("Sheet5").Cells(24 + i, 2) = "    Deck " & i
   Worksheets("Sheet5").Cells(24 + i, 6) = cdd(i)
   Worksheets("Sheet5").Cells(24 + i + 2 + ndeck, 2) = "    Deck " & i
   Worksheets("Sheet5").Cells(24 + i + 2 + ndeck, 6) = wsc(i)
Next i

Worksheets("Sheet5").Cells(24 + ndeck + 2, 2) = "Wind Speed Coefficients, Cs:"

End Sub
```

```
'
' CNOIDAL WAVE KINEMATICS
'
' Created by: Zhaohui Jin
' Created on: 3/31/98
'
' The subroutine and supporting functions in this module determine water
' particle horizontal velocities under the wave crest using cnoidal wave
' theory.
'
' Last modified by:  James Stear
' Last modified on:  3/31/97
'
' Nature of last modification:
'
' Adapted program into TOPCAT program.
'

Dim q(2, 5)

Sub cnoidal()

judge = 0
If (pltype = 2 Or pltype = 3 Or pltype = 4) And mtpltype <> 1 Then
   judge = 1
Else If mtpltype = 1 Then judge = 2
End If

depth = sdep + wdep
eps = wavh / depth
ki = solve_ki(wavh, depth, wavp, eps)

If ki = -1 Then
   MsgBox "Cnoidal theory does not converge.  Check wave parameters."
   Workbooks(1).Sheets(1).Activate
   End
Else
   eta0 = E0(ki) / K0(ki)
   hdep = depth * (1 - eps * h_1(ki, eta0) - eps ^ 2 * h_2(ki, eta0))  'trough depth
End If

cncelerity = Sqr(g * depth) * (1 + eps * c_1(ki, eta0) + eps ^ 2 * c_2(ki, eta0))
wn = 2 * Pi / cncelerity / wavp     'wave number

cn2 = cn_2(0, ki, eta0)
cn4 = cn_4(0, ki, eta0)

crest = hdep + wavh
wvcrest = H_velocity(wavh, depth, 0, ki, eta0, eps, crest, hdep, cn2, cn4) * ds


 For i = 1 To 100
    If elev(i) > crest Then
       wvel(i) = 0
    Else
       wvel(i) = H_velocity(wavh, depth, 0, ki, eta0, eps, elev(i), hdep, cn2, cn4) * ds
    End If
 Next i

 For i = 1 To 2      'initialization not dispensible
    For j = 1 To 5
       For k = 1 To 100
          spwvel(i, j, k) = 0
       Next k
    Next j
 Next i

 If spatial = True And judge = 1 Then
    If pltype = 2 Then
       q1 = ki / Pi * wn * (bcw(1) / 2 + tcw(1) / 4) 'BS side cylinder
       q2 = ki / Pi * wn * (bcw(2) + tcw(2)) / 4   'EO side cylinder
       surelev(1, 1) = sur(eps, ki, eta0, q1, depth)  'surface elevation
```

```
    surelev(2, 2) = sur(eps, ki, eta0, q2, depth)  'EO side
    surelev(2, 1) = sur(eps, ki, eta0, 0, depth)   'EO center
    For j = 1 To 100
      spwvel(1, 1, j) = H_velocity(wavh, depth, q1, ki, eta0, eps, elev(j), hdep, cn2, cn4) * ds 'BS side
      spwvel(2, 1, j) = H_velocity(wavh, depth, 0, ki, eta0, eps, elev(j), hdep, cn2, cn4) * ds 'EO center
      spwvel(2, 2, j) = H_velocity(wavh, depth, q2, ki, eta0, eps, elev(j), hdep, cn2, cn4) * ds 'EO side
    Next j
  Else
    q1 = ki / Pi * wn * (bcw(1) / 2 + tcw(1) / 4) 'BS side cylinder
    q2 = ki / Pi * wn * (bcw(2) + tcw(2)) / 4 'EO side cylinder
    q3 = ki / Pi * wn * mcw / 2        'EO center cylinder
    surelev(1, 1) = sur(eps, ki, eta0, q1, depth)  'surface elevation: BS side
    If pltype = 4 Then surelev(1, 2) = sur(eps, ki, eta0, 0, depth) 'BS center
    surelev(2, 1) = sur(eps, ki, eta0, q3, depth) 'EO center
    surelev(2, 2) = sur(eps, ki, eta0, q2, depth)  'EO side
    For j = 1 To 100
      spwvel(1, 1, j) = H_velocity(wavh, depth, q1, ki, eta0, eps, elev(j), hdep, cn2, cn4) * ds
      If pltype = 4 Then spwvel(1, 2, j) = H_velocity(wavh, depth, 0, ki, eta0, eps, elev(j), hdep, cn2, cn4) * ds 'BS center
      spwvel(2, 1, j) = H_velocity(wavh, depth, q3, ki, eta0, eps, elev(j), hdep, cn2, cn4) * ds 'EO center
      spwvel(2, 2, j) = H_velocity(wavh, depth, q2, ki, eta0, eps, elev(j), hdep, cn2, cn4) * ds
    Next j
  End If


ElseIf spatial = True And judge = 2 Then
  For i = 1 To 2
    For j = 1 To cylnum(i)
      If cylnum(i) * 2 - jacnum(i) = 0 Then
        q(i, j) = (j - 0.5) * totlen(i) / jacnum(i) * ki / Pi * wn
      Else
        q(i, j) = (j - 1) * totlen(i) / jacnum(i) * ki / Pi * wn
      End If
      'cn2=
      'cn4=
      For k = 1 To 100
        spwvel(i, j, k) = H_velocity(wavh, depth, q(i, j), ki, eta0, eps, elev(k), hdep, cn2, cn4) * ds
      Next k
    Next j
  Next i
End If

End Sub


'$$$$$$
Function sur(eps, ki, eta0, q, d)
  cn2 = cn_2(q, ki, eta0)
  h1 = h_1(ki, eta0)
  h2 = h_2(ki, eta0)
  sur = eps * (cn2 - h1) - eps ^ 2 * (0.75 * cn2 * (1 - cn2) + h2)
  sur = sur * d
End Function
'
'  Function solve_ki()
'
Function solve_ki(hh, dd, tt, eps)

Dim ki(3), vlu(3)

ki(1) = 0.0000001
ki(3) = 0.9999999
vlu(1) = equ(hh, dd, tt, ki(1), eps)
vlu(3) = equ(hh, dd, tt, ki(3), eps)

If vlu(1) = 0 Then solve_ki = ki(1)
If vlu(3) = 0 Then solve_ki = ki(3)
If vlu(1) * vlu(3) > 0 Then
  solve_ki = -1
Else
  ki(2) = 0.5
  dx = (ki(3) - ki(1)) / 2
  vlu(2) = equ(hh, dd, tt, ki(2), eps)
```

```
  Do
    dx = dx / 2
    If vlu(2) = 0 Then
      solve_ki = ki(2)
    ElseIf vlu(1) * vlu(2) > 0 Then
      ki(1) = ki(2)
      ki(2) = ki(1) + dx
    Else
      ki(3) = ki(2)
      ki(2) = ki(1) + dx
    End If
    vlu(1) = equ(hh, dd, tt, ki(1), eps)
    vlu(2) = equ(hh, dd, tt, ki(2), eps)
    vlu(3) = equ(hh, dd, tt, ki(3), eps)
  Loop Until Abs(dx) <= 0.00001
  solve_ki = ki(2)
End If
End Function


'
' Function E0()
'
Function E0(ki)

dx = Pi / 2 / 100 / 3
E0 = dx * (Sqr(1 - ki ^ 2 * (Sin(0)) ^ 2) + Sqr(1 - ki ^ 2 * (Sin(Pi / 2)) ^ 2))
For i = 1 To 99 Step 2
  E0 = E0 + 4 * dx * Sqr(1 - ki ^ 2 * (Sin(i / 100 * Pi / 2)) ^ 2)
Next i
For i = 2 To 98 Step 2
  E0 = E0 + 2 * dx * Sqr(1 - ki ^ 2 * (Sin(i / 100 * Pi / 2)) ^ 2)
Next i

End Function


'
' Function K0(ki)
'
Function K0(ki)

dx = Pi / 2 / 1000 / 3
K0 = dx * (1 / Sqr(1 - ki ^ 2 * (Sin(0)) ^ 2) + 1 / Sqr(1 - ki ^ 2 * (Sin(Pi / 2)) ^ 2))
For i = 1 To 999 Step 2
  K0 = K0 + 4 * dx / Sqr(1 - ki ^ 2 * (Sin(i / 1000 * Pi / 2)) ^ 2)
Next i
For i = 2 To 998 Step 2
  K0 = K0 + 2 * dx / Sqr(1 - ki ^ 2 * (Sin(i / 1000 * Pi / 2)) ^ 2)
Next i

End Function


'
' Function c_1()
'
Function c_1(ki, eta0)

ki2 = 1 - ki ^ 2
c_1 = (2 - ki ^ 2 - 3 * eta0) / ki ^ 2 / 2

End Function


'
' Function c_2()
'
Function c_2(ki, eta0)

ki2 = 1 - ki ^ 2
c_2 = (-5 * eta0 * (15 * eta0 + 19 * ki ^ 2 - 38) - 18 * ki ^ 4 - 88 * ki2) / 120 / ki ^ 4
```

```
End Function

'
' Function l_1()
'
Function l_1(ki, eta0)

l_1 = (12 * eta0 + 5 * ki ^ 2 - 10) / 8 / ki ^ 2

End Function

'
' Function equ()
'
Function equ(hh, dd, tt, ki, eps)

eta0 = E0(ki) / K0(ki)
aa = dd / g / tt ^ 2
bb = 3 * eps / (16 * ki ^ 2 * (K0(ki)) ^ 2)
cc = ((1 + eps * c_1(ki, eta0) + eps ^ 2 * c_2(ki, eta0)) / (1 - eps * l_1(ki, eta0))) ^ 2
equ = aa - bb * cc

End Function


'
' Function cn_2()
'
Function cn_2(qq, ki, eta0)

ki2 = 1 - ki ^ 2
j = 1
kk1 = K0(ki)
kk2 = K0(Abs(Sqr(ki2)))
r = Exp(-1 * Pi * kk2 / kk1)
cn_2 = (eta0 - ki2) / ki ^ 2

Do
   aa = 2 * Pi ^ 2 / ki ^ 2 / (kk1) ^ 2 * (j * r ^ j / (1 - r ^ (2 * j))) * Cos(j * qq * Pi / kk1)
   cn_2 = cn_2 + aa
   j = j + 1
Loop Until Abs(aa) < 0.00001

End Function


'
' Function cn_4()
'
Function cn_4(qq, ki, eta0)

cn_4 = (cn_2(qq, ki, eta0)) ^ 2

End Function


'
' Function h_1(ki,eta0)
'
Function h_1(ki, eta0)

ki2 = 1 - ki ^ 2
h_1 = (eta0 - ki2) / ki ^ 2

End Function


'
' Function f_1(ki,eta0)
'
```

```
Function f_1(ki, eta0)

ki2 = 1 - ki ^ 2
f_1 = (-1 * eta0 * (6 * eta0 + 11 * ki ^ 2 - 16) + ki2 * (9 * ki ^ 2 - 10)) / 12 / ki ^ 4

End Function


'
' Function f_2(ki,eta0)
'
Function f_2(ki, eta0)

f_2 = (2 * eta0 + 7 * ki ^ 2 - 6) / 4 / ki ^ 2

End Function


'
' Function H_velocity()
'
Function H_velocity(hh, dd, qq, ki, eta0, eps, ss, h1, cn2, cn4)
'h1 is the trough depth, hh is the wave height

If ss > hh + h1 Then 'ss is the balance position, but current elev(i) method is not correct, so use old method temporily
   H_velocity = 0
   Exit Function
End If

'cn2 = cn_2(qq, ki, eta0)
'cn4 = cn_4(qq, ki, eta0)
ki2 = 1 - ki ^ 2

aa = cn2 - h_1(ki, eta0)
bb = f_1(ki, eta0) + f_2(ki, eta0) * cn2 - cn4
cc = 3 / 4 / ki ^ 2 * (ss / dd) ^ 2 * (ki2 + 2 * (2 * ki ^ 2 - 1) * cn2 - 3 * ki ^ 2 * cn4)
H_velocity = Sqr(g * dd) * (eps * aa + eps ^ 2 * (bb - cc))

End Function


'
' Function h_2()
'
Function h_2(ki, eta0)

ki2 = 1 - ki ^ 2
h_2 = (eta0 * (ki ^ 2 - 2) + 2 * ki2) / 4 / ki ^ 4

End Function
'==========================================================================================
'
' Subroutine stokes
'
'
' Determination of 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.
'
Sub stokes()  ' Start STOKES

' Worksheets(4).Cells(8, 1) = wdep + sdep
' Worksheets(4).Cells(8, 3) = wavp
' Worksheets(4).Cells(8, 4) = wavh
' Worksheets(4).Cells(14, 1) = 0
' Worksheets(4).Calculate
' 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)
```

```
depth = wdep + sdep

'
' First find k, using relationship established by Fenton (1985); thesis eqn 3.14
'

wavek = 0.00001
deltak = 0.01
zeroed1 = False

Do
  s_st = 1 / Application.Cosh(2 * wavek * depth)
  C0_st = Sqr(Application.Tanh(wavek * depth))
  C2_st = C0_st * (2 + 7 * s_st ^ 2) / (4 * (1 - s_st) ^ 2)
  C4_st = C0_st * (4 + 32 * s_st - 116 * s_st ^ 2 - 400 * s_st ^ 3 - 71 * s_st ^ 4 + 146 * s_st ^ 5) / 32 / (1 - s_st) ^ 5

  zero_out1 = 100000 * ((wavek * wavh / 2) ^ 2 * C2_st + (wavek * wavh / 2) ^ 4 * C4_st + (Application.Tanh(wavek * depth)) ^ 0.5 - 2 * Pi / wavp / (g * wav
ek) ^ 0.5)

  If zero_out1 > 0 Then
    wavek = wavek - deltak
    deltak = deltak / 2
    If deltak < 0.000000001 Then zeroed1 = True
  End If

  wavek = wavek + deltak

Loop While zeroed1 = False

'
' Now solve for lambda_st, using the relation from Skjelbreia and Hendrickson (1961)
'

hcos = Application.Cosh(wavek * depth)
hsin = Application.Sinh(wavek * depth)

C1lambda = (8 * hcos ^ 4 - 8 * hcos ^ 2 + 9) / 8 / hsin ^ 4
C2lambda = (3840 * hcos ^ 12 - 4096 * hcos ^ 10 - 2592 * hcos ^ 8 - 1008 * hcos ^ 6 + 5944 * hcos ^ 4 - 1830 * hcos ^ 2 + 147) / 512 / hsin ^ 10 / (6 * hcos ^ 2
- 1)

lambda_st = 1
' deltast = 0.000001

If (1 - 4 * Pi ^ 2 / g / wavp ^ 2 / wavek / Application.Tanh(wavek * depth)) > 0 Then
  MsgBox "Wave kinematics solution will not converge.  Try cnoidal theory."
  Workbooks(1).Sheets(1).Activate
  End
End If

Do

  part1 = 1 + C1lambda * lambda_st ^ 2 + C2lambda * lambda_st ^ 4 - 4 * Pi ^ 2 / g / wavp ^ 2 / wavek / Application.Tanh(wavek * depth)
  part2 = 2 * C1lambda * lambda_st + 4 * C2lambda * lambda_st ^ 3

'  If zero_out2 > 0 Then
'    lambda_st = lambda_st - deltast
'    deltast = deltast / 2
'    If deltast < 0.000001 Then zeroed2 = True
'  End If

'  lambda_st = lambda_st + deltast

  lambda_test = lambda_st
  lambda_st = lambda_st - part1 / part2

' Loop While zeroed2 = False

Loop While Abs(lambda_test - lambda_st) > 0.00001

' stk = Worksheets(4).Cells(8, 6)
' stlambda = Worksheets(4).Cells(14, 1)
```

```
stk = wavek
stlambda = lambda_st

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 * stc
osh ^ 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 - 10
50 * 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


For i = 1 To 2     'initialization not dispensible
  For j = 1 To 5
    For k = 1 To 100
      spwvel(i, j, k) = 0
```

```
    Next k
  Next j
Next i

If spatial = True And mtpltype = 0 Then
  If pltype = 2 Then
    phsangle1 = (bcw(1) + tcw(1)) / 4 * stk
    phsangle2 = (bcw(2) + tcw(2)) / 4 * stk
    surelev(1, 1) = St_Sur(stk, phsangle1)
    surelev(2, 1) = St_Sur(stk, phsangle2)
    For i = 1 To 100
      If elev(j) > crest Then 'need modification here
        spwvel(1, 1, i) = 0 'BS side
        spwvel(2, 1, i) = 0 'EO center
        spwvel(2, 2, i) = 0 'EO side
      Else
        spwvel(1, 1, i) = St_Vel(stk, phsangle1, celerity, elev(i)) * ds
        spwvel(2, 1, i) = St_Vel(stk, 0, celerity, elev(i)) * ds    'center
        spwvel(2, 2, i) = St_Vel(stk, phsangle2, celerity, elev(i)) * ds  'side
      End If
    Next i
  Else
    phsangle1 = (bcw(1) + tcw(1)) / 4 * stk
    phsangle2 = (bcw(2) + tcw(2)) / 4 * stk
    phsangle3 = msw / 2 * stk
    surelev(1, 1) = St_Sur(stk, phsangle1)  'BS side
    If pltype = 4 Then surelev(1, 2) = St_Sur(stk, 0)
    surelev(2, 2) = St_Sur(stk, phsangle2)  'EO side
    surelev(2, 1) = St_Sur(stk, phsangle3)  'EO center
    For i = 1 To 100
      If elev(j) > crest Then 'need modification here
        spwvel(1, 1, i) = 0 'BS side
        spwvel(1, 2, i) = 0
        spwvel(2, 1, i) = 0 'EO center
        spwvel(2, 2, i) = 0 'EO side
      Else
        spwvel(1, 1, i) = St_Vel(stk, phsangle1, celerity, elev(i)) * ds 'BS side
        If pltype = 4 Then spwvel(1, 2, i) = St_Vel(stk, 0, celerity, elev(i)) * ds 'BS center
        spwvel(2, 1, i) = St_Vel(stk, phsangle3, celerity, elev(i)) * ds    'center
        spwvel(2, 2, i) = St_Vel(stk, phsangle2, celerity, elev(i)) * ds  'side
      End If
    Next i

  End If

ElseIf spatial = True And mtpltype = 1 Then
  For i = 1 To 2
    For j = 1 To cylnum(i)
      If cylnum(i) * 2 - jacnum(i) = 0 Then
        q(i, j) = (j - 0.5) * totlen(i) / jacnum(i) * stk
      Else
        q(i, j) = (j - 1) * totlen(i) / jacnum(i) * stk
      End If
      For k = 1 To 100
        If elev(k) > crest Then
          spwvel(i, j, k) = 0
        Else
          spwvel(i, j, k) = St_Vel(stk, q(i, j), celerity, elev(k)) * ds
        End If
      Next k
    Next j
  Next i
End If

End Sub ' End STOKES

Function St_Sur(stk, phsangle)
  If stk = 0 Then Exit Function
  dummy = 0
  For i = 1 To 5      'eta(i) public variables
    dummy = dummy + eta(i) * Cos(i * phsangle)
  Next i
```

```
  St_Sur = dummy / stk
End Function

Function St_Vel(stk, phsangle, c, s)
  dummy = 0
  For i = 1 To 5
    dummy = dummy + i * phi(i) * Application.Cosh(i * stk * s) * Cos(i * phsangle)
  Next i
  St_Vel = dummy * c
End Function
```