Development of a Computational Method for the Prediction of Wave Induced Longitudinal Bending in Ships

Charles Rogers
University of New Orleans

Follow this and additional works at: http://scholarworks.uno.edu/honors_theses

Recommended Citation

This Honors Thesis-Unrestricted is brought to you for free and open access by the Undergraduate Showcase at ScholarWorks@UNO. It has been accepted for inclusion in Senior Honors Theses by an authorized administrator of ScholarWorks@UNO. For more information, please contact scholarworks@uno.edu.
Development of a Computational Method for the Prediction of Wave Induced Longitudinal Bending in Ships

An Honors Thesis
Presented to
The Department of Naval Architecture and Marine Engineering
of the University of New Orleans

In Partial Fulfillment
of the Requirements for the Degree of
Bachelor of Science, with University Honors
and Honors in Naval Architecture and Marine Engineering

by
Charles Rogers
May 2012
Acknowledgements

Many thanks to all of the professors and faculty in the Naval Architecture and Marine Engineering department at UNO for all the lessons I have learned over the past four years. This thesis would not be possible without them. I also must thank the UNO Office of Research and Sponsored Programs’ Summer Undergraduate Experience for helping to lay the seeds of this thesis.

Thanks to Dr. Lothar Birk for his help on the review committee of this thesis.

Very special thanks to Dr. Brandon Taravella, my thesis director. Without you I would have been lost at sea.
## Contents

1 Introduction .................................................. 5

2 Background ................................................... 6

3 Physics of Vessel Response ................................. 8

4 Physics of Wave Forcing ....................................... 12

5 Equation Derivation ........................................... 14

6 Program Development .......................................... 16

7 Data and Discussion ........................................... 32

A Program Code ................................................ 37
Abstract

This thesis documents the development of a computational method for wave induced longitudinal bending in ships. First, there will be a discussion about the importance of longitudinal bending in ship design. The paper will then outline the basic physics at work in the system. It will review the wave forcing computation as well as the response of the vessel. It will then document the progression of the program, which was constructed in Fortran 90, as it solves the linear differential equation for the vessel bending caused by an incoming wave. The entire program then appears at the end of the paper. While the current program is not complete the theory behind it is valid and the code can be augmented to include non-linear components in the future.

Keywords: Longitudinal Bending, Wave Loads, Bending Moment Prediction, Computational Method, Ships
Chapter 1

Introduction

In today’s world there has been a desire to build everything bigger and better. Ships have progressively gotten larger and larger, sometimes outgrowing the very facilities designed for their use, such as Panamax container vessels. As these vessels grow and grow they become more susceptible to longitudinal bending from waves. That is, the motion of a wave underneath a vessel is a kind of forcing that has to be explicitly thought about and compensated for when building a vessel. Naval architects have simplified models and understand the physics of these forces, but predictive calculations are not as common. This thesis documents the concept and programming of a computational method to predict these bending forces.
Chapter 2

Background

Wave forces represent a constant problem to a ship over its life. The constant uneven distribution on the vessel bends and twists it over time, and this continued abuse necessitates careful consideration when the structure of the vessel is designed. As such longitudinal loads are generally part of rules and guides for ship classification. One example is in the American Bureau of Shipping (ABS) Rules for Building and Classing Steel Vessels. Part 3 of the rules cover Hull Construction and Equipment, and the first section of this involves calculations for longitudinal strength, first in Still Water and then for Wave Loads. The early appearance of these formulas is a testament to how fundamental the prevention of these forces is in ship design.

Since these longitudinal forces are such drivers of design there have been many forays into predicting them. Indeed many papers have been written about the topic, each looking at different facets of the wave forces and resulting ship motions. It is interesting to look at the different ways the problem has been approached. Going chronologically, an interesting starting point is the Ship Structure Committee paper comparing model and full-scale bending trends from 1972. The purpose was to see if model tests, in conjunction with ocean data, could yield accurate predictions of long-term wave bending in ships. The end conclusions were that the model predictions were valid, provided that full scale data and ocean data was available [1]. A similar report, issued a year later in 1973, is purely a model test developed at an earlier stage of design. Rather than compare to existing full-scale ship data this model test, on a SL-7 containership in oblique waves, was purely a predictive method. It was done to gather data that could be used in the future for correlations to ship data, but the report does not mention if any such correlations were carried out. It is therefore interesting to note
that this is purely predictive data, i.e. it was taken before hand to see what the motions would be like, and all fell into place with expected results [2].

Computational predictive methods have also been worked on over the years. Jensen and Pedersen wrote a paper involving a quadratic theory of bending moment prediction, and it also included non-linear components [4]. The general topic of wave induced bending can be branched into many other subtopics as well, as evidenced by Wu et al. and their paper on structural responses on a fast catamaran [9]. Many others have also taken the time to do research and write on various traits of wave induced bending in ships.

Presently there are some established prediction programs. Both Shipmo, using strip theory, and WAMIT, using linear and second order potential theory, predict vessel motions in a seaway while being excited by a wave force. These methods can be used to predict wave loading, and are useful in early stages of design.
Chapter 3

Physics of Vessel Response

As mentioned previously a model of the physics involved in the problem is already understood by engineers. The wave force induces a vibration in the ship, and mechanical vibration is a subject well documented and referenced. There are many kinds of vibrations present in a vessel, but for the purpose of this thesis focus will solely on the wave vibrations in heave (vertical movement) and the accompanying motions.

There are two basic vessel responses from wave excitation: hogging and sagging. Hogging is the mode in which a wave crest is at the midship of the vessels with the ends unsupported. The opposite of this is called sagging, where the vessels bow and stern are supported and the midbody sags inward. It can be visualized how a wave moving past a ship will result in a combination of both hogging and sagging over time, with the magnitude of the response tied to the mass and elasticity of the vessel and the wave forcing.

Two main ways exist of approaching the problem of ship vibrations. Continuous analysis involves treating the ship as one long continuous beam. The basic responses of the vessel can be best understood at this level in terms of node vibration. For low frequencies an example Euler beam model can be used to understand the problem [8].

![Continuous model of an Euler beam](image)

Figure 3.1: Continuous model of an Euler beam [8]

At this stage it is worthwhile to define the components of the simplified model. The
springs in the system are the hydrostatic stiffness of the vessel, i.e. its buoyancy. The mass of the vessel is easy enough to understand, but in a ship/wave system it also includes the added mass of water that moves with the vessel. The dashpot in the diagram is the damping in the system, the friction force between the vessel and the water as well as energy leaving the system as radiant waves. The forcing function in the example above is for vibrations created from the propeller, whereas in the case studied here it will come from the waveform.

The graphic above is a representation of an Euler beam, but it is not a complete model of the system. An Euler beam does not include rotational inertia or shear stiffness. These parts can be augmented onto the Euler beam, resulting in a Timoshenko beam. However the equations governing Timoshenko beams are complicated and difficult to solve analytically. Therefore, instead of this continuous analysis for the vessel, a discrete methodology was used.

Discrete analysis of the vibration problem allows for mathematical models that can be solved across non-uniform beams, a more realistic depiction of the vessel since it does not have a constant structural cross-section along its whole length, parallel midbody notwithstanding. Instead of thinking of the vessel as one continuous beam, it can be thought of as many smaller beams connected at points called nodes. These nodes are assigned their own mass and inertia so changes along the length can be reflected in the model. The excitation and restoring forces all act on this node.

This discrete model is solved by balancing the equations of motion on the vessel. Now in general a vessel is given 6 degrees of freedom; translational motion on each axis (surge, sway, and heave) as well as rotation around each axis (roll, pitch, and yaw). To simplify the seakeeping problem the vessel is usually restricted to just a few of these freedoms, in this case just translational motion in the z-direction (heave) and the rotation fore and aft about the y-axis (pitch). It is naval architect convention to label these forces with different numeric subscripts to help differentiate between components in a complex formula. From this the subscript $3$ is used to denote values having to do with heave motion, and $33$ means that
the heave motion is uncoupled from the other degrees of freedom. This is used to describe several of the coefficients in the problem.

Some other variables that need to be defined now are the vessel’s added mass, damping, and stiffness. The added mass is the mass of water that moves with the vessel. It has been accelerated due to the motion of the vessel, and needs to be considered in conjunction with the vessel’s own mass. Added mass is represented by $A_{33}$, with the subscript that the added mass in question is due only to heave forces. Damping, $B_{33}$, is the creation of radiant waves and the friction between the hull and the water. Both serve to slow and reduce the vessel’s overall motion. Finally the hydrostatic stiffness $C_{33}$ is a function of the inertia and elasticity of the vessel.

Returning now to our simple model of the vessel, it has been restricted so that it can only move in a vertical plane. As such it becomes 2D motion problem, that as mentioned previously can be solved analytically. Looking at Figure 3.2 we can see how all of the pieces discussed can come together to describe the response of a vessel, related to a much more simple mechanical model [3].

![Figure 3.2: Boat mass-spring oscillation [3]](image-url)
Now the basic foundation of the ship model has been built, the full model can be described. By separating the vessel model in a finite number of discrete sections a solution can be found that governs their motion. By connecting these discrete beams end to end and making the end connection a node, similar to what is described in Figure 3.2, the equations of dynamic equilibrium can be written and the equations for the whole vessel can be solved together.
Chapter 4

Physics of Wave Forcing

The vessel response has been touched upon, but that is only part of the equation. The wave forces also need to be discussed. In general the open ocean has an entire spectrum of waves of many different heights and frequencies. These irregular seas are difficult to define simply, instead being a sum of many simple waves. For the purposes of vessel motion prediction it is easiest to therefore rely on regular waves, i.e. those that are modelled with harmonic motion.

The exciting force created by these waves is a Froude-Krylov force. This force can be solved for at individual sections of the vessel by using (4.0.1).

\[
F_{EX_3} = \bar{\zeta} \int_L e^{ikx} e^{-kT^*(x)} \left[ c_{33}(x) - \omega_0 \left[ \omega_e a_{33}(x) - ib_{33}(x) \right] \right] dx
\] (4.0.1)

The derivation of (4.0.1) can be found at length in [5]. Even though the details of its derivation are not discussed here, the variables still need to be defined for it to be used. The variable \( \bar{\zeta} \) is the amplitude of the wave, and the integral is taken along the length \( L \) of the vessel. Other wave properties are present in the formula as well, such as wave number \( k \). The mean draft \( T^* \) is found by dividing the sectional area of the section by that section’s beam.

There are two variables that have yet to be discussed, \( \omega_0 \) and \( \omega_e \). These two variables are the wave frequency and encounter frequency, respectively. The wave frequency \( \omega_0 \) is easily understood; it is the frequency of the wave in question. The encounter frequency \( \omega_e \), however, takes a bit more to understand. If the vessel is moving then the waves will not affect the vessel at the wave frequency. Instead the meeting of waves and vessel will occur
at a different frequency depending on direction and magnitude of the vessel’s motion. This relation can be seen in (4.0.2).

\[ \omega_e = \omega - kV \cos(\mu) \]  

\hspace{1cm} (4.0.2)

Again \( k \) here is the wave number, and \( V \) is the forward speed of the vessel. The angle between the wave and the vessel is \( \mu \). Normally this angle is given relative to the vessel, with 0 degrees as following seas (waves coming from astern) and 180 degrees as head seas (waves coming head on to the vessel’s path).
Chapter 5

Equation Derivation

Now that the basic gist of the physics involved has been established some time can be taken to focus on the equations themselves. As mentioned previously wave interactions with a vessel are a vibrations problem. Using Figure 3.2 as the free body diagram the individual parts of this forced motion problem can be established. The general form of motion for vibration motion of this kind is given in (5.0.1).

\[ m\ddot{x} + c\dot{x} + kx = F(t) \]  

(5.0.1)

Simple substitution can then change this general equation into one that represents the motion of the vessel due to waves. The resulting equation will have components very similar to some found in (4.0.1). As mentioned above, the mass of the vessel as well as the added mass of the water needs to be considered. Damping and hydrostatic stiffness also need to be included. Looking at all these substitutions together (5.0.1) becomes (5.0.2).

\[ (M_{33} + A_{33}(\omega_e)) \ddot{x}_3 + B_{33}(\omega_e)\dot{x}_3 + C_{33}x_3 = F_3(t) \]  

(5.0.2)

The right hand side of the equation, the forcing function, is in (5.0.2) the wave force as a function of time. In this case it is the Froude-Krylov excitation force found above in (4.0.1).

At this stage we have the main equation that can be solved to find the vessel’s response to certain wave forcing. It is a linear, second order, non-homogeneous differential equation. The only unknowns are the acceleration \( \ddot{x}_3 \), the velocity \( \dot{x}_3 \), and the position \( x_3 \). It grows in complexity, though, when applied to the discretized model of the vessel. The equation must be extended to include all of the individual nodes that were used to define the vessel. This
changes (5.0.2) into a matrix equation, all solved simultaneously because the pieces need to be in dynamic equilibrium [8]. This new equation can easily be written as seen in (5.0.3).

\[
\left[ M_{33} + A_{33}(\omega_e) \right] |\ddot{x}_3| + \left[ B_{33}(\omega_e) \right] |\dot{x}_3| + \left[ C_{33} \right] |x_3| = |F_3(t)|
\] (5.0.3)

It can be seen how the complexity of the equation has just been increased, since now there are multiple differential equations that all need to be solved at the same time. The exact size of the matrices is known at this stage from some earlier simplifications. The vessel, now divided into many smaller beams, has been given just two degrees of freedom. For the purposes of this program the vessel will be divided into 21 evenly spaced sections. Therefore the matrices for the mass, added mass, damping, and stiffness will be 42 by 42, square matrices with dimensions twice the number of sections (since each has two degrees of freedom). The vessel responses acceleration, velocity, position, as well as the excitation force will then by 42 by 1 matrices.

However, the process for solving the equation is still the same. Numerically the differential can be solved by using the classical fourth-order Runge-Kutta method. This is a recursive formula that can approximate solutions of ordinary differential equations [7]. This Runge-Kutta method is also very easily programmed, which is helpful in solving (5.0.3).

Simply solving (5.0.3) by the Runge-Kutta method is not a complete answer though. Waves move in time and space, so the wave force needs to be time-stepped across the length of the vessel, and at each step the force needs to be computed. This again adds more complexity to the problem, but again it can be programmed.
Chapter 6

Program Development

With the physics of the problem and math understood, focus can turn to the crux of the thesis: creating a program to solve the problem. Fortran 90 was used for its accessibility and backwards compatibility. The program flows from data input to solving for each of the required matrices, before using the Runge-Kutta method to solve for the resulting force on the vessel (as described previously). In each case the coefficient matrices are solved by using dynamic equilibrium conditions.

The main program itself is not that long, but it calls on several subroutines to help in its completion. The main program is seen below.

```fortran
program bendmom

integer :: Nx
integer, dimension(50) :: Ny
real*8 :: moe, grav, rho, wavnum, mu, U, omegao, omegae, zeta, L, S, meanT
real*8 :: dt, T1, T2, T3, T4
real*8, dimension(21) :: x, Iyy, mass, Lc, c33, c33L, a33, b33
real*8, dimension(42) :: zz, zzdot, F1, F2, F3, F4, X1, X2, X3, X4, Y1, Y2, Y3, Y4, &
                        force1, force2, force3, force4
real*8, dimension(5000) :: t
real*8, dimension(21,150) :: y, z
real*8, dimension(42,42) :: K, mm, bm, inversemm
complex*16 :: imag, addedm

! Nx = total number of stations
! Ny(i) = total number if points at station i
! moe = modulus of elasticity (N/m**2)
! grav = acceleration due to gravity (m/sec**2)
! rho = mass density of salt water (kg/m**2)
! Iyy(i) = area moment of inertia (m**4) between sta. i and i+1
```
! mass(i) = mass distribution (kg) between sta. i and i+1
! Lc(i) = distance (m) between sta. i and i+1, used for error check (Length check)
! L = distance (m) between all stations; comes from l
! B(i) = beam at station i (m)
! K = stiffness matrix
! S = sectional area of stat. i (m**2)
! meantT = meant draft at stat. i (m)
! imag = imaginary number i, square root of -1
! wavnum = k, wave number based on omega (rad/m) !UNITS CHECK ON WAVE DATA
! omegao = wave frequency
! omegae = encounter frequency
! U = forward speed
! mu = heading
! zeta = wave amplitude
! a33(i) = added mass / length
! b33(i) = damping / length
! addedm = added mass in complex form (with damping)
! mm = mass matrix
! bm = damping matrix
!
grav = 9.81 !9.81 m/s**2
moe = 2.0d11 !200 GPa
rho = 1.025 !1.025 kg/m**3
imag = (0,1)
!
! Hull geometry input
! y is positive starboard
! z is negative down, 0 at still waterline

call readgeom (Nx, Ny, x, y, z, Iyy, mass, B, L)

call readwaveinfo (wavnum, omegao, omegae, zeta, mu, U)

call stiffness (K, Nx, moe, Iyy, L, rho, grav, c33, c33L, y, Ny)

do i=1, Nx
    call hmasse(y(i,:), -z(i,:), Ny(i), omegae, 1, addedm)
    a33(i) = dreal(addedm) * rho
    b33(i) = rho * (-aimag(addedm)) * omegae
end do

call massmatrix (mass, a33, L, Nx, mm)
! call MIGS (mm, 42, inversemm)
call dampmatrix (b33, L, Nx, bm)
dt = 0.05
t(1) = 0
zz = 0.0
zzdot = 0.0
do j=1, 100
  i = 0
  force1 = 0.0
  force2 = 0.0
  force3 = 0.0
  force4 = 0.0
  T1 = t(j)
  T2 = t(j) + dt/2
  T3 = t(j) + dt/2
  T4 = t(j) + dt
do i=1, 2*(Nx+1)
  call HeaveForce(i, Nx, wavnum, omegao, omegae, zeta, B(i), force1(i), &
    x(i), y, z, Ny(i), c33(i), a33(i), b33(i), T1, L)
  call HeaveForce(i, Nx, wavnum, omegao, omegae, zeta, B(i), force2(i), &
    x(i), y, z, Ny(i), c33(i), a33(i), b33(i), T2, L)
  call HeaveForce(i, Nx, wavnum, omegao, omegae, zeta, B(i), force3(i), &
    x(i), y, z, Ny(i), c33(i), a33(i), b33(i), T3, L)
  call HeaveForce(i, Nx, wavnum, omegao, omegae, zeta, B(i), force4(i), &
    x(i), y, z, Ny(i), c33(i), a33(i), b33(i), T4, L)
end do

do i=1, 2*(Nx)
  write(6,*)T1, i, force1
end do

F1 = matmul(inversemm,(force1 - matmul(K,zz) - matmul(bm,zzdot)))
Y1 = zzdot
X1 = zz

F2 = matmul(inversemm,(force2 - matmul(K,zz) - matmul(bm,zzdot)))
Y2 = zzdot + F1*(dt/2)
\[
X_2 = z z + Y_1 \cdot (d t / 2)
\]

\[
F_3 = \text{matmul}(\text{inversemm}, (\text{force3} - \text{matmul}( K, z z) - \text{matmul}( b m, z zdot)))
\]

\[
Y_3 = z zdot + F_2 \cdot (d t / 2)
\]

\[
X_3 = z z + Y_2 \cdot (d t / 2)
\]

\[
F_4 = \text{matmul}(\text{inversemm}, (\text{force4} - \text{matmul}( K, z z) - \text{matmul}( b m, z zdot)))
\]

\[
Y_4 = z zdot + F_3 \cdot d t
\]

\[
X_4 = z z + Y_3 \cdot d t
\]

\[
zz = z z + (d t / 6) \cdot (Y_1 + 2 \cdot Y_2 + 2 \cdot Y_3 + Y_4)
\]

\[
zzdot = z zdot + (d t / 6) \cdot (F_1 + 2 \cdot F_2 + 2 \cdot F_3 + F_4)
\]

! Output F so that every other value is picked up – send this to results loop

end do

write(6, '(A)') "Program complete"

! Subroutine retained to check for problems

! call diagnostic (Nx, x, Ny, y, z, B, L, Iyy, mass, K, c33L, wavnum, omegao, &

omegae, zeta, F3, mm, bm, a33, b33)

end program

The main program begins by defining all of the variables that will be used throughout and passed from different subroutines. For user convenience these variables are then all defined in the following comment section. The constants for gravity, modulus of elasticity of steel, density of water, and the imaginary number are defined next for future use. The main program then begins calling on some of the various subroutines used to organize the program.

The first subroutine the main program calls upon reads in the file for the vessel geometry. This subroutine is seen below.

```fortran
subroutine readgeom (Nx, Ny, x, y, z, Iyy, mass, B, L)
    integer :: i, j, Nx, errorflag
    integer, dimension(50) :: Ny
    real*8 :: L
    real*8, dimension(21) :: x, Iyy, mass, Lc, B, wgt
```
REAL*8, DIMENSION(21,150) :: Y, Z
CHARACTER :: INPUT*(48)

WRITE(6,'(A)') "Geometry data filename:"
READ(5,'(A)') INPUT

OPEN(10, FILE=INPUT)

READ (10,*) Nx

DO I=1, Nx

READ (10,*) X(I), NY(I) !Reversed to give more order to input file.
          !Reads long 'l position then number of points
    DO J=1, NY(I)
        READ (10,*) Y(I,J), Z(I,J)
    END DO

    B(I) = 2*Y(I,NY(I)) !Uses last y number as beam. Input written CL to WL (going up)
END DO

DO I=1, Nx-1 !Careful with this; edit depending on input file direction
    Lc(I) = X(I+1) - X(I) !(assumes bow stat. first then moving aft)
END DO

!ERRORFLAG = 0

DO I=1, Nx-2 !TEST THIS LOOP check to make sure it works properly
    IF (Lc(I) /= Lc(I+1)) THEN
        !ERRORFLAG = 1
    ELSE
        !ERRORFLAG = 0
    END IF
    IF (ERRORFLAG == 1.0) THEN
        WRITE(6,'(A)') "Length between stations changes. Data will be incorrect."
    END IF
END DO
This subroutine first asks the user to input the name of the file that defines the geometry of the hull. This geometry file is a normal text file organized in a specific manner. The first piece of information required from the input file is the number of sections that define the vessel, \( N_x \). The individual sections are then defined by a longitudinal position \( x(i) \), the forward perpendicular at 0 and larger values aft, then the number of points that define that section \( N_y(i) \). These variables are arrays, and the index \( i \) is used to differentiate between different sections. For each section of the vessel the \( y \) and \( z \) offsets are then read into arrays of the same name. The convention for these values is \( y \) is positive port and \( z \) is positive upward, but with the waterline of the vessel as 0 (so baseline is negative). The subroutine then uses the last point, which corresponds to the waterline of the section, to calculate the beam at that section. The subroutine then uses a similar procedure to read in the inertia \( I_{yy} \) and mass \( \text{mass} \) of each section.

The next subroutine is in a similar vein to the previous one, and reads in the wave data.

```fortran
subroutine readwaveinfo (wavnum, omegao, omegae, zeta, mu, U)

real*8 :: wavnum, omegao, omegae, zeta, mu, U
character :: input*(48)

write(6,'(A)') "Wave_data_filename:"
read(5,'(A)') input

open(12, file=input)
read (12,*) wavnum, omegao, mu, U, zeta !Note setup needed in input file
omegae = omegao - (wavnum*U*cos(mu)) !Calculation from note packet section 8
close (12)
```
This is a simpler subroutine, and the input file is only a few numbers. The subroutine reads in the wave number, wave frequency, angle relative to the vessel, ship speed, and wave amplitude. It then uses (4.0.2) to calculate the encounter frequency.

The main program now has everything necessary to begin calculations. The first coefficient matrix calculated by the program is the stiffness matrix, as seen below.
The first step in this subroutine is to calculate the hydrostatic stiffness for the vessel \( c_{33} \), then convert it to stiffness per length \( c_{33}L \). Now recall that the vessel has been divided (discretized) into equal sections (beams) by nodes, and that at each of these nodes we have established a geometry, mass, and inertia value. The program will use the value at a node to describe half of the beam before and after that node, and thereby describe the whole vessel. As such some of the values in the stiffness matrix will overlap from section to section, as is required for all of the beams to be in dynamic equilibrium.

Next, the main program reaches one of the more complicated subroutines. The \texttt{hmasse} subroutine calculates both the added mass and damping for each section, as seen below.
mp = 3 − 2 * ns

ciome = (0., 1.) * ome

rk = ome ** 2 / g

hend = 2 * pi / (11 * rk)

h = sqrt((yk(nk) − yk(nk−1))**2+(zk(nk) − zk(nk−1))**2)

! compute surface patch points yk, zk and all source points yi, zi

1 do k = 1, nk

   yi(k) = (yk(k) + yk(k+1)) / 2 + (zk(k+1) − zk(k)) / 20
   zi(k) = (zk(k) + zk(k+1)) / 2 − (yk(k+1) − yk(k)) / 20

2 end do

y(i) = 0
z(i) = 0

3 do k = nk, nk + 69

   h = min(h * 1.5, hend)
   y(k+1) = y(k) + h
   z(k+1) = z(k)
   y(i) = y(k) + 0.5 * h
   z(i) = z(k) − 1.0 * h

4 end do

! body boundary condition

5 do k = 1, nk

   if (mp.eq.1) then
      a(k, nk+70) = − ciome * (yk(k+1) − yk(k))
   else
      a(k, nk+70) = + ciome * (zk(k+1) − zk(k))
      a(k, nk+71) = − ciome * (yk(k+1)**2 − yk(k)**2 + zk(k+1)**2 − zk(k)**2) / 2
   endif

6 do i = ns − 1, nk+70 − 1

7      a(k, i) = pqsn(y(k), zk(k), y(k+1), zk(k+1), yi(i), zi(i), mp)

8 end do

9 end do

! surface boundary condition near body

10 do k = nk, nk + 25

11      a(k, nk+70) = 0
12      a(k, nk+71) = 0

13 dy = y(k+1) − y(k)
14 dz = z(k+1) − z(k)

15 do i = ns − 1, nk+70 − 1

16      a(k, i) = 2-point integration (.316, .684) makes integral source potential correct at k=i

17 end do

18 end do

19 end do
\[
a(k, i) = pqsn(yk(k), zk(k), yk(k+1), zk(k+1), yi(i), zi(i), mp) & \\
+ \text{ome}^{*2/g} \text{dy} \left( pqsn(yk(k)+0.316* \text{dy}, zk(k)+0.316* \text{dz}, yi(i), zi(i), mp) & \\
+ pqsn(yk(k)+0.684* \text{dy}, zk(k)+0.684* \text{dz}, yi(i), zi(i), mp) \right)/2
\]
\end{align*}

end do
end do

! ** surface condition far away **
do k=nk+26,nk+70−1
\[
a(k, nk+70)=0
\]
a(k, nk+71)=0
dy=yk(k+1)−yk(k)
dz=zk(k+1)−zk(k)
fak =((k−nk)/70.)**2
\[
do i=ns−1,nk+70−1
\]
a(k, i)=ome**2/g & 
* \text{dy} \left( pqsn(yk(k)+0.316* \text{dy}, zk(k)+0.316* \text{dz}, yi(i), zi(i), mp) & \\
+ pqsn(yk(k)+0.684* \text{dy}, zk(k)+0.684* \text{dz}, yi(i), zi(i), mp) \right)/2 - (0.,1.) & \\
( pqsn(yk(k+1)−fak* \text{dy}, zk(k+1), yi(i), zi(i), mp) & \\
- pqsn(yk(−fak* \text{dy}, zk(k), yi(i), zi(i), mp) )
\end{align*}
\end{do}
end do

! ** Equation: Sum of all source strength =0 for ns=1 **
if (ns.eq.1) then
do i=ns−1,nk+70−1
\[
a(0, i)=1.
\]
end do
a(0, nk+70)=0.
end if

call simqcd(a(ns−1,ns−1),nk+71−ns , ns, 121, is , 1, d−5, detl)

if (is.ne.0) stop '*** singular system. 2 contour points identical?'

! ** force integration. n=1 for heave, n=2 for sway and roll **
! For ns=2: sway and roll motion (l=1,2), force and moment (m=1,2)
do m=1,ns
do l=1,ns
f(m, l)=0
\[
do k=1,nk−1
\]
dy=yk(k+1)−yk(k)
dz=zk(k+1)−zk(k)
if (mp.eq.1) then
  fakt=-dy
else
  if (m.eq.1) fakt=dz
  if (m.eq.2) fakt=-0.5*(yk(k+1)**2-yk(k)**2 &
    +zk(k+1)**2-zk(k)**2)
endif
doi=ns-1,nk+69
  f(m,l)=f(m,l)+a(i,nk+69+l)*fakt*0.5* &
    (pq(yk(k)+0.316*dy,zk(k)+0.316*dz,y(i),zi(i),mp) &
    +pq(yk(k)+0.684*dy,zk(k)+0.684*dz,y(i),zi(i),mp))
end do
end do
addedm = -2*ciome*f(m,1)/ome**2
end do
end do

contains

! pq source potential. dy, dz contour coordinate – source coordinate
real*8 function pq(dy,dz)
  real*8, intent(in) :: dy, dz
  pq=0.5*dlog(dy**2+dz**2)
end function

! pqn integral normal velocity (to right) between contour points P1,P2
real*8 function pqn(dy1,dz1,dy2,dz2)
  real*8, intent(in) :: dy1, dz1, dy2, dz2
  pqn=datan2(dy2*dz1-dz2*dy1,dy1*dy2+dz1*dz2)
end function

! pq, pqsn with ←mirror images other side
real*8 function pq(yk,rzk,ryi,rzi,mp)
  real*8, intent(in) :: yk, rzk, ryi, rzi
  pq=pq(yk-ryi,rzk-rzi)+mp*pq(yk+ryi,rzk-rzi)
end function

real*8 function pqsn(yk1,zk1,yk2,zk2,ryi,rzi,mp)
  real*8, intent(in) :: yk1, zk1, yk2, zk2, ryi, rzi
  pqsn=pqn(yk1-ryi,zk1-rzi,yk2-ryi,zk2-rzi) &
    +mp*pqn(yk1+ryi,zk1-rzi,yk2+ryi,zk2-rzi)
end function
SUBROUTINE SIMQCD (A, N, NR, IM, I, S, DETL)

! GAUSS algorithm to solve complex system of linear equations
! with possibly more than 1 right-hand side
! full unsymmetrical matrix
! A and DETL are complex
! A=coefficient matrix + negative r.h.sides; after execution, instead of r.h.s.
! there are the solutions, the coefficient matrix is destroyed
! 1. index: row (from 1 to N)
! 2. index: column (from 1 to N+NR)
! N=number of equations (rows)
! NR=number of right-hand sides
! IM=maximum declaration of 1. index in dimension statement of calling routine
! I=0 usually; otherwise pivot element less then S .
! S = 1.E-5 recommended.
! DETL is the LN(determinant of coefficient matrix)

IMPLICIT COMPLEX*16 (A-H,O-Z)
REAL*8 S
DIMENSION A(*)
NNR=(N+NR)+IM
MMR=N+IM
K1=-IM
DETL=0.
DO 50 I = 1, N
   K1=K1+IM+1
   BIGA=A(K1)
   IMAX=K1
   DO 30 J=K1+1,K1+N-I
      IF (ABS(BIGA).LT.ABS(A(J)))/THEN
         BIGA=A(J)
         IMAX=J
      ENDIF
   CONTINUE
IF (ABS(BIGA) .LT. S) RETURN

DETL = DETL + LOG(BIGA)

A(IMAX) = A(K1)

DO 50 K = K1 + IM, NNR, IM

IMAX = IMAX + IM

SAVE = -A(IMAX) / BIGA

A(IMAX) = A(K)

A(K) = SAVE

I = K2 = K1

DO 50 J = K + 1, K + N - 1

I = IM

A(J) = A(J) + A(K2) * SAVE

DO 80 I = MMR, NNR, IM

K1 = MMR + 1

DO 80 J = N, 2, -1

K1 = K1 - IM

K2 = I

SAVE = A(I + J)

DO 80 K = K1, K1 + J - 2

K2 = K2 + I

A(K2) = A(K2) + A(K) * SAVE

I = 0

END

It should be noted that hmasse also calls on another subroutine, this one used to solve a complex system of linear equations. The added mass and damping are returned as part of a complex number of the form $a + ib$, with the added mass as the real portion and the damping as the imaginary part. Both values, once extracted need to multiplied by the density $\rho$, and the damping also needs to be multiplied by the encounter frequency.

Now that the added mass and damping values are known at each section, they can be input into the correct matrices for solving (5.0.3). First, the mass matrix is compiled as seen below.

```fortran
subroutine massmatrix (mass, a33, L, Nx, m)

integer :: Nx, R, C
real*8 :: L
real*8, dimension(21) :: mass, a33, m
```
The subroutine is straightforward. The added mass found from \textit{hmasse} is added to the mass input from the geometry file, and again it is applied across the node similar to the stiffness. The interesting part of this matrix is what occurs at the ends of the vessel. The bow and stern are each only given half of the weight of the nearest section instead of the average mass used for the midbody sections. These mass matrix values \(m\) are then loaded into the matrix \(mm\).

The subroutine to load the damping matrix uses similar ideas to those implemented in the mass matrix, as can be seen below.
Again, the damping is included differently depending on whether or not one of the ends (bow or stern) is being considered. The damping value found in the subroutine \textit{hmasse} is already per unit length, so it needs to be multiplied by the length between sections. This is not true at the ends, where only half of the length is used. Once the subroutine has found the values of the damping for all the sections they are loaded into the damping matrix \textit{bm}.

The wave forcing also needs to be calculated. It is used within the time step portion of the main program. The subroutine used is shown below.
Note that the heave force $F_3$ is calculated using the formula from (4.0.1). An estimation of the sectional area $S$ is found by using the trapezoidal rule, and this is used to find the mean draft $meanT$. Like many of the other values found at a section in the program the heave force is found per unit length and therefore needs to be multiplied by the length between sections, or by half the length if it is the bow or stern section.

The inverse of the mass matrix is also needed for the calculation. However, the subroutine to calculate it does not do so correctly, so it is not shown here [6]. The main program also includes a diagnostic subroutine for error checking and troubleshooting, again not shown here. Both of these subroutines can be seen in the full program code, found in Appendix A.
Chapter 7

Data and Discussion

The current findings of this thesis research are mixed. The physics theory behind the thesis is sound and understood. However, failures in computer science have resulted in an incomplete program at this stage. The program has issues with the inversion of the mass matrix, as discussed earlier. However, comparisons can still be made based on how the program solves the problem.

Previous model tests were done at UNO with a model of the KRISO container ship. These were done to experimentally find the bending moment RAO for the vessel. To clarify, this is not an exact RAO (response amplitude operator). Instead of being non-dimensional the bending RAO has units of $\frac{kN\cdot m}{m}$, that is units of bending moment over the amplitude of the exciting wave. At the same time as the experimental data was gathered the full scale vessel was run through a motions program to compare data. The program used was SEAWAY, a module of the program Octopus and similar to the motion programs discussed previously.

Throughout the course of this thesis the KRISO data was run through Shipmo for motions analysis. The bending moments of the vessel were calculated at three longitudinal locations along the vessel: Stations 5, 10, and 15. This coincides with the locations tested during the previously discussed model test. As such the model, SEAWAY, and Shipmo data can all be compared. The comparison at Station 5 of the multiple programs can be seen in Figure 7.1.

The sets of points are the model tests with both a 1.0 and 1.5 inch amplitude wave. The two different SEAWAY lines represent an actual and ideal vessel parameters, mainly the mass distribution along the vessel length. The final line is the output from Shipmo.
Figure 7.1: Comparison of prediction methods with model data at Station 5

It is immediately apparent that there are some discrepancies between the Shipmo and SEAWAY calculations, as well as with the model data. While all have the same basic shape, the magnitudes of the calculated and measured responses are different. As such the output from the program developed in this thesis would be different as well. As seen above the stiffness matrix has been augmented to include not just the hydrostatic stiffness but the structural stiffness as well. This helps the accuracy of the program, as the strength of the steel structure of the vessel is taken into account.

Similar graphs also were created for the bending moment RAO at Station 10 and Station 15. This can be seen in Figure 7.2 and Figure 7.3, respectively. Just as with the graphs from Station 5 there are differences in magnitudes with the data above, but comparable basic shapes.

There is also, much room for improvement. The program design at this stage is just a linear solution. While this is a good approximation, it can be improved by adding non-linear components, such as changes to the damping caused by bow flare. This would improve the
accuracy of the results and therefore the applicability of the program. Also, at this stage the program is designed primarily for use with the KRISO container ship. By changing some of the input characteristics the program can be generalized to work with any monohull form, greatly increasing applicability.
Figure 7.3: Comparison of prediction methods with model data at Station 15
Bibliography


Appendix A

Program Code

```fortran
program bendmom

integer :: Nx
integer, dimension(50) :: Ny
real*8 :: moe, grav, rho, wavnum, mu, U, omegao, omegae, zeta, L, S, meanT
real*8 :: dt, T1, T2, T3, T4
real*8, dimension(21) :: x, Iyy, mass, Lc, B, c33, c33L, a33, b33
real*8, dimension(42) :: zz, zzdot, F1, F2, F3, F4, X1, X2, X3, X4, Y1, &
                          Y2, Y3, Y4, force1, force2, force3, force4
real*8, dimension(5000) :: t
real*8, dimension(21,150) :: y, z
real*8, dimension(42,42) :: K, mm, bm, inversemm
complex*16 :: imag, addedm

! Nx = total number of stations
! Ny(i) = total number of points at station i
! moe = modulus of elasticity (N/m**2)
! grav = acceleration due to gravity (m/sec**2)
! rho = mass density of salt water (kg/m**2)
! Iyy(i) = area moment of inertia (m**4) between sta. i and i+1
! mass(i) = mass distribution (kg) between sta. i and i+1
! Lc(i) = distance (m) between sta. i and i+1, used for error check (Length check)
! L = distance (m) between all stations; comes from l
! B(i) = beam at station i (m)
! K = stiffness matrix
! S = sectional area of stat. i (m**2)
! meanT = mean draft at stat. i (m)
! imag = imaginary number i, square root of -1
! wavnum = k, wave number based on omega (rad/m) !UNITS CHECK ON WAVE DATA
! omegao = wave frequency
! omegae = encounter frequency
! U = forward speed
```

37
! mu = heading
! zeta = wave amplitude
! a33(i) = added mass / length
! b33(i) = damping / length
! addedm = added mass in complex form (with damping)
! mm = mass matrix
! bm = damping matrix

grav = 9.81 !9.81 m/s**2
moe = 2.0d11 !200 GPa
rho = 1.025 !1.025 kg/m**3
imag = (0,1)

! Hull geometry input
! y is positive starboard
! z is negative down, 0 at still waterline
call readgeom (Nx, Ny, x, y, z, Iyy, mass, B, L)
call readwaveinfo (wavnum, omegao, omegae, zeta, mu, U)
call stiffness (K, Nx, moe, Iyy, L, rho, grav, c33, c33L, y, Ny)
do i=1, Nx
  call hmasse(y(i,:), -z(i,:), Ny(i), omegae, 1, addedm)
a33(i) = dreal(addedm) * rho
b33(i) = rho * (-aimag(addedm)) * omegae
end do
call massmatrix (mass, a33, L, Nx, mm)
call MIGS (mm, 42, inversemm)
call dampmatrix (b33, L, Nx, bm)
dt = 0.05
t(1) = 0
zz = 0.0
zzdot = 0.0
do j=1, 100
  i = 0
force1 = 0.0
force2 = 0.0
force3 = 0.0
force4 = 0.0
T1 = t(j)
T2 = t(j) + dt/2
T3 = t(j) + dt/2
T4 = t(j) + dt
do i=1, 2*(Nx+1), 2
call HeaveForce(i, Nx, wavnum, omegao, omegae, zeta, B(i), force1(i), &
   x(i), y, z, Ny(i), c33(i), a33(i), b33(i), T1, L)
call HeaveForce(i, Nx, wavnum, omegao, omegae, zeta, B(i), force2(i), &
   x(i), y, z, Ny(i), c33(i), a33(i), b33(i), T2, L)
call HeaveForce(i, Nx, wavnum, omegao, omegae, zeta, B(i), force3(i), &
   x(i), y, z, Ny(i), c33(i), a33(i), b33(i), T3, L)
call HeaveForce(i, Nx, wavnum, omegao, omegae, zeta, B(i), force4(i), &
   x(i), y, z, Ny(i), c33(i), a33(i), b33(i), T4, L)
end do
do i=1, 2*(Nx)
write(6,*)T1, i, force1
end do
F1 = matmul(inversemm,( force1 - matmul(K,zz)-matmul(bm,zzdot )))
Y1 = zzdot
X1 = zz
F2 = matmul(inversemm,( force2 - matmul(K,zz)-matmul(bm,zzdot )))
Y2 = zzdot + F1*(dt/2)
X2 = zz + Y1*(dt/2)
F3 = matmul(inversemm,( force3 - matmul(K,zz)-matmul(bm,zzdot )))
Y3 = zzdot + F2*(dt/2)
X3 = zz + Y2*(dt/2)
F4 = matmul(inversemm,( force4 - matmul(K,zz)-matmul(bm,zzdot )))
Y4 = zzdot + F3*dt
X4 = zz + Y3*dt
zz = zz + (dt/6)*(Y1 + 2*Y2 + 2*Y3 + Y4)
zzdot = zzdot + (dt/6) * (F1 + 2*F2 + 2*F3 + F4)
end do
write(6, '(A)') "Program complete"
! Subroutine retained to check for problems
! call diagnostic (Nx, x, Ny, y, z, B, L, Iyy, mass, K, c33L, wavnum, &
! omegao, omegae, zeta, F3, mam, cm, a33, b33)
end program

subroutine readgeom (Nx, Ny, x, y, z, Iyy, mass, B, L)
integer :: i, j, Nx, errorflag
integer, dimension(50) :: Ny
real*8 :: L
real*8, dimension(21) :: x, Iyy, mass, Lc, B, wgt
real*8, dimension(21,150) :: y, z
character :: input*(48)
write(6, '(A)') "Geometry data filename:"
read(5, '(A)') input
open(10, file=input)
read (10,*) Nx
do i=1, Nx
  read (10,*) x(i), Ny(i) !Reversed to give more order to input file.
  !Reads long 'l position then number of points
  do j=1, Ny(i)
    read (10,*) y(i,j), z(i,j)
  end do
  B(i) = 2*Y(i,Ny(i)) !Uses last y number as beam. Input written CL to WL (going up)
end do
!do i=1, Nx−1
!    read (10,*) Iyy(i), mass(i) !Reads mass in kg
!end do

do i=1, Nx−1 !Careful with this; edit depending on input file direction
    Lc(i) = x(i+1) − x(i) ! (assumes bow stat. first then moving aft)
end do

!errorflag = 0

do i=1, Nx−2 !TEST THIS LOOP written on 10−19−10 check to make sure it works properly
    if (Lc(i) /= Lc(i+1)) then
        errorflag = 1
    else
        errorflag = 0
    end if
    if (errorflag == 1.0) then
        write (6, '(A)') "Length between stations changes. Data will be incorrect."
    end if
end do

close (10)

end subroutine

subroutine readwaveinfo (wavnum, omegao, omegae, zeta, mu, U)

real*8 :: wavnum, omegao, omegae, zeta, mu, U
character :: input*(48)

write(6, '(A)') "Wave data filename:"
read(5, '(A)') input
open(12, file=input)
read (12,*) wavnum, omegao, mu, U, zeta ! Note setup needed in input file
omegae = omegao − (wavnum∗U∗cos(μ))  //Calculation from note packet section 8

close (12)

end subroutine

subroutine stiffness (K, Nx, moe, Iyy, L, rho, grav, c33, c33L, y, Ny)

integer :: R, C, Nx
integer, dimension(50) :: Ny
real*8 :: rho, grav, moe, L
real*8, dimension(21) :: Iyy, c33, c33L
real*8, dimension(42,42) :: K
real*8, dimension(21,150) :: y

do i = 1, Nx
  c33(i) = 2*y(i,Ny(i))*rho*grav  //Points read from input file start
  c33L(i) = c33(i)*(L/2)  //at CL and go up to WL (Ny is at WL)
end do

i = 1
K = 0.0

do R=1, ((Nx−1)*2)−1, 2

  C = R
  K(R,C) = 12.0*moe*Iyy(i)/L**3 + K(R,C) + c33L(i)
  K(R,C+2) = −12.0*moe*Iyy(i)/L**3 + K(R,C+2)
  K(R,C+3) = 6.0*moe*Iyy(i)/L**2 + K(R,C+3)
  K(R+1,C) = 6.0*moe*Iyy(i)/L**2 + K(R+1,C)
  K(R+1,C+1) = 4.0*moe*Iyy(i)/L + K(R+1,C+1)
  K(R+1,C+2) = −6.0*moe*Iyy(i)/L**2 + K(R+1,C+2)
  K(R+1,C+3) = 2.0*moe*Iyy(i)/L + K(R+1,C+3)
  K(R+2,C) = −12.0*moe*Iyy(i)/L**3 + K(R+2,C)
  K(R+2,C+1) = −6.0*moe*Iyy(i)/L**2 + K(R+2,C+2)
  K(R+2,C+2) = 12.0*moe*Iyy(i)/L**3 + K(R+2,C+3) + c33L(i+1)
  K(R+2,C+3) = −6.0*moe*Iyy(i)/L**2 + K(R+3,C)
  K(R+3,C) = 6.0*moe*Iyy(i)/L**2 + K(R+3,C+1)
\[ K(R+3,C+1) = 2.0 \times \text{moe} \times I_{yy}(i)/L + K(R+3,C+2) \]
\[ K(R+3,C+2) = -6.0 \times \text{moe} \times I_{yy}(i)/L^{**2} + K(R+3,C+2) \]
\[ K(R+3,C+3) = 4.0 \times \text{moe} \times I_{yy}(i)/L + K(R+3,C+3) \]
\[ i = i + 1 \]

end do

end subroutine

subroutine HeaveForce (i, Nx, wavnum, omegao, omegae, zeta, B, F3, x, &
    y, z, Ny, c33, a33, b33, t, L)

    integer :: i, Ny
    real*8 :: wavnum, omegao, omegae, zeta, F3, S, B, meanT, x, c33, b33, a33, t, L
    real*8, dimension(21,150) :: y, z
    complex*16 :: imag

    imag = (0,1)
    S = 0.0
    
    do j=1, Ny-1
        S = S + ((y(i,j+1)+y(i,j))/2) * (z(i,j+1)-z(i,j)) ! Points go from CL to WL
    end do
    
    meanT = (S*2) / B
    
    F3 = zeta * (\text{cdexp} (\text{imag} *\text{wavnum} * x) * \text{dexp} (-(\text{wavnum}) * \text{meanT}) ) * (c33-\text{omegao} &
        ((\text{omegae}*a33)-(\text{imag}*b33))) * \text{cdexp}(\text{imag} * \text{omegae} * t))

    if (i==1 . or. i==N) then
        F3 = F3 * L/2
    else
        F3 = F3 * L
    end if

end subroutine
subroutine hmasse(yk, zk, nk, ome, ns, addedm)

! computes complex hydrod. mass (= mass−i/omega*damping constant)
! for symmetric cross sections at water surface on deep water
! ome = circular frequency of motion (encounter circular frequency ship−wave)
! ns=2 for antisymmetric motions and forces, =1 for symmetric.
! yk, zk must be arrays with 120 elements; elements 1 to nk<=50
! contain contour points of the starboard (right) half of the cross section
! starting from keel to top
! result is addedm (complex mass matrix 1*1 for ns=1, 2*2 for ns=2)

parameter (pi=3.14159, g=9.81)
complex*16 a(0:120,0:122), ciome, detl, f(2,2), addedm !(ns,ns)
real*8 yk(120), zk(120), yi(0:120), zi(0:120)
real*8 ome

if(nk.gt.50)stop'***too many points on section. 50 allowed.'
if(ns.ne.1.and.ns.ne.2)stop'***wrong parameter, ns(1 or 2)'
! +y to starboard; +z downward. last point z gives CWL
mp=3−2*ns
ckm=(0.,1.)*ome
rkm=ome**2/g
hend=2*pi/(11*rkm)
hy=sqrt((yk(nk)−yk(nk−1))**2+(zk(nk)−zk(nk−1))**2)
! compute surface patch points yk, zk and all source points yi, zi
1 do k=1,nk−1
   yi(k)=(yk(k)+yk(k+1))/2+(zk(k+1)−zk(k))/20
   zi(k)=(zk(k)+zk(k+1))/2−(yk(k+1)−yk(k))/20
end do
yi(0)=0
zi(0)=0
2 do k=nk, nk+69
   h=min(h*1.5, hend)
   yk(k+1)=yk(k)+h
   zk(k+1)=zk(k)
   yi(k)=yk(k)+0.5*h
   zi(k)=zk(k)−1.0*h
end do
! body boundary condition

do k=1,nk-1
  if (mp, eq. 1) then
    a(k, nk+70) = -ciome * (yk(k+1) - yk(k))
  else
    a(k, nk+70) = +ciome * (zk(k+1) - zk(k))
    a(k, nk+71) = -ciome * (yk(k+1)**2 - yk(k)**2 + zk(k+1)**2 - zk(k)**2) / 2
  endif
  do i=ns-1,nk+70-1
    a(k, i) = pqsn(yk(k), zk(k), yk(k+1), zk(k+1), yi(i), zi(i), mp)
  end do
end do

! surface boundary condition near body

do k=nk, nk+25
  a(k, nk+70) = 0
  a(k, nk+71) = 0
  dy = yk(k+1) - yk(k)
  dz = zk(k+1) - zk(k)
  do i=ns-1, nk+70-1
    a(k, i) = ome**2 / g * dy * (pqs(yk(k)+0.316*dy, zk(k)+0.316*dz, yi(i), zi(i), mp) &
                              +pqs(yk(k)+0.684*dy, zk(k)+0.684*dz, yi(i), zi(i), mp)) / 2
    end do
end do

! surface condition far away

do k=nk+26, nk+70-1
  a(k, nk+70) = 0
  a(k, nk+71) = 0
  dy = yk(k+1) - yk(k)
  dz = zk(k+1) - zk(k)
  fak = ((k-nk) / 70.)**2
  do i=ns-1, nk+70-1
    a(k, i) = ome**2 / g &
    *dy * (pqs(yk(k)+0.316*dy, zk(k)+0.316*dz, yi(i), zi(i), mp) &
            +pqs(yk(k)+0.684*dy, zk(k)+0.684*dz, yi(i), zi(i), mp)) / 2 - (0..1.) &
    (*pqs(yk(k+1)-fak*dy, zk(k+1), yi(i), zi(i), mp) &
     -pqsn(yk(k)-fak*dy, zk(k), yi(i), zi(i), mp))
  end do
! Equation: Sum of all source strength = 0 for ns=1
if (ns.eq.1) then
  do i=ns-1,nk+70-1
    a(0,i)=1.
  end do
  a(0,nk+70)=0.
end if

call simqcd(a(ns-1,ns-1),nk+71-ns,ns,121,is,1,d-5,detl)
if (is.ne.0) stop '*** singular system, 2 contour points identical?'
! force integration. n=1 for heave, n=2 for sway and roll
! For ns=2: sway and roll motion (l=1,2), force and moment (m=1,2)
do m=1,ns
  do l=1,ns
    f(m,1)=0
    do k=1,nk-1
      dy=y(k+1)-y(k)
      dz=z(k+1)-z(k)
      if (mp.eq.1) then
        fakt=-dy
    else
      if (m.eq.1) fakt=dz
      if (m.eq.2) fakt=-0.5*(y(k+1)**2-y(k)**2 &
      +z(k+1)**2-z(k)**2)
endif
    do i=ns-1,nk+69
      f(m,1)=f(m,1)+a(i,nk+69+1)*fakt*0.5* &
      (pq(y(k)+0.316*dy,z(k)+0.316*dz,y(i),z(i),mp) &
      +pq(y(k)+0.684*dy,z(k)+0.684*dz,y(i),z(i),mp))
    end do
  end do
  addedm = -2*ciome*f(m,1)/ome**2
end do
end do
contains

! pq source potential. dy, dz contour coordinate – source coordinate
real*8 function pq(dy, dz)
   real*8, intent(in) :: dy, dz
   pq=0.5*dlog(dy**2+dz**2)
end function

! pqn integral normal velocity (to right) between contour points P1,P2
real*8 function pqn(dy1, dz1, dy2, dz2)
   real*8, intent(in) :: dy1, dz1, dy2, dz2
   pqn=datan2(dy2*dz1-dz2*dy1,dy1*dy2+dz1*dz2)
end function

! pqs, pqsn with +− mirror images other side
real*8 function pqs(ryk, rzk, ryi, rzi, mp)
   real*8, intent(in) :: ryk, rzk, ryi, rzi
   pqs=pq(ryk-ryi, rzk-rzi)+mp*pq(ryk+ryi, rzk-rzi)
end function

real*8 function pqsn(yk1, zk1, yk2, zk2, ryi, rzi, mp)
   real*8, intent(in) :: yk1, zk1, yk2, zk2, ryi, rzi
   pqsn=pqn(yk1-ryi, zk1-rzi, yk2-ryi, zk2-rzi) &
      +mp*pqn(yk1+ryi, zk1-rzi, yk2+ryi, zk2-rzi)
end function

end subroutine

SUBROUTINE SIMQCD(A,N,NR, IM, I , S ,DETL)

! GAUSS algorithm to solve complex system of linear equations
! with possibly more than 1 right-hand side
! full unsymmetrical matriz
! A and DETL are complex
! A=coefficient matrix + negative r.h.sides; after exectution, instead of r.h.s.
! there are the solutions, the coefficient matrix is destroyed
! 1. index: row (from 1 to N)
! 2. index: column (from 1 to N+NR)
! N=number of equations (rows)
! NR=number of right-hand sides
! IM=maximum declaration of 1. index in dimension statement of calling routine
! $I=0$ usually; otherwise pivot element less then $S$ .
! $S=1.E^{-5}$ recommended.
! DETL is the LN(determinant of coefficient matrix)

IMPLICIT COMPLEX*16 (A-H,O-Z)
REAL*8 S
DIMENSION A(*)
NBR=(N+NR)+1M
MMR=N*IM
K1=-IM
DETL=0.
DO 50 I=1,N
K1=K1+1M+1
BIGA=A(K1)
IMAX=K1
DO 30 J=K1+1,K1+N-I
IF (ABS(BIGA).LT.ABS(A(J))) THEN
BIGA=A(J)
IMAX=J
ENDIF
30 CONTINUE
IF (ABS(BIGA).LT.S) RETURN
DETL=DETL+LOG(BIGA)
A(IMAX)=A(K1)
DO 50 K=K1-1M,NBR,IM
IMAX=IMAX+IM
SAVE=-A(IMAX)/BIGA
A(IMAX)=A(K)
A(K)=SAVE
K2=K1
DO 50 J=K1+1,K1+N-I
K2=K2+1
50 A(J)=A(J)+A(K2)*SAVE
DO 80 I=MMR,NBR-1,IM
K1=MMR+1
DO 80 J=N,2,-1
K1=K1-1M
K2=1
SAVE=A(I+J)
80 K=K1,K1+J-2
K2=K2+1
A(K2) = A(K2) + A(K) * SAVE

I = 0

END

subroutine massmatrix (mass, a33, L, Nx, mm)

integer :: Nx, R, C
real*8 :: L
real*8, dimension(21) :: mass, a33, m
real*8, dimension(42,42) :: mm

do i = 1, Nx
  if (i == 1) then
    m(i) = mass(1)/2 + (a33(1) * L/2)
  else if (i == Nx) then
    m(i) = mass(Nx-1)/2 + (a33(Nx) * L/2)
  else
    m(i) = (mass(i) + mass(i-1))/2 + (a33(i) * L)
  end if
end do

mm = 0.0

do i = 1, Nx
  R = 2*i - 1
  C = R
  mm(R,C) = m(i)
end do

end subroutine

Subroutine to invert matrix $A(N,N)$ with the inverse stored in $X(N,N)$ in the output. Copyright (c) Tao Pang 2001.

```fortran
IMPLICIT NONE
INTEGER, INTENT (IN) :: N
INTEGER :: I, J, K
INTEGER, DIMENSION (N) :: INDX
REAL*8, DIMENSION (N,N) :: A
REAL*8, DIMENSION (N,N) :: X
REAL, DIMENSION (N,N) :: B

DO I = 1, N
  DO J = 1, N
    B(I, J) = 0.0
  END DO
END DO

DO I = 1, N
  B(I, I) = 1.0
END DO

CALL ELGS (A, N, INDX)

DO I = 1, N-1
  DO J = I+1, N
```
DO K = 1, N
    B(INDX(J), K) = B(INDX(J), K) - A(INDX(J), I) * B(INDX(I), K)
END DO
END DO
END DO

! DO I = 1, N
    X(N, I) = B(INDX(N), I) / A(INDX(N), N)
DO J = N-1, 1, -1
    X(J, I) = B(INDX(J), I)
DO K = J+1, N
    X(J, I) = X(J, I) - A(INDX(J), K) * X(K, I)
END DO
    X(J, I) = X(J, I) / A(INDX(J), J)
END DO
END DO
END SUBROUTINE MIGS

SUBROUTINE ELGS (A, N, INDX)

! Subroutine to perform the partial-pivoting Gaussian elimination.
! A(N,N) is the original matrix in the input and transformed matrix
! plus the pivoting element ratios below the diagonal in the output.
! INDX(N) records the pivoting order. Copyright (c) Tao Pang 2001.

IMPLICIT NONE
INTEGER, INTENT (IN) :: N
INTEGER :: I, J, K, ITMP
INTEGER, INTENT (OUT), DIMENSION (N) :: INDX
REAL :: C1, PI, PI1, PJ
REAL*, 8, INTENT (INOUT), DIMENSION (N,N) :: A
REAL, DIMENSION (N) :: C

! Initialize the index

! DO I = 1, N
!    INDX(I) = I
END DO

! Find the rescaling factors, one from each row

!
DO I = 1, N
    CI = 0.0
  END DO
DO J = 1, N
    C1 = AMAX1(C1, DABS(A(I, J)))
END DO
C(I) = C1
END DO

! Search the pivoting (largest) element from each column
!
DO J = 1, N-1
    PI1 = 0.0
  DO I = J, N
    PI = DABS(A(INDX(I), J))/C(INDX(I))
    IF (PI .GT. PI1) THEN
        PI1 = PI
        K = I
    ENDIF
  END DO
END DO
!
! Interchange the rows via INDX(N) to record pivoting order
!
ITMP = INDX(J)
INDX(J) = INDX(K)
INDX(K) = ITMP
DO I = J+1, N
    PJ = A(INDX(I), J)/A(INDX(J), J)
!
! Record pivoting ratios below the diagonal
!
A(INDX(I), J) = PJ
!
! Modify other elements accordingly
!
DO K = J+1, N
    A(INDX(I), K) = A(INDX(I), K) - PJ*A(INDX(J), K)
END DO
END DO
END SUBROUTINE ELGS
subroutine inversematrix (mm, inversemm)

integer :: i, j, R, C
real*8, dimension(42,42) :: mm, inversemm
real*8, dimension(42,84) :: augm

i = 1
j = 1
R = 0
C = 0
augm = 0.0

put mm values into augm
do i=1, 42
  R = i
do j=1, 42
    C = j
augm(R,C) = mm(R,C)
end do
end do

i = 1
j = 43
R = 0
C = 0

add identity matrix to augm such that [augm|I]
do i=1, 42
  R = i
  do j=43, 84
    C = j
    if (j==(i+42)) then
      augm(R,C) = 1.0
    else
      augm(R,C) = 0.0
    end if
  end do
end do
! end do
!
!
! end subroutine

subroutine dampmatrix (b33, L, Nx, bm)

integer :: Nx, R, C
real*8 :: L
real*8, dimension(21) :: b33, cc
real*8, dimension(42,42) :: bm

do i = 1, Nx
  if (i==1 .or. i==Nx) then
    cc(i) = b33(i) * (L/2)
  else
    cc(i) = b33(i) * L
  end if
end do

cm = 0.0

do i = 1, Nx
  R = 2*i - 1
  C = R
  bm(R,C) = cc(i)
end do
end subroutine

subroutine diagnostic (Nx, x, Ny, y, z, B, L, Iyy, mass, K, c33L, &
  wavnum, omegao, omegae, zeta, F3, mm, cm, a33, b33)

integer :: Nx
integer, dimension(50) :: Ny
real*8 :: L, wavnum, omegao, omegae, zeta
real*8, dimension(21) :: x, B, Iyy, mass, c33L, F3, a33, b33
real*8, dimension(21,150) :: y, z
real*8, dimension(42,42) :: K, mm, cm !10x10 is for test; real array is 42x42

open(20, file="output.txt")
write(20, '(A)') "Input file is:"
write(20, *) Nx

do i=1,Nx
write(20, *) x(i), Ny(i)
do j=1, Ny(i)
write(20, *) y(i,j), z(i,j)
end do
write(20, *) B(i)
end do

write(20, *) L

do i=1, Nx-1
write(20, *) Iyy(i), mass(i)
end do
write(20, '(A)') "Stiffness matrix is:"
write(20, *) K

do i=1, Nx
write(20, *) c33L(i)
end do
write(20, *) wavnum, omegao, omegae, zeta

do i=1, Nx
write(20, *) F3(i)
end do
write(20,'(A)') "Mass matrix is:"
write(20,*), mm

write(20,'(A)') "Damping matrix is:"
write(20,*), cm

write(20,'(A)') "a33 values are:"
do i = 1, Nx
write(20,*), a33(i)
end do

write(20,'(A)') "b33 values are:"
do i = 1, Nx
write(20,*), b33(i)
end do

close(20)
end subroutine
This is to certify that Charles Thomas Rogers has successfully completed his Senior Honors Thesis, entitled:

Development of a Computational Method for the Prediction of Wave Induced Longitudinal Bending in Ships

Brandon M. Taravella
Director of Thesis

Lothar Birk
for the Department

Carl D. Malmgren
for the University Honors Program

May 2, 2012
Date