# METHODS FOR REMOVING INERTIAL FORCE FROM MEASURED WAVE IMPACT FORCE SIGNALS

FRASER N. WINSOR

0-612-55546-1

# Methods for Removing Inertial Force from Measured Wave Impact Force Signals

by

© Fraser N. Winsor

A thesis submitted to the
School of Graduate Studies
in partial fulfilment of the
requirements for the degree of
Master of Engineering

Faculty of Engineering and Applied Science
Memorial University of Newfoundland

August 2000

St. John's Newfoundland

This thesis is dedicated to
my beautiful daughters,
Lauren and Andie.

## Abstract

This report discusses the implementation of three methods for removing or mitigating the so-called inertial force from measured signals resulting from wave impact on components of offshore structure models. The wave impact causes vibration or acceleration in the modeled component. The acceleration is perceived as a force by the measurement transducers. This inertial force component is not scaleable, and must be removed in order to obtain the actual applied force.

A mitigation method based on the solution of the system equation of motion using normalized coordinates, known as the normal mode method, is investigated. A technique based on the division of the fast Fourier transform of the measured force by the system frequency response, known as the inverse Fourier transform method, is implemented. Finally, the use of digital low pass and band reject filters is examined.

These techniques are applied to wave impact and decay test measurements from experiments conducted on jacket type fixed offshore structure models.

The results prove to be less than ideal. The reasons for this are discussed, and recommendations are made for future investigations.

## Acknowledgements

A number of people have provided me great assistance and support in my efforts to complete the Master of Engineering program, and particularly, this thesis. Specifically, I would like to thank:

1. my supervisor, Dr. Michael Booton, for his advice and guidance, good nature and patience;
2. IMD technical staff who were involved in the design, fabrication, and testing of the models discussed in this report. Although many hands were involved in this process, I would like to mention Gary Fudge, John Bell, Trent Slade, Andy Wallace, Jim Everard, Howard Mesh, Shane McKay, Terry Lindstrom, Peter Hackett, and Vaughan Benson;
3. former and present IMD Directors General, Mr. Norm Jeffery, Dr. Roger Foxall, and Dr. Tom LeFeuvre, who provided financial and moral support;
4. Michael Sullivan, for his advice, encouragement, and gentle prodding;
5. Dr. John Murray, for his expert tutelage;
6. my wife, Donna, for her love, patience, encouragement, and accepting that things were indeed "fine" at work today.

# Table of Contents

# List of Tables

## List of Figures

## List of Symbols

### Symbols Used in Section 4.0

| | |
|---|---|
| $[M]$ | Mass matrix. |
| $[K]$ | Stiffness matrix. |
| $\{F(t)\}$ | Applied force vector. |
| $\{\ddot{x}(t)\}$ | Acceleration vector. |
| $\{x(t)\}$ | Displacement vector. |
| $[I]$ | Identity or unit matrix. |
| $[A]$ | System or dynamic matrix. |
| $\omega$ | Natural frequency in radians per second. |
| $\lambda$ | Eigenvalues (equals $\omega^2$). |
| $\{F(t)\}_{\text{measured}}$ | Force measured by transducers. |
| $\{F\}_{\text{measured decay}}$ | Force magnitude determined from decay tests. |
| $\{F(t)\}_{\text{inertial}}$ | Force due to structural acceleration. |
| $S_n$ | Scalar used in eigenvector normalization. |
| $\{X_n\}$ | Normalized eigenvector. |
| $\{\tilde{X}_n\}$ | Non-normalized eigenvector. |
| $[X]$ | Mode shape matrix. |
| $\{x\}$ | Geometric or physical coordinate vector. |
| $\{z\}$ | Modal amplitude or normal coordinate vector. |

| | |
|---|---|
| $\{\ddot{z}\}$ | Modal acceleration in normal coordinates. |
| X. Y. and Z | Coordinate system axes. |

## Symbols Used in Section 5.0

| | |
|---|---|
| $f_m\ (t)$ | Represents measured force. |
| $f_a\ (t)$ | Represents actual force. |
| $h\ (t-\tau)$ and $h(t)$ | Represent impulse response. |
| $H\ (i\,\omega)$ | Represents frequency response or the Fourier transform of $h(t)$. |
| $F_m\ (i\,\omega)$ | Represents the Fourier transform of $f_m\ (t)$. |
| $F_a\ (i\,\omega)$ | Represents the Fourier transform of $f_a\ (t)$. |
| $s1(t)$ and $s2(t)$ | Step response vectors selected from measured decay force. |
| $h1(t)$ and $h2(t)$ | Impulse response obtained by differentiating step response. |
| FM and $FM\ (i\omega)$ | FFT of the measured force. |
| H1 and $H1(i\omega)$ | FFT of the impulse response. |

| | |
|---|---|
| FA1 and $FA1(i\omega)$ | FFT of the actual force signal. |
| $H1S(i\omega)$ | Scaled version of H1 and $H1(i\omega)$ |
| $FA1S(i\omega)$ | FFT of the actual force signal determined using $H1S(i\omega)$. |
| $fa1s(t)$ | Actual force time series determined from $FA1S(i\omega)$. |
| "$a$", "$c$", "$q$", "$r$", a[n], b[n], c[n] | Illustrative variables used in Matlab *deconv* command. n is an vector element index. |
| $M, N$ | Vector lengths. |
| $h1s(t)$ | Scaled version of $h1(t)$, used in deconvolution procedure. |
| $f_m2(t)$ | Version of $f_m(t)$ padded with zeros, used in deconvolution procedure. |
| "h_x_fx" | Represents the impulse response vector of a force measured in the X direction, due to a load applied in the X direction. |
| "h_y_fy" | Represents the impulse response vector of a force measured in the Y direction, due to a load applied in the Y direction. |
| "h_z_fz" | Represents the impulse response vector of a force measured in the Z direction, due to a load applied in the Z direction. |
| "fm_x" | Measured force signal developed using the impulse response signal "h_y_fx" in deconvolution. |

## Symbols Used in Section 6.0

### Symbols Related to the Moving Average Filter

| | |
|---|---|
| $x[i]$ | Input signal. |
| $y[i]$ | Output signal. |
| $M$ | Filter length. |
| $p$ | $(M-1)/2$. |
| $q$ | $p+1$. |

### Symbols Related to the Windowed Sinc Filter

| | |
|---|---|
| $h[i]$ | Filter kernel or impulse response. |
| $M$ | $\approx 4/BW$. |
| $BW$ | Transition bandwidth between the passband and stopband. |
| $f_c$ | Cutoff frequency. |
| $K$ | Normalization constant. |

**Symbols Related to the Single Pole Filter**

| | |
|---|---|
| $x[i]$ | Input signal. |
| $y[i]$ | Output signal. |
| $a_0$ | Recursion coefficient. |
| $b_1, b_2, b_3, b_4$ | Recursion coefficients. |
| $z$ | Used in calculation of recursion coefficients. |
| $f_c$ | Cutoff frequency. |

**Symbols Related to the Band Reject Filter - Recursive**

| | |
|---|---|
| $x[i]$ | Input signal. |
| $y[i]$ | Output signal. |
| $a_0, a_1, a_2,$ $b_1, b_2$ | Recursion coefficients. |
| $K, R$ | Used in calculation of recursion coefficients. |
| $BW$ | Bandwidth of the transition region. |
| $f$ | Centre frequency of the transition region. |

**Symbols used in Appendix A**

| | |
|---|---|
| $[I]$ | Identity or unit matrix. |
| $[A]$ | System or dynamic matrix. |

|  |  |
|---|---|
| $\lambda$ | Eigenvalues (equals $\omega^2$). |
| $[B]$ | Equals $[A - \lambda I]$. |
| $\{\tilde{X}_n\}$ | Non-normalized eigenvector. |

## Symbols used in Appendix B

|  |  |
|---|---|
| $F_{ext,i}$ | External force. |
| $m$ | Deck mass (including added mass). |
| $k$ | Dynamometer stiffness. |
| $x_i$ | Deck displacement. |
| $\ddot{x}_i$ | Deck acceleration. |

## Symbols used in Appendix C

|  |  |
|---|---|
| $Pn$ | Scaled FFT Magnitude. |
| $x$ | Input time series. |

## Symbols used in Appendix D

|  |  |
|---|---|
| $P_a$ *and* $P_b$ | Separate sets of applied loads. |
| $x_a$ and $x_b$ | Deflections, or displacements. |
| $f_{im}$ *and* $f_{in}$ | Inertial forces for separate mode shapes m and n. |

| | |
|---|---|
| $x_n$ *and* $x_m$ | Deflections, or displacements. |
| $\omega_m^2, \omega_n^2$ | Modal eigenvalues. |
| $[M]$ | Mass matrix. |
| $[K]$ | Stiffness matrix. |

**Symbols used in Appendix E**

| | |
|---|---|
| $x$ | Time series to be differentiated. |
| $\Delta x$ | Time series increment. |
| $y$ | Ordinate of input signal. |
| $i$ | Ordinate index. |
| $h$ | Increment of abscissa. |
| $y''$ | Second derivative. |
| $n$ | Order of filter. |
| $f$ | Frequency range vector. |
| $a$ | Amplitude vector. |
| $b$ | Output filter coefficients. |

**Symbols used in Appendix F**

| | |
|---|---|
| $\xi_n$ | Damping ratio. |
| $a_0 . a_1$ | Proportionality constants. |
| $C$ | Damping coefficient matrix. |
| $m$ | Mass matrix. |
| $k$ | Stiffness matrix. |
| $\omega_m$ and $\omega_n$ | Fundamental frequency and the highest relevant frequency. |

**Symbols used in Appendix G**

| | |
|---|---|
| det_pul | Dectected pulse. |
| des_pul | Desired pulse. |
| DET_PUL | FFT of detected pulse. |
| DES_PUL | FFT of desired pulse. |
| RFR | Required frequency response. |
| rfk | Required frequency kernel. |
| M and N | Length or number of points in a time series. |

**Symbols used in Appendix H**

|  |  |
|---|---|
| "$a$" . "$c$" ."$q$". "$r$". a[n]. b[n]. c[n] | Illustrative variables used in Matlab *deconv* command. n is an vector element index. |

**Symbols used in Appendix J**

|  |  |
|---|---|
| $[M]$ | Mass matrix. |
| $[C]$ | Damping coefficient matrix. |
| $[K]$ | Stiffness matrix. |
| $\{x(t)\}$ | Displacement vector. |
| $\{\dot{x}(t)\}$ | Velocity vector. |
| $\{\ddot{x}(t)\}$ | Acceleration vector. |
| $u_1(t), u_2(t), \dot{u}_1(t), \dot{u}_2(t)$ | First order differential equations used in solution of second order equation of motion. |
| $[F]_{\text{measured decay}}$ | Force amplitudes from measured decay test. |
| $[\tilde{X}]$ | Non-normalized eigenvectors. |

# 1.0 Introduction

The topic of wave impact loading of offshore structures and their components has received a great deal of attention over the past twenty-five years. Components, such as structural cross-members, located just above the mean water level are subjected to intermittent large magnitude impulsive wave loads. Over time these loads can lead to fatigue overstressing and result in failure. Other components, such as deck beams and plating, are typically located at elevations higher than the expected extreme wave. That is, they are designed to avoid impact rather than withstand it.

Despite this, special conditions sometimes occur that require re-evaluation of the loading of these components. For fixed offshore structures, seabed subsidence or the observation of wave heights beyond design or predicted extreme values can raise the concerns of operators. Similarly, for floating structures, the unexpected reduction of air-gap due to extreme waves or vessel damage can expose decks, superstructure, and other equipment to higher than anticipated loads.

In recent years, older offshore structures have been subjected to assessments for re-certification. These vessels or structures were often originally designed without adequate consideration of extreme load effects. Updated estimates, and subsequent remedial action, are required for the structures to remain in operation.

Many offshore structures are now the subject of decommissioning plans. Regulators in many jurisdictions require that obsolete structures be removed from the field at the end of their operational life. Operators, for a variety of reasons, often prefer to delay complete removal, and propose partial decommissioning. This can expose certain structural components to wave loads for which they were not specifically designed. Estimates are then required in order to demonstrate that the structures can withstand the wave loading.

Designers have come to realize that reliable wave load estimates for structures located above the mean water level are an important part of the overall vessel design. Damage to structure or equipment can have costly economic and safety implications, so there has been a push in recent years develop a better understanding of the wave impact process.

Numerical and analytical estimates of wave loading have typically been made using Morison's equation or diffraction based formulations. These are suitable for the case of continuous wave loading of submerged structures, but will lead to an under estimation of loads under partial submergence, or extreme non-linear wave conditions. This has led to the increased use of model tests to establish load estimates and to calibrate numerical models. Over the years, simplified structures have been subjected to a variety of test procedures. The most popular modeled structure has been the horizontal circular cylinder, chosen because it can represent the shape of common structural members, and because it is amenable to analytical and numerical modeling. These cylinders have been subjected to drop or plunge tests, generated waves in flumes or wave tanks, and oscillating free surfaces in U-tubes. The results have shown considerable scatter and there remains some disagreement regarding the appropriate value for the non-dimensional slam coefficient.

More recently, model tests have focussed attention on horizontal and vertical flat plates, which can represent decks and superstructure. Again, these structures have been subjected to drop-type tests, and to tests in generated waves. The instrumentation has included accelerometers, pressure transducers, and load transducers.

Regardless of the type of modeled structure, or instrumentation scheme, the wave impact process will invariably lead to dynamic response. The structure accelerates or vibrates at its resonant frequency, and this is observed as force by the load transducers. This causes an unwanted amplification and oscillation in the measured signal, which masks the actual or applied force. To obtain an accurate estimate of the full scale wave impact load, this inertial force must be removed.

Experimenters have proposed many methods to eliminate or mitigate the measured inertial force. Several of the techniques will be examined, implemented, and compared in this report.

Finally, recommendations will be made for areas that require further investigation in order to improve the inertial force mitigation techniques.

# 2.0 Review of Literature

## 2.1 Introduction

There is an abundance of published literature that focus on model test studies of wave impacts on components of offshore structures. These components include horizontal and vertical oriented circular cylinders, plates and deck structures. Test techniques include the use of generated waves in basin or flume facilities, drop tests, experiments in oscillating U-tubes, and even full-scale measurements.

Many of these publications present their measured model test results in comparison with numerical predictions. The prediction methods are often the main focus of these papers. A discussion of numerical prediction techniques is beyond the scope of this report. As a result, this review of literature has focussed on gathering details related to the physical model tests and data processing techniques. The topics of interest include:

1. Particulars of data acquisition and instrumentation used during model experiments.
2. Details regarding wave parameters that influence impact measurement results.
3. Information on techniques used to mitigate or remove inertial components from wave impact measurements.

## 2.2 Horizontal Circular Cylinders Subjected to Regular Generated Waves

A number of authors have reported on experimental and theoretical work related to small horizontal circular cylinders subjected to generated regular waves. Dalton and Nash (1976), Miller (1977) and (1980), Isaacson and Subbiah (1990), and Isaacson and Prasad (1992) and (1994) have presented papers describing experimental setup, problems encountered and general observations on the topic. Measured results are often compared to results obtained using analytical or numerical methods.

Some early experimenters report problems with their test measurements, which they attribute (explicitly or implicitly) to inadequacies with the test equipment or wave generation system. Dalton and Nash (1976) reported that their load measurements did not display the expected impulsive characteristics. They imply that this was related to the size of the cylinder used. They provide no discussion of data acquisition, sampling rates, or signal processing techniques. They do not provide a discussion of the effects of dynamic amplification.

Most of the experiments described involve the use of small cylinders, where the measurements can be complicated by the effects of scale. Few provide details of the data acquisition systems.

Miller (1977) and (1980) reported on experiments where the support structure for the cylinder model and measurement system had a low natural frequency. which caused corruption of the load measurement signals. The quality of the wave generation was erratic. which led to problems with the estimation of surface velocity needed for theoretical calculations. The shape of the input wave was reported by several investigators. Dalton and Nash (1976) and Miller (1977) and (1980). to have a significant effect on the resulting load measurement. Parameters such as wave steepness have been shown to affect the characteristics of the measured loads. particularly the rise-time. Dalton and Nash (1976) described how the load measurements resulting from the more irregular waves (i.e. steep) in the ramp-up portion of the wave signal displayed characteristics that were more impulsive than the remainder of the load measurement.

Cylinder misalignment. with either the water surface. or the wave front was observed (Dalton and Nash (1976). Miller (1977) and (1980). and Isaacson and Prasad (1992) and (1994)) to have an influence on the rise-time. and therefore on the dynamic response of the systems. These misalignments were caused. in some degree. by inadequate adaptability of the test equipment. but mostly by the poor wave generation quality.

Problems with alignment. and low wave vertical velocity were cited as the main disadvantages of generated wave tests as compared with drop-type tests (to be discussed later). Dalton and Nash (1976) claim that drop tests differ from generated wave tests due to the effects of virtual mass.

Isaacson and Prasad (1992) and (1994), and Miller (1977) reported that load rise-time was affected by air entrainment, fluid compressibility, cylinder roughness, cylinder inclination, and motion of the cylinder. They claimed that rise-time affects the dynamics response of the cylinder, which in turn affects the fluid particle kinematics. That is, the motion of the cylinder affects the characteristics of the wave that is causing the initial cylinder vibration.

The influence of entrapped air will be discussed later, but has been shown to cushion the slam loading.

Miller (1977) and (1980) reported that the dynamic amplification factor could be as high as 2.0. He reported that dynamic amplification depends on system natural frequency, acceleration, and damping. He felt that in future experiments, the dynamic characteristics of the system should be varied to develop a better understanding of the slam process.

Miller stated that slamming was not Reynold's number dependent. The influence of viscosity was negligible, being associated with inertia instead. Although the form of the slam coefficient is reminiscent of drag, the physics are associated with accelerated flow.

Isaacson and Prasad (1992) and (1994) stated that during the stage of partial submergence it was not possible to distinguish between load contributions due to slamming and drag.

Miller (1980) felt that it was usually not possible to deduce the input responsible for a particular slam response. Wellicome, in his discussion of Miller (1977) remarked that it was only possible to work backwards from load measurement to the actual impact. These discussions imply the existence of dynamic amplification or resonance in the measured signals.

Miller (1977) and (1980) mentioned that computer models were quite useful in developing an understanding of the impact process.

Most of the early authors provide little information of their data acquisition systems. Isaacson and Prasad (1992) and (1994) are the exception; quoting very high sample rates (20 kHz).

## 2.3 Horizontal Circular Cylinders Tested in Oscillating U-Tube Tanks

Sarpkaya (1978) reported on experiments using aluminum horizontal circular cylinders, 0.076m to 0.2m in diameter, subjected to the impact of an oscillating free surface in a U-tube. The instrumentation measured the vertical force and the vertical acceleration. The free surface oscillated at a period of 5.5 seconds. The cylinders, which were considered to be elastically supported, were reported to have natural frequencies of 358 Hz and 628 HZ for diameters of 0.076m and 0.15m, respectively.

The author stated that the system natural frequency would have to approach infinity for the response to approach that of a rigid body. This is interesting since several other experimenters report having used rigid cylinders (Faltinsen et al (1977)). Sarpkaya reiterated a point made by other authors that the response of the system to an impulsive force is heavily dependent on the exact nature of the applied force, as well as the system natural frequency.

Sarpkaya stated that random factors such as the nature of the disturbances at the wave surface, the orientation of the structural member relative to the given wave, currents, three-dimensional nature of the waves, and spray, affect the determination of impact force magnification in the (full scale) ocean environment. In short, the rise time is affected by these non-deterministic factors and there is no way to predict the reaction forces on circular members even when damping, natural frequency and the ideal values of the slam coefficient are known.

The author used several derivations of the slam coefficient. For one coefficient, the inertial force of the cylinder was removed, based on the acceleration measurement. He reported obtaining experimentally derived values for the slam coefficient that are close to the theoretical value of $\pi$. The slamming force may be amplified by as much as 1.7 due to the dynamic response of the structure. He reported that the slam coefficient can reach a value as high as 6.3 due to dynamic response. The fluid force acting on the cylinder-transducer system can amplify or attenuate the dynamic response.

Sarpkaya (1978) described the total slamming process in a manner similar to Garrison (1996). After the initial impact, the net force decreases due to the decreased contribution of added mass. The cylinder undergoes damped oscillation at the natural frequency. Following this the buoyant force increases and separation effects give rise to large drag forces.

He highlighted some of the difficulties of tests using generated waves, saying that there is a limitation in the range of wave amplitudes that can be achieved, and there is a difficulty in measuring the fluid velocities at the instant of impact.

## 2.4 Horizontal Circular Cylinders Subjected to Drop Tests

Numerous authors (Campbell, Wellicome, and Weynberg (1977), Faltinsen, Kjaerland, Nottveit, Vinje (1977), Campbell, and Weynberg (1979), Miao (1990), Garrison (1996)), have reported on experiments where horizontal circular cylinders are dropped or plunged via mechanical apparatus into a still water surface.

The advantages of this form of experiment relative to generated wave tests are that larger diameter cylinders can be utilized, reducing the effects of scale. Cylinder diameters ranged from 0.05m (Miao (1990)) to 0.35m (Faltinsen et al (1977)). In addition the cylinders can be subjected to higher, and more controlled velocities, something that cannot be easily achieved using generated waves. The angle of the cylinders relative the free surface is more readily controlled, avoiding the effects of misalignment, especially on impact rise time. Campbell and Weynberg (1979) report that their test rig allowed the cylinder angle to be controlled to ±0.5 minutes.

One disadvantage of this form of experiment is that the effects of wave shape on impact force cannot be studied. The cylinders impact a flat surface, rather than a wave profile with steepness.

The instrumentation has ranged from force transducers connecting the cylinders to the test mechanism, to high frequency pressure transducers installed at various locations on the cylinder, and strain gauges installed directly on the cylinder to measure stress. Faltinsen et al (1977) reported measurement uncertainty to be ±10 percent, using a strain-gauged system. The pressure transducers suffered from gauge slip and required calibration prior to each run.

The cylinders have been described as being either rigid or elastic. The rigid cylinders were constructed of aluminum, while the elastic cylinders were constructed of PVC tube. Faltinsen et al (1977) used the results from rigid cylinder tests to compare with theory. They were surprised to find poor agreement between the theoretical and experimental values in the impact phase. This may have been related to the difficulty in achieving a truly rigid cylinder. Using elastic cylinders, the results showed fair agreement with a numerical model, though the authors cautioned that their scaling methods might have been questionable. The length, diameter, thickness, and mass distribution of the elastic cylinders were varied. The experimenters attempted to maintain similarity using three non-dimensional parameters. This necessitated drop velocities higher than specified by Froude, and resulted in stresses higher than yield stress for realistic extreme wave velocities.

The degree of cylinder-end fixity can influence the measured results. Miao (1990) reported his cylinders to have fixed end conditions. Faltinsen et al (1977) reportedly used pinned-end conditions, but felt the fixed-end conditions would only result in higher stresses.

Several authors (Faltinsen et al (1977), Garrison (1996)) reported that Froude scaling was most appropriate, due to the inertial and gravitational nature of impact. Reynold's number scaling was inappropriate since it was felt that viscous effects were not relevant. With deeper cylinder submergence, drag, buoyancy, and viscous effects dominate.

Cylinder and test system dynamic response was an important issue to all the experimenters. Campbell and Weynberg (1979) reported that the natural frequency of their test rig was 500 Hz. Both the force transducers and the pressure sensors used in that experiment exhibited oscillations associated with the test rig vibration. Miao (1990) reported that the dominant component of the dynamic response was at the fundamental frequency.

Miao (1990) stated that the response of an elastic member to an impulsive load depends on the characteristics of the applied force, and the vibrational characteristics of the member. Garrison (1996) stated that slam forces are characterized by a high spike upon contact with the water surface, the magnitude of which depends on the velocity and beam dynamics.

Campbell et al (1977) used curve-fitting techniques to deal with cylinder dynamics. This involved fitting a cubic or quadratic polynomial to the measured signal. There were often difficulties with the fit to the first cycle, which affected the interpretation of the slam. Faltinsen et al (1977), Campbell and Weynberg (1979), and Miao (1990) all present computer models based on the equation of motion of the cylinder, which help shed light on the effects of dynamic response on the force measurement.

Miao (1990) reported that peak dynamic stress was lower than that derived using quasi-static methods, and felt that the effects of slamming load from design wave conditions were not a serious as conventionally thought. He states that the dynamic load factor for a given impulse shape is dependent on the ratio of the impulse duration and the natural period of the structure. Therefore, in a heavy sea condition, where the impact duration is short, part of the load is resisted by the inertia of the structure and the stresses produced are smaller than for a longer loading of the same magnitude. He felt, therefore, that the effects of dynamic amplification could be ignored.

Garrison (1996) reanalyzed the data presented by Faltinsen. He stated that real fluid effects influence the impact process, including those related to the free surface. The impact process is highly non-linear and transient, but becomes steady flow at deeper immersion, and related to viscous effects.

He felt that in the impact phase, mass, stiffness, and Froude scaling should be maintained between model and prototype. Garrison (1996) divided that entire slamming process into three regions. Region 1 being the initial impact where inertial effects are dominant and viscous (Reynolds) and gravitational (Froude) effects are negligible. Region 2 is a transitional phase where beam dynamics, Froude and to a minor degree Reynolds effects are important. Region 3 is a large submergence, steady state region where forces on the cylinder are due largely to buoyancy and drag.

## 2.5 Horizontal Circular Sections on Full Scale OTS

Kaplan (1979) reported on full scale measurements from a horizontal circular cylinder installed on the Ocean Tests Structure (OTS). The OTS was an instrumented platform located in the Gulf of Mexico in the late 1970's.

The instrumented horizontal circular member was located 5 feet above the mean water surface, and had an outer diameter of 12.75 inches and a length of 25.5 inches. The force measurements were filtered prior to being recorded, using an analog 4-pole Butterworth filter having a cutoff frequency of 3 Hz. These signals were digitized and re-sampled at a rate of 0.1 sec.

The measured vertical force did not display the shape predicted by theory. However, the filtered theoretical time trace did resemble the physical measurement, displaying suppressed rapid impulsive rise characteristics, and magnitude. The author felt that this indicated that the initial filtering might have corrupted the full-scale measurements. Similarly he felt that the re-sampling rate of 0.1 seconds might have been inadequate to capture the rapid rise characteristic.

Kaplan (1979) described a typical vertical impact as having a region of positive force, followed by a region of negative force. A typical horizontal force was described as having a region of negative force, followed by a region of positive force.

He reported that Miller (1977) showed the slam coefficient to be $3.6\pm1$, where the variation was due to experimental uncertainties and the influence of measuring system dynamics. Miller (1977) in the discussion reply felt that the variation was more likely due to variation in input rise time.

Kaplan (1979) was reluctant to apply the slam coefficient method to the OTS data due to uncertainties in the evaluation of the peak force, and determination of the vertical velocity.

Further proof that the initial filtering of the measurements may have been detrimental was demonstrated when the instrumented member was "hit sharply". The unfiltered measurement displayed a sharp initial pulse, followed by a lesser negative region with a shorter time extent, and lingering high frequency component. Conversely the filtered record was less sharp and the time extent of the impulse was larger.

He reported that the natural frequency of the measurement system was very close to the estimated peak frequency of the input wave, and that this probably led to dynamic amplification.

## 2.6 Horizontal Circular Cylinders in Breaking Waves

Chan (1993) and Prasad, Chan, and Isaacson (1994) reported on experiments using horizontal circular cylinders in breaking waves. Chan (1993) used a relatively large cylinder, with a diameter of 0.216m instrumented with pressure transducers. The transducers had a natural frequency of 100kHz, and were sampled at 16700Hz. Prasad et al (1994) used a smaller diameter cylinder, 0.042m instrumented to measure vertical and horizontal forces. The sample rate was 10000Hz, low-pass filtered at 1000Hz.

Both authors reported that impact forces due to breaking waves are greater than those due to non-breaking waves. Chan (1993) reported that the increase in force can be more than double, while Prasad et al (1994) reported that these forces can be 4 to 20 times greater.

The position of the cylinder relative to the wave profile has a significant effect on the peak impact forces.

Prasad (1994) reported that when considering the wave impact phenomenon, the variability of the hydrodynamic loading must be considered and the effect this will have on dynamic response. Previous experimenters observed scatter in the slam coefficient that was attributed to the effect cylinder and measurement system dynamic response had on the applied and measured force. Other factors such as entrained air, ripples on the water surface, cylinder roughness, and cylinder inclination also affect the measurements. Chan (1993) also pointed out that in earlier experiments, the cylinder diameter was often less than a fifth of the incident wave height, and these smaller models were affected by the response of the measurement system.

According to Chan (1993) existing numerical models based on rate of change of momentum of a mass of water past a cylinder assume that the water surface impacting the surface to be flat. This may be appropriate for small cylinders, but will not be relevant for steep wave impacts, where the diameter will be comparable to the dimension of the crest. For drop type tests the added mass is assumed to be half that of a long flat plate in an infinite volume of fluid, having the same water contact width. Again this assumption is not appropriate for the breaking wave situation.

In the breaking wave situation, the effects of entrapped air and wave profile trajectory are important. The horizontal force component may be as large as the vertical component (this could lead to high overturning moments on the structure). The presence of entrained air and splash affects the local water particle kinematics and pressures, and consequently influence the magnitude of the impact force. When there is little entrapped air, the impact pressures are high, however, the peaks are not simultaneous over the entire zone (Chan (1993)). As a result the impact forces may not be as high. When the wave curvature is smaller, and air entrapment is greater, the pressures are not as impulsive and there are synchronous high pressures over a wide zone. This yields an oscillatory impact force that may influence the structural dynamics significantly. Pressure variations in both space and time are strongly dependent on the location of the cylinder relative to the incident wave profile.

Prasad et al (1994) reported that their measurement setup had a natural frequency of 29Hz in the horizontal direction. In the vertical direction, the dominant frequency was 290Hz with minor components at 130 and 190Hz. In this case the effect of dynamic response is expected to have a greater influence on the horizontal measurement, due to the low natural frequency. The recorded and applied forces will differ, and there may be a phase difference.

The authors introduced a numerical correction method based on the equation of motion of the cylinder and supports, which is solved for displacement. The measured force was assumed to be equal to the stiffness times the displacement. The method was demonstrated using the measured horizontal force. The measured signal was first low-pass filtered (125Hz) to remove high frequency noise that would amplify the predicted force. The assumptions made for this method was that the cylinder-dynamometer system was single DOF, and that the cylinder motion did not affect the applied force.

This method was not applicable to the vertical measurement due to the presence of the minor components below the dominant frequency. An alternate method, based on Isaacson and Prasad (1992) assuming an idealized symmetric triangular impact applied to a single DOF system, was used. This was a closed-form solution where dimensionless rise-time was varied.

Chan's experiments used pressure sensors. The pressures fluctuated significantly for almost identical wave conditions, but the impulses (the pressures integrated over the duration of the impact) were repeatable. Chan suggested that the impulse may be a much better variable to use in the numerical simulation and may be utilized in predicting the structural responses and stresses. Usage of the impulse is only possible if the time scales of the impulsive pressures are much smaller than the response time scales of the structure.

Both authors suggested that due to the complexity of the breaking wave impact it was unlikely that a single closed form expression, similar to that for the slam coefficient, would be developed to predict this type of impact.

## 2.7 Vertical Plates in Breaking Waves

Several authors reported on experiments involving vertical plates, subjected to breaking or plunging wave conditions (Kirkgoz and Mengi (1987), Chan and Melville (1988), Toumazis et al (1989), and Chan et al (1991)). These plates were surface piercing and non-piercing, and were instrumented with fast response pressure transducers. Sample rates were very high, from 3.6 to 20 kHz.

All authors were unanimous that impact loads from breaking waves were significantly higher than those from comparable non-breaking waves. Chan and Melville (1988) reported impact pressures 2 to 5 times higher, with fluid velocities 2 times higher.

The impact pressure time series was characterized by short rise times, high maxima, and pressure oscillations. Only Chan noted the pressure oscillations, which he attributed to the dynamics of trapped air.

Chan and Melville (1988), Toumazis et al (1989), and Chan et al (1991) all reported on the affect that entrapped air had on impact measurements. Toumazis suggested that the effects of trapped air were not fully understood, and therefore limited the development of numerical and analytical models. Air entrapment was observed to have a cushioning effect, reducing the peak impact pressure. Chan observed two ranges of pressure oscillation, 2000 to 5000 Hz and 300 to 800 Hz. He claimed that increased air entrapment led to a reduction of the oscillation frequency.

Chan and Melville (1988) conducted experiments at two scales. They reported that in the smaller scale model tests the low frequency range oscillations were not observed, however the other pressure characteristics and distributions were comparable with the larger scale results.

All the authors discuss the influence of wave profile on the impact measurements. Kirkgoz and Mengi (1987) observed significant dynamic load resulting from so-called perfect impact, where the vertical wave front impacts a vertical wall. They reported that the pressures in this case were simultaneous over the entire interface. They describe the pressure distribution as being linear from the point of impact to the crest, and parabolic from the point of impact to the bottom of the plate.

Chan and Melville (1988) stated that other investigators correlated high impact pressure to steep incident wave fronts and fluid velocities. Chan et al (1991) described the effects of five wave profiles on pressure measurement. The first profile was inclined away from the wall and resulted in non-impulsive pressures with low magnitudes and long rise times. The second profile was more vertical and resulted in high peak pressures, short rise times, and low amplitude, high frequency oscillations. The third profile had a more developed, higher speed wave crest. The peak pressures may not have been as high, in this case, but the total integrated force could be damaging. There was an initial pronounced oscillation at 200Hz, with reduced to low amplitudes after two or three periods, followed by a slow modulated decay. The fourth profile resulted in significant trapped air. The peak pressures were simultaneous and of comparable magnitude over the region. The pressure oscillation magnitudes were more pronounced and in phase. The rise times were longer and the oscillation frequency was lower, at 100Hz. The peak pressures resulting from the crest were higher. In the fifth case the crest has developed into a well-defined jet, which hit the lower transducer resulting in high impulsive pressures. The oscillation frequency was 1000Hz.

Toumazis et al (1989) had similar observations. When the wave does not break, the measured pressures are small, because there is no entrapped air. He stated that this can be considered a two dimensional problem and scaling can be based on Froude. When hit by the tip of a breaking wave, high pressure measurements were recorded. Above and below this point, the pressures were lower. In the overturning stage, the breaking wave jet was in free-fall. In addition to gravity forces, surface tension and inertial forces are important. The larger the radius of curvature of the wave front, the lesser the air entrainment, the higher the peak pressures.

In the entrapped air region, uniform pressures were observed, with prolonged rise times. A high degree of measurement variability was also observed. Due to the randomness of wave breaking, the characteristics of the trapped air region will vary even for repeated runs. Even the frequency of pressure oscillation varied. Toumazis et al (1989) made similar observations, citing that air entrainment results in higher scatter, longer impact duration, and lower impact pressure magnitudes.

Chan et al observed that as the air pocket size increased both the peak pressure and oscillation frequency decreased. The pressures within the air pocket were simultaneous and of equal magnitude. When integrated over the contact area, the air pocket pressures may be large enough to induce significant structural response.

Chan raised the question as to whether the observed pressure oscillations were the direct result of oscillation of the entrapped air, or whether it is merely structural vibration. Chan and Melville (1988) stated that the fact that the low frequency oscillations were in phase over several impact elevations (for a particular test), ruled out structural vibrations and supported its association with trapped air.

Chan et al (1991) used a one-dimensional numerical model to study whether oscillations were due to entrapped air or structural vibration. The model consisted of a coupled system where the inertia of the incident wave, the dynamics of the structure, and the compression and leakage of the trapped air were taken into account. Results showed that the entrapped air affected the pressure characteristics more significantly than the structural parameters. Low air pocket thickness resulted in high peak pressure magnitude. Higher air pocket thickness resulted in lower frequency pressure oscillation. Impulsive pressures were observed irrespective of the variations in structural mass or stiffness properties. The authors stated however, that following the first few pressure oscillations, the modulated decay oscillation could be attributed to structural vibrations.

Kirkgoz and Mengi (1987) developed a formulation for the dynamic behavior of a vertical plate based on the classical theory of elastic plates. Spatial integration of this equation was done using finite element methods.

Chan and Melville (1988) discussed the use of a superposition averaging technique to "filter" off pressure oscillations.

Chan et al (1991) noted that non-submerged vertical plates would have likely have an impulsive total force, due to the integration of the impulsive pressures. Surface piercing plates may display non-impulsive integrated total force due to the attenuation by non-impulsive loading below the still water line.

Chan and Melville (1988) highlighted a number of questions they felt needed to be addressed. (1) What are the important temporal and spatial characteristics of the impact pressures? (2) How do the pressure characteristics correlate with the incident wave kinematics? (3) What is the cause of the high variability in impact pressures? (4) What is the cause of the pressure oscillations? (5) How can model test results be extrapolated to prototype conditions?

## 2.8 Model Jacket Structures Subjected to Generated Waves

Several authors have reported on model tests of jacket type structures. Broughton and Horn (1987). Kaplan (1992). Kaplan et al (1995). Murray et al (1995). Murray et al (1997) discussed model set-up. test procedures. and methods of dealing with extraneous force measurements.

Most of these authors recognized that resonance effects. caused by structural acceleration. lead to measurement of unwanted inertial forces. A number of methods were proposed for removal of these loads.

Murray et al (1997) pointed out that resonant responses will be dominant at the resonant frequencies of the structure and are related to the inertial and damping (hydrodynamic and structural) characteristics. In an ideal situation. a hydro-elastic model would be used for testing This. however. can be prohibitively expensive. A more reasonable option is to construct a model with sufficiently high stiffness that its resonant frequency is above the maximum frequency of interest to the designer and can therefore be ignored. This involves a trade off between stiffness and load measurement sensitivity. A high data acquisition sampling rate is also required.

Four methods were described for the removal of the extraneous forces. Kaplan (1992) described an inverse Fourier transform method that can be used to relate the measured force to the actual force. In order to establish the actual impact force time history from a measured result, it is possible to relate them by a convolution integral expression using the impulse response function of the measuring system. Applying Fourier transform operations to this relation leads to an expression that relates the measured and actual forces by the Fourier transform of the impulse response. An inverse Fourier transform will yield the actual force time history. Murray et al (1995) contend that in a multiple degree-of-freedom system where cross-coupling can occur, implementation of this method can be difficult.

Another method described by Murray et al (1995), involved using accelerometer measurements to estimate the inertial force, which is then subtracted from the total measured force. The problem with this method is the estimation of the added mass and added moments of inertia due to the variation of water impacting the deck. Also this method assumes that the damping is negligible (which may not be the case particularly for fluid damping) and that the deck is rigid.

Murray (1995) and Kaplan (1995) describe a method of digital filtering whereby through careful selection of the cutoff frequency, the high frequency inertial forces are removed from the measured force signals. The risk in using a filtering technique is that valid high frequency components will be removed. The authors demonstrated that the while the low frequency force components for the spatially varied load cells are out of phase, the high frequency components are in phase with each other. They conclude that the high frequency components are not phase locked to the low frequency components. and can be legitimately removed through filtering. Murray et al (1997) used this filtering technique.

Andersen et al (1998) recommended a method for removing inertial load. or structural dynamics. from measured wave impact signals. based on the so-called normal mode approach. The procedure establishes an equation of motion for the system. determines the eigenvectors using measured decay test data. and solves the equations using normal coordinates. The accelerations determined in the normal coordinate system are then converted to physical coordinates. and used to calculate the inertial force. which is subtracted from the measured force.

Several of the authors pointed out that the shape and extent of the impacting wave had a great influence on the measured force. Broughton and Horn (1987) correlated crest height with measured force. Kaplan (1992) used a 2nd order equation of motion for the structure to establish the influence of pulse duration and natural period on response amplitude. He showed that when the pulse time is close to the natural period. the peak response can be 1.7 times the actual value. Murray (1995) reported a direct relationship between wave steepness and resonance effects.

Kaplan (1992) stated that measurement procedures. inertial effects. system dynamics. filter effects must be considered when determining the shape and magnitude of impact forces.

Kaplan (1992) developed a numerical formulation for vertical loads on flat deck structures, based on his previous work on circular cylinders.

Broughton and Horn (1987) reported on model tests of a jacket structure. Four tri-axial load cells were sampled at a rate of 20 Hz. Model tests reported by Murray et al (1995) and (1997) used a dynamometer constructed of eight uniaxial load cells sampled at 200 Hz and 500 Hz.

Broughton and Horn (1987) developed finite element models for the jacket and deck with load estimates based on the rate of change of momentum of the wave relative to the structure (the typical method for this type of load estimation) and from model test results. One of these FE models was non-linear.

## 2.9 Summary

Clearly the subject of wave impact on models of offshore structures or their components is complicated. Publications covering more than twenty years have been reviewed here, and despite the variations in their topics and techniques there are some common themes.

Most experimenters agree that model and measurement systems should have a high natural frequency. A value significantly higher than the frequency of the impacting wave is required. This is complicated by the fact that waves impacting above the still water line (or breaking waves) are thought to have high frequency energy content. This leads to dynamic response or amplification, sometimes in multiple modes of vibration.

Early experiments suffered from the limitations of test equipment and facilities, particularly in the areas of wave generation and data acquisition. Poor wave quality limited the usefulness of measured force results. Low data sampling (or resampling) and inappropriate filtering choices led to misinterpretations. Model and measurement systems with natural frequencies in the range of wave frequencies led to corrupted force measurements.

Most of the recent experimenters recommended very high sample rates, in the kilohertz range, with appropriate hardware filter cutoffs to prevent aliasing. The practicality of using this range of sample rate depends on the number of instrumentation channels and the data storage capacity. Further work needs to be done to establish the suitable sampling rates for wave impact experiments. Experimenters recommended that digital signal processing or filtering be done offline. This allows the filter selection to be based on the particulars of the measured signal.

Besides the natural frequency and damping characteristics of the model and measuring system, the nature of the dynamic response depends greatly on a number of other factors. Wave related parameters such as steepness, air entrainment, surface ripple, fluid compressibility, surface current, and the three-dimensional nature of waves can affect dynamic response. Wave steepness has been shown to affect response rise time and entrapped air can influence response magnitude. The presence of breaking waves has been shown to greatly increase measured response. Breaking waves are related to wave steepness, and their effects are influenced by the position of the model relative to the wave profile. Other factors such as model surface roughness, and inclination can contribute to measurement results.

Many of these factors are considered to be non-deterministic, so even when the physical characteristics of the model and measuring system are known, predictions and interpretation of results can be difficult. Work has been done to quantify the effects of many of these factors, though the type and shape of the modeled structure often dictate the influence of a particular factor.

Many types of instrumentation have been used in wave impact experiments. Force transducers, typically combined to measure in multiple degrees-of-freedom, quantify impact loads. Pressure transducers are used to measure localized pressures. Accelerometers are used to measure structural vibrations. Strain gauges are used to directly measure stress in a structure. Photographic or video records are used to document the impact process.

The experimenters agree that Froude scaling is most appropriate, particularly at the initial phase of impact, where inertial and gravitational forces are deemed to be dominant. As the impact process proceeds, particularly where the structure is subjected to deep submergence, viscous effects begin to dominate.

## 2.10 Objectives of the Present Study

A number of methods are presented to mitigate or eliminate the inertial force components that are typically present in measured wave impact signals. The inertial force is the result of structural accelerations caused by the wave impact, and recorded by the measurement system. The inertial force corrupts the actual applied force, and should be removed prior to scaling. Of all the problems associated with the measurement of wave impacts, the question of how to deal with inertial force is the most pressing. For this reason, this report will examine techniques to mitigate this force.

Several authors reported that computer models based on the equation of motion of the model were used to provide insight into the impact process. These were typically single degree-of-freedom models, which generally assumed damping to be negligible. Others applied curve fitting or averaging techniques to smoothen impact data.

Accelerometer measurements have been used to directly calculate inertial force, which is then subtracted from the total measured force.

One author describes the so-called normal mode method to reduce inertial force components. The equation of motion is solved in the normalized coordinate system. The eigenvectors are determined from the measured decay tests. This eliminates the need to develop a stiffness matrix.

Another author presents an inertial force mitigation technique based on a convolution integral involving the measured and actual forces and the system impulse response. The equation is solved in the frequency domain, and an inverse Fourier transform is used to obtain the actual force time series.

Digital filtering is also recommended as a method to remove the inertial force component from measured wave impact signals.

These last three methods (normal mode technique, inverse Fourier transform technique, and digital filtering) will be explored in greater detail in this report.

## 3.0 Discussion of Model Jacket Structure Experiments

The sample wave impact measurements used throughout this report were taken from experiments conducted at the Institute for Marine Dynamics (IMD). located in St. John's. Newfoundland. IMD is an institute of the National Research Council of Canada (NRC).

IMD has conducted numerous experiments over the years measuring wave induced impact loads on components of offshore structures. These components are generally located above the still water line. and include deck plating. deck beams. support columns. wave deflectors. deck equipment. and superstructure.

Instrumentation used is generally confined to load and pressure transducers. When these are attached to structural components. they measure localized forces and pressures. Sets of load transducers used in combination are known as dynamometers. These are used typically to measure global loads on the large structural components. though they can be designed for smaller components.

IMD's typical dynamometer design consists of a support frame. known as a strongback. upon which strain gauge based load transducers are mounted. The structural component under consideration is attached to the load transducers via flexural members (flexures) that ensure uni-directional loading of the individual transducers. The load transducers are oriented orthogonally with respect to each other. This allows force measurement in three degrees-of-freedom. Moments may also be deduced from the force measurements, relative to an assumed origin. A typical dynamometer design is shown in Figure 3.1. In this case the structural component is the deck of a jacket structure. Note that the deck plating is not shown, for clarity.

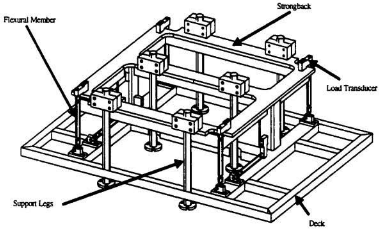

**Figure 3.1 Typical Dynamometer**

A jacket is a type of fixed offshore structure designed using space frame/truss components. It rests on the ocean floor and is secured using piles. The structure provides a protective "jacket" for the production risers, thus the name. IMD has conducted model tests on several jacket designs. In-house research experiments were performed on a 1:36

scale model of a generic jacket structure. Some results of that test program were reported in Murray et al (1995) and (1997). Another set of experiments were performed as part of a fee-for service contract, using a 1:54 scale model of an existing North Sea jacket-type platform Winsor (1998). A model of a typical jacket structure is shown in Figure 3.2.

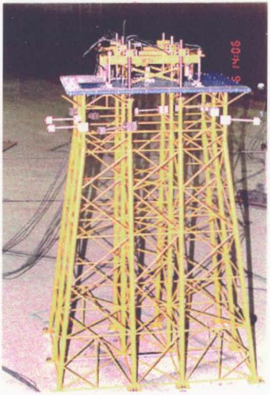

**Figure 3.2 Typical Jacket Structure Model**

The models for these experiments were constructed in similar fashion. Each consisted of three sections. base, strongback, and deck. The base was a space frame/truss structure designed to maintain geometric similarity between model and prototype. The strongback was attached to the top of the base, and was used to support the force transducers. The deck was connected to the transducers using flexural members (see Figure 3.1).

In both experiments the deck was suspended from the strongback using seven transducers. Four transducers were oriented to measure vertical force only, two were oriented to measure transverse force, and one was oriented to measure longitudinal force. The deck was connected to the individual transducers using aluminum flexures. These flexures were in the form of cylinders, with sections of reduced diameter, that permitted only uniaxial force to be transmitted to the transducers. This arrangement of transducers allowed for the measurement of six degree-of-freedom forces and moments applied to the deck. Both models utilized Interface SSB-250, cantilever type transducers.

The bases of the jacket structures were constructed of steel round stock and tubing. A typical elevation is shown in Figure 3.3. The deck of the 1:36 model was constructed using square aluminum closed-tubing with sheathing fixed to its top and underside. The interior cavity of the deck was packed with Styrofoam to prevent water from becoming trapped inside. Similarly, the 1:54 scale model deck was fabricated using box and round tube members covered top and bottom with thin plate. The interior spaces were filled with urethane foam to limit water ingress. Attached to the perimeter of the deck were

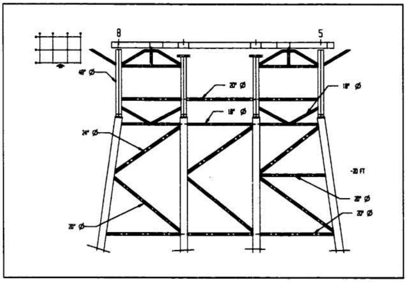

**Figure 3.3 Elevation of Jacket Structure Model**

cantilever deck sections, fabricated with steel box section material and covered with thin plate and mesh. The cantilever sections were removable, but were in place for the experiments. A typical deck is shown in Figure 3.4

The experiments were conducted in IMD's Offshore Engineering Basin (OEB). The models were securely bolted to the basin floor during testing. The 1:36 scale model tests were conducted in a water depth representing 77.0 m full scale. The 1:54 scale model tests were conducted in a water depth representing 81.1 m full scale. The wave conditions

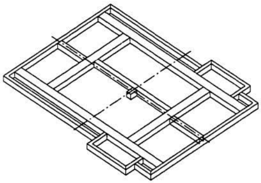

**Figure 3.4 Typical Jacket Structure Deck**

for test measurements used in this report are summarized in Table 3.1. The wave spectra were based on the JONSWAP formulation. For the purposes of this report the 1:36 scale model will be referred to as Jacket A, the 1:54 scale model will be known as Jacket B.

Figure 3.5 shows the coordinate system used for both sets of experiments. The coordinate system was right-hand-rule, with the origin located at the geometric centre of the dynamometer, at the top elevation of the deck. Moments were measured positive clockwise about each axis, viewed from the origin. The load transducer locations are indicated on the figure.

Table 3.1 – Summary of Wave Conditions for Jacket Structure Experiments

| Jacket Structure | Wave Direction | Water Depth (m) | Hs (m) | Tp (s) |
|---|---|---|---|---|
| Jacket A (Scale 1:36) | 0 Degrees | 77.0 | 19.5 | 14.5 |
| Jacket B (Scale 1:54) | 0 Degrees | 81.1 | 16.4 | 17.4 |

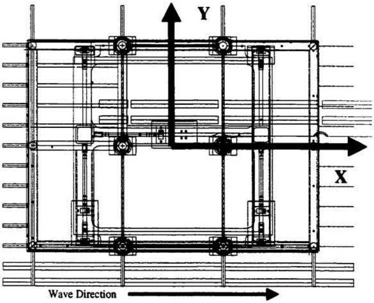

**Figure 3.5 Coordinate System**

The tests for Jacket A were sampled at a model scale rate of 500 Hz, with a hardware cutoff frequency of 100 Hz. The tests were for Jacket B sampled at a rate of 200 Hz, with a hardware cutoff of 100 Hz. Decay tests were conducted to determine the lowest natural frequency of the models. This was done by applying a known load to the deck and releasing it instantaneously. The decay time traces were measured using the dynamometer transducers. The natural frequency and damping characteristics were determined from traces fitted to these measurements, or by locating the resonant peak of the signal in frequency domain. Table 3.2 summarizes the natural frequencies. Figures 3.6 to 3.11 show the measured decay time traces.

Table 3.2 – Jacket Structure Natural Frequencies

| Structure | Degree-of -Freedom | Natural Frequency in Air (Hz) | Natural Frequency in Water (Hz) |
|---|---|---|---|
| Jacket A | X | 2.28 | NA |
| | Y | 2.48 | NA |
| | Z | 7.24 | NA |
| Jacket B | X | 3.41 | 3.41 |
| | Y | 3.61 | 3.59 |
| | Z | 10.39 | 4.77 |

The linearity of the dynamometers was established by applying a range of known loads to the deck, parallel to each coordinate axis. The applied loads were measured using a

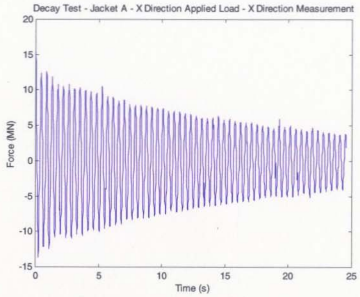

**Figure 3.6** **Decay Force – Jacket A – X Direction**

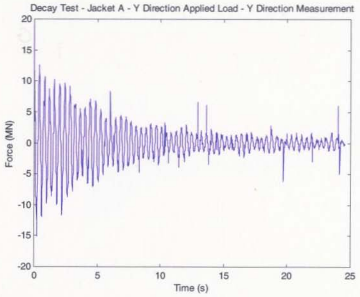

**Figure 3.7** **Decay Force – Jacket A – Y Direction**

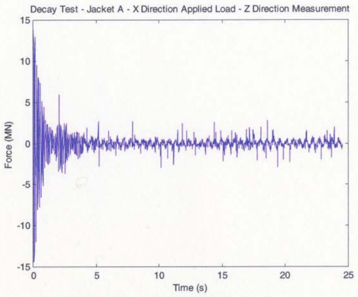

**Figure 3.8 Decay Force – Jacket A – Z Direction**

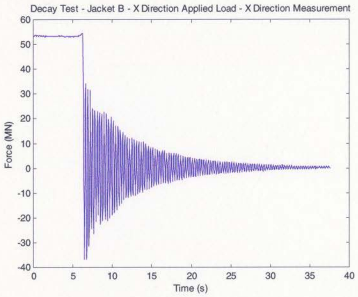

**Figure 3.9 Decay Force – Jacket B – X Direction**

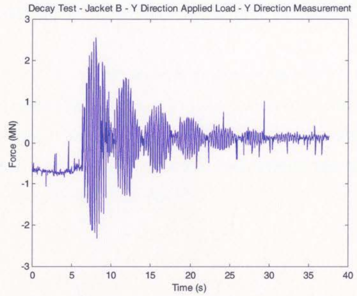

**Figure 3.10 Decay Force – Jacket B – Y Direction**

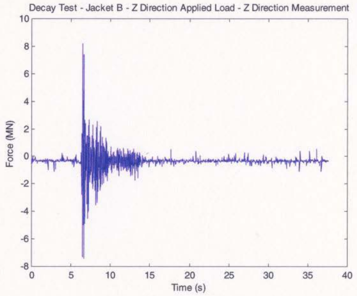

**Figure 3.11 Decay Force – Jacket B – Z Direction**

calibrated in-line load transducer. In all cases the measured load was at least 98% of the applied load.

Figures 3.12 to 3.17 show the wave impact force measurements obtained from the tests in wave conditions defined in Table 3.1. The signals are presented in full scale, with units of mega-Newtons. These are single impact events selected from the full time trace. In each case the three degree-of-freedom measurements are shown. These signals will be used extensively throughout this report.

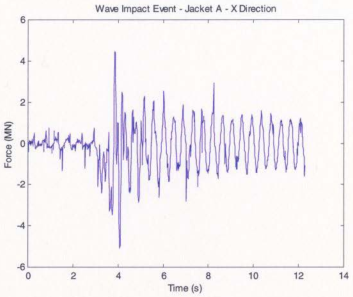

**Figure 3.12 Wave Impact Event – Jacket A – X Direction**

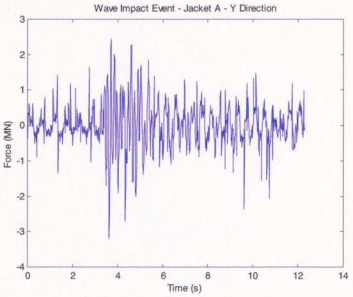

**Figure 3.13 Wave Impact Event – Jacket A – Y Direction**

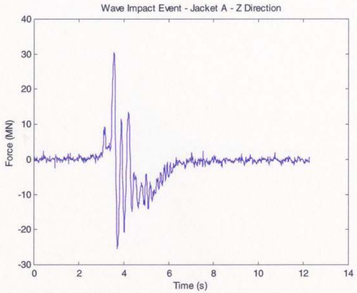

**Figure 3.14 Wave Impact Event – Jacket A – Z Direction**

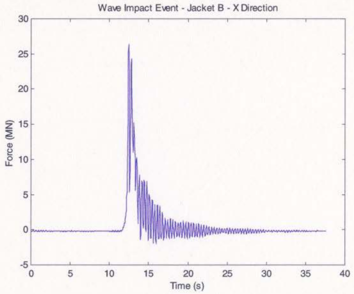

**Figure 3.15 Wave Impact Event – Jacket B – X Direction**

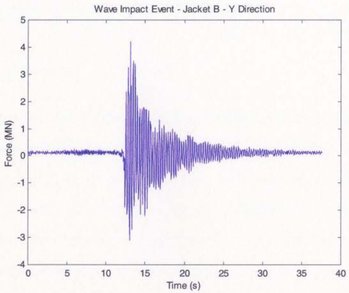

**Figure 3.16 Wave Impact Event – Jacket B – Y Direction**

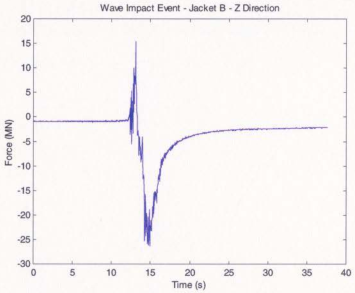

**Figure 3.17 Wave Impact Event – Jacket B – Z Direction**

# 4.0 Removal of Inertial Forces Using the Normal Mode Approach

## 4.1 Introduction

Andersen et al. (1998) recommend a method for removing the effects of structural dynamics (inertial load) from measured wave impact signals, based on the so-called normal mode approach. The procedure, which is outlined in Table 4.1, establishes an equation of motion for the system, determines the eigenvectors using measured decay test data, and solves the equations using normal coordinates. The accelerations determined in the normal coordinate system are then converted to physical coordinates, and used to calculate the inertial force, which is subtracted from the measured force.

This procedure, in general, is described in standard structural dynamics or vibrations texts such as Thomson, (1981).

Table 4.1 – Procedure for Application of Normal Mode Approach

| Step | Description |
|---|---|
| 1 | Establish equation of motion. |
| 2 | Establish equations for eigenvalues and eigenvectors. Determine eigenvalues. |
| 3 | Determine eigenvectors from measured decay tests. |
| 4 | Normalize eigenvectors. |
| 5 | Convert system from physical to normal coordinate system. |
| 6 | Solve for accelerations in normal coordinate system. |
| 7 | Convert back to physical coordinate system. |
| 8 | Calculate inertial force and subtract from measured force to obtain actual force. |

## 4.2 Methodology – Normal Mode Approach

### 4.2.1 Establish Equation of Motion

A simple equation of motion for the jacket structure described in Section 3.0 is defined as,

$$[M]\{\ddot{x}(t)\} + [K]\{x(t)\} = \{F(t)\} \qquad \textbf{4.1}$$

where,

| | |
|---|---|
| $[M]$ | mass matrix |
| $[K]$ | stiffness matrix |
| $\{F(t)\}$ | applied force vector |
| $\{\ddot{x}(t)\}$ | acceleration vector |
| $\{x(t)\}$ | displacement vector |

In this case damping is assumed to be negligible, so no damping term is included in the equation. For the case of free vibration the equation becomes:

$$[\mathrm{M}]\{\ddot{x}(t)\} + [K]\{x(t)\} = 0 \qquad \textbf{4.2}$$

### 4.2.2 Determine Eigenvalues

Equation 4.2 is used to determine the natural frequencies (eigenvalues) and eigenvectors of the structure, in the following manner. Pre-multiplying each term by the inverse of [M], results in

$$[M]^{-1}[M]\{\ddot{x}(t)\}+[M]^{-1}[K]\{x(t)\}=0 \qquad 4.3$$

which can be written as

$$[I]\{\ddot{x}(t)\}+[A]\{x(t)\}=0 \qquad 4.4$$

where,

$[I]=[M]^{-1}[M]=$ identity or unit matrix
$[A]=[M]^{-1}[K]=$ system or dynamic matrix

Assuming harmonic motion, the displacement, velocity, and acceleration can be written as follows.

$$x(t)=e^{i\omega t} \qquad 4.5$$

$$\dot{x}(t)=i\omega e^{i\omega t} \qquad 4.6$$

$$\ddot{x}(t)=(i\omega)^2 e^{i\omega t}=-\omega^2 e^{i\omega t} \qquad 4.7$$

Substitution of equations 4.5 and 4.7 into equation 4.4 leads to

$$-\omega^2 [I]\{x(t)\}+[A]\{x(t)\}=0 \qquad 4.8$$

or

$$-\lambda [I]\{x(t)\}+[A]\{x(t)\}=0 \qquad 4.9$$

where

$\lambda = \omega^2$ = eigenvalues

Equation 4.9 can then be written in the following form.

$$[A-\lambda I]\{x(t)\}=0 \qquad 4.10$$

The eigenvalues are determined by evaluating the determinant when set equal to zero. This is known as the characteristic equation, the roots of which are the eigenvalues.

$$|A-\lambda I|=0 \qquad 4.11$$

### 4.2.3 Determine Eigenvectors

The eigenvalues are substituted into Equation 4.10 to obtain the eigenvectors $\{\tilde{X}_i\}$ where '$i$' represents the particular degree-of-freedom or mode shape. This results in the following equation.

$$[A-\lambda_i\, I]\{\tilde{X}_i\}=0 \qquad \textbf{4.12}$$

The eigenvalues represented by $\{\tilde{X}_i\}$ are referred to as non-normalized. The eigenvectors may also be determined using the adjoint matrix of the system as discussed in Appendix A.

After substitution of the following

$$\begin{aligned}[I]&=[M]^{-1}[M] \qquad \textbf{4.13}\\ [A]&=[M]^{-1}[K]\\ \lambda&=\omega^2\end{aligned}$$

into Equation 4.12 and pre-multiplication by $1/[M]^{-1}$ Equation 4.14 is obtained.

$$-\omega_i^2[M]\{\tilde{X}_i\}+[K]\{\tilde{X}_i\}=0 \qquad \mathbf{4.14}$$

where

| | |
|---|---|
| $\omega_i$ | natural frequency |
| $\tilde{X}_i$ | eigenvector (non - normalized) |

Equation 4.14 provides a convenient method to determine the eigenvectors using measured decay test data. During the model test experiments, described in Section 3.0, the following forces were measured.

$$\{F(t)\}_{measured}=[K]\{x(t)\}=\{F(t)\}-[M]\{\ddot{x}(t)\} \qquad \mathbf{4.15}$$

Equation 4.15 indicates that the force measured by the force transducers equals the transducer displacement multiplied by its stiffness. For impact type force measurements the displacement will inherently include the dynamic or inertial component. So the measured force is a combination of the actual force (due to the applied load) and the inertial load.

$$\{F(t)\}_{measured}=\{F(t)\}+\{F(t)\}_{inertial} \qquad \mathbf{4.16}$$

where the inertial force is given by the expression

$$\{F(t)\}_{\text{inertial}} = -[M]\{\ddot{x}(t)\} \qquad \textbf{4.17}$$

In a practical sense it is difficult to measure the physical displacements of the structure directly. The accelerations can be measured, using accelerometers, and this technique has been used in the past to establish inertial loads, as discussed in Appendix B. Without the ability to accurately measure the physical displacement, it is difficult to directly determine the stiffness or flexibility matrices.

An alternate method can be used to establish the physical response of a structure. This involves the use of experimentally determined force decay time traces. The model structure is given an initial displacement or offset in the degree-of-freedom of interest. It is allowed to vibrate freely, and the resulting inertial forces are measured directly from the force transducers installed on the structure. The inherent assumption is that following the initial excitation, no other external force acts on the structure and measured forces are purely inertial.

Based on this assumption, and using Equation 4.14, it is seen that the modal force vector obtained from the decay tests is given by

$$\{F\}_{\text{measured decay}} = [K]\{\tilde{X}_i\} = \omega_i^2 [M]\{\tilde{X}_i\} \qquad \textbf{4.18}$$

Utilizing the relationship given in Equation 4.18 circumvents the inconvenience of not knowing the stiffness matrix. The non-normalized eigenvectors can be determined using Equation 4.19.

$$\{\tilde{X}_i\}=\frac{1}{\omega_i^2}[M]^{-1}\{F\}_{\text{measured decay}} \qquad \textbf{4.19}$$

The natural frequencies can be determined from the characteristic equation or measured. The mass and mass inertia properties are used to develop the mass matrix. The measured force vector is determined from the fast Fourier transform of the decay time trace. A discussion of the procedure to obtain the matrix of measured decay force vectors is provided in Appendix C.

### 4.2.4 Eigenvector Normalization

The non-normalized eigenvectors are determined using only the measured decay force signals, the mass matrix, and the vector of natural frequencies. The shapes of the eigenvectors are uniquely defined, but their amplitudes are arbitrary. This allows the eigenvectors to be normalized. There are several methods to do this.

One method involves setting an arbitrary degree-of-freedom (typically the first), for a particular mode shape, to unity. The remaining degrees-of-freedom are set relative to this reference value.

A variation of this procedure is to normalize the shapes to the maximum displacement value in each mode rather than with respect to an arbitrary coordinate. The maximum coordinate becomes unity.

The most popular normalization procedure for computer applications involves adjusting each modal amplitude such that it that satisfies the expression.

$$\{X_n\}^T [M]\{X_n\}=1 \qquad \textbf{4.20}$$

where,

$\{X_n\}$ vector of normalized mode shape amplitudes.

A scalar factor is used to normalize the eigenvectors. This factor is determined using this equation.

$$\{\tilde{X}_n\}^T [M]\{\tilde{X}_n\}=S_n \qquad \textbf{4.21}$$

where $S_n$ = scalar

The normalized eigenvector, $\{X_n\}$, is achieved by applying the scalar factor to the non-normalized eigenvector in the following manner.

$$\{X_n\}=\{\tilde{X}_n\}S_n^{-1/2} \qquad \textbf{4.22}$$

This, along with the orthogonality condition, (see Appendix D) for the mass matrix leads to the following equation.

$$[X]^T[M][X]=[I] \qquad \textbf{4.23}$$

where,

| | |
|---|---|
| $[X]$ | matrix of normalized eigenvectors |

The eigenvectors or mode shapes are said to be orthonormal relative to the mass matrix.

### 4.2.5 Conversion to Normal Coordinates

With the eigenvectors established, it is necessary to convert the system from physical to normal coordinates. In matrix notation the modal displacement vector is related to the mode shape and modal amplitude as follows.

$$\{x(t)\}=[X]\{z(t)\} \qquad \textbf{4.24}$$

where,

| | |
|---|---|
| $\{x(t)\}$ | geometric or physical coordinate vector, |
| $[X]$ | $\mathrm{N}\times\mathrm{N}$ mode shape matrix, |
| $\{z(t)\}$ | modal amplitude or normal coordinate vector. |

The mode shape matrix serves to transform the normal coordinate vector to the physical coordinate vector.

Due to the orthogonality properties of mode shapes, it is possible to evaluate the normal coordinates simply by pre-multiplying both sides of Equation 4.24 by $\{X_n\}^T[M]$. Since $\{X_n\}^T\,[M]\{X_m\}=0$ when $n \neq m$, this leads to the following relationship for a normal coordinate vector.

$$\{z_n(t)\}=\frac{\{X_n\}^T\,[M]\{x(t)\}}{\{X_n\}^T\,[M]\{X_n\}} \qquad \textbf{4.25}$$

Substituting Equation 4.24 into Equation 4.1, and pre-multiplying by $\{X_n\}^T\,[M]$ leads to

$$[X]^T\,[M][X]\{\ddot{z}(t)\}+[X]^T\,[K][X]\{z(t)\}=[X]^T\,\{F(t)\} \qquad \textbf{4.26}$$

Further substitution of Equation 4.27 leads to Equation 4.28.

$$[X]^T\,[K][X]=\{\omega^2\}[X]^T\,[M][X] \qquad \textbf{4.27}$$

$$[X]^T\,[M][X]\{z(t)\}=\frac{1}{\omega^2}[X]^T\,(\{F(t)\}-[M][X]\{\ddot{z}(t)\}) \qquad \textbf{4.28}$$

Rearranging this leads to

$$[X]^T[M][X]\{\ddot{z}(t)\}+\{\omega^2\}[X]^T[M][X]\{z(t)\}=[X]^T\{F(t)\} \quad \textbf{4.29}$$

Substituting Equation 4.30 and 4.31 into Equation 4.29 leads to Equation 4.32.

$$[X]^T[M][X]=[I] \quad \textbf{4.30}$$

$$(\{F(t)\}-[M][X]\{\ddot{z}(t)\})=\{F(t)\}_{\text{measured}} \quad \textbf{4.31}$$

$$\{z(t)\}=\frac{1}{\omega^2}[X]^T\{F(t)\}_{\text{measured}} \quad \textbf{4.32}$$

Here $\{z(t)\}$represents the displacement time-series vector in normal coordinates. This vector can then be differentiated numerically to yield the acceleration time-series vector in normal coordinates. A discussion of numerical differentiation is given in Appendix E.

### 4.2.6 Re-conversion to Physical Coordinates

The acceleration vector in physical or geometric coordinates is obtained by applying Equation 4.33. Finally, the inertial force is obtained by the use of Equation 4.17. The inertial force is subtracted from the force measured using the force transducers to obtain the actual applied force on the structure.

$$\{\ddot{x}(t)\} = [X]\{\ddot{z}(t)\} \qquad \textbf{4.33}$$

## 4.3 Implementation of Normal Mode Approach

The normal mode approach outlined in Section 4.2 is implemented for a three degree-of-freedom system in Matlab script *normal_mode_v3_3dof_nodamping.m*, which is shown in Appendix K.

The program requires the input of the system mass matrix, the measured decay force vectors, the natural frequency vector, and the implicit time series spacing. The mass matrix, shown in Table 4.2, is assumed to be diagonal. Each diagonal term represents the deck mass for the jacket structure model described in Section 3.0. For a higher DOF system the additional diagonal terms would represent added moment of inertia.

Table 4.2 – Mass Matrix for Jacket Structure Model (B) (kg full scale)

| | | |
|---|---|---|
| 5.53e06 | 0.0 | 0.0 |
| 0.0 | 5.53e06 | 0.0 |
| 0.0 | 0.0 | 5.53e06 |

The method to determine the matrix of measured decay force vectors is described in Appendix C. For this investigation, three different decay force vector matrices were used. This was done to illustrate the sensitivity of the results to these parameters. Table 4.3 shows the three, 3x3 matrices used. These contain the so-called magnitudes of the signals.

Table 4.3 – Matrix of Measured Decay Force Vectors
Jacket B

Decay Force Vector Matrix 1

| Measurement DOF | Excitation Mode 1 (X) | Excitation Mode 2 (Y) | Excitation Mode 3 (Z) |
|---|---|---|---|
| X | 23.3 | 0.85 | 1.7 |
| Y | 1.1 | 22 | 2 |
| Z | -0.052 | 0.104 | 20 |

Decay Force Vector Matrix 2

| Measurement DOF | Excitation Mode 1 (X) | Excitation Mode 2 (Y) | Excitation Mode 3 (Z) |
|---|---|---|---|
| X | 23.75 | 0.475 | 1.95 |
| Y | 1.247 | 15 | 1.158 |
| Z | -0.094 | 0.047 | 20.6 |

Decay Force Vector Matrix 3

| Measurement DOF | Excitation Mode 1 (X) | Excitation Mode 2 (Y) | Excitation Mode 3 (Z) |
|---|---|---|---|
| X | 22.303 | 1.325 | -0.525 |
| Y | 1.021 | 26.235 | 0.870 |
| Z | -0.069 | 0.085 | -5.838 |

The results obtained by using the three decay force vectors were for all intents and purposes the same. The actual force signals were identical, based on visual inspection. For that reason, decay force vector matrix 2 was chosen for all remaining processing.

Table 4.4 shows the natural frequency vector defining the frequencies of the model jacket structure deck for each degree-of-freedom, in radians per second. These frequencies were determined from the measured decay test time traces.

Table 4.4 – Vector of Natural Frequencies – Jacket B (radians/s)

| X | Y | Z |
|---|---|---|
| 21.4 | 23.1 | 66.7 |

The time step is 0.0367 seconds, which is equivalent to a 200 Hz (model scale) sampling rate.

Following this input, the non-normalized eigenvectors are determined using the measured decay force vectors. The normalizing scalar is then established, and the eigenvectors are normalized.

Next the input force vectors are specified. These are the measured signals from which the inertial force component is to be removed.

Using the force vectors and the normalized eigenvectors, the displacements in the normalized coordinate system are determined. These displacements are then differentiated to obtain the accelerations in the normalized coordinate system. The differentiation is based on the central difference technique described in Appendix E. These accelerations are converted back to the physical coordinate system, and then used to determine the inertial force component. This force is subtracted from the measured total force, to obtain the actual force.

The input signals are shown in Figures 4.1 to 4.9. Figures 4.1 to 4.3 are decay test measurements. These signals are completely inertial in nature, and the normal mode approach should (in theory) reduce them to steps or flat lines as appropriate. These signals represent the forces measured in the X, Y, and Z directions, due to a load applied in the X direction. See Section 3.0 for a description of the coordinate system. Figures 4.4 to 4.9 are individual wave impact events.

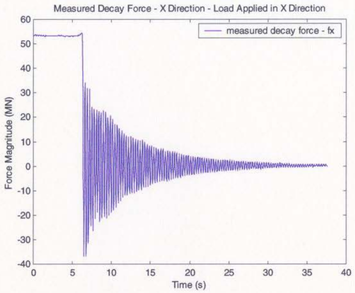

**Figure 4.1 Measured Decay Force – Jacket B – X Direction**

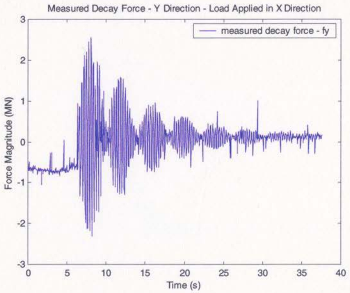

**Figure 4.2 Measured Decay Force – Jacket B – Y Direction**

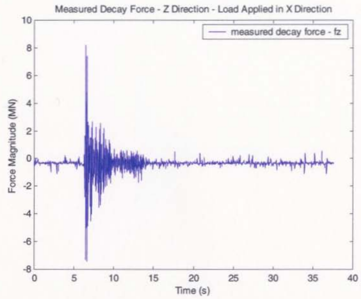

**Figure 4.3** **Measured Decay Force – Jacket B – Z Direction**

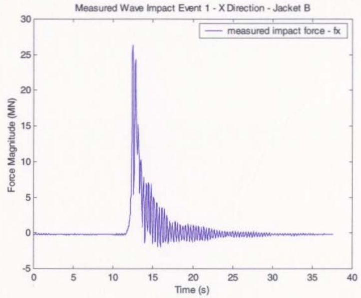

**Figure 4.4** **Measured Wave Impact Event 1 – Jacket B – X Direction**

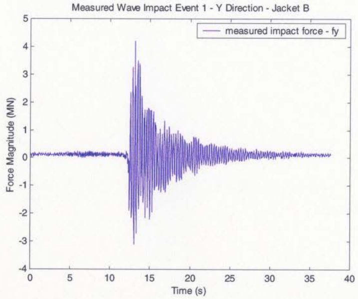

**Figure 4.5 Measured Wave Impact Event 1 – Jacket B – Y Direction**

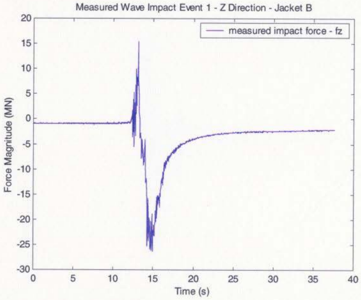

**Figure 4.6 Measured Wave Impact Event 1 – Jacket B – Z Direction**

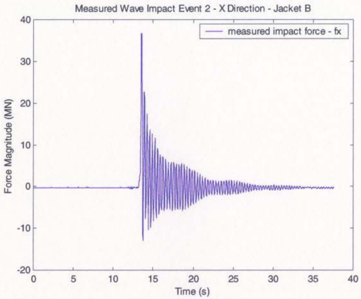

**Figure 4.7** **Measured Wave Impact Event 2 – Jacket B – X Direction**

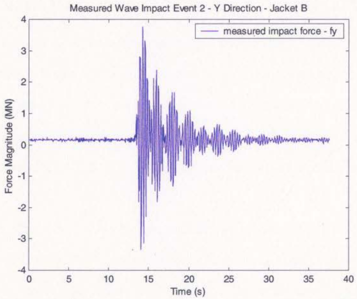

**Figure 4.8** **Measured Wave Impact Event 2 – Jacket B – Y Direction**

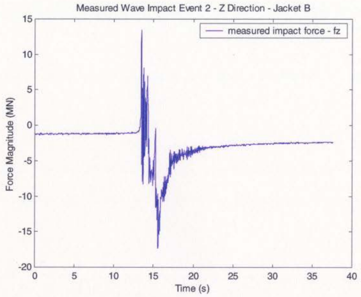

**Figure 4.9 Measured Wave Impact Event 2 – Jacket B – Z Direction**

Figures 4.10 to 4.12 compare the measured and actual forces for a decay test. Ideally, the actual force, in Figure 4.10, would appear as a step. Instead, considerable high frequency vibration is observed at the discontinuity. Despite this, there is clearly some signal reduction. Figure 4.12 shows some minor signal removal from the Z direction measurement. Figure 4.11 shows no signal reduction from the Y direction measurement.

Figures 4.13 to 4.15 compare the measured and actual force results for a single wave impact event. Again, the X and Z direction plots show reasonable actual force signals, with some reduction of inertial force. The Y direction plot shows no signal reduction, however, the overall scale or magnitude of the signal is quite small compared to those in the other degrees-of-freedom.

Figures 4.16 to 4.18 show similar results for another wave impact event from the same test.

This method was also implemented in a six degree-of-freedom form, but the results were equally unsatisfactory, so they will not be presented here.

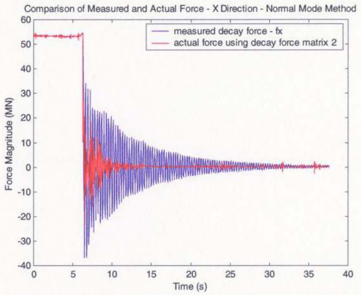

**Figure 4.10 Measured and Actual Force – X Dir. – Normal Mode**

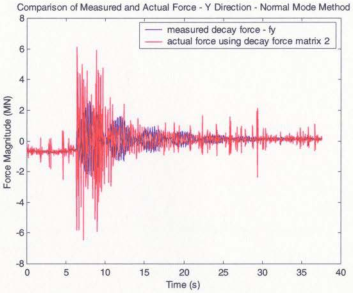

**Figure 4.11 Measured and Actual Force – Y Dir. – Normal Mode**

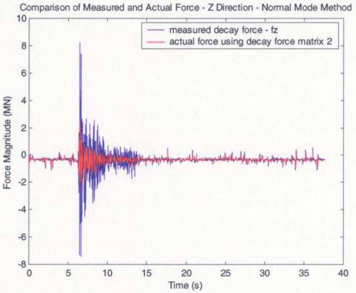

**Figure 4.12 Measured and Actual Force – Z Dir. – Normal Mode**

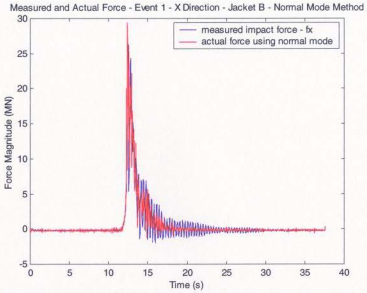

**Figure 4.13 Wave Event 1 and Actual Force – X Dir. – Normal Mode**

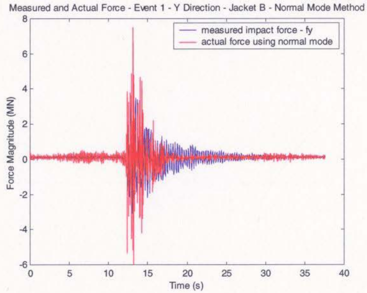

**Figure 4.14 Wave Event 1 and Actual Force – Y Dir. – Normal Mode**

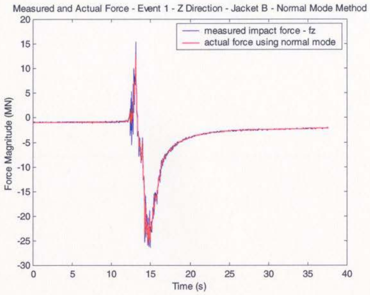

**Figure 4.15 Wave Event 1 and Actual Force – Z Dir. – Normal Mode**

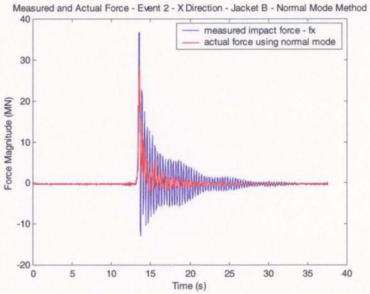

**Figure 4.16 Wave Event 2 and Actual Force – X Dir. – Normal Mode**

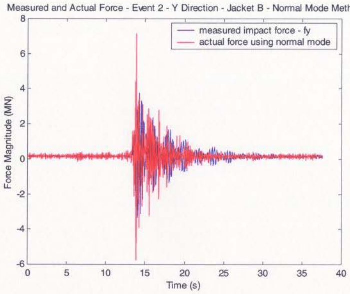

**Figure 4.17 Wave Event 2 and Actual Force – Y Dir. – Normal Mode**

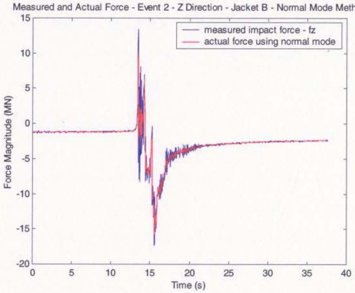

**Figure 4.18 Wave Event 2 and Actual Force – Z Dir. – Normal Mode**

## 4.4 Limitations of the Normal Mode Approach

Clearly the results obtained from this method fall short of expectations. This section will list and discuss some of the limitations of the normal mode approach.

1. The method is time consuming to implement. It is not a method that can be utilized without significant preparation. Its usefulness depends on the availability of quality decay test measurements, and accurate model physical properties (such as mass, radii of gyration, and natural frequency).

2. The success of the approach is subject to the vagaries of the system measuring system. For example, does the dynamometer have some inherent cross-talk? Were the loads applied to excite the decay tests in-line with the model coordinate system? These are questions that will plague a user of this technique, particularly if they are not involved in the model test program.

3. The major problem with this technique, when applied to wave impact cases, is the determination of the added mass. As the wave impacts the deck of the model, it will contribute to the mass of the structure. This added mass will vary spatially and temporally as the wave travels along the deck. The result is a fluctuating system natural frequency, and the introduction of fluid damping. The quantification of added

mass for structures of this type (subjected to wave impact loading) is an area of research that needs significant attention.

4. The method used to establish the matrix of decay force vectors using fast Fourier transforms, described in Appendix C, is really intended for use with sinusoidal type signals. Application to decay type signals likely introduces a source of error.

5. The normal mode technique makes the simplifying assumption that there is no damping, and this is reflected in the defined equation of motion. Clearly structural damping is present, as well as damping related to fluid-structure interaction. This is another topic recommended for future investigation. An attempt was made to introduce a damping term to the present equation of motion, based on Rayleigh damping. The results of this are presented in Appendix F.

6. A good indicator that this method has not been successful is the failure to satisfy the orthogonality condition, as defined in Equation 4.23. The result should be an identity matrix, with values of 1.0 along the diagonal, and zeros elsewhere. As seen in Table 4.5, the off diagonal terms are non-zero, and are too large to be considered due to round off.

7. The greatest limitation with this method, and indeed all the methods to be discussed here, is the difficulty in quantifying whether a particular actual force result is

satisfactory. Other than a subjective visual assessment of the signal and perhaps its FFT, in comparison with the original signal, there are no obvious, quantifiable parameters to gauge the suitability of the modified signal. This is an area that requires further research.

Table 4.5 – Orthogonality Condition Not Satisfied (see Equation 4.23)

| | | |
|---|---|---|
| 1.0000 | 0.0840 | 0.0930 |
| 0.0840 | 1.0000 | 0.0619 |
| 0.0930 | 0.0619 | 1.0000 |

# 5.0 Inverse Fourier Transform Method

## 5.1 Background

Kaplan (1992), Kaplan et al (1995), and Murray et al (1995) discussed the use of the Inverse Fourier Transform method to remove inertial forces from measured force signals. This can be applied to force measurements from the deck dynamometer of a model jacket structure subjected to wave impact, as described in Section 3.0.

Kaplan (1992) stressed the importance of removing the effects of measurement system dynamics from the total force measurement. These so-called inertial forces are due to structural acceleration, and are seen as forces by the load measuring devices. The removal of these extraneous force components is necessary in order to obtain the actual applied force. The manner in which the actual force differs from the measured force will depend on the time extent and form of the applied impulsive force, and the natural frequency of the measurement system.

According to Kaplan (1992), the measured force is related to the actual force using a convolution integral expression, as shown in Equation 5.1. The convolution integral and its use in the determination of the response to an arbitrary excitation is discussed in general terms in Thomson (1981).

$$f_m(t) = \int_0^t h(t-\tau)\, f_a(\tau) d\tau \qquad \textbf{5.1}$$

where

$f_m(t)$ represents measured force,

$f_a(\tau)$ represents actual force,

$h(t-\tau)$ represents impulse response.

Applying a Fourier transform operation leads to the relationship in Equation 5.2.

$$F_m(i\omega) = H(i\omega)\, F_a(i\omega) \qquad \textbf{5.2}$$

where

$H(i\omega)$ represents frequency response or the Fourier transform of $h(t)$,

$F_m(i\omega)$ represents the Fourier transform of $f_m(t)$,

$F_a(i\omega)$ represents the Fourier transform of $f_a(t)$.

Knowing $F_m(i\omega)$ *and* $H(i\omega)$, Equation 5.2 can be rearranged to obtain $F_a(i\omega)$.

$$F_a(i\omega) = F_m(i\omega) / H(i\omega) \qquad \textbf{5.3}$$

The actual force time series can then, in theory, be determined by taking the inverse Fourier transform of $F_a(i\omega)$.

Kaplan et al (1995) and Murray et al (1995) stated that the implementation of this method is complicated. This is particularly true for multi-degree-of-freedom (MDOF) systems, due to the presence of mode coupling. They claimed that a matrix of cross-coupling terms would be required, and developing this would be extremely difficult. In addition, the effects of added mass and fluid damping further complicate the situation when considering wave induced impact loads.

## 5.2 Implementation of Inverse Fourier Method

Despite these warnings, an attempt has been made here to implement this procedure, first in a single degree-of-freedom (SDOF) form, and then in MDOF form. The measured force signals, and the impulse response signals will be taken from decay tests conducted for the experiments described in Section 3.0. The reason for this is two-fold. First, the use of in-air measurements eliminates the effects of added mass and fluid damping. Second, the signal from a decay test will be purely inertial following the release of the initially applied load. The successful removal of the inertial component will result in a signal resembling a rectangular pulse.

Murray et al (1995) described how the impulse response for a system could be developed from the derivative of the response to a pulse or step. This serves as the rationale for using the decay test to obtain the impulse response. The decay test is performed by applying a known load to the system, in line with a particular degree-of-freedom (DOF). The statically applied load is released instantaneously, and the decay force time trace is measured through the system instrumentation. The measured response is typically used to obtain the natural frequency and damping characteristics of the system. Decay tests are repeated for each relevant degree-of-freedom (DOF).

### 5.2.1 SDOF Implementation – Frequency Domain Division

This section will describe the implementation of the inverse Fourier transform method in SDOF form, using measured and simulated data. It will be seen that the application of the technique is not straightforward. The signal modifications required to facilitate the procedure will be illustrated. See Table 5.1 for a summary of the procedure in SDOF format.

A set of decay tests was conducted on the instrumented deck of a jacket structure model (Section 3.0). A typical decay test force measurement, from the dynamometer, is shown in Figure 5.1. For this demonstration the signal will serve as the source for the measured force signal, $f_m(t)$, and the impulse response, $h(t)$. The measured force signal, $f_m(t)$, is selected from the original signal, and is shown in Figure 5.2. Points from 330.71 to

Table 5.1 - Procedure for Application of the
Inverse Fourier Transform Method (SDOF Format)

| Step | Description |
|---|---|
| 1 | Establish step or pulse response vectors. These can be taken from measured decay force tests. The exact starting point is a matter of trial and error, though in theory it should coincide with the release of the applied load (in the decay test). |
| 2 | Differentiate the step response vector to obtain the impulse response vector. |
| 3 | Take the Fast Fourier Transform (FFT) of the measured input vector. |
| 4 | Take the FFT of the impulse response vector, to obtain the frequency response vector. |
| 5 | Normalize the frequency response vector by its own initial value. This is required to maintain the same DC component between the measured input vector, and the derived output vector. |
| 6 | Divide the FFT of the measured input vector by the normalized frequency response vector to obtain the FFT of the actual force vector. |
| 7 | Take the inverse Fourier transform of the actual force vector FFT to obtain the actual force time series. |

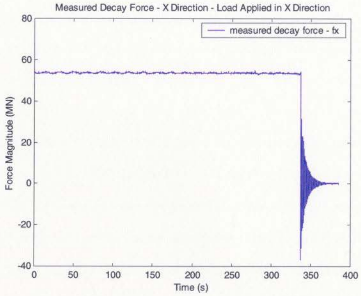

**Figure 5.1 Total Measured Decay Force – X Direction**

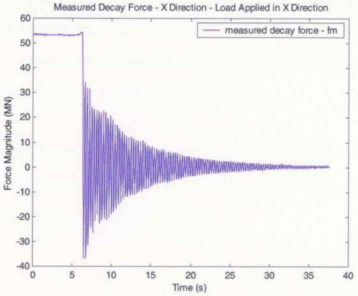

**Figure 5.2 Selected Measured Decay Force – X Direction**

368.30 seconds of the original signal are selected. Note that the length of the $f_m(t)$ signal is 1024 points. It is recommended that that when signals are subjected to a Fast Fourier Transform (FFT) that they be a power of 2 in length. This ensures the most efficient application of the FFT.

In a similar manner, the step or pulse response used to derive the impulse response function, $h(t)$, is also selected from the original signal. In order to demonstrate the sensitivity of the overall procedure to the signal selection, two pulse responses have been created. Signals $s1(t)$ and $s2(t)$ represent selections from times 336.9976 to 374.5847 seconds and 337.0344 to 374.6214 seconds, respectively. These selections are one time step in the difference. They are shown in Figures 5.3 and 5.4. Note that these signals are also 1024 points in length. This is required to ensure efficient application of the FFT algorithm, and also to facilitate division of the signals in the frequency domain. It is necessary that the signals involved in the vector division be the same length. This could be considered a minor shortcoming of this method.

Next the pulse response signals are differentiated, using the algorithm described in Appendix E. These derivatives, defined $h1(t)$ and $h2(t)$, represent the response to an impulse, and are shown in Figures 5.5 and 5.6.

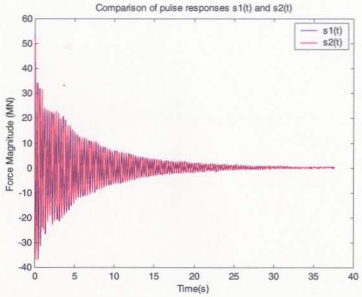

**Figure 5.3 Step or Pulse Responses**

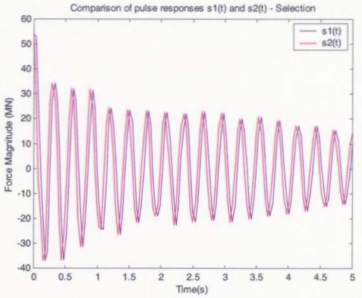

**Figure 5.4 Step or Pulse Responses – Selected View**

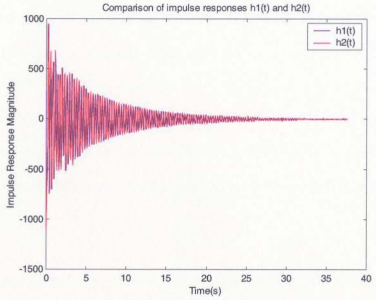

**Figure 5.5 Impulse Responses**

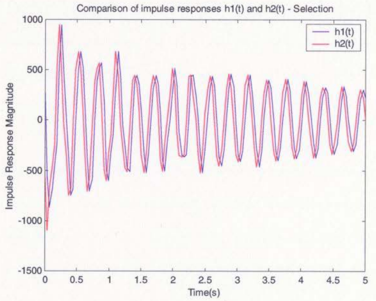

**Figure 5.6 Impulse Responses – Selected View**

The next step is to apply an FFT to each signal. Then the division of the FFT of $f_m(t)$ by the FFT of $h(t)$, is done to obtain the FFT of $f_a(t)$, as per Equation 5.3. The FFT of an impulse response is commonly known as a frequency response. The MATLAB operator for array division ( ./ ) is required here since each vector element must be divided individually. The command would be "FA1=FM./H1;". FM represents $F_m(i\omega)$, the FFT of the measured force, H1 represents $H1(i\omega)$ the FFT of the impulse response, and FA1 represents, the FFT of the actual force signal. Figures 5.7 to 5.9 show the resulting signals.

At first examination the result for $F_a1(i\omega)$ looks promising. The resonant peak has been removed and the remaining signal is reminiscent of a sinc function, as would be expected. However, if the inverse Fourier transform of this is obtained, the result does not resemble the expected rectangular pulse. The result is shown in Figure 5.10. The signal appears to be inverted, the overall magnitude of the signal does not compare with the original $f_m(t)$, and there is an unexpected high frequency component.

In the ideal situation, the inertial component would be completely removed from the $f_m(t)$ signal, to reveal an actual force ( $f_a(t)$ ) signal resembling a rectangular pulse. The mean of these signals would be identical. This can only be achieved if the DC component of their respective FFT's is identical. Therefore, for the frequency domain division

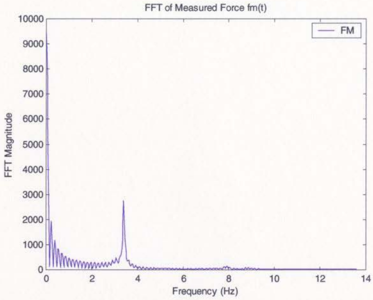

**Figure 5.7 FFT of Measured Decay Force**

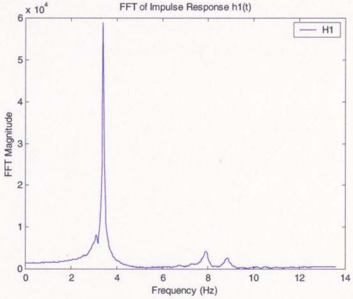

**Figure 5.8 Frequency Response**

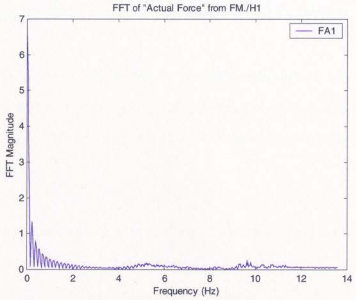

**Figure 5.9** **FFT of Actual Force**

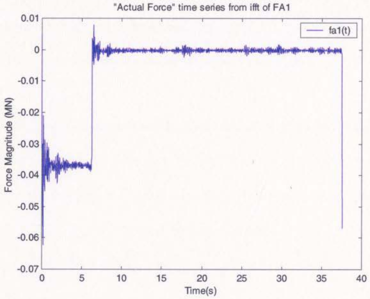

**Figure 5.10** **Actual Force Time Series**

described above, it is necessary that the DC component of the frequency response vector have a value of 1.0.

The DC component of a signal (frequency domain) is related to the mean of the signal time series multiplied by the number of points in the time series. Strictly speaking, the DC component of the frequency domain vector is equal to the sum of the time domain vector elements.

The example from above can be re-done using this modification. The frequency response vector is scaled as follows, "H1S=H1/H1(1);". Figure 5.11 shows the modified frequency response vector. The DC component now has a value of 1.0. Dividing $F_m(i\omega)$ by $H1S(i\omega)$ yields $F_a1S(i\omega)$, as shown in Figure 5.12. Taking the inverse Fourier transform of this leads to the signal shown in Figure 5.13. Figure 5.14 plots the actual force, $fa1s(t)$, in comparison with the measured force, $f_m(t)$.

The scaling procedure has inverted the signal, and adjusted the overall magnitude so that it resembles that of $f_m(t)$. Note however that this $f_a(t)$ signal shows an unexpected high frequency component. This high frequency component is not apparent in the $f_m(t)$ signal or its FFT, $F_m(i\omega)$. This will be discussed later.

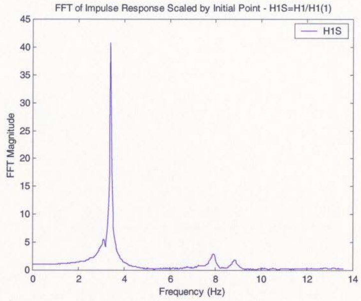

**Figure 5.11 Scaled Frequency Response**

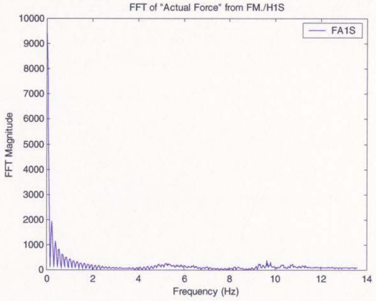

**Figure 5.12 FFT of Actual Force - Using Scaled Frequency Response**

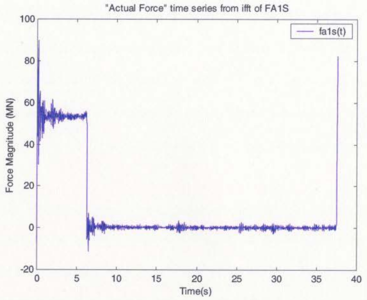

**Figure 5.13 Actual Force Vector - Using Scaled Frequency Response**

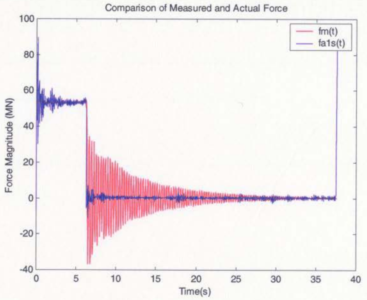

**Figure 5.14 Measured Decay and Actual Force – IFT Method**

In order to demonstrate the sensitivity of the process to the selection of the impulse response $h(t)$, the procedure is repeated using $h2(t)$, which is shifted from $h1(t)$ by one time step (i.e. its selection started at time 337.0344 seconds of the total measured force signal, rather than 336.9976). Figures 5.15 and 5.16 clearly show that using $h2(t)$ as the impulse response produces an actual force signal that differs considerably from that produced by $h1(t)$, showing even more high frequency contamination.

The shift in the step response selection (by one point) causes an increase in the element sum (sum of all the time series vector elements) of the subsequent impulse response. As a result the DC component of the frequency response is higher. When the normalization is done, the entire vector is divided by the larger number and therefore has smaller values in the high frequency range.

Subsequent division with the FFT of the measured force signal $f_m(t)$ causes the high frequency range of the resultant $F_a(i\omega)$ to be amplified.

Ultimately the selection of the "proper" step response / impulse response will be a matter of trial and error.

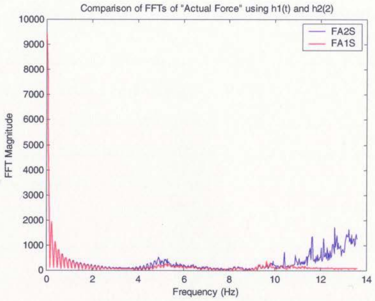

**Figure 5.15 Comparison of Alternate Frequency Responses**

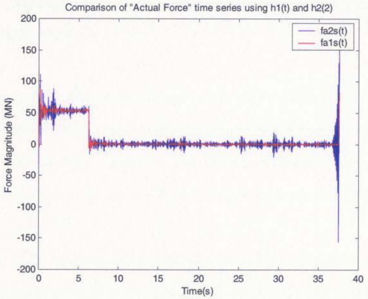

**Figure 5.16 Actual Force Vectors From Alternate Freq. Responses**

### 5.2.2 SDOF Implementation - Use of Convolution and Deconvolution Techniques

Convolution and deconvolution are procedures to combine signals in the time domain. They are equivalent to multiplication and division (in the frequency domain), respectively. The procedure defined by Kaplan (1992), and shown in Equation 5.3, is equivalent to a deconvolution in the time domain. A brief description of convolution is given in Appendix G.

It would be of some interest to implement the procedure as a deconvolution (time domain) and compare it to the result from the frequency domain division. MATLAB conveniently provides scripts for this purpose (*conv* and *deconv*). As with the frequency domain division, however, it will be seen that the implementation of the deconvolution is not straightforward.

The same signals employed for the frequency domain division will be used. Both the measured force signal, $f_m(t)$, and the pulse response will be selected from the original decay test signal shown in Figure 5.1. The pulse response is differentiated to obtain the impulse response, $h1(t)$, shown in Figure 5.5. Both signals have a length of 1024 points.

The MATLAB deconvolution command (*deconv*) is implemented as follows, "*[q,r]=deconv(c,a);*", where vector "*a*" is deconvolved out of vector "*c*" to obtain the vector "$q$", the quotient, and the vector "$r$", the remainder. In this example, the $f_m(t)$ will be "$c$", and $h1(t)$ will be "$a$".

Implementation of the deconvolution, with the signals unmodified, results in a "$q$" vector that contains only one point. Smith (1999) describes how when two signals, of length $M$ and $N$ respectively, are convolved, the resulting signal has a length ($M$+$N$ – 1). It would be reasonable to assume that the signal "$c$" in the deconv command, be larger than the signal "$a$". Since $h1(t)$ has a length of 1024 points, $f_m(t)$ would need a length of 2047 point in order to deconvolve a signal "$q$" of length 1024 points. The solution is to pad the signal "$c$" ( $f_m(t)$ in this example) with zeros, to a length of 2047 points. The result of doing this, and deconvolving, yields a "$q$" as shown in Figure 5.17. Clearly this does not resemble the expected rectangular pulse signal. Some other modification must be required.

One modification that facilitates the deconvolution process is the addition of an impulse of magnitude 2.0 to the first point of the defined impulse response vector. This procedure is discussed in Appendix H.

This procedure was applied to the measured $f_m(t)$ and $h1(t)$ signals from above. The $h1(t)$ signal must first be scaled by the factor (-mean( $h1(t)$ )*1024), where 1024 is the number of points in the $h1(t)$ signal, to yield $h1s(t)$. This is identical to the normalization done in the frequency domain to achieve a DC component of 1.0. The scaled impulse response vector is then deconvolved out of $f_m 2(t)$, the version of $f_m(t)$ padded with zeros. This yields the still unrealistic signal for "q", shown in Figure 5.17.

The value of the first point of $h1s(t)$ (a[1]) is then altered, iteratively, and the deconvolution quotient, q[n], is recalculated until the its mean matches the mean of $f_m(t)$ (note that this is the mean of the non-padded signal). For this example, the original value of the initial point of the scaled impulse response (h1s (1)), was ultimately changed,

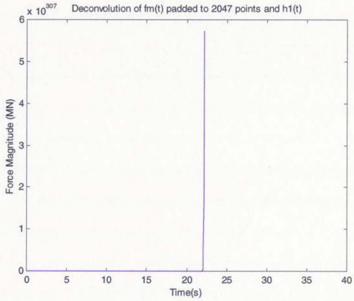

**Figure 5.17 Result of Deconvolution**

from 0.1936 to 2.19331 before the mean of q[n] matched the mean of $f_m(t)$. q[n] then resembles what could reasonably be considered the actual force signal, $f_a(t)$, as shown in Figure 5.18.

What has happened here? The vector element sum of the original unmodifed impulse response ( $hl(t)$ ) had a value of $-1.0$. The modified impulse response ( $hls(t)$ ) had a sum equal to 1.0. Essentially an impulse of magnitude 2.0 has been added to the time series. The absolute value of the DC component in frequency domain remains unchanged at 1.0. There is however, a shift in the frequency response vector, which is particularly noticeable in the high frequency range. As an aside, the FFT of a single impulse of amplitude 2.0 would be a flat line of magnitude 2.0.

Figure 5.19 compares the actual force determined using the inverse Fourier transform method to that from the deconvolution. It is seen that the general shape and magnitude of the signals are comparable, but that the deconvolved signal is considerably less noisy. The noise reduction is due to the signal shift in the high frequency range of the frequency response, the result of the addition of the impulse. The appropriateness of this technique may be questionable, however, it could be argued that it is legitimate since it merely filters out high frequency components that are themselves not legitimate, the result of the numerical process. Otherwise it could be viewed as a form of low-pass filter. The deconvolved signal does show some oscillation or noise near the discontinuity, see Figure 5.20. This may be caused by some peculiarity of the deconvolution process, or by Gibb's

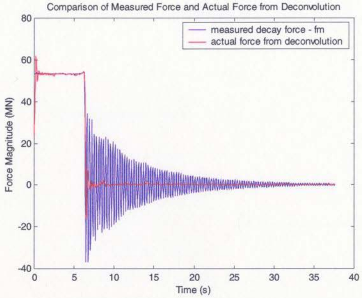

**Figure 5.18 Measured Decay and Actual Force from Deconvolution**

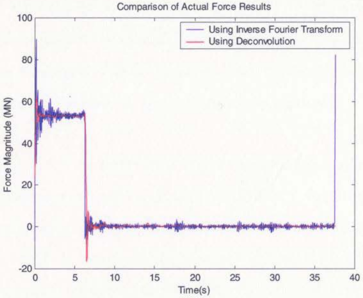

**Figure 5.19 Actual Forces from IFT and Deconvolution Methods**

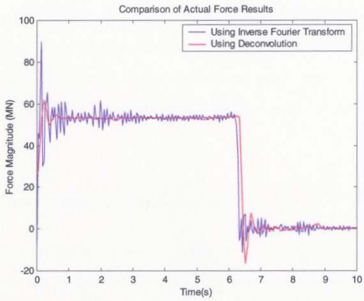

**Figure 5.20 Actual Forces from IFT and Deconvolution Methods**

Effect (see Appendix I). In either case this illustrates the single biggest drawback of this technique when applied to impact type processes; that it is likely impossible to separate the actual impact force from the signal corruption at the discontinuity.

Figures 5.21 to 5.24 compare the results of the inverse Fourier transform and deconvolution methods applied to simulated data. Figure 5.21 compares the computed actual force time series, using the deconvolution technique (q), and the inverse Fourier transform (fa), to the original simulated "measured" force signal. Figure 5.22 shows the Fourier transforms for these signals. This example uses a low frequency sinusoidal input signal. Note the output signals are rounded at the discontinuity. The question is whether this rounding represents Gibbs Effect, since the anticipated oscillation is not evident. Figures 5.23 and 5.24 show similar plots using a higher frequency simulated "measured" force signal. Note that in this case the rounding is no longer evident at the discontinuities, instead, the expected oscillation typical of Gibbs Effect occurs.

### 5.2.3 MDOF – Frequency Domain Division

Up to this point the discussion has focussed to the application of these techniques to single degree-of-freedom systems. In a realistic situation the force measurements would likely be made (using a dynamometer) in at least three degrees-of-freedom simultaneously. So it is necessary to investigate the implementation of the inverse Fourier transform method, and the deconvolution method in a MDOF situation.

The investigation will be limited to a three degree-of-freedom system, where orthogonal force measurements represent the X, Y and Z directions. As before, the force measurements used here are taken from decay test results from the jacket structure model tests described in Section 3.0. Figure 3.5 defines the coordinate system.

Appendix K shows the MATLAB script for the implementation of the inverse Fourier transform method, in matrix form, called *FFT_matrix_attempt1b.m.*

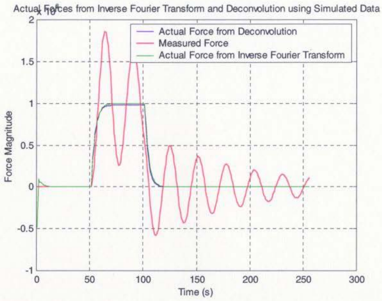

**Figure 5.21 Comparison of Methods – FFTs of Simulated Time Series**

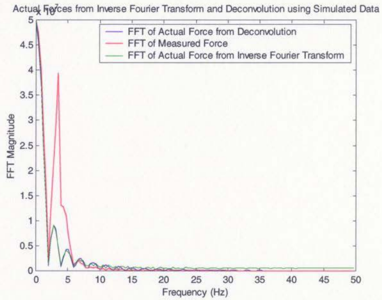

**Figure 5.22 Comparison of Methods – FFTs of Simulated Time Series**

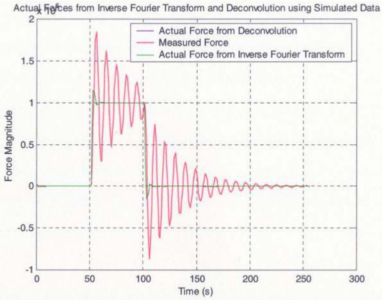

**Figure 5.23 Comparison of Methods Using Simulated Time Series**

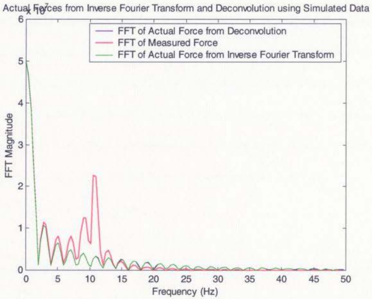

**Figure 5.24 Comparison of Methods – FFTs of Simulated Time Series**

Nine time traces, from three separate decay tests, are loaded into the MATLAB workspace. From these the so-called measured force signals are selected. The selections have a length of 1024 points, though any other power of two could have been used. Also from the original decay test signals, the step response signals are selected. As with the single degree-of-freedom implementation, these are selected at the point where the applied load is released. The step response signals are also 1024 points in length. The files are saved to the computer hard drive in ASCII format. This is required so that they can be differentiated using the script *First_deriv.m*, (Appendix K) to obtain the impulse response vectors.

A fast Fourier transform (FFT) is then applied to the measured force vectors and the impulse response vectors using the MATLAB command *fft.m*. The frequency response vectors (the ffts of the impulse response vectors) are then normalized in the following manner. Each frequency response vector is normalized by the initial point value of its relevant diagonal term vector. For example, the vectors (three in total) resulting from load applied in the X direction are divided by the first point of the X direction measurement. Likewise for the vector sets due to loads applied in the Y and Z directions. This ensures that the matrix of initial points (from the nine frequency response vectors) will have 1.0's along the diagonal.

The division of the measured force FFTs by the frequency responses in matrix form is done in the following manner. First , three of the nine available measured force signals are chosen, and stored in a common matrix, size 3x1024. In the script shown in Appendix K, the signals measured from the test where the load was applied in the X direction were used. Next, in a "for loop", the frequency response vectors are stored in a 3x3 matrix, and used to divide the 3x1 column vector from the measured force matrix. That is, the first points from each of the nine frequency response vectors are used to develop a 3x3 matrix. Similarly, the first points from each of the measured force signals are used to develop a 3x1 column vector. The matrix division of these yields a 3x1column vector representing the first points of three "actual force" vectors. This is repeated 1024 times (in this case), until the three complete "actual force" vectors are generated.

The inverse Fourier transform is then taken of each row vector in the "actual force" matrix, to obtain the "actual force" time series. Figures 5.25 to 5.26 compare the "actual force" time series to the original "measured force" time series. The results appear to be as good as those using the SDOF method, at least in the X direction. In the Y and Z directions, there appears to be little or no signal suppression. The overall magnitude of the Y and Z signals is, however, quite small compared to the X signal, as would be expected in this loading condition. Perhaps it is optimistic to expect any signal removal in these degrees-of-freedom?

Several other attempts were made to implement this procedure, using different approaches to the development of the frequency response matrix and its application in the

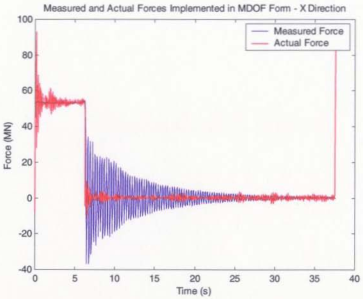

**Figure 5.25 Result of Inverse Fourier Transform in MDOF Form**

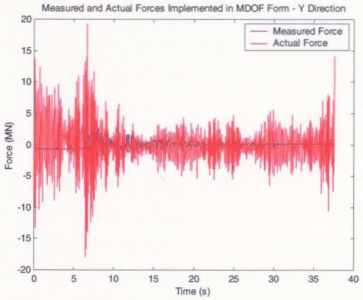

**Figure 5.26 Result of Inverse Fourier Transform in MDOF Form**

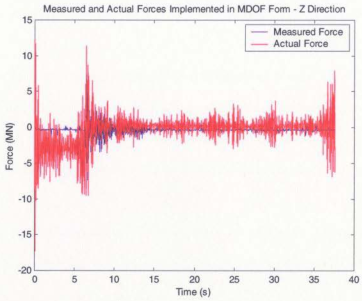

**Figure 5.27 Result of Inverse Fourier Transform in MDOF Form**

generation of the "actual force" vectors. These provided no improvement over the method outlined above, so they will not be presented here.

### 5.2.4 MDOF – Use of Deconvolution Techniques

The next implementation attempt is conducted using time domain signals, making use of the MATLAB deconvolution command (*deconv*), as was done for the SDOF case. The initial portion of the script is identical to the frequency domain attempt, to the point where the impulse response vectors are normalized. The concept here is to deconvolve the impulse response signals out of the measured force signals to obtain nine vector components of actual force.

The first problem with this approach becomes apparent when the deconvolutions are attempted. The quotient signals produced bear no resemblance to measured force signals from which they are deconvolved. As with the SDOF implementation, the impulse response signals require that the first point be modified to facilitate the deconvolution.

For the so-called "diagonal term" vectors, where the DOF of the applied load and measurement coincide, the addition of a 2.0 magnitude impulse works well.

When the value of the first point of the impulse response vector is modified, the comparison between the "measured force" and the deconvolved quotient signal can be improved. To demonstrate this, the impulse response vectors representing collinear applied and measured loads, have had their first points modified by the addition of an impulse of magnitude 2.0. As with the SDOF example this changes the sum of the vector elements from –1.0 to 1.0. The comparisons are shown in Figures 5.28 to 5.30. The vector "h_x_fx" represents the impulse response vector of a force measured in the X direction, due to a load applied in the X direction. Similarly, "h_y_fy" and "h_z_fz" represent collinear applied and measured loads. Table 5.2 shows the modifications to the initial point values of the these impulse response  signals.

The comparison in Figure 5.28 shows excellent signal removal. This should be expected since both the measured force and the impulse response vector originate from the same signal. The comparisons in Figures 5.29 and 5.30 show only moderate signal removal. In these cases the measured force and impulse response vectors were derived from different decay tests. Note however that the force scale is considerably smaller than for that shown in Figure 5.28.

Table 5.2 Impulse Response Vector Modifications

| Impulse Response Signal File | Description | Original Initial Point Value | Modified Initial Point Value |
|---|---|---|---|
| h_x_fx | Impulse response vector measured in X direction due to load applied in the X direction. | 0.1936 | 2.1933 |
| h_y_fx | Impulse response vector measured in Y direction due to load applied in the Y direction. | -0.3115 | 1.6900 |
| h_z_fz | Impulse response vector measured in Z direction due to load applied in the Z direction. | -0.8669 | 1.1330 |

It should be noted that the signals in Figure 5.28 are identical to those shown in Figure 5.18. The quotient signal shown in Figure 5.28 is one component of the total X direction force, while the quotient signal shown in Figure 5.18 was developed as a single DOF.

For the cases where the impulse response vectors and the measured force vectors are not collinear, reasonable results are not obtained by the modification technique. Even using an iterative approach, changing $h(1)$ until the mean of the convolved signal matches the original measured force signal does not work. Instead of iterating towards the "correct actual force", with only minor alteration of the impulse response signal, ridiculously large values need to be substituted to generate a signal that is reasonable, and then in shape only. The magnitude (quantified by the mean) in these cases will differ from that of the measured force signal by a scale factor which is equal to the value substituted for the first point of the impulse response signal.

It is not clear what justification could be used to apply such a scale factor. In addition, the scaled quotient signal not only matches the mean of the "measured force" signal, the shape is identical. When subtracted from "measured force" signal (to obtain the inertial force), the signal would essentially be zero. This might not seem unreasonable, since with this type of 3 DOF system, the amount of X direction measured force (for example) due to an applied load in the Y or Z directions would be expected to be negligible.

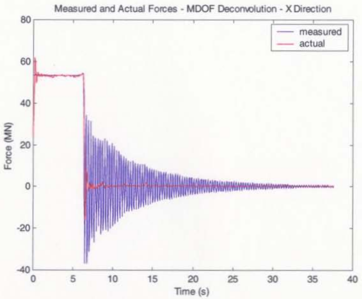

**Figure 5.28 Result of Deconvolution Method in MDOF Form**

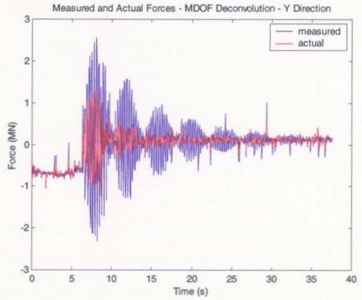

**Figure 5.29 Result of Deconvolution Method in MDOF Form**

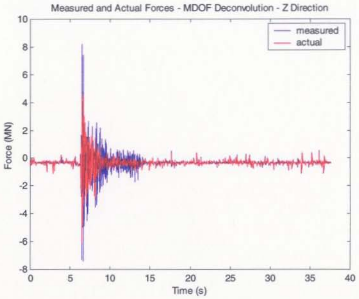

**Figure 5.30 Result of Deconvolution Method in MDOF Form**

As an example, Figure 5.31 compares the measured force signal "fm_x" to a deconvolved quotient "q", that has been developed using the impulse response signal "h_y_fx" (impulse response in the X direction to a load applied in the Y direction). The first point was iterated to a value of 22000 (from the original value of –1.6026). The quotient was then scaled by 22000 so that its magnitude (mean) matches that of the "measured force".

Appendix K contains the MATLAB script developed for this implementation, *FFT_matrix_deconv_a.m*. The impulse response vectors are normalized by the mean of the relevant "diagonal term" vector (multiplied by its length, 1024). Each vector then has

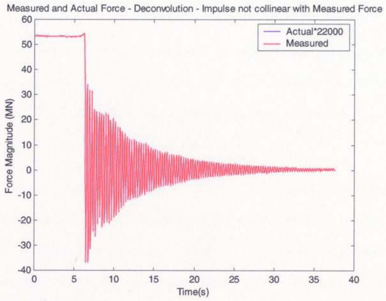

**Figure 5.31 Result of Deconvolution-Impulse and Force not Collinear**

its first point value modified by adding (or subtracting) the absolute value of two times the vector element sum of the original unmodified vector. For the "diagonal term" vectors, this means an addition (or subtraction) of 2.0. The "off-diagonal" terms would see the addition (or subtraction) of a lesser value. The concept here is to maintain the signal ratio between the relevant "diagonal" and "off-diagonal" vectors. Unfortunately, this procedure fails to produce reasonable results for "off-diagonal" term vectors.

## 5.3 Application to Measured Wave Impacts

These procedures were applied to measured wave impact signals, gathered from the model test experiments described in Section 3.0. This was implemented in SDOF and MDOF form. The MATLAB scripts for these are *ift_wave_sdof.m* and *FFT_matrix_wave.m*, and are shown in Appendix K.

The results were less than satisfactory. Figures 5.32 to 5.37 compare the measured and actual force time series, for Jacket B and Jacket A, for the SDOF case. In neither DOF was there significant signal suppression. In fact, in many, the actual force signal was amplified. There are two reasons for this. Figures 5.38 to 5.43 compare the FFTs of the measured and actual force signals. In each DOF it can be seen that the high frequency region of the actual force signal is amplified in sections. This does not represent real frequency components, but is the result rather of the division of the small measured force values by smaller values in the frequency response. Figures 5.44 to 5.46 compare the measured force FFTs and the frequency response, for Jacket A, in the X and Y degrees-of-freedom. These show the relative scales of the force signal and the frequency response in the higher frequency region. The measured signal displays considerable high frequency noise, which is amplified in the division process.

Figure 5.46 compares the FFT of the force signal and the frequency response for Jacket A in the Z DOF. This illustrates the other reason for the poor performance of this method. The resonant peak of the measured signal has been shifted from that of the frequency

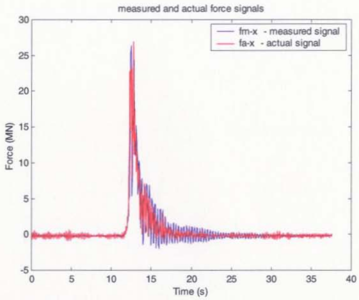

**Figure 5.32 IFT Method - Jacket B Wave Impact – SDOF – X Dir.**

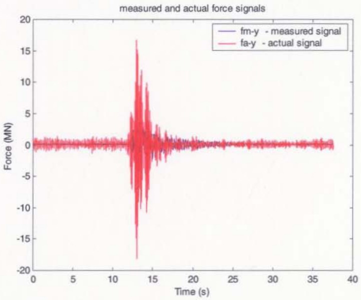

**Figure 5.33 IFT Method - Jacket B Wave Impact – SDOF – Y Dir.**

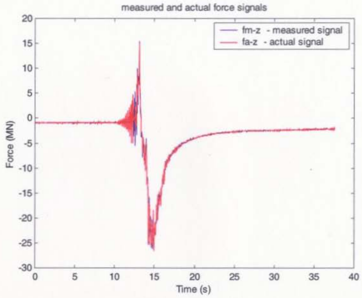

**Figure 5.34 IFT Method - Jacket B Wave Impact – SDOF – Z Dir.**

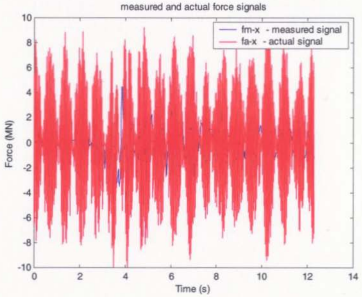

**Figure 5.35 IFT Method - Jacket A Wave Impact – SDOF – X Dir.**

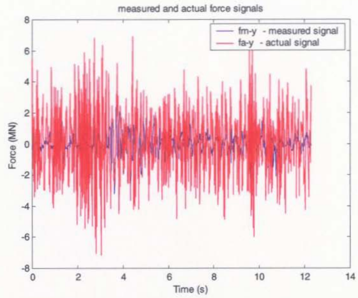

**Figure 5.36 IFT Method - Jacket A Wave Impact – SDOF – Y Dir.**

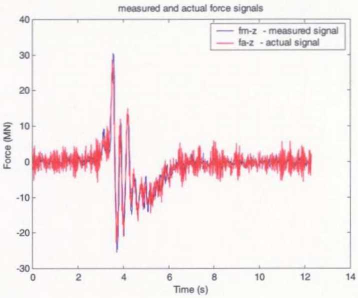

**Figure 5.37 IFT Method - Jacket A Wave Impact – SDOF – Z Dir.**

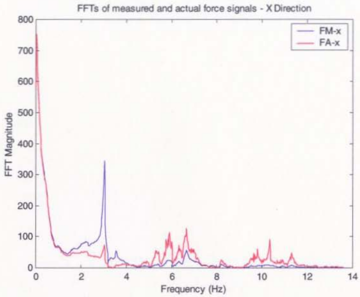

**Figure 5.38 IFT Method - Jacket B FFT of Wave Impact – SDOF – X**

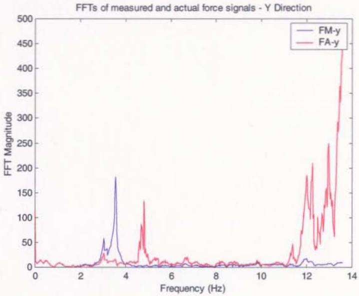

**Figure 5.39 IFT Method - Jacket B FFT of Wave Impact – SDOF – Y**

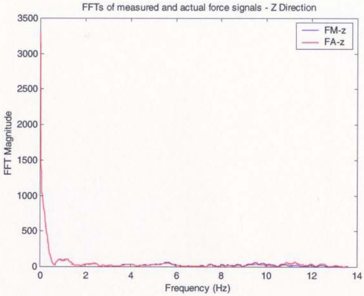

**Figure 5.40 IFT Method - Jacket B FFT of Wave Impact – SDOF – Z**

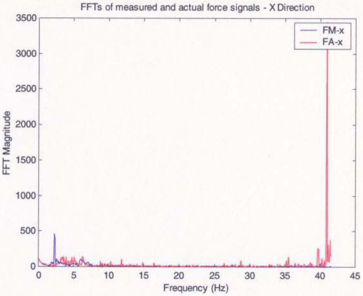

**Figure 5.41 IFT Method - Jacket A FFT of Wave Impact – SDOF – X**

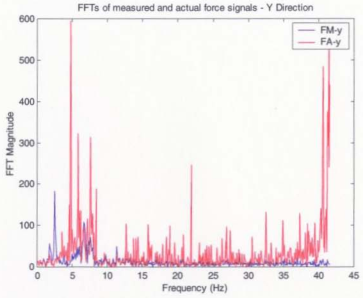

**Figure 5.42 IFT Method - Jacket A FFT of Wave Impact – SDOF – Y**

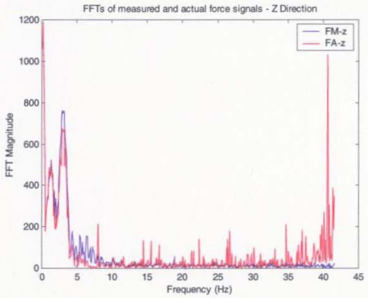

**Figure 5.43 IFT Method - Jacket A FFT of Wave Impact – SDOF – Z**

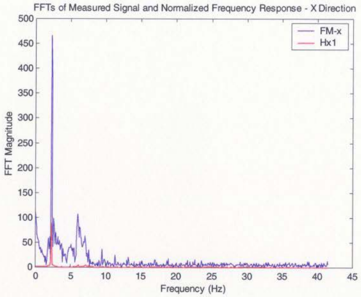

**Figure 5.44 FFT of Measured Signal and Freq. Response – Jacket A**

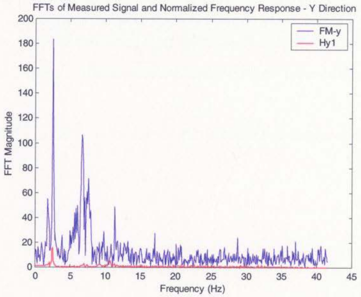

**Figure 5.45 FFT of Measured Signal and Freq. Response – Jacket A**

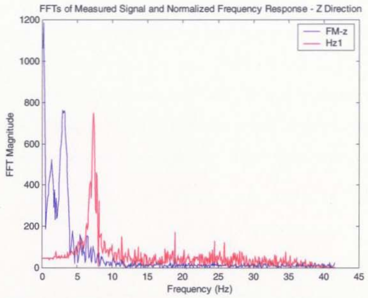

**Figure 5.46 FFT of Measured Signal and Freq. Response – Jacket A**

response. This is an effect of the added mass caused by the impacting wave. Since the peaks no longer coincide, the procedure cannot be properly implemented.

Similar poor results were obtained from the MDOF implementation, so will not be presented here.

## 5.4 Blind Deconvolution

An interesting technique for removing unwanted convolutions from signals is described by Smith (1999). This technique is known as Blind Deconvolution. The impulse response of the system is unknown, and must be estimated. The appropriateness of the method may be dubious, but it does provide some interesting results. A discussion of this method, and its application to measured wave impact signals is provided in Appendix G.

## 5.5 Summary

Based on the application of these techniques to measured decay data, it is clear that they have some merit. In both the single DOF, and multiple DOF implementations the technique did a reasonable job of suppressing the inertial component, when the impulse response vector and the measured force vector were collinear. Otherwise, the results were not impressive.

The technique is best suited for single DOF measurement systems, where crosstalk and contamination from multiple modes of vibration are less an issue. For MDOF systems, further work needs to be done to develop methods to establish the matrix of cross coupling terms. This needs to be tied into studies of overall dynamometer design. There has been little work done to quantify the effects of system dynamics on these important measuring devices.

Further work needs to be done to establish criteria for the selection of pulse (step) responses from measured decay tests. These are differentiated to generate the impulse responses. The sensitivity of the results to the selection has been demonstrated here, but the process is still based on trial and error at this point.

Additional study is required to understand the differences observed between the results obtained using the frequency domain division and the deconvolution in time domain. Why does the impulse response vector require the addition of an impulse to the first point to facilitate the deconvolution? Is this modification even appropriate?

Gibbs effect seems to be an important and unavoidable feature of this technique. Some authors have described methods to mitigate Gibb's effect, and these are discussed briefly in Appendix I. Effort needs to be made to investigate these mitigation techniques and to establish their effect and appropriateness applied to the inverse Fourier transform methods.

The application of the techniques to measured wave impact signals highlights the need for further study in two areas. First, more work needs to be done to understand the effects of spatially and temporally varying added mass on the measured wave impact signals. Second, the appropriateness of filtering to remove the erroneous high frequency components from the actual force signals needs to be examined.

The Blind Deconvolution technique, discussed in Appendix G requires further experimentation with the selection of desired pulse shapes, the application of the Custom Filter method, and consideration of Gibbs effect.

# 6.0 Use of Digital Filters to Remove Inertial Force

## 6.1 Introduction

Murray and Kaplan (1995) described the use of digital filters in post-experiment processing to remove inertial loads from measured force signals. They demonstrated the use of a low-pass Kaiser window to remove high frequency components from measured wave impact loads on the instrumented deck of a jacket-type offshore structure. Their justification for applying a low-pass filter was that the high frequency components measured by the vertical measuring load transducers (five in this case) were not phase locked to the low frequency components. That is, the low frequency component time traces for the individual transducers that comprise the dynamometer were out of phase with each other. The time traces of the high frequency components were in phase. The authors stated that this indicates that the high frequency components are completely inertial in origin, and can be legitimately removed. In the conclusion section of their paper they recommend further investigation using band reject type filters.

An attempt was made to apply this low/high pass separation technique to measured vertical force signals from the experiments described in Section 3.0. The results were not as conclusive as those presented by Murray and Kaplan (1995). The signals were low and high pass filtered with appropriate frequency cutoffs, using a Kaiser window. The low pass signals showed the expected phase relationships, with the forward load transducer signals out of phase with rear load transducers. When high pass filtered though, the signals could not definitively be described as being in phase. Indeed, if regions of the high frequency range are band passed, there are sections that show the same phase relationships as seen in the low pass case. This is not to suggest that the hypothesis put forward by Murray and Kaplan (1995) is invalid, but it does emphasize the need to careful application of low pass filters, and the need for further investigation of the use of band reject filters.

## 6.2 Filter Types

There are, of course, many types of low pass and band reject filters that can be applied to a measured signal. The choice can be overwhelming to someone not versed in the mysteries of digital signal processing (DSP). The selection of an inappropriate filter, or its improper application can lead to incorrect interpretation of results.

Smith (1999) provides an excellent discussion of filter types going a long way to demystify their use. He basically divides filters into two categories, those optimized for use with time-domain encoded signals, and those optimized for use with frequency-domain encoded signals. The former are best at removing noise from signals, while the latter are best at separating bands of frequencies. The time domain filters typically have sharp step response, while having poor frequency response characteristics. The frequency domain filters, on the other hand, have excellent frequency response characteristics, with fast roll-off, and good stopband attenuation, but they can have slow step responses, with significant overshoot.

Which type of filter best applies to the measured wave impact signals being examined here? This question will be examined by implementing a number of the filters recommended by Smith (1999). The algorithms and scripts for each filter will be presented, and their characteristics and limitations when applied to a typical wave impact measurement will be discussed

Smith (1999) also categorizes filters as to whether they are applied in convolution or in recursion. These classifications essentially influence the speed of the applied filter. Since the signals under examination here are post processed, speed is not an issue.

Figures 6.1 and 6.2 show a typical wave impact time trace and its FFT. These will be used for this examination of filters. This signal represents a wave impact on the deck dynamometer measured in the horizontal (X) direction. This will be the common signal to which the filters will be applied. The filters that are to be explored are the moving average filter, the windowed sinc filter, the single pole filter, the Chebychev type I, and the Kaiser window. These will all be implemented as low pass filters, and several will be implemented in band reject form.

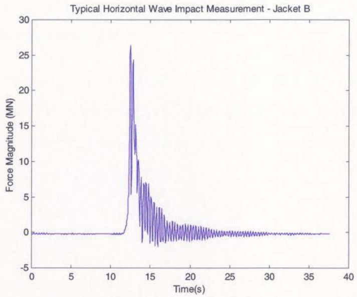

**Figure 6.1 Horizontal Wave Impact – X Direction – Jacket B**

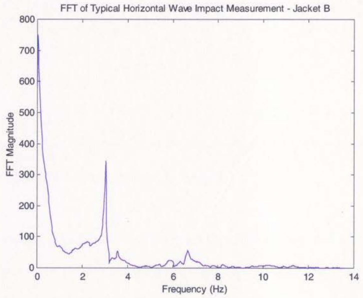

**Figure 6.2 FFT of Horizontal Wave Impact – X Direction – Jacket B**

## 6.3 Low Pass Filters

### 6.3.1 Moving Average Filter

The moving average filter is the simplest of all filters. It is classified as finite impulse response (FIR). It is implemented by averaging sections of the time series using successive windows of arbitrary length. The algorithms in non-recursive and recursive form are given in Equations 6.1 and 6.2.

$$y[i]=\frac{1}{M}\sum_{j=0}^{M-1}x[i+j] \qquad \textbf{6.1}$$

$$y[i]=y[i-1]+x[i+p]-[i-q] \qquad \textbf{6.2}$$

Where, $p=(M-1)/2$ and $q=p+1$. $M$ is the filter length, or the length of the averaging window. $x[i]$ and $y[i]$ are the input and output signals respectively. The Matlab scripts for these are given in Appendix K.

The moving average filter is the best for reducing random noise in a signal. It has a sharp step response, with little or no overshoot. Its frequency response characteristics are very poor, with extremely slow roll-off, and very poor stopband attenuation. Figures 6.3 and 6.4 show the frequency and step responses for a range of filter lengths.

Increasing the filter length has the effect of improving the roll-off and stopband attenuation of the frequency response, while lengthening the rise-time of the step response.

The frequency responses are sinc functions. This means that the impulse responses are rectangular pulses, where the width of the pulse is the same as the filter length. The same filtered output could be obtained by convolving the input signal with the rectangular pulse.

Other than altering the filter length, the user of the moving average filter has no control over location of the cutoff frequency. For the example input signal, a filter length of 9 points causes the first zero crossing of the frequency response to coincide with the resonant peak, resulting in the best case for this filter type, see Figures 6.5 and 6.6. A filter length of 19 points causes the second zero crossing to coincide with the resonant peak, resulting in the second best case.

The 9 point moving average filter, for this wave input example, does a good job of removing the resonant peak without corrupting the lower frequency (wave induced) components. The higher frequency components are not reduced to zero, but are reduced

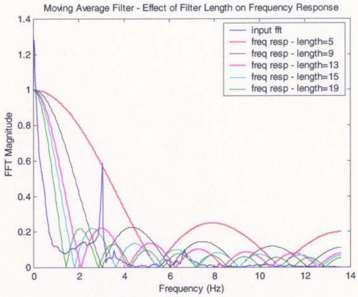

**Figure 6.3 Effect of Filter Length on Freq. Response – Moving Avg.**

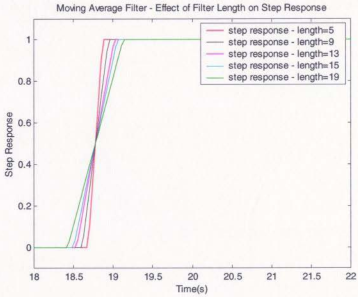

**Figure 6.4 Effect of Filter Length on Step Response – Moving Avg.**

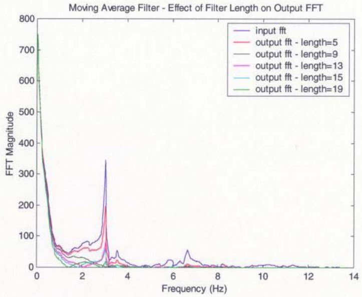

**Figure 6.5 Effect of Filter Length on Output FFT – Moving Avg.**

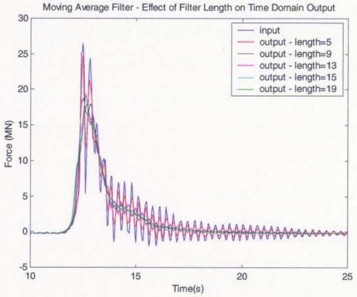

**Figure 6.6 Effect of Filter Length on Time Series – Moving Avg.**

to less than 20 percent of their original values. Common wisdom would suggest that this type of filter is not appropriate for this type of signal. However, based on these interesting observations, and the ease of implementation, use of the moving average filter should at least be explored.

Blackman and Gaussian windows, applied in convolution, provide step responses that are almost as good as the moving average filter, while producing better stopband attenuation in the frequency response.

### 6.3.2 Windowed Sinc Filter

The windowed sinc filter is also classified as FIR, since it is implemented in convolution. It is known for its excellent frequency response characteristics: fast roll-off, low passband ripple and good stopband attenuation. The step response shows short rise-time, but an overshoot in the range of 9 to 10 percent. Increasing the filter length improves the roll-off in the frequency response, but has little effect on the step response rise-time or overshoot. Figures 6.7 and 6.8 illustrate the effect of filter length on frequency and step responses.

The filter kernel or impulse response for the windowed sinc filter Smith (1999) is given in Equation 6.3.

$$h[i] = K\frac{\sin\left(2\pi f_c\left(i - M/2\right)\right)}{i - M/2}\left[0.42 - 0.5\cos\left(\frac{2\pi i}{M}\right) + 0.08\cos\left(\frac{4\pi i}{M}\right)\right] \qquad \textbf{6.3}$$

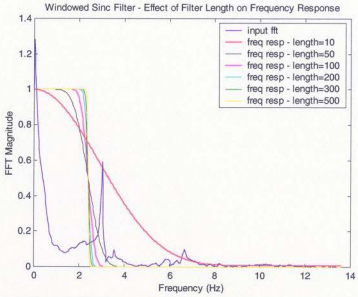

**Figure 6.7 Effect of Filter Length on Freq. Response – Windowed Sinc**

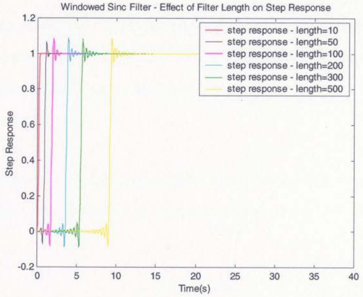

**Figure 6.8 Effect of Filter Length on Step Response – Windowed Sinc**

$M \approx 4/BW$ where $BW$ is the transition bandwidth between the passband and stopband. $M$ is required to be even. $f_c$ is the cutoff frequency, and $K$ is a normalization constant to set unity gain at DC. The cutoff frequency used for this and all examples was 0.088 where the Nyquist frequency was 0.5. The filter kernel is convolved directly with the input signal to produce the filtered output signal. Figures 6.9 and 6.10 show the effects of filter length on the output signal and FFTs. Beyond $M$ equal to 50, there is little visible difference between the output signals and their FFTs. The Matlab script implementing this algorithm is shown in Appendix K

As a result of their excellent frequency response, windowed sinc filters are best used to separate bands of frequencies.

The impulse response of an ideal filter is a sinc function. An ideal filter cannot be implemented on computer. The windowed sinc filter is the result of truncating, shifting and windowing (using a Blackman window) the ideal impulse response. This filter kernel is shown in Figure 6.11.

Stopband attenuation can be improved by using the filter in multiple stages. Either by applying the filter successively or by convolving the kernel with itself.

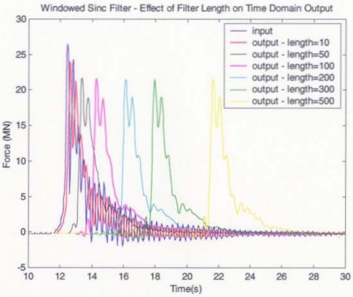

**Figure 6.9 Effect of Filter Length on Time Series – Windowed Sinc**

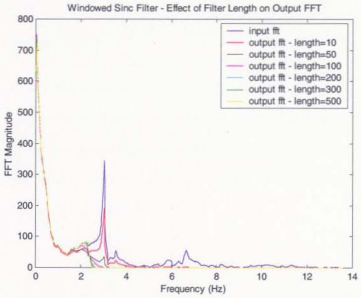

**Figure 6.10 Effect of Filter Length on Output FFT – Windowed Sinc**

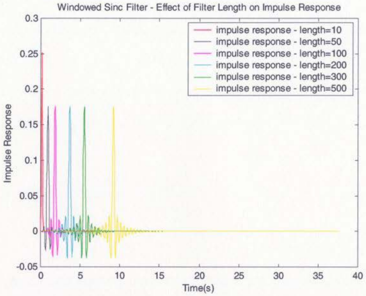

**Figure 6.11 Effect of Filter Length on Imp. Response – Windowed Sinc**

### 6.3.3 Single Pole Filter

The single pole filter is classified as a recursive, time domain filter. It has very good step response characteristics, with quick rise-time, and no overshoot. However, it has terrible frequency response characteristics, with extremely slow roll-off and poor stopband attenuation. Figure 6.12 and 6.13 show the frequency and step responses for non-cascaded and cascaded versions.

Recursive filters are referred to as infinite impulse response (IIR), since their impulse responses are composed of decaying exponentials. They use previously calculated output values, along with the present input value to determine subsequent output values.

The recursive algorithm given in Equation 6.4 gives the output signal for the non-cascaded single pole low pass.

$$y[i] = a_0\, x[i] + b1\, y[i-1] \qquad \textbf{6.4}$$

where $a_0 = 1 - z$, $b_1 = z$, $z = e^{-2\pi f_c}$, $f_c$ is the cutoff frequency (0.088 in this case). In four stage cascaded form, the recursion equation would be as shown in Equation 6.5.

$$y[i] = a_0 x[i] + b_1 y[i-1] + b_2[i-2] + b_3[i-3] + b_4[i-4] \qquad \textbf{6.5}$$

where $a_0 = (1-z)^4$, $b_1 = 4z$, $b_2 = -6z^2$, $b_3 = 4z^3$, $b_4 = -z^4$, $z = e^{-14.45 f_c}$, $x$ is the input signal, and $y$ is the output signal. $z$ is the amount of decay between adjacent samples. Figures 6.14 shows the filtered output signal. Clearly in this form, the algorithms have done little to remove inertial loads.

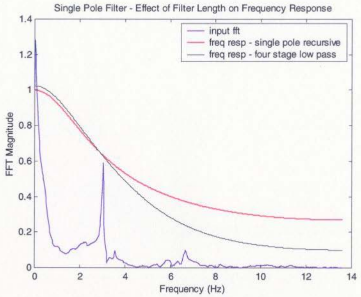

**Figure 6.12 Effect of Filter Length on Freq. Response – Single Pole**

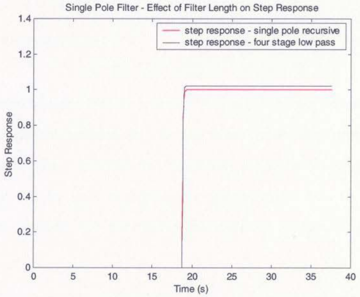

**Figure 6.13 Effect of Filter Length on Step Response – Single Pole**

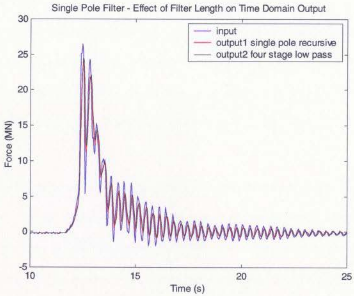

**Figure 6.14 Effect of Filter Length on Time Series – Single Pole**

### 6.3.4 Chebychev Type 1

The Chebychev type 1 filter is a recursive. The type 1 designation indicates that ripple is allowed only in the passband. This filter is optimized for use in the frequency domain, in that it is designed to separate bands of frequencies. It is not considered to be as good as the windowed sinc filter. The purpose of the ripple in the passband is to improve the speed of the roll-off. With zero percent ripple, it is known as a Butterworth filter.

The parameters used to design a Chebychev filter include cutoff frequency, percent ripple in the passband, and number of poles. Generally, the more poles used, the better. The recursion coefficients can be selected from published tables, based on the cutoff frequency and number of poles or efficiently using computer programs. Such a program is provided by Smith (1999) and is implemented in a Matlab script in Appendix K. Otherwise the z-transform would be utilized to determine the coefficients.

The larger the number of poles, the better the frequency response. The step response is quite sharp, but the overshoot can reportedly be in the range of 5 to 30 percent. Beyond 8 to 10 poles, there is little difference in the shape of the frequency response, see Figure 6.16. Figure 6.16 shows the frequency response using 10 poles and a variation from 0 to 5 in the percent ripple. Figure 6.17 shows an overshoot of more than 20 percent, with number of poles equal to 10, and a percent ripple of 5.

The effect of the number of poles on the output time series is shown in Figure 6.18.

### 6.3.5 Kaiser Window

Murray and Kaplan (1995) employed a non-recursive Kaiser window in the filtering of their wave impact measurements. Matlab conveniently provides scripts to design such filters. The Kaiser window is an FIR filter, which produces an impulse response that can be applied in convolution.

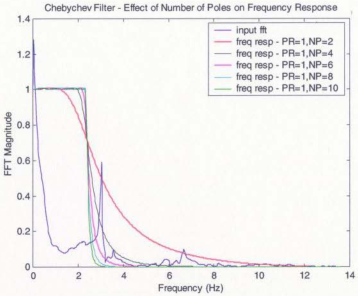

**Figure 6.15 Effect of Number of Poles on Freq. Response – Chebychev**

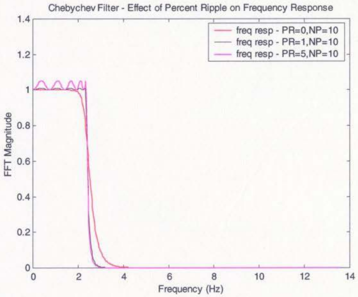

**Figure 6.16 Effect of Percent Ripple on Freq. Response – Chebychev**

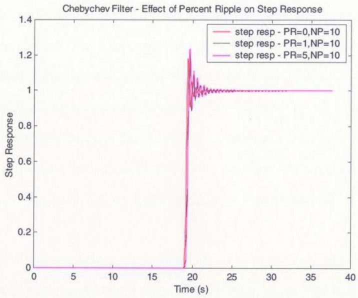

**Figure 6.17 Effect of Percent Ripple on Step Response – Chebychev**

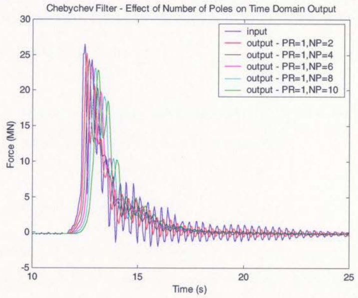

**Figure 6.18 Effect of Number of Poles on Time Series – Chebychev**

The Matlab script "Kaiserord.m" requires as input the frequency vector defining the transition band (the section between the passband and stopband). The amplitude of this frequency vector is required. Generally a value of 1 is used for the passband, and 0 for the stopband. The allowable amplitude deviation in the passband and stopband is required. This is the same as percent ripple in other filter designs. Finally, the sample rate of the input signal is needed. This can be normalized so that the Nyquist frequency is 0.5. The script produces the order and cutoff frequency required for use in scripts "Kaiser.m" or "fir1.m". The command procedure for this is given in Appendix K.

Figure 6.19 shows the effect percent ripple has on the frequency response. The roll-off is almost identical. The transition band has been set very narrow. This allows for a clean separation of frequencies, as shown in Figure 6.20. The step responses are also very similar, each showing a reasonably sharp response with approximately 10 percent overshoot. See Figure 6.21.

Given the similarity of the frequency and step responses, one would expect the output time series to be similar. This is demonstrated in Figure 6.22.

### 6.3.6 Comparison of Low Pass Filters

One of the goals of this exercise is to determine which filter or type of filter is the best to apply to a measured wave impact, in order to remove inertial forces. Will a filter

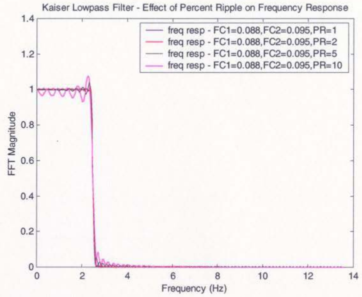

**Figure 6.19 Effect of Percent Ripple on Freq. Response – Kaiser LP**

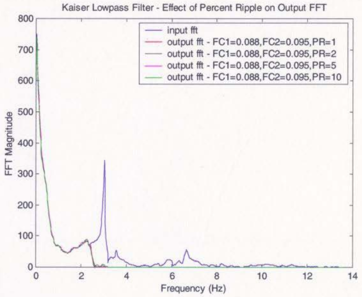

**Figure 6.20 Effect of Percent Ripple on Output FFT – Kaiser LP**

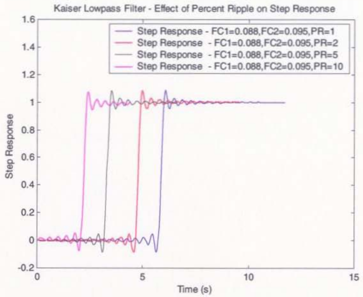

**Figure 6.21 Effect of Percent Ripple on Step Response – Kaiser LP**

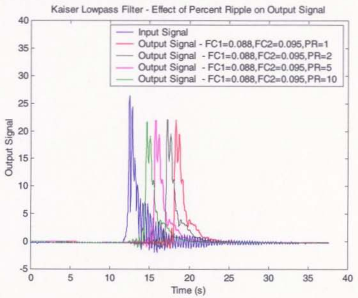

**Figure 6.22 Effect of Percent Ripple on Time Series – Kaiser LP**

optimized for the time domain provide better results than one optimized for the frequency domain? Is there a certain filter from either category that stands out? The results from the filters just discussed will be compared to shed some light on these questions.

Table 6.1 summarizes the input parameters for each filter design. An attempt was made to keep the cutoff frequencies the same.

Table 6.1 – Low Pass Filter Input Parameters

| Filter Name | Filter Type | Normalized Frequency Cutoffs | Other Input Parameters | Notes |
|---|---|---|---|---|
| Moving Average | Time Domain, Convolution | N/A | Filter Length=9 | Symmetrical Averaging |
| Windowed Sinc | Frequency Domain, Convolution | 0.088 | Filter Length=300 | |
| Single Pole | Time Domain, Recursive | 0.088 | | Single Stage |
| Chebychev | Frequency Domain, Recursive | 0.088 | Percent Ripple =1<br><br>Number of Poles = 10 | Type 1 |
| Kaiser Window | Non-recursive | 0.088 to 0.095 | Sample Rate = 1<br><br>Passband Amplitude=1<br><br>Stopband Amplitude =0<br><br>Percent Ripple=1 | Transition frequencies defined<br><br>Percent ripple the same in passband and stopband |

Figure 6.23 compares the frequency response. Figure 6.24 compares the step response. Figures 6.25 and 6.26 compare output signal time series and their FFTs, respectively.

A moving average filter of length 9, based on symmetrical averaging, resulted in a frequency response with a zero crossing located at the resonant peak of the input signal. This focussed signal removal resembles a band reject filter in some aspects. Despite its slow roll-off, and poor stopband attenuation, the signal removal is reasonable. From the output signal FFT it is seen that the low frequency wave components have not been dramatically altered, and the energy in the stopband region is not significant. The step response is very sharp, and has no overshoot.

The single pole filter was implemented as a single stage. The roll-off and stopband attenuation are atrocious. The step response is excellent, with extremely sharp rise time, and no overshoot. The single pole filter, however, removes almost none of the inertial component.

The frequency response for the other filters is extremely good, with little to choose between them. The roll-offs are quick, and the attenuation in the stopband is good. The Chebychev displays the worst passband ripple.

For clarity, Figure 6.27 to 6.30 compare the frequency response, step response, output signals and FFTs for the three filters deemed to be the best. The windowed sinc and

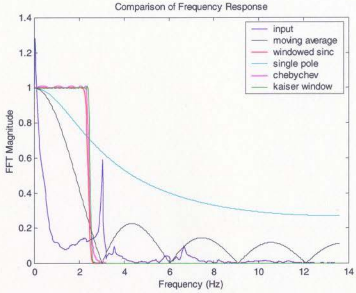

**Figure 6.23 Comparison of Frequency Response for Low Pass Filters**

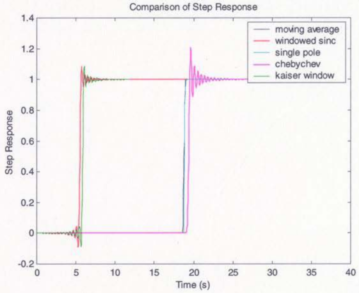

**Figure 6.24 Comparison of Step Response for Low Pass Filters**

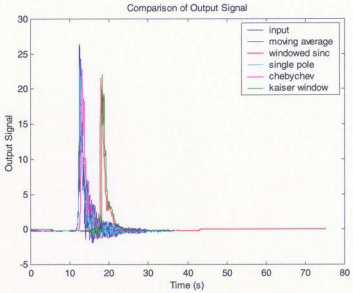

**Figure 6.25 Comparison of Time Series for Low Pass Filters**

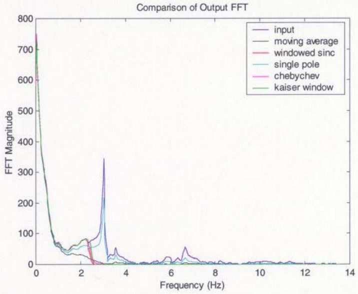

**Figure 6.26 Comparison of Output FFTs for Low Pass Filters**

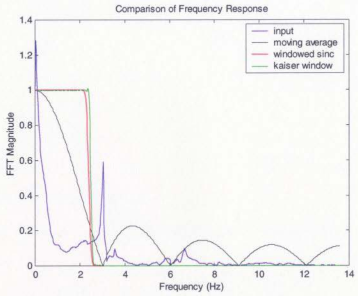

**Figure 6.27 Frequency Response for Selected Low Pass Filters**

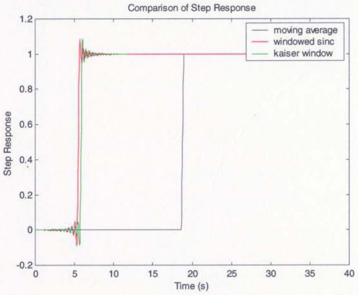

**Figure 6.28 Step Response for Selected Low Pass Filters**

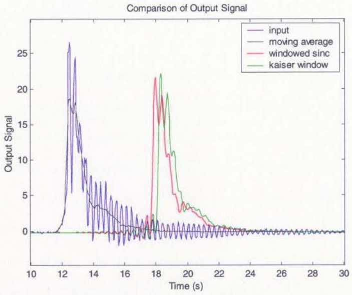

**Figure 6.29 Time Series for Selected Low Pass Filters**

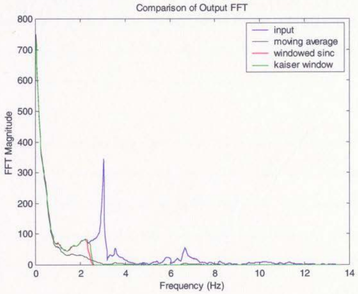

**Figure 6.30 Output FFTs for Selected Low Pass Filters**

Kaiser window filters have comparable frequency and step responses and result in nearly identical output signals.

If only frequency domain filters are considered, the choice of filters would likely be based on ease of implementation or other personal preference. The choice between time domain and frequency domain filters is not so straightforward. The moving average filter, by most standards, would be labeled as having useless frequency response characteristics. It is really only luck that the input signal resonant peak is narrow enough to be eliminated by the first zero crossing region of the frequency response. However, a narrow resonant peak is typical of wave impact signals, so the moving average filter should always be considered as a filtering tool for these types of signals. Due to the nearly ideal step response the output signal is not corrupted by overshoot. So what you see, is what you get. The frequency domain filters, on the other hand, can separate close proximity frequency bands. However, due to the overshoot in the step response the output signal will be corrupted, particularly in the impact region.

So which type of filter should be used? There is no clear or easy answer. A reasonable suggestion would be to implement both types when examining wave impact signals. Application of moving average and windowed sinc filters would produce output signals that help define a range of peak load. From Figure 6.29, the moving average and windowed sinc filters would estbalish a peak load range from 19 to 22 (MN). For engineering purposes this would be quite reasonable. It would be foolish to expect better results given the inherent uncertainties of all filters.

## 6.4 Band Reject Filters

If one is uncomfortable with the assertion by Murray and Kaplan (1995) that the high frequency components in a wave impact signal are entirely inertial, then the use of band reject filters should be considered.

A band reject filter acts like a combination low pass and high pass filter, maintaining low and high frequency ranges while eliminating the transition zone that separates them. As with the low pass filters, care must be taken in their design. Three types will be examined and compared.

### 6.4.1 Recursive Algorithm

Smith (1999) provides algorithms for the recursion coefficients of a band reject filter. These are shown in Equations 6.6 and 6.7.

$$\begin{aligned}
a_0 &= K \\
a_1 &= -2K\cos(2\pi f) \\
a_2 &= K \\
b_1 &= 2R\cos(2\pi f) \\
b_2 &= -R^2
\end{aligned}$$

**6.6**

where

$$K = \frac{1-2R\cos(2\pi f)+R^2}{2-2\cos(2\pi f)}$$
$$R = 1-3BW$$

$BW$ and $f$ are the bandwidth and centre frequency of the transition region, respectively.

The recursion equation is as follows.

$$y(i) = a_0\, x(i) + a_1\, x(i-1) + a_2\, x(i-2) + b_1\, y(i-1) + b_2\, y(i-2) \qquad \textbf{6.7}$$

The centre frequency is selected to be 0.1113. This is the location of the resonant peak of the input signal under examination. Recall that the frequency scale has been normalized so that the Nyquist frequency is 0.5. A range of transition bandwidths, from 0.01 to 0.1 was examined. Figure 6.31 compares the frequency responses. This filter produces a V-shaped frequency response. This allows for a very focussed rejection band selection. The speed of the roll-off increases with decreasing transition bandwidth. The level of corruption of the high frequency band increases with increasing bandwidth. For example, with a bandwidth of 0.1, the amplification of the high frequency components is almost 20 percent. The effect of this on the output signal and its FFT is negligible, as shown in Figures 6.32 and 6.33. The transition bandwidth of 0.0467 establishes the lower limit of the transition zone at 0.088, which is comparable with the cutoff values of the low pass filters shown previously.

Figure 6.34 shows that the level of step response overshoot is in the range of 10 to 15 percent.

### 6.4.2 Windowed Sinc Band Reject

Applying a spectral inversion technique to a low pass windowed sinc filter generates the windowed sinc band reject filter. Spectral inversion involves modifying the filter kernel by changing the sign of all samples, and then adding 1.0 to the point of symmetry. This essentially flips the frequency response of the low pass filter vertically, making it a high pass filter. The procedure is to establish the low pass filter, with a frequency cutoff at the lower limit of the desired transition zone. Similarly, develop a low pass filter with a frequency cutoff at the upper limit of the desired transition zone. The second low pass filter is then spectrally inverted to obtain a high pass filter, with its cutoff at the upper limit of the desired transition zone. The two filter kernels are then added. The frequency

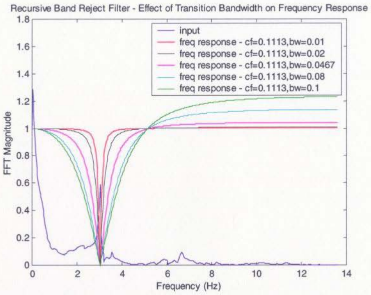

**Figure 6.31 Effect of Bandwidth on Freq. Response - Recursive BR**

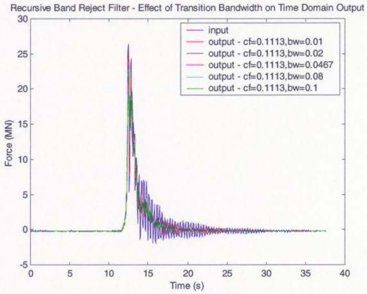

**Figure 6.32 Effect of Bandwidth on Time Series - Recursive BR**

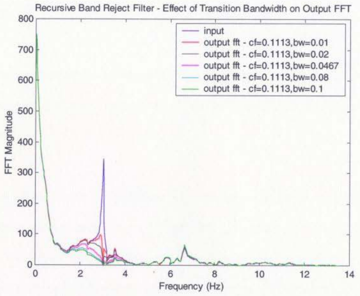

**Figure 6.33 Effect of Bandwidth on Output FFT - Recursive BR**

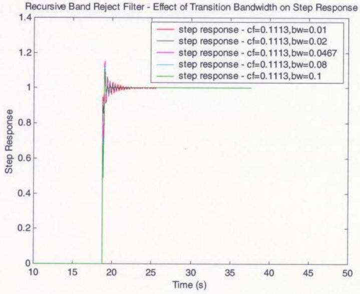

**Figure 6.34 Effect of Bandwidth on Step Response - Recursive BR**

response for the new filter kernel will be a band reject. If one wished to obtain a band pass filter, the two filter kernels would have to be convolved.

There are two limitations to the spectral inversion method. First the filter kernels must have left-right symmetry. Second, the unit impulse must be added at the centre of symmetry.

Filters developed using this technique were applied to the sample input signal. The low and high frequency cutoffs were 0.088 and 0.1346, respectively. The filter length was varied from 100 to 500. The effect of this variation of frequency response is shown in Figure 6.35. The roll-offs are quite sharp, increasing with filter length, and there is no amplification in either the low or high frequency passbands. As seen in Figure 6.36, the frequency content in the reject zone is essentially reduced to zero. This could be considered a problem since, in theory, there could be some wave induced (non-inertial) energy in that region.

The step responses show similar levels of overshoot, approximately 10 percent, see Figure 6.37. Figure 6.38 shows the filtered output signals which, to the eye, appear quite similar.

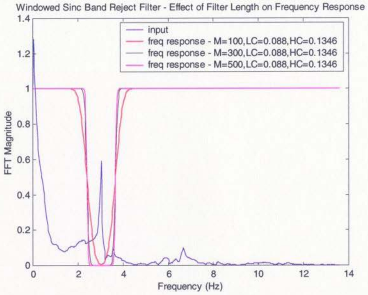

**Figure 6.35 Effect of Filter Length on Freq. Response – Win. Sinc BR**

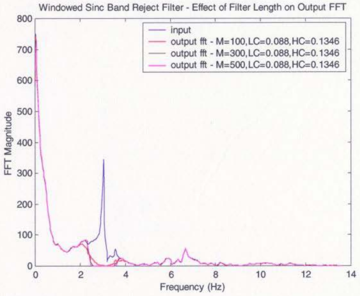

**Figure 6.36 Effect of Filter Length on Output FFT – Win. Sinc BR**

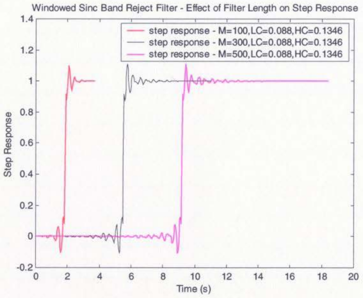

**Figure 6.37 Effect of Filter Length on Step Response – Win. Sinc BR**

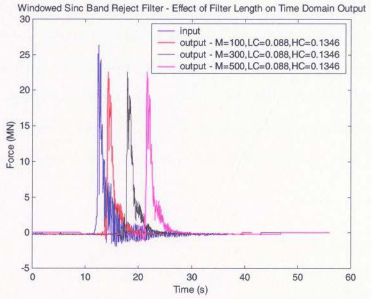

**Figure 6.38 Effect of Filter Length on Time Series – Win. Sinc BR**

### 6.4.3 Kaiser Band Reject

The scripts described in Section 6.3.5 to develop Kaiser low pass filters can used to generate band reject filters. Instead of a two point vector defining the transition between the low frequency passband and stopband, a four point vector defining the transitions between the the reject region and the low and high passbands is used. For this example, this vector of normalized frequencies would be [0.088 0.104 0.1186 0.1346]. This vector defines a rejection zone that centres on the resonant peak of the input signal. The corresponding amplitude vector would be [1 0 1], and a similar three point vector would define the allowable deviation in each zone. For this demonstration, the allowable percent ripple is varied from 1 to 6 percent. Figure 6.39 illustrates the effect of percent ripple on the frequency response. The roll-off for each is quite fast, and seemingly unaffected by the percent ripple. The frequency response using 6 percent ripple shows the largest deviation in the low and high passbands, as expected. But it also shows an amplification in the stopband, which is unexpected and unwanted. The effect of this is evident in the output signal FFT shown in Figure 6.40. The output time-series, however, are essentially identical, as seen in Figure 6.41. The step response are also identical, each showing an overshoot of approximately 10 percent, as seen in Figure 6.42.

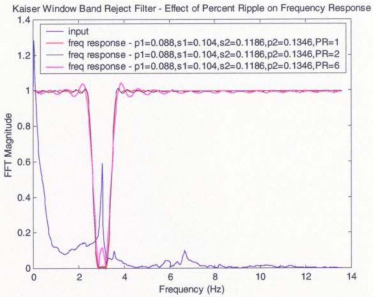

**Figure 6.39 Effect of Percent Ripple on Freq. Response – Kaiser BR**

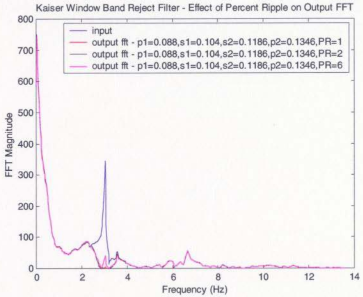

**Figure 6.40 Effect of Percent Ripple on Output FFT – Kaiser BR**

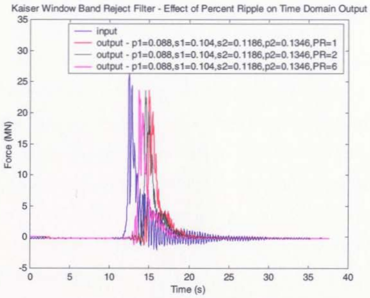

**Figure 6.41 Effect of Percent Ripple on Time Series – Kaiser BR**

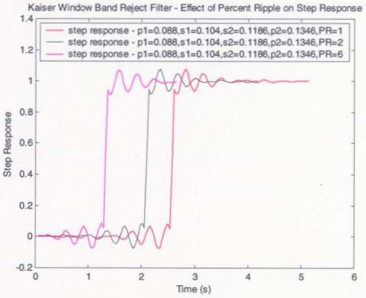

**Figure 6.42 Effect of Percent Ripple on Step Response – Kaiser BR**

### 6.4.4 Comparison of Band Reject Filters

From the filters discussed above, one of each type was designated to be the best. These were the recursive with a bandwidth of 0.0467, the windowed sinc with a filter length of 100, and the Kaiser window with a percent ripple of 1. The frequency response of each is compared in Figure 6.43. The Kaiser window displays the fastest roll-off, but shows the greatest deviation in the passbands. The windowed sinc shows no passband ripple, but has a slightly slower roll-off. The recursive filter has the slowest roll-off, but the reject zone is very focussed, a feature that is advantageous when dealing with a narrow banded resonant peak. The recursive filter does show an amplification in the order of 0.5 percent in the high frequency passband.

The step responses are quite similar, each showing sharp rise time, with overshoot in the 10 to 15 percent range, as seen in Figure 6.44. The output time series are very similar, as seen in Figure 6.45. The peak values range from approximately 22.5 to 24, with the recursive filter producing the highest, and the windowed sinc the lowest. The output signal FFTs are shown in Figure 6.46. An interesting result from the recursive filter is that the energy is never reduced to zero in the stopband region (well perhaps at one point). This "simulates" the retention of the wave induced component in this region.

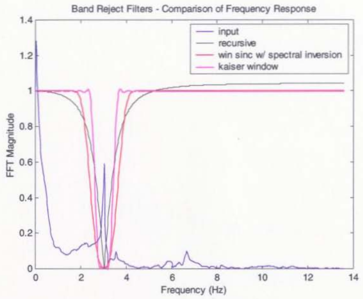

**Figure 6.43 Comparison of Freq. Response – Band Reject Filters**

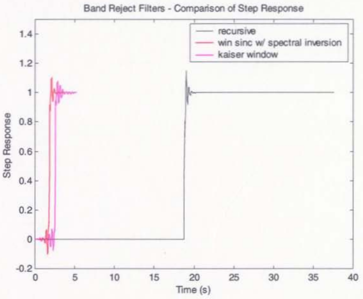

**Figure 6.44 Comparison of Step Response – Band Reject Filters**

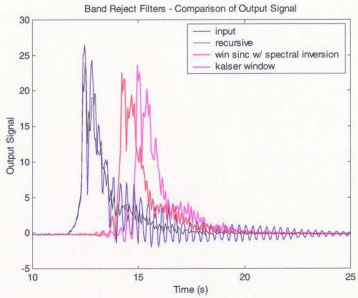

**Figure 6.45 Comparison of Time Series – Band Reject Filters**

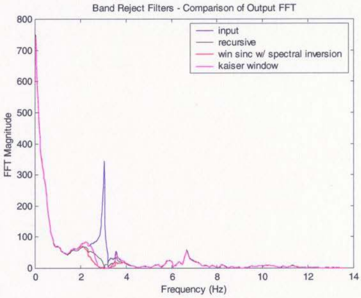

**Figure 6.46 Comparison of Output FFT – Band Reject Filters**

As with the low pass filters, the selection of filter from this group will depend on ease of implementation and unique features of the input signal, such as width of the resonant peak.

## 6.5 Comparison of Low Pass and Band Reject Filters

Figure 6.47 to 6.50 compare the frequency response, step response, output time series and output FFTs for the low pass and band reject filters discussed above. Further work needs to be done to test the hypothesis put forward by Murray and Kaplan (1995) regarding the phase relationships of low and high passed signals measured from dynamometers. The appropriateness of low pass filters versus band reject filters needs to be established.

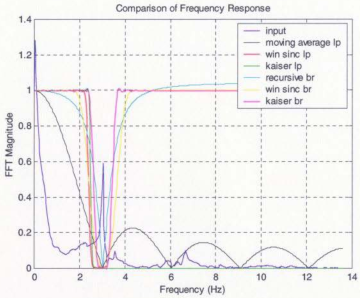

**Figure 6.47 Comparison of Freq. Response – Low Pass & Band Reject**

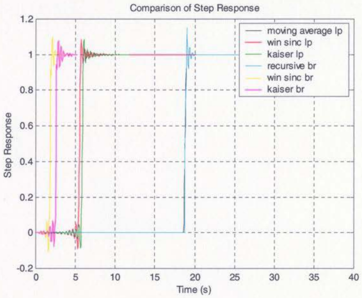

**Figure 6.48 Comparison of Step Response – Low Pass & Band Reject**

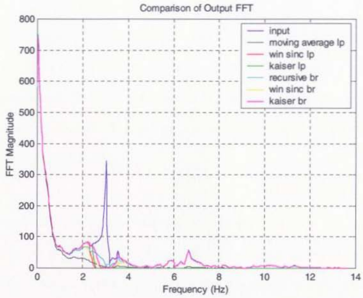

**Figure 6.49 Comparison of Output FFT – Low Pass & Band Reject**

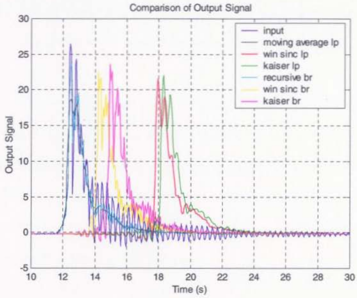

**Figure 6.50 Comparison of Time Series – Low Pass & Band Reject**

## 7.0 Comparison of Methods

For completeness, this section compares results obtained using the methods discussed above. This includes the normal mode approach, the inverse Fourier transform implemented in SDOF format, and selected filters. The methods were applied to three wave impact events, from Jacket B experiments in the X and Z directions, and Jacket A experiments in the Z direction.

The normal mode approach was applied using script *Normal_mode_v3_3dof_nodamping.m*. The inverse Fourier transform method was implemented using script *Ift_wave_sdof.m*. The selected filters were the low pass Moving Average, the low pass Windowed Sinc, the band reject using the recursive algorithm, and the band reject Windowed Sinc. These are implemented in scripts *Mov_avg_v2.m*, *Win_sinc_v2.m*, *Band_reject_recur_v2.m*, and *Band_reject_win_sinc_v2.m*, respectively.

Tables 7.1 to 7.4 outline the input parameters used in the program implementation. Figures 7.1 to 7.21 compare the results. These illustrate, once again, the deficiencies of the normal mode and inverse Fourier transform methods. The output signals from these methods display considerable high frequency contamination and little reduction of inertial component. These problems are related to added mass estimation and the inherent shift of structural natural frequency in water.

Table 7.1 – Mass Matrix for Jacket Structure Model (A)
(kg full scale)

| | | |
|---|---|---|
| 3.19e6 | 0.0 | 0.0 |
| 0.0 | 3.19e6 | 0.0 |
| 0.0 | 0.0 | 3.19e6 |

Table 7.2 – Matrix of Measured Decay Force Vectors
Jacket A

| Measurement DOF | Excitation Mode 1 (X) | Excitation Mode 2 (Y) | Excitation Mode 3 (Z) |
|---|---|---|---|
| X | 9.020 | 0.275 | -0.180 |
| Y | 0.443 | 3.803 | -0.298 |
| Z | -0.036 | -0.059 | 0.940 |

Table 7.3 – Vector of Natural Frequencies – Jacket A
(radians/s)

| X | Y | Z |
|---|---|---|
| 14.32 | 15.85 | 45.51 |

**Table 7.4 – Filter Input Parameters**

| | Input Parameters |
|---|---|
| Filter Type | Jacket B – X Direction |
| Low Pass Moving Average | Filter Length: 9 |
| Low Pass Windowed Sinc | Filter Length: 300<br>Cutoff Frequency: 0.088 (Nyquist = 0.5) |
| Band Reject – Recursive | Centre Frequency: 0.111 (Nyquist = 0.5)<br>Transition Bandwidth: 0.047 |
| Band Reject – Windowed Sinc | Filter Length: 100<br>Low Freq. Cutoff: 0.088 (Nyquist = 0.5)<br>High Freq. Cutoff: 0.135 |
| Filter Type | Jacket B – Z Direction |
| Low Pass Moving Average | Filter Length: 9 |
| Low Pass Windowed Sinc | Filter Length: 300<br>Cutoff Frequency: 0.331 (Nyquist = 0.5) |
| Band Reject – Recursive | Centre Frequency: 0.357 (Nyquist = 0.5)<br>Transition Bandwidth: 0.056 |
| Band Reject – Windowed Sinc | Filter Length: 100<br>Low Freq. Cutoff: 0.331 (Nyquist = 0.5)<br>High Freq. Cutoff: 0.3858 |
| Filter Type | Jacket A – Z Direction |
| Low Pass Moving Average | Filter Length: 27 |
| Low Pass Windowed Sinc | Filter Length: 300<br>Cutoff Frequency: 0.024 (Nyquist = 0.5) |
| Band Reject – Recursive | Centre Frequency: 0.360 (Nyquist = 0.5)<br>Transition Bandwidth: 0.058 |
| Band Reject – Windowed Sinc | Filter Length: 100<br>Low Freq. Cutoff: 0.007 (Nyquist = 0.5)<br>High Freq. Cutoff: 0.048 |

The results obtained from the filters are much cleaner. The Windowed Sinc filters cause a shift in the location of the peak. This is a result of the convolution process utilized in that filter. The choice of filter would likely be based on personal preference, and ease of implementation. The main problem with the use of filters is the uncertainty regarding the legitimacy of the frequency components being removed. There is always the fear that too much or not enough of the offending signal has been removed.

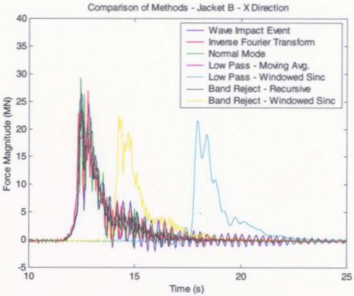

**Figure 7.1 Results – All Methods – Jacket B – X Dir.**

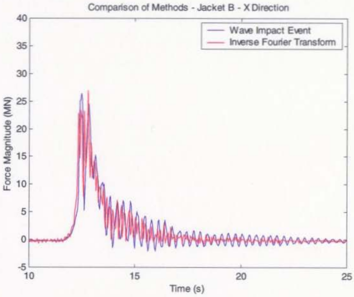

**Figure 7.2 Results – Inverse Fourier Transform – Jacket B – X Dir.**

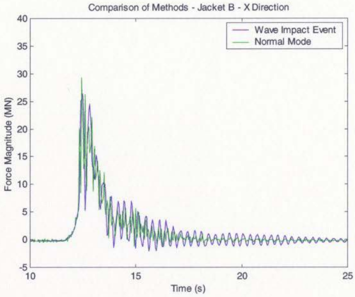

**Figure 7.3 Results – Normal Mode – Jacket B – X Dir.**

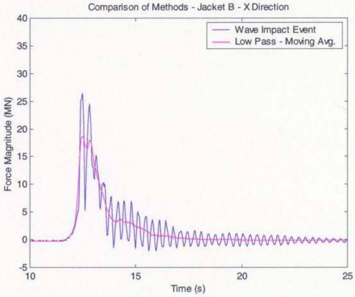

**Figure 7.4 Results – Low Pass Moving Average – Jacket B – X Dir.**

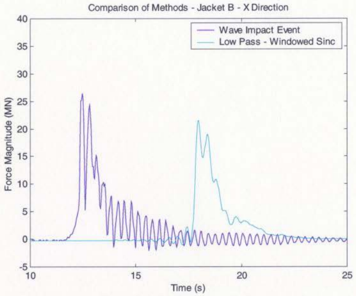

**Figure 7.5 Results – Low Pass Windowed Sinc – Jacket B – X Dir.**

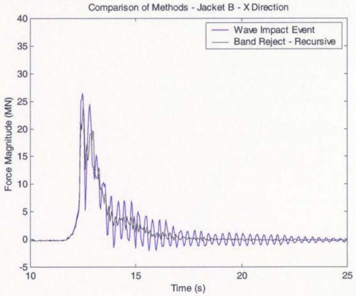

**Figure 7.6 Results – Band Reject Recursive – Jacket B – X Dir.**

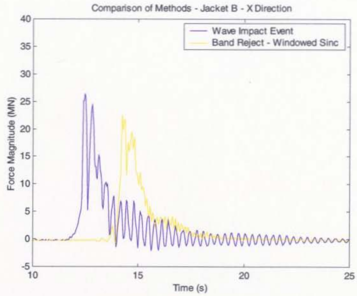

**Figure 7.7 Results – Band Reject Win. Sinc – Jacket B – X Dir.**

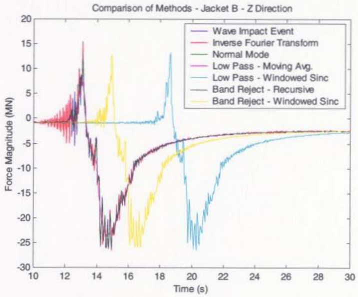

**Figure 7.8 Results – All Methods – Jacket B – Z Dir.**

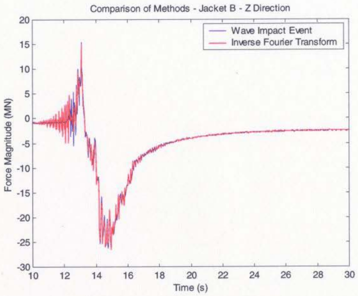

**Figure 7.9 Results – Inverse Fourier Transform – Jacket B – Z Dir.**

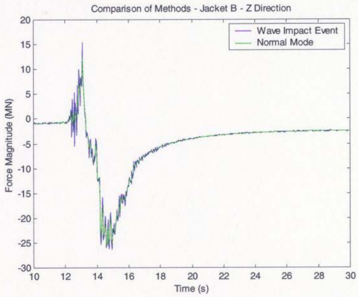

**Figure 7.10 Results – Normal Mode – Jacket B – Z Dir.**

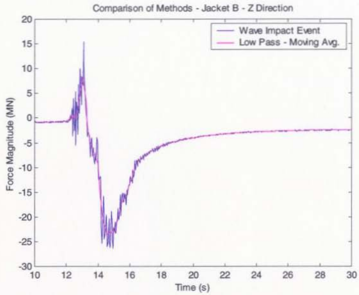

**Figure 7.11 Results – Low Pass Moving Average – Jacket B – Z Dir.**

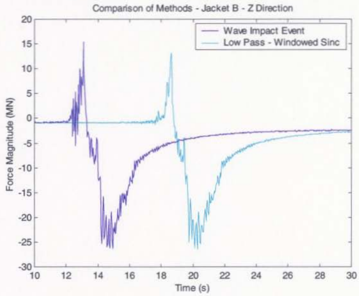

**Figure 7.12 Results – Low Pass Windowed Sinc – Jacket B – Z Dir.**

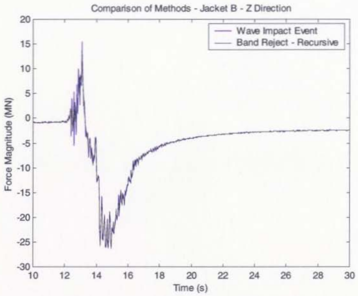

**Figure 7.13 Results – Band Reject Recursive – Jacket B – Z Dir.**

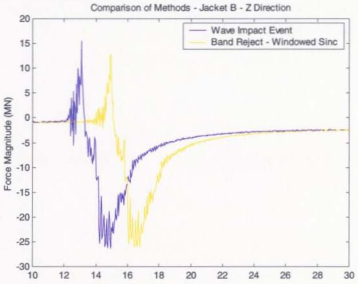

**Figure 7.14 Results – Band Reject Win. Sinc – Jacket B – Z Dir.**

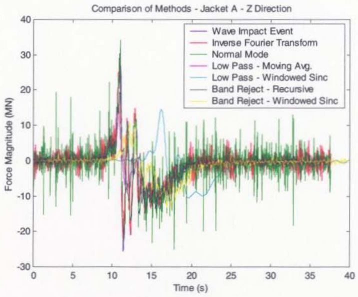

**Figure 7.15 Results – All Methods – Jacket A – Z Dir.**

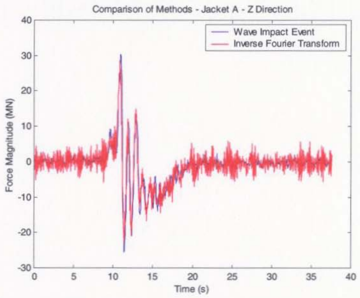

**Figure 7.16 Results – Inv. Fourier Transform – Jacket A – Z Dir.**

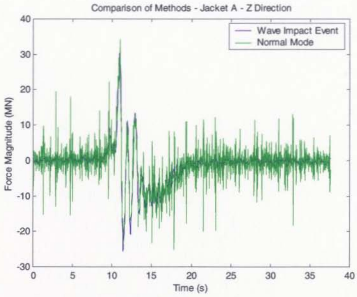

**Figure 7.17 Results – Normal Mode – Jacket A – Z Dir.**

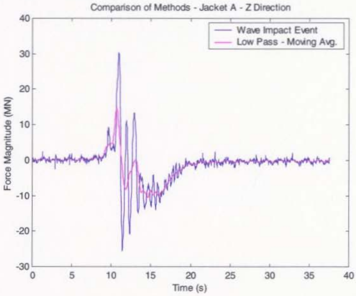

**Figure 7.18 Results – Low Pass Moving Average – Jacket A – Z Dir.**

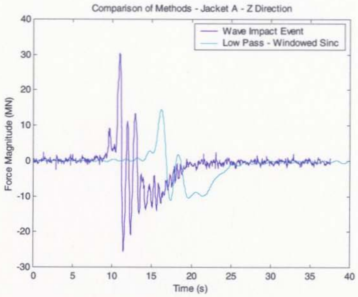

**Figure 7.19 Results – Low Pass Windowed Sinc – Jacket A – Z Dir.**

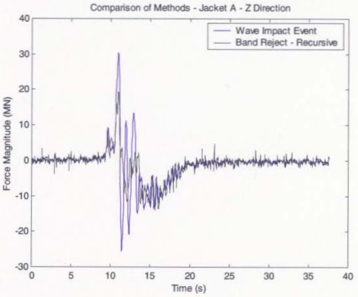

**Figure 7.20 Results – Band Reject Recursive – Jacket A – Z Dir.**

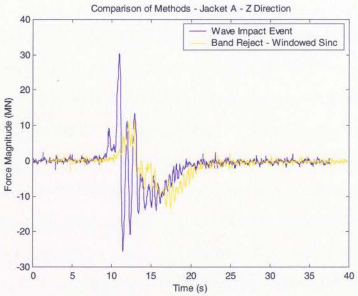

**Figure 7.21 Results – Band Reject Win. Sinc – Jacket A – Z Dir.**

# 8.0 Conclusions

The subject of wave impact on offshore structures and their components is important to vessel designers and operators for many reasons. They are often required to quantify these impact loads. Standard methods for wave load prediction will underestimate the forces on these structures due to intermittent loading. This necessitates the use of physical model tests to establish wave impact loads. The model measurement systems are designed to have high stiffness. This ensures that the natural frequency of the structure is above the wave frequency. However, it is widely believed that impacting waves (where the contact is well above the mean water line) contain high frequency energy components that cause the structure to vibrate at its modal frequencies. This impact-induced vibration is recorded by the measuring system as a force (inertial force), and corrupts the actual applied force measurement. Before scaling can occur, the inertial force must be removed from the measured signal.

A number of techniques for removing inertial force from measured signals have been described in published literature. Three of the major methods have been discussed, implemented, and compared in this report. Their algorithms have been presented, and the steps required to produce functioning code (using MATLAB), have been given. Each of the techniques contained inherent and unique problems that made their implementation challenging. Some of the problems were common to all the methods.

None of the techniques implemented produced results that were considered satisfactory. The main problem was the presence of unwanted high frequency content after the application of the methods. Another problem that plagues the comparison of the techniques is the lack of a method to quantify the performance. How can the success of a technique to remove inertial load be judged when it is not known how much inertial load there is to remove in the first place? These problems, and others, have led to the compilation of a list of topics that require further examination (Section 9.0).

While neither of the implemented methods were seen as completely satisfactory, the use of digital filtering techniques are recommended based on their relative performance, and the ease of implementation.

## 9.0 Summary of Topics That Require Further Work

Throughout this report, topics considered worthy of further investigation have been mentioned. These will be summarized here, in no particular order of importance.

- Many factors (such as air entrainment, model surface roughness, etc.) were cited as having an effect on the dynamic response of a model and its measuring system. The type and shape of the particular model will, in turn, affect many of these factors. This means that for each new model test, some investigation of the influence of these factors needs to be done.
- An appropriate data sampling rate for wave impact studies needs to be established. Early experimenters likely used analog techniques to record test data, and later digitized for computer analysis. Their sampling rates, when reported, seem quite low compared to recent experiments, which were often in the kilohertz range. Adequate values are likely somewhere in between. Further effort is needed to identify appropriate data acquisition sampling rates, and hardware filter values.
- The main problem associated with both the inverse Fourier transform method and the normal mode method is the effect of added mass. As the wave impacts a structure, the level of added mass varies both spatially and temporally. Without an accurate accounting of this parameter, both methods fail to produce satisfactory results. Techniques need to be developed to provide better estimates of the added mass during wave impact events.

- The normal mode technique makes the simplifying assumption that there is no damping. This is reflected in the defined equation of motion. Clearly damping is present in structural form, as well as damping related to fluid-structure interaction. This is another topic recommended for further investigation.
- The developers of the inverse Fourier transform method claimed that a matrix of cross-coupling terms would be required in order to properly implement the method. They claim that this would be difficult, if not impossible to achieve. Still it is a topic that warrants further inquiry.
- A study of cross-coupling terms could be linked to a study of overall dynamometer design. There has been little work done to quantify the effects of system dynamics on these important measuring devices.
- For the inverse Fourier transform method, it was demonstrated that the selection of the step response (which was differentiated to obtain the impulse response) had a major influence on the result. A systematic procedure needs to be established for this important part of the technique. In addition, a method to directly and accurately measure the stiffness matrix of a system needs to be developed.
- The inverse Fourier transform method was implemented using a time domain based deconvolution. The result was shown to be different from that obtained using frequency domain division. This was attributed to the impulse added to the impulse response to facilitate the deconvolution. The appropriateness of this addition needs to be investigated.

- Gibbs effect seems to be an unavoidable feature of many DSP techniques. A number of authors have proposed methods to mitigate this phenomenon, typically using windowing techniques. Further study needs to be done to evaluate the effectiveness of these techniques. This could be linked to a study of the use of filters to remove the erroneous high frequency content introduced when using the inverse Fourier transform method, or the normal mode method.
- The Blind Deconvolution technique, discussed in Appendix G requires further experimentation with the selection of desired pulse shapes, the application of the Custom Filter method, and consideration of Gibbs effect.
- Further work needs to be done to test the hypothesis put forward by Murray and Kaplan (1995) regarding the phase relationships of low and high passed signals measured from dynamometers. They stated that the high frequency components from the separate vertical dynamometer load cells were not phase locked to the low frequency components. This meant, they claimed, that the high frequency components were inertial in origin and could be removed using a low pass filter.
- The value of using a band reject filter versus a low pass filter to remove inertial force from measured signals needs to be established. In addition, a band reject filter that does not remove wave-induced energy from the rejection zone would be ideal.

These are a few of the topics that came to light during the preparation of this report. Some of these would be relatively simple to investigate, while others would require some complex model testing.

The subject of wave impact on offshore structures and their components will continue to be important to designers and operators. The development of numerical methods will depend on the availability of reliable model test methods, and the techniques to ensure credible results. As each of the topics above is investigated, the level of comfort with measured results will increase. This will lead to improved estimates of wave impact loads on offshore structures.

## 10.0 References

Andersen, H., Sterndorff, M.J., Dahl, C., (1998), EVALUATION OF BMT/IMD MODEL TEST RESULTS FOR EKOFISK, Project 98-50026, April 23, 1998.

Broughton, P. and Horn, E., (1987), EKOFISK PLATFORM 2/4C: RE-ANALYSIS DUE TO SUBSIDENCE, Proc. Inst. civil Eng., London, October 1987.

Campbell, I.M.C. Wellicome, J.F., & Weynberg, P., (1977), AN INVESTIGATION INTO WAVE SLAMMING LOADS ON CYLINDERS (OSFLAG 2A) FINAL REPORT, Wolfson Marine Craft Unit Report No. 317, Technology Reports Centre No. OT-R-7743, March 1977.

Campbell, IMC. and Weynberg, PA (1979), MEASUREMENT OF PARAMETERS AFFECTING SLAMMING, Rept No 440, Wolfson Unit for Marine Tech and Industrial Aerodyn, Univ Southampton.

Chan, E.S. and Melville, W.K., (1988), PLUNGING WAVE PRESSURES ON A VERTICAL PLANE WALL, Proc. R. Soc. A 417, 95-131, 1988.

Chan, E. S. Cheong, H. F. and Gin, K.Y.H., (1991), WAVE IMPACT LOADS ON HORIZONTAL STRUCTURES IN THE SPLASH ZONE, Proc. 1st International Offshore and Polar Engineering Conference, Aug 1991.

Chan, S., (1993), EXTREME WAVE ACTION ON LARGE HORIZONTAL CYLINDERS LOCATED ABOVE STILL WATER LEVEL, Proc 3rd (1993) International Offshore and Polar Conference, Singapore, 6-11 June 1993.

Clough, R.W., & Penzien, J., (1975), DYNAMICS OF STRUCTURES, McGraw-Hill Book Company.

Dalton, C., Nash, J., (1976), WAVE SLAM ON HORIZONTAL MEMBERS OF AN OFFSHORE PLATFORM, Proc. Offshore Tech Conf. Paper No. 2500, Houston, 1976.

Faltinsen, O., Kjaerland, O.,Nottveit, A., and Vinje, T., (1977), WATER IMPACT LOADS AND DYNAMIC RESPONSE OF HORIZONTAL CIRCULAR CYLINDERS IN OFFSHORE STRUCTURES, Offshore Technology Conference, OTC 2741, 1977.

Garrison, C.J., (1996), WATER IMPACT LOADS ON CIRCULAR STRUCTURAL MEMBERS, Applied Ocean Research, 18, pp45-54, 1996.

Hamming, R.W., (1983), DIGITAL FILTERS, Second Edition, Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

Isaacson, M. and Subbiah, K., (1990), RANDOM WAVE SLAMMING ON CYLINDERS, Journal of Waterway, Port, Coastal, and Ocean Engineering, Vol. 116, No. 6, Nov/Dec, 1990.

Isaacson, M., Prasad, S., (1992), WAVE SLAMMING ON A HORIZONTAL CIRCULAR CYLINDER, Civ. Eng. In Oceans V, ASCE, 1992.

Isaacson, M., Prasad, S., (1994), WAVE SLAMMING ON A HORIZONTAL CIRCULAR CYLINDER, Intl. Journal of Offshore and Polar Engineering, Vol. 4, No. 2,June 1994

Jackson, L.B., (1989), DIGITAL FILTERS AND SIGNAL PROCESSING, Second Edition, Kluwer Academic Publishers, Norwell MA.

James, M.L., Smith, G.M., Wolford, J.C., (1977), APPLIED NUMERICAL METHODS FOR DIGITAL COMPUTATION WITH FORTRAN AND CSMP, Second Edition, Harper & Row, Publishers Inc.

Kaplan P., (1979), IMPACT FORCES ON HORIZONTAL MEMBERS, Proc. Civ. Eng. In Oceans IV, September 1979

Kaplan, P., (1992), WAVE IMPACT FORCES ON OFFSHORE STRUCTURES: RE-EXAMINATON AND NEW INTERPRETATIONS, Offshore Technology Conference, OTC 6814, 1992

Kaplan P., Murray, J.J., Yu, W.C., (1995), THEORETICAL ANALYSIS OF WAVE IMPACT FORCES ON PLATFORM DECK STRUCTURES, OMAE 1995, Vol. 1A, Offshore Technology, ASME 1995.

Kirkgoz, M.S., & Mengi, Y., (1987), DESIGN OF A CAISSON PLATE UNDER WAVE IMPACT, Ocean Engineering, Vol.14, No. 4, pp. 275-283, 1987.

Miao, G., (1990), DYNAMIC RESPONSE OF ELASTIC HORIZONTAL CIRCULAR ELEMENTS OF OFFSHORE STRUCTURES, Proc. 9th Int. Conf. on Offshore Mech. and Arctic Eng., Vol. 1, 1990, pp 239-245

Miller, B., (1977), WAVE SLAMMING LOADS ON HORIZONTAL CIRCULAR ELEMENTS OF OFFSHORE STRUCTURES, Spring Meeting, Royal Inst. of Naval Architects, Paper No., 5.

Miller, B., (1980), WAVE SLAMMING ON OFFSHORE STRUCTURES, NMI report R81, March 1980.

Mogridge, G.R., & Cornett, A.M., (1989), MULTIDIRECTIONAL WAVE LOADING ON THE JACKET AND DECK OF A SPACE-FRAME OFFSHORE PLATFORM, CTR-HY-035, Hydraulics Laboratory, Division of Mechanical Engineering, National Research Council Canada, 1989/07.

Murray, J.J., Kaplan, J.J., Yu, W.C., (1995), EXPERIMENTAL AND ANALYTICAL STUDIES OF WAVE IMPACT FORCES ON EKOFISK PLATFORM STRUCTURES. Offshore Technology Conference 1995, OTC 7782.

Murray, J.J., Winsor, F.N., & Kaplan, P., (1997), IMPACT FORCES ON A JACKET DECK IN REGULAR WAVES AND IRREGULAR WAVE GROUPS. Offshore Technology Conference, OTC 8360, Houston, Texas, May 5-8, 1997.

Prasad, S., Chan, E.S., & Isaacson, M., (1994), BREAKING WAVE IMPACT ON A SLENDER HORIZONTAL CYLINDER, Proceedings of the Fourth International Offshore and Polar Engineering Conference, Osaka, Japan April 10-15, 1994.

Sarpkaya, T., (1978), WAVE IMPACT LOADS ON CYLINDERS, Offshore Technology Conference, OTC 3065, 1978

Smith, S.W., (1999), THE SCIENTIST AND ENGINEER'S GUIDE TO DIGITAL SIGNAL PROCESSING, Second Edition, California Technical Publishing, San Diego, CA.

Terrell, T.J., (1980), INTRODUCTION TO DIGITAL FILTERS, The Macmillan Press Ltd, London.

The Mathworks, Inc., (1996) SIGNAL PROCESSING TOOLBOX USER'S GUIDE, The Mathworks Inc., 24 Prime Park Way, Natick, MA 01760-1500.

Thomson, W.T., (1981), THEORY OF VIBRATION WITH APPLICATIONS, Second Edition, Prentice-Hall, Inc., Englewood Cliffs, N.J., 07632.

Toumazis, A.D., Shih, W.K., & Anastasiou, K.A., (1989), WAVE IMPACT LOADING ON HORIZONTAL AND VERTICAL PLATES, International Association for Hydraulic Research, XXII Congress, August 21-25, 1989, Ottawa, Canada.

Walker, J.S., (1991), FAST FOURIER TRANSFORMS, CRC Press, Inc. Boca Raton, Florida.

Winsor, F., (1998), EKOFISK CESSATION PROJECT / SUBSIDENCE MITIGATION PROJECT – MODEL TEST AND DATA REPORT, Institute for Marine Dynamics, National Research Council Canada, TR-1998-04, June 1998.

# Appendix A

## Use of Adjoint Matrix

## to Determine Eigenvectors

## A.1.0 Method to Establish Eigenvectors

The eigenvectors discussed in Section 4.2.3 may also be determined using the adjoint matrix of the system as follows.

Let

$$[A-\lambda I]=[B] \qquad \textbf{A.1}$$

The inverse of $[B]$ is given by,

$$[B]^{-1}=\frac{1}{|B|}adj[B] \qquad \textbf{A.2}$$

The adjoint of a square matrix is the transpose of its cofactor matrix. Multiplying both sides of Equation A.2 by $|B|[B]$ leads to

$$|B|[I]=[B]adj[B] \qquad \textbf{A.3}$$

By substituting Equation A.1 into Equation A.3, Equation A.4 is obtained.

$$|A-\lambda I|[I]=[A-\lambda I]adj[A-\lambda I] \qquad \textbf{A.4}$$

Substitution of a root or eigenvalue $(\lambda_i)$ into Equation A.4 results in Equation A.5.

$$[0] = [A - \lambda_i I] adj [A - \lambda_i I] \qquad \textbf{A.5}$$

When compared to Equation 4.12 it is seen that for each eigenvalue, the eigenvector can be determined as follows.

$$\{\tilde{X}_i\} = adj [A - \lambda_i I] \qquad \textbf{A.6}$$

# Appendix B

# Use of Accelerometer Measurements to Determine Inertial Force

## B.1.0 Use of Accelerometer Measurements to Determine Inertial Force

Several authors have discussed the use of accelerometer measurements to determine and remove inertial force components from measured force data.

Mogridge and Cornett (1989) applied the inertial force correction outlined in Equation B.1 to the total force measured by a jacket structure deck dynamometer subjected to wave loading.

$$F_{ext,t} = k\,x_t + m\,\ddot{x}_t$$

where

| | | |
|---|---|---|
| $F_{ext,t}$ | external force | **B.1** |
| $m$ | deck mass (including added mass) | |
| $k$ | dynamometer stiffness | |
| $x_t$ | deck displacement | |
| $\ddot{x}_t$ | deck acceleration. | |

The deck dynamometer for their experiment consisted of four vertical load transducers, and three horizontal load transducers (one in the X direction, two in the Y direction). Four accelerometers were mounted on the deck, three measuring translational modes, and one to estimate torsional moments.

The sample rate was 500 Hz model scale, with a hardware filter of 100 Hz to prevent aliasing. The force measurements were further filtered in post-processing, applying a Kaiser filter with a cutoff of 9 Hz full scale (47.6 Hz model scale). The signals were decimated at a rate of 2:3. The authors reported that the advantage of post-experiment filtering was that it allowed the objectives of the filtering to be easily identified, and the appropriate filter chosen.

The corrections were applied only to the horizontal force measurements because the accelerations were not measured in six DOFs. The authors quantify the signal reduction merely by noting the percentage reduction in the peak, and making some general observations regarding the reduction of oscillations after the impact peak has passed.

Murray and Kaplan (1995) discuss the limitations of this method. First, the equation of motion assumes no damping. This ignores the effects of both structural and fluid damping. The deck structure is assumed to behave as a rigid plate. This is likely not the case. It can be confirmed by comparing the phase differences between accelerometers. The main problem with this method is in the estimation of added mass and added moment of inertia. As the wave crest passes the underside of the deck, the amount of added mass varies spatially and temporally. The water contact area continually changes, making an added mass estimate difficult. As a result the corrected force signal will contain unwanted oscillations related to the poor added mass estimate. Mogridge and

Cornett (1989) apparently make no allowance for this, which undoubtedly led to erroneous results.

# Appendix C

## Use of Fast Fourier Transform to Develop Matrix of Decay Force Vectors

## C.1.0 Use of Fast Fourier Transform to Develop Matrix of Decay Force Vectors

This section will discuss the methodology used to develop the decay force vector used in Equation 4.19, of Section 4.2.3. This square matrix represents the amplitude of force response in all degrees-of-freedom (DOF), due to excitation in each DOF. Ideally, the off diagonal matrix terms would be zero, indicating that the structure responded only in the direction of applied excitation.

Free-vibration decay tests were conducted on the model described in Section 3.0, in each degree-of-freedom measured by the dynamometer. For example, in the X direction, the decay test was conducted by applying a static load to the model, colinear with the model coordinate system. The load was released instantaneously, and force measurements were made in six degrees-of-freedom. The load measurements were used to develop a force amplitude vector for the X DOF. This procedure was repeated for each degree-of-freedom.

Development of the matrix of decay force vectors involved the use of the fast Fourier transform (FFT). For each force vector, or excitation DOF, the measured response signals for each degree-of-freedom were subjected to a FFT. Care needs to be taken in the application of the FFT. The signal time step should be selected such that aliasing is avoided. When dealing with model test data, this is typically considered in the data acquisition setup. The length or size of the FFT should match the number of data points in the time series. For the most efficient FFT processing, its length should be a power of two. When the time series length is shorter than the FFT length, it is recommended that the time series be padded with zeros, to the length of the FFT. It is important that the time series not be truncated by using a FFT length shorter than the number of time series data points. An effort should also be made to ensure that the product of the FFT length and the signal time step be an integer multiple of the time series natural period. This ensures that periodicity is maintained.

The magnitude of the FFT is scaled using a procedure described the Matlab Signal Processing Toolbox User's Guide. As per the Matlab script in Equation C.1, the scaled magnitude is obtained by dividing the absolute value of the FFT by the number of data points in the time series, all multiplied by a factor of two.

$$Pn = abs\,(FFT\,(x)) \times 2\,/\,length(x) \qquad \textbf{C.1}$$

where,

$Pn =$ scaled magniude

$x =$ input time series

The sign of the phase is used to determine the sign of the magnitude. Matlab command *angle.m* is used to determine the phase of the input signal.

## C.2.0 Implementation

The procedure used here is summarized in Table C.1. The procedure is implemented in the Matlab script *FFT_modulus_general.m*, which is shown in Appendix K

When applying this method to a set of decay signals, the question arises as to what the appropriate signal length is. Is it sufficient to use 128 points, or are 512 or 1024 points preferable? This was explored by examining three decay force vectors (from the model tests described in Section 3.0), using three sample lengths, 128, 256, and 512 points. Samples of the selected signals are shown in Figures C.1 to C.6. These represent the measurements in six degrees-of-freedom, to an excitation load applied in the X direction. Similar plots could be generated for excitation in the other degrees-of-freedom.

Table C.2 shows the scaled magnitudes and phase signs obtained for each signal length, obtained using the appropriate identical FFT length. These represent results for three decay force vectors corresponding to initial excitation in the three translational degrees-of-freedom. Figure C.7 shows a sample magnitude (modulus) and phase plot. The magnitudes for each vector were normalized relative to their respective largest value. The largest term was then represented by unity. The normalized vectors obtained for each FFT length (or signal length) were compared, as shown in Figures C.8 to C.10, and Table C.3. These plots indicate that the selected signal length (and therefore FFT length) arguably has little influence on the shape of the vector, and therefore either FFT length value may be used.

This method is best suited for determining the magnitude of sinusoidal signals. It is a simplifying assumption to apply it to a decaying signal.

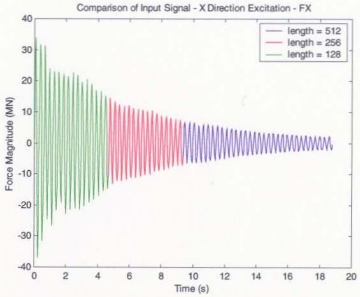

**Figure C.1 Input Signals of Various Power Length – X Direction**

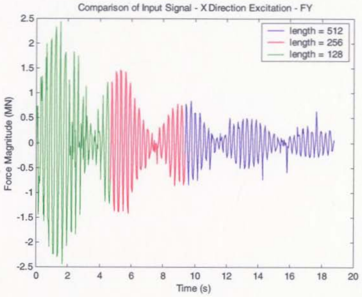

**Figure C.2 Input Signals of Various Power Length – Y Direction**

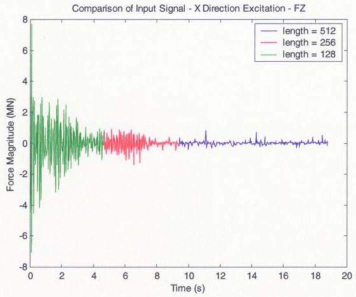

**Figure C.3 Input Signals of Various Power Length – Z Direction**

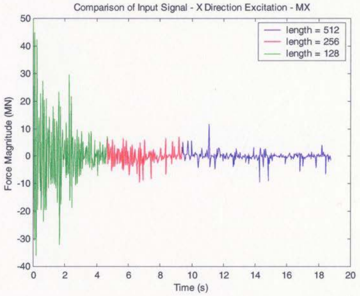

**Figure C.4 Input Signals of Various Power Length – Moment about X**

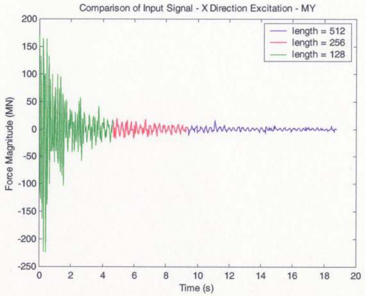

**Figure C.5 Input Signals of Various Power Length – Moment about Y**

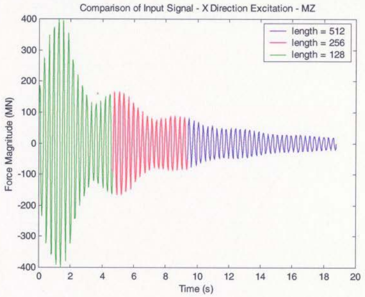

**Figure C.6 Input Signals of Various Power Length – Moment about Z**

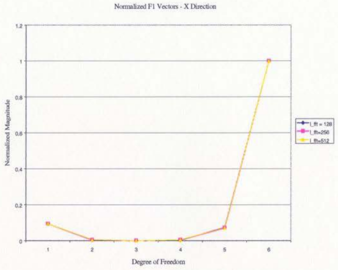

**Figure C.7 Normalized Force Magnitude vs. DOF – X Excitation**

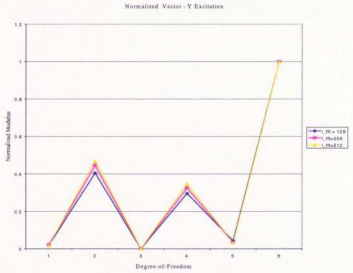

**Figure C.8 Normalized Force Magnitude vs. DOF – Y Excitation**

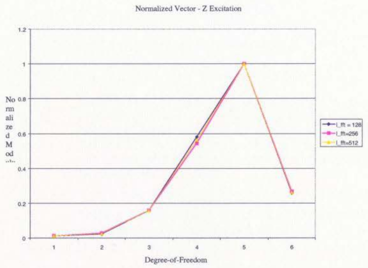

**Figure C.9 Normalized Force Magnitude vs. DOF – Z Excitation**

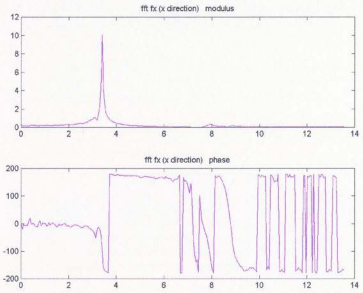

**Figure C.10 Sample FFT Magnitude (Modulus) and Phase vs. Freq. in Hertz**

Table C.1 - Procedure for Developing the Decay Force Vectors

| Step | Procedure Description |
|---|---|
| 1 | Establish the input signal time step. It is important to ensure that aliasing does not occur. For model test data, the time step would be established during the data acquisition configuration. For simulated data, the time step can be set experimentally by iteratively adjusting the step increment and calculating the FFT until the magnitude no longer changes. |
| 2 | Establish the natural period for each degree of freedom by performing decay or a spectral analysis. |
| 3 | Examine the measured signal that is in-line with the direction of applied load. From this select a representative segment, ensuring that the number of data points in the selection is a power of two (i.e. 64, 128, 256, etc.). If it is not possible to match one of these values, use a smaller value. The signal can then be padded with zeros to attain the power of two. The Matlab command *FFT.m* will automatically pad a signal with zeros to a power of two. |
| 4 | Check the periodicity of the selected signals. The signal time step (T) multiplied by the number of data points in the signal (N) should be an integer multiple of the individual signal periods. |
| 5 | Perform the FFT. Scale the magnitude using the using the technique defined in Equation C.1. |
| 6 | Examine the FFT of the measured signal that is in-line with the direction of applied load. Record the magnitude, phase (sign only), and frequency of the resonant peak. At the same frequency, record the magnitude, and phase (sign only) of the other responses. The sign of the phase determines the sign of the magnitude. The phase can be can be determined using the Matlab command *angle.m* .<br><br>These magnitudes represent the decay force amplitude vectors. |

| Table C.2 – Scaled Magnitude and Phase for Three Vectors | | | | | | |
|---|---|---|---|---|---|---|
| | Vector F1 – X Direction | | Vector F2 – Y Direction | | Vector F3 – Z Direction | |
| DOF | Scaled Magnitude (MN) | Phase Sign | Scaled Magnitude (MN) | Phase Sign | Scaled Magnitude (MN) | Phase Sign |
| FFT Length 128 | | | | | | |
| FX | 22.303 | negative | 1.325 | negative | 0.525 | positive |
| FY | 1.021 | negative | 26.235 | negative | 0.870 | negative |
| FZ | 0.069 | positive | 0.085 | negative | 5.838 | positive |
| MX | 0.870 | negative | 19.150 | negative | 21.500 | negative |
| MY | 17.400 | positive | 2.775 | positive | 37.100 | positive |
| MZ | 237.750 | negative | 64.750 | negative | 9.600 | negative |
| FFT Length 256 | | | | | | |
| FX | 16.190 | negative | 0.872 | negative | 0.274 | positive |
| FY | 0.832 | negative | 17.584 | negative | 0.570 | positive |
| FZ | 0.038 | positive | 0.040 | negative | 3.200 | positive |
| MX | 0.720 | negative | 12.850 | negative | 11.075 | negative |
| MY | 12.370 | positive | 1.356 | positive | 20.300 | positive |
| MZ | 172.500 | negative | 39.450 | negative | 5.395 | negative |
| FFT Length 512 | | | | | | |
| FX | 10.002 | negative | 0.370 | negative | 0.141 | positive |
| FY | 0.530 | negative | 9.493 | negative | 0.275 | positive |
| FZ | 0.027 | positive | 0.021 | negative | 1.602 | positive |
| MX | 0.500 | negative | 7.035 | negative | 5.720 | negative |
| MY | 7.600 | positive | 0.685 | positive | 10.150 | positive |
| MZ | 105.800 | negative | 20.280 | negative | 2.670 | negative |

Table C.3 – Normalized Decay Force Vectors

Vector F1 – X Direction

| FFT Length | 128 | 256 | 512 |
|---|---|---|---|
| FX | 0.093808 | 0.093855 | 0.094532 |
| FY | 0.004294 | 0.004823 | 0.005009 |
| FZ | 0.000288 | 0.000219 | 0.000255 |
| MX | 0.003659 | 0.004174 | 0.004726 |
| MY | 0.073186 | 0.07171 | 0.071834 |
| MZ | 1 | 1 | 1 |

Vector F2– Y Direction

| FFT Length | 128 | 256 | 512 |
|---|---|---|---|
| FX | 0.020463 | 0.022104 | 0.018245 |
| FY | 0.405174 | 0.445729 | 0.468102 |
| FZ | 0.001313 | 0.001019 | 0.001036 |
| MX | 0.295753 | 0.325729 | 0.346893 |
| MY | 0.042857 | 0.034373 | 0.033777 |
| MZ | 1 | 1 | 1 |

Vector F3– Z Direction

| FFT Length | 128 | 256 | 512 |
|---|---|---|---|
| FX | 0.014151 | 0.013478 | 0.013911 |
| FY | 0.02345 | 0.028079 | 0.027074 |
| FZ | 0.157345 | 0.157635 | 0.157833 |
| MX | 0.579515 | 0.545567 | 0.563547 |
| MY | 1 | 1 | 1 |
| MZ | 0.25876 | 0.265764 | 0.263054 |

## Appendix D

## Discussion of Orthogonality Condition

## D.1.0 Discussion of Orthogonality Condition

Orthogonality relationships are special properties of free-vibration mode shapes. The concept of orthogonality is based on Betti's law. Betti's law implies that the work done by one set of loads on the deflections due to a second set of loads is equal to the work of the second set of loads acting on the deflections due to the first. Equation D.1 illustrates this concept.

$$[P_a]^T \{x_b\}_{=} = [P_b]^T \{x_a\} \qquad \textbf{D.1}$$

Where $P_a$ *and* $P_b$ are separate sets of applied loads, and $x_a$ **and** $x_b$ are the respective deflections, or displacements.

When considering free vibration, Betti's law can be re-written to represent the relationship between deflections and inertial forces.

$$-[f_{im}]^T \{x_n\} = -[f_{in}]^T \{x_m\} \qquad \textbf{D.2}$$

where $f_{im}$ *and* $f_{in}$ represent inertial forces for separate mode shapes m and n, while $x_n$ *and* $x_m$ are the respective deflections, or displacements.

The inertial force can be represented by $\omega_n^2[M]\{x_n\}$. When substituted into Equation D.3 this leads to

$$\omega_m^2\{x_m\}^T[M]\{x_n\} = \omega_n^2\{x_n\}^T[M]\{x_m\} \qquad \textbf{D.3}$$

This can be re-arranged to give

$$(\omega_m^2 - \omega_n^2)\{x_m\}^T[M]\{x_n\} = 0 \qquad \textbf{D.4}$$

since the product of the matrices is a scalar.

When $\omega_n \neq \omega_m$ we see that

$$\{x_m\}^T[M]\{x_n\} = 0 \qquad \textbf{D.5}$$

A similar relationship can be developed for the stiffness matrix.

$$\{x_m\}^T[K]\{x_n\} = 0 \qquad \textbf{D.6}$$

These are known as the orthogonality conditions with respect to mass and stiffness. These can be expressed in terms of normalized eigenvectors as follows.

$$[X_m]^T [M][X_n] = 0 \quad \text{when} \quad m \neq n \qquad \textbf{D.7}$$

$$[X_m]^T [K][X_n] = 0 \quad \text{when} \quad m \neq n \qquad \textbf{D.8}$$

In a practical sense, when these conditions are satisfied, it means that the eigenvectors are independent of each other.

# Appendix E

## Discussion of Numerical Differentiation Techniques

## E.1.0 Numerical Differentiation Techniques

The normal mode approach to removing inertial forces from measured signals requires a numerical differentiation to obtain acceleration signals from displacements. The numerical differentiation of an experimentally acquired signal, or other noisy signal, can be a dubious operation. Rapid changes in signal value from point to point are amplified in the differentiation process, resulting in serious problems with accuracy. The differentiation algorithms used here are described in James (1977).

The definition of the first derivative, shown in Equation E.1, illustrates the inherent problem with numerical differentiation.

$$\frac{dy}{dx} = \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x} \qquad \textbf{E.1}$$

The denominator $\Delta x$ is required to be as small as possible. This means that the difference between the numerator parameters will be smaller. Taken to extremes, the operation will ultimately be limited by the precision of the computer. In other words, the subtraction of two small (and similar) numbers, and subsequent division by another small number may result in a derivative with large error.

To mitigate this it is sometimes recommended that an analytical expression be fit through experimentally obtained data. The expression could then be differentiated analytically. Alternately, points generated from the analytically defined curve can be differentiated numerically. Otherwise, the experimental data must be differentiated point by point.

The data obtained from the experiments discussed in Section 3.0 A were not amenable to curve fitting techniques, therefore analytical differentiation was not an option. Instead a numerical technique based on Taylor-series expansions was used. These techniques are commonly known as Central-Difference expressions.

The Matlab scripts that implement the normal mode method utilize the Central-Difference expression, with error of order $h^4$. The algorithm shown in Equation E.2 produces the second derivative.

$$y_i'' = \frac{-y_{i+2} + 16\,y_{i+1} - 30\,y_i + 16\,y_{i-1} - y_{i-2}}{12\,h^2}$$

where, **E.2**

| | |
|---|---|
| $y$ | ordinate of the input signal |
| $i$ | ordinate index |
| $h$ | increment of the abscissa |
| $y''$ | second derivative |

This expression requires knowledge of the values previous and subsequent to the point indicated by the ordinate index. This is dealt with by using the Forward-Difference, and Backward-Difference expressions as shown in Equations E.3 and E.4, respectively.

$$y_i'' = \frac{-y_{i-3} + 4\,y_{i-2} - 5\,y_{i-1} + 2\,y_i}{h^2} \qquad \textbf{E.3}$$

$$y_i'' = \frac{2\,y_i - 5\,y_{i-1} + 4\,y_{i-2} - y_{i-3}}{h^2} \qquad \textbf{E.4}$$

These expressions have been used in normal mode method scripts ( i.e. *normal_mode_v3_3dof.m* ) to differentiate the normal coordinate displacement vectors. Script *first_deriv.m* is a stand-alone differentiation program that also utilizes these algorithms.All scripts are located in Appendix K.

An alternate procedure for differentiation involves the multiplication of a signal's Fourier transform by an imaginary ramp function. This can be simulated by subjecting the time-series signal to a filter that has a response $H(\omega) = j\,\omega$ . The Matlab script *remez.m* can be used to generate such a filter. A typical implementation of the *remez.m* command is shown in Equation E.5.

$b = remez(n, f, a, ftype)$

where

**E.5**

| | |
|---|---|
| $n$ | order of filter |
| $f$ | frequency range vector |
| $a$ | amplitude vector |
| $b$ | output filter coefficients |

Example:
$b = remez(21, [0\ 1], [0\ pi*Fs], 'd')$

The order of the filter $n$, determines the fit to the imaginary ramp function. The frequency range vector must be expressed in values from 0 to 1, where 1 represents the Nyquist frequency (i.e. half the sampling frequency $Fs$). The amplitude vector defines the amplitude for the frequency pairs defined in the frequency range vector. The *ftype* option defines the filter type, in this case '*d*' signifies differentiation.

This differentiation method was applied in the normal mode method in script *normal_mode_remez.m* (see Appendix K ). The results displayed transients at the beginning of the signal, the length of which were related to the order of the filter $n$. The first $n + 1$ data points were truncated from each *remez.m* output signal, in order to maintain alignment with the other signals utilized in the program.

A comparison of results using *remez.m* and the central-difference technique showed negligible differences.

## Appendix F

## Use of Rayleigh Damping Method to Estimate a Damping Force Component

## F.1.0 Use of Rayleigh Damping Method to Estimate a Damping Force Component

The normal mode approach as described in Section 4.0 assumes that damping is negligible. This assumption ignores the inherent structural damping present in the model measuring system. This section will discuss the use of the Rayleigh damping method to provide an estimation for the damping term in the system equation of motion. The method will be applied to simulated and measured force signals, and the effect of the damping term will be assessed.

## F.2.0 Description of Rayleigh Damping

The Matlab script *Normal_Mode_V3_3dof.m*, shown Appendix K, includes an estimate for the damping term in the equation of motion. The damping coefficient is determined

$$\xi_n = \frac{a_0}{2\omega_m} + \frac{a_1 \omega_n}{2} \qquad \textbf{F.1}$$

based on the Rayleigh Damping method, as described by Clough and Penzien (1975). The Rayleigh damping method assumes that the damping ratio is proportional to mass and stiffness. The damping ratio is expressed as shown in Equation F.1.

The first term on the right hand side defines the mass proportionality, while the second term defines the stiffness proportionality. The parameters $a_0$ and $a_1$ are known as proportionality constants. The damping matrix is then determined using the Equation F.2.

$$C = a_0\, m + a_1\, k \qquad \text{F.2}$$

where,

| | |
|---|---|
| $C$ | damping coefficient matrix, |
| $m$ | mass matrix, |
| $k$ | stiffness matrix. |

Equation F.1 can be expressed in matrix form for multi-degree-of-freedom systems, and rearranged to allow determination of the proportionality constants, as shown in Equation F.3. This requires some knowledge of the damping ratio for each degree of freedom. Generally, this information is not available, so it is assumed that the damping ratio is constant for each frequency or degree-of-freedom, and Equation F.3 can be simplified as shown in Equation F.4.

$$\begin{Bmatrix} a_0 \\ a_1 \end{Bmatrix} = 2\frac{\omega_m \omega_n}{\omega_n^2 - \omega_m^2}\begin{bmatrix} \omega_n & -\omega_m \\ -1/\omega_n & 1/\omega_m \end{bmatrix}\begin{Bmatrix} \xi_m \\ \xi_n \end{Bmatrix} \qquad \text{F.3}$$

$$\begin{Bmatrix} a_0 \\ a_1 \end{Bmatrix} = \frac{2\xi}{\omega_m + \omega_n}\begin{Bmatrix} \omega_m\, \omega_n \\ 1 \end{Bmatrix} \qquad \text{F.4}$$

The parameter $\xi$ is the assumed damping ratio. The frequencies $\omega_m$ and $\omega_n$ are the fundamental frequency and the highest relevant frequency respectively, in radians per second. The modes between the fundamental and highest frequencies will have damping ratios slightly lower than the estimated value. The frequencies above and below the fundamental and highest values will have increasingly higher values for the damping ratio.

## E.3.0 Implementation of Rayleigh Damping Method

A set of simulated decay test force signals has been generated using the method described in Appendix J. The normal mode approach with and without damping was applied to these signals using scripts *normal_mode_v3_3dof.m* and *normal_mode_v3_3dof_nodamping.m* (Appendix K), respectively. Without inclusion of a damping term the resulting actual force signal contains evidence of a damping force. When the Rayleigh estimate is included, the normal mode approach successfully removes both the inertial and damping force components from the simulated actual force signals.

Figures F.1 to F.3 compare the simulated measured and actual forces and show the effect of the Rayleigh damping estimate. The measured force signals were generated using a damping ratio of 0.05, and the same value was used in the Rayleigh estimate. Clearly, inclusion of the damping term results in a better representation of the expected actual force.

Figure F.4 illustrates the effect of changing the damping ratio when generating the Rayleigh damping estimate. The ability to remove the damping force is significantly impaired. Recall that the simulated signals were generated using a damping ratio of 0.05. Use of other values in the Rayleigh damping estimate causes the normal mode script to produce actual force signals that contain a damping component. Note that when a damping ratio of 0.05 is used, the resulting actual force is 90 degrees out of phase with the signal generated without consideration of damping. This makes sense since under the assumption of harmonic motion, acceleration and displacement are 180 degrees out of phase with each other, while velocity is 90 degrees out of phase.

The oscillation remaining in the could be the result of the numerical differentiation process used in the normal mode approach.

Figures F.5 to F.10 compare measured and actual force signals, from the experiments described in Section 3.0. The actual force signals were generated using *normal_mode_v3_3dof.m*, with a damping ratio of 0.05 assumed. Clearly, the signal reduction has not been improved in this case. Figures F.11 to F.16 show very little difference between the damped and non-damped cases. This could be due to the assumption of a damping ratio of 0.05. The examples using simulated signals demonstrated the sensitivity of the result to correct damping ratio. However, Figures F.17 to E.19 show little improvement in the actual force result when a more realistic damping ratio of 0.007 was used.

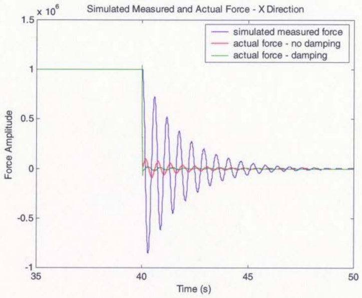

Figure F.1 Simulated Measured and Actual Force – X Direction

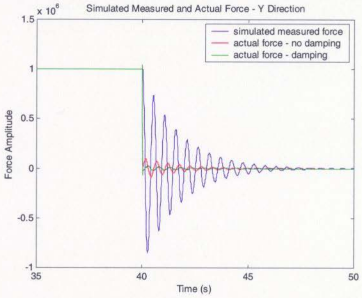

Figure F.2 Simulated Measured and Actual Force – Y Direction

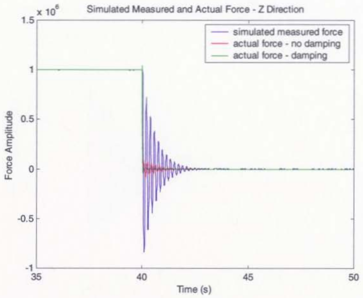

**Figure F.3 Simulated Measured and Actual Force – Z Direction**

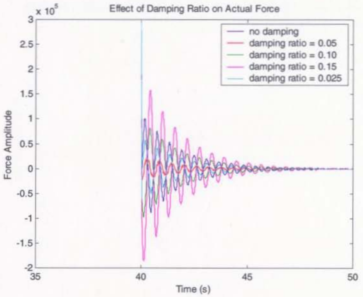

**Figure F.4 Effect of Damping Ratio on Actual Force**

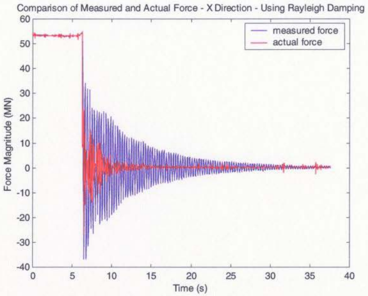

**Figure F.5 Measured and Actual Force – Rayleigh Damping – X Dir.**

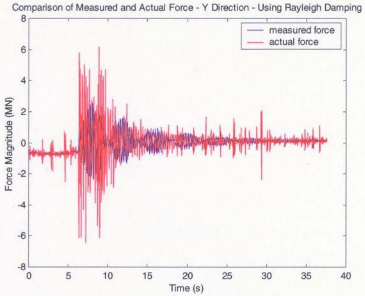

**Figure F.6 Measured and Actual Force – Rayleigh Damping – Y Dir.**

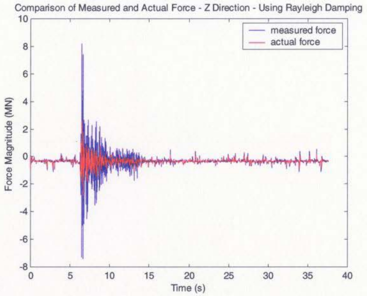

**Figure F.7 Measured and Actual Force – Rayleigh Damping – Z Dir.**

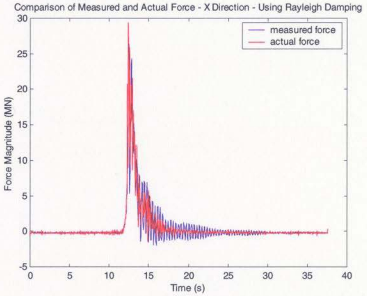

**Figure F.8 Measured and Actual Impact – Rayleigh Damping – X Dir.**

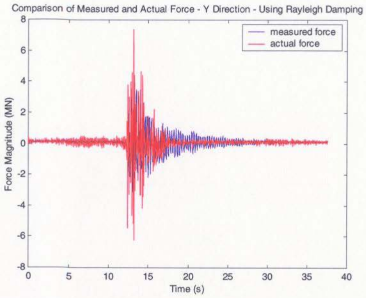

**Figure F.9 Measured and Actual Impact – Rayleigh Damping – Y Dir.**

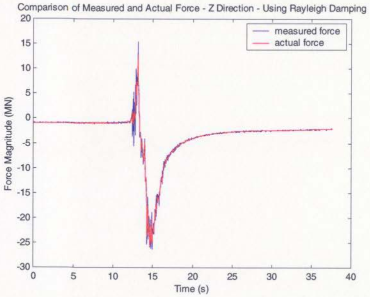

**Figure F.10 Measured and Actual Impact – Rayleigh Damping – Y Dir.**

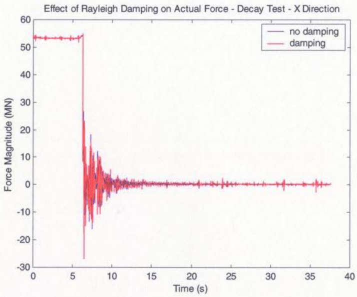

**Figure F.11 Effect of Rayleigh Damping on Actual Force – X Dir.**

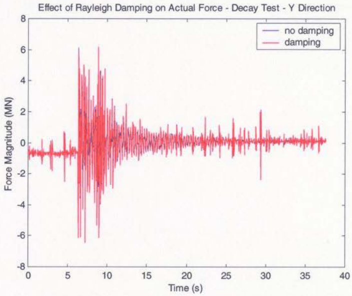

**Figure F.12 Effect of Rayleigh Damping on Actual Force – Y Dir.**

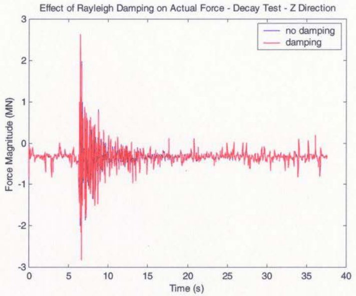

**Figure F.13 Effect of Rayleigh Damping on Actual Force – Z Dir.**

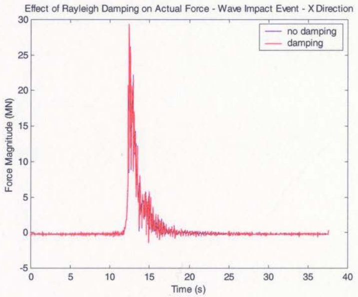

**Figure F.14 Effect of Rayleigh Damping on Actual Force – X Dir.**

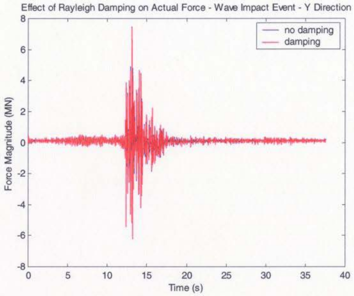

**Figure F.15 Effect of Rayleigh Damping on Actual Force – Y Dir.**

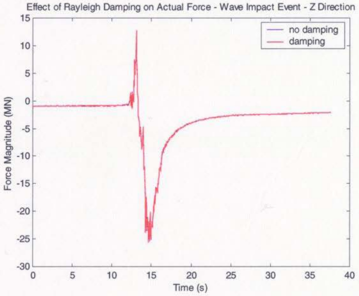

**Figure F.16 Effect of Rayleigh Damping on Actual Force – Z Dir.**

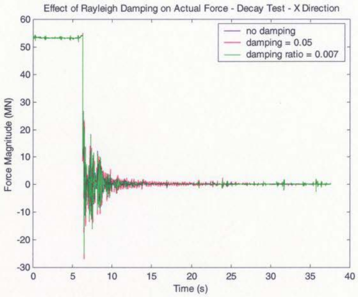

**Figure F.17 Effect of Damping Ratio on Actual Force – X Dir.**

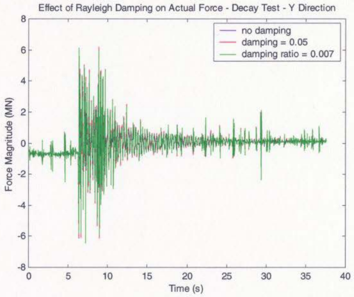

**Figure F.18 Effect of Damping Ratio on Actual Force – Y Dir.**

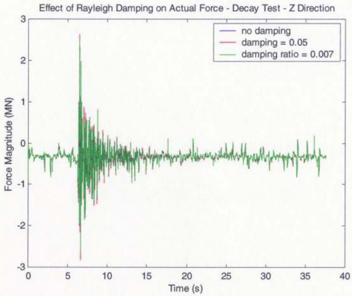

**Figure F.19 Effect of Damping Ratio on Actual Force – Z Dir.**

# Appendix G

## Discussion of Blind Deconvolution Applied to Measured Wave Impacts

## G.1.0 Blind Deconvolution Applied to Measured Wave Impacts

### G.1.1 Introduction

An interesting procedure to remove unwanted convolutions from measured signals, known as Blind Deconvolution, is presented in Smith (1999). The author describes the case of a gamma-ray detector, which records pulses due to the impact of gamma rays. The pulses are however convolved with the unknown impulse response of the system. The Blind Deconvolution involves the estimation of the system impulse response that is then deconvolved from the detected pulses, resulting in a filter kernel that can then be used to reveal the true nature of other detected pulses.

## G.2.0 Implementation of Method

In this section the implementation of this procedure using wave induced impact data, as well as other types of signals, will be presented.

As with the methods previously discussed, the procedure is really quite simple. However, questions can be raised about the implementation. The first step is to identify the so-called detected pulse. For the example described in Smith (1999), the detected pulse was the response to an impulse, measured by the light detector. For the present case the detected pulse will be represented by the total vertical force measurement from the deck dynamometer of a jacket structure model (described in Section 3.0) subjected to wave loading, see Figure G.1. From this, a representative selection will be made and used as the detected pulse (det_pul), see Figure G.2.

The remaining steps seem simple on paper. Generate a signal that represents the desired pulse (des_pul), a pulse that has not been corrupted by unwanted convolution. Take the FFT of these signals, and divide the desired (DES_PUL) by the detected (DET_PUL) to obtain the required frequency response (RFR). Take the inverse Fourier transform of this to obtain the required filter kernel (rfk). The rfk is equivalent to an impulse response, and will be used in convolution with other detected pulses to obtain the non-corrupted measurement.

Before the rfk can be used in a convolution it must be conditioned using the so-called Custom Filter Technique, which involves shifting, truncating, and windowing the signal.

The requirement for truncation of the rfk is related to the fact that a computer cannot represent a continuous signal. By necessity it requires a sampled filter kernel. The filter kernel is, therefore, inherently truncated, no matter what its length. As a result it will be subject to the problems associated with the discontinuity, namely overshoot, ringing, and aliasing. This is unavoidable. The question then is how much should be truncated? Despite the fact that a longer filter kernel will produce a better representation of the frequency response, it will increase computation time. Also, the practical filter kernel length is limited by the length of the input signal (the signal it will be convolved with) and the problems related to end effects (to be discussed later).

The application of a window (i.e. Blackman or Hamming) will relieve the effects caused by the truncation discontinuity.

The shift allows for the easy indexing of vectors by the computer programs performing the task. This allows for simpler application of the window. The only effect of the shift is to produce a similar shift in the output signal, which can be compensated later if required.

A number of questions are immediately raised regarding the proper implementation of the Blind Deconvolution method. These are summarized below.

1. What shape should be used for the desired pulse? What should its length be, and does it need to be aligned with the detected pulse? Does its peak need to be scaled to match the peak of the detected pulse?
2. Do the frequency spectra of the desired pulses need to be normalized in order to maintain the DC component?
3. How do you decide where to truncate the required filter kernel when applying the custom filter method?
4. How much does the required filter kernel need to be shifted when applying the custom filter method?

5. Are there restrictions regarding the type, size, and location of the window used when applying the custom filter method?

Some of these issues will be addressed in the following paragraphs.

In his gamma-ray detector example, Smith (1999) used a Blackman window as his desired pulse. This seems a reasonable starting point for the present example. He recommends that the width of the window be approximately 1/3 the length of the impulse event. For the present example the window width will be approximately ½ the length of the wave impact event. The effects of other shapes and window lengths will be examined later.

Figure G.3 shows the selected wave impact event (the detected pulse) plotted against two versions of the desired pulse, one shifted to match the detected pulse location and the other not shifted. The desired pulses have been padded with zeros to 2500 points to match the length of the detected pulse. This is required to facilitate the subsequent frequency domain division. Note that each desired pulse has been scaled to match the height of the wave impact. An attempt will be made without this scaling later.

The FFTs of the desired pulse signals are divided by the FFT of the detected pulse. This produces the required filter responses (RFR). The inverse FFT of each RFR produces the required filter kernels (rfk). These are shown in Figure G.4. Close examination reveals that the two signals are identical, but shifted relative to each other by the amount applied to the desired pulses. The Fourier transform process considers the signals to be periodic, allowing them wrap around. The shift in the desired pulse causes only a shift in phase, while not affecting the magnitude of the FFT.

The next step is to apply the so-called Custom Filter method, where the rfk is shifted, truncated, and windowed. Since the signals are identical, either can be selected for further processing. In this case it is recommended that the signal (rkf1) developed from the non-shifted desired pulse be used. The reason for this is that two steps are eliminated from the process – the initial shift to align the desired pulse with the detected pulse, and the shift required to facilitate windowing in the Custom Filter method.

The truncation of the required filter kernel is an arbitrary process, dependant on the nature of the filter kernel signal, and the length of the detected pulse signal. In this case the region from 15.180 to 23.364 seconds will be kept, and the remainder of the signal will be set to zero. Figure G.5 shows the truncated (and inherently shifted) rfk.

A Blackman window is than applied to the truncated rfk. It is centred on the peak of the filter kernel and has a length twice that of the non-zero portion of the truncated rfk, as shown in Figure G.6. It is the signal shifting that allows the windowing to be applied easily. The purpose of the windowing is to smooth the signal in regions of discontinuity, thereby reducing aliasing. The windowed rfk is shown in Figure G.7.

If the windowed rfk is then convolved with the original wave impact event vector (det_pul) the "actual force" vector is produced. This is shown in Figure G.8. Note that the convolved signal appears to be reasonable, but its main peak is shifted from that of the detected signal. This shift is comparable to that inherent in the rfk, Figure G.7. If the shift is removed from the rfk (Figure G.9) and the convolution reapplied, the peak of convolved signal now aligns with that of the detected pulse, see Figure G.10. The corrupted sections indicated are known as end effects and are a natural result of the convolution process. This will be discussed later.

So it has been shown that the Blind Deconvolution technique can produce results that appear reasonable, in that they remove high frequency inertial component from the measured signal. It has also been demonstrated that in its application, it is not necessary to align the peak of the desired pulse with the peak of the detected pulse, in fact there are advantages to not doing so.

If the peak desired pulse is not scaled to match that of detected pulse, the resulting convolved signal is reduced by that very amount, as shown in Figure G.11. Similarly, if the FFT of the desired pulse is normalized by its own initial value, prior to division with the detected pulse spectrum, it results in a convolved signal that is reduced by the same scale factor. The reason for suggesting this is that in the Inverse Fourier Transform method, it is necessary to normalize the frequency response in order to maintain the mean values between measured and actual forces. In summary, it can be said that as a rule it is best to scale the desired pulse peak to match the peak of the detected pulse, but it is not necessary to normalize the desired pulse spectrum by its own initial value.

As mentioned above, the truncation of the rfk can be somewhat arbitrary. The dilemma is this. A longer filter kernel will allow better reproduction of the frequency response, while increasing the length of the corruption due to end effects. The simple solution is to increase the length of the detected pulse so that the region of interest (the wave impact) is not affected by the corruption.

## G.3.0 End Effects

End effects are the corrupted sections that occur at the beginning and end of all convolved signals. When two signals of length M and N are convolved, the resulting signal has a length of M+N – 1. The first and last M-1 points of the convolved signal (where M is the length of the impulse response) are considered to be corrupted. According to Smith (1999) this is due to the fact that the M length impulse response is not "fully immersed" in the signal being convolved. He describes the convolution process this way. Each point in the output signal is the summation of the products of the M impulse response points, and N relevant input signal points. When calculating points at the beginning and end of the output signal, the impulse response signal tries to use (input signal) points that are outside the range of the input signal. Since this is impossible, the

M-1 points at the beginning and end of the output signal are developed using incomplete information.

Jackson (1989) suggests two techniques to deal with end effects in a convolution. These are known as the overlap-add and overlap-save techniques. In each, the input signal being convolved is divided into segments (either overlapping or non-overlapping depending on the method) which are then convolved with the impulse response. The resulting convolved segments are then overlapped or concatenated (as the case may be).

These methods were applied to some simple sinusoidal signals and compared to the results obtained using the Matlab convolution command (conv) and frequency domain multiplication. The results were identical, as shown in Figure G.12. This implies either that the overlap-add and overlap-save methods are not effective in this application, or that the MATLAB command already utilizes these or similar techniques. In either case, the beginning and end sections of a convolved signal should be disregarded.

## G.4.0 Effect of Desired Pulse Shape

Next the effect of desired pulse shape and length on the final convolved result will be examined. Figure G.13 shows desired pulse signals created using Blackman windows of different lengths. With the remainder of the procedure unchanged, the resulting convolved signals are presented in Figure G.14. The windows with lengths of 159 and

100 points result in similar signals. The result using the 50 point window is significantly different. Smith (1999) warned of this saying that if the desired pulse is made too narrow, the final result will not be satisfactory. Experimentation with window length is recommended.

Examination of the wave impact event (detected pulse in Figure G.2) shows that it consists of a region of positive force, followed immediately by a region of negative force. It would be of some interest to investigate the effect of using a desired pulse with similar characteristics. Figure G.15 shows such a signal. When applied to the detected pulse, the resulting convolved signal again appears reasonable, see Figure G.16. The Blind Deconvolution technique seems to be quite flexible, as long as the desired pulse vectors are not forced to extremes (i.e. made too narrow).

## G.5.0 Application of Blind Deconvolution to Decay Test Results

While investigating the previous methods (inverse Fourier transform and normal mode method), attempts have been made to retrieve the pulse signal (actual applied force) responsible for the measured signal from a decay test. It would be of some interest to apply the Blind Deconvolution to this type of signal.

Figure G.17 shows the measured decay test signal, which represents the detected pulse, and a rectangular pulse, which represents the desired pulse. The FFTs of these signals are obtained (Figure G.18), and divided to obtain the required frequency response (Figure G.19). The inverse FFT is performed to obtain the rfk (Figure G.20), which is then shifted, truncated, and windowed (Figures G.21 to G.23). The modified filter kernel is the unshifted (Figure G.24), and convolved with detected pulse to obtain the signal shown in Figure G.25. The convolved signal is subject to end effects, as shown in Figure G.26. In Figure G.27 the convolved and detected signals have been arbitrarily aligned by matching the location of the discontinuity.

The convolved result is comparable to that obtained using the inverse Fourier transform method. The transients observed at the discontinuities are caused by the truncation of the filter kernel (or impulse response) and are commonly referred to as Gibbs Effect, see Appendix I. These are unavoidable. Figure G.28 compares convolution results using three different filter kernel truncation lengths. Although the signals display different means in the offset region, the shape and magnitude of the transients are almost identical. This is a characteristic of Gibbs Effect. The amplitude of the transient will remain essentially unchanged for all levels of truncation.

Gibbs Effect was also observed in the inverse Fourier transform method, and is a significant hindrance to the usefulness of these methods to remove inertial loads, since it is frequently the discontinuity region that is of most interest to investigators.

As an aside, the similarities between the inverse Fourier transform method, and the Blind Deconvolution method show be pointed out. In the inverse Fourier transform method the system frequency response can be obtained by dividing the FFT of the measured force by the FFT of the actual force. The Blind Deconvolution method is the reciprocal of this, with the desired pulse FFT being divided by the detected pulse FFT to obtain the required filter response. These responses are then used in a deconvolution and a convolution process, respectively.

## G.6.0 Summary

In summary it can be said that the Blind Deconvolution technique can be considered another useful tool in the determination of actual force signals, from measured signals corrupted by measurement system dynamics. The method does, however, require experimentation with the selection of desired pulse shapes, the application of the Custom Filter method, and consideration of Gibbs effect.

Table G.1 summarizes the basic steps required in the application of the Blind Deconvolution technique.

Table G.1 – Procedure to Apply the Blind Deconvolution Method

| Step | Description |
|---|---|
| 1 | Determine the detected pulse. For the case of wave impact this will be the measured force. |
| 2 | Generate the desired pulse. This represents a pulse that has not been corrupted by unwanted convolution. The exact shape is a matter of trial and error. |
| 3 | Take the FFT of both these signals. |
| 4 | Divide the FFT of the desired pulse by the FFT of the detected pulse to obtain the required filter response. |
| 5 | Take the inverse FFT of the required filter response to obtain the required filter kernel. |
| 6 | Modify the required filter kernel (i.e. shift, truncate, and window). |
| 7 | Apply the modified filter kernel in convolution with other detected pulses to obtain the actual force signals. |

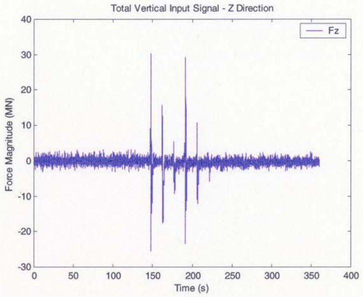

**Figure G.1 Total Z Direction Wave Impact – Jacket A**

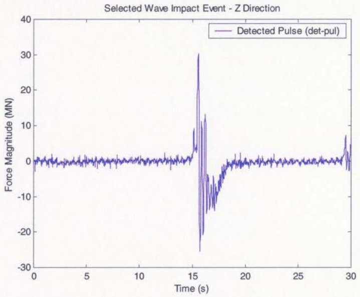

**Figure G.2 Selected Z Direction Wave Impact Event – Jacket A**

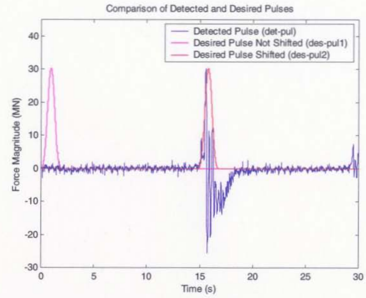

**Figure G.3 Comparison of Detected and Desired Pulses**

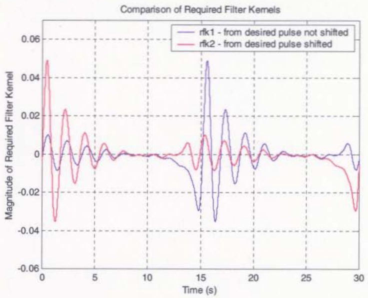

**Figure G.4 Comparison of Required Filter Kernels**

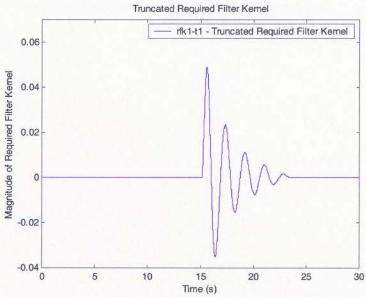

**Figure G.5 Truncated Required Filter Kernel**

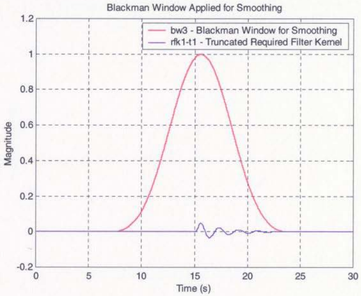

**Figure G.6 Blackman Window Used for Smoothing**

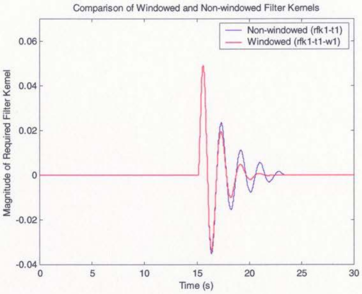

**Figure G.7 Smoothing Effect of Blackman Window**

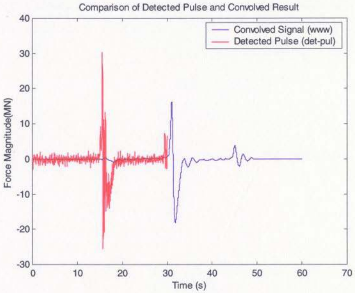

**Figure G.8 Comparison of Detected Pulse and Convolved Result**

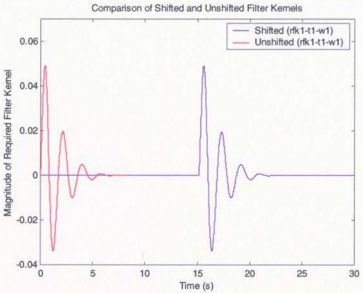

**Figure G.9 Comparison of Shifted and Unshifted Filter Kernels**

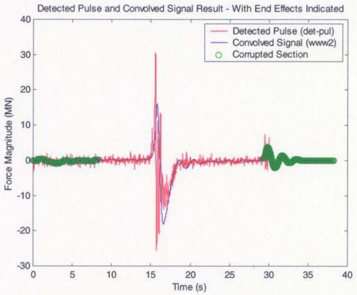

**Figure G.10 Detected Pulse and Convolved Signal – With End Effects**

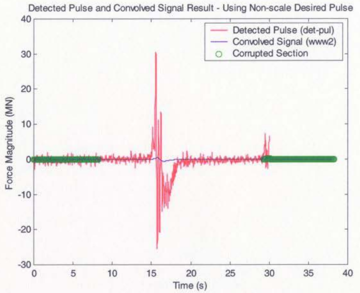

**Figure G.11 Convolved Signal – Using Non-scaled Desired Pulse**

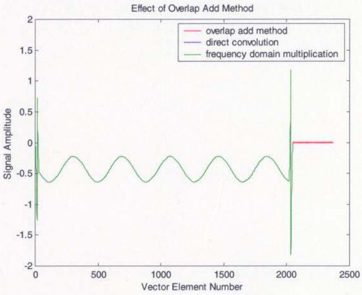

**Figure G.12 Effect of Overlap Add Method**

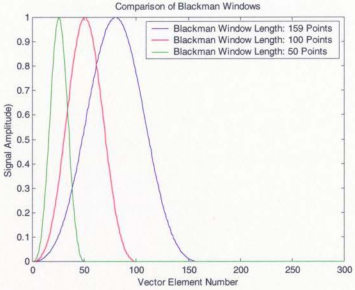

**Figure G.13 Comparison of Blackman Windows**

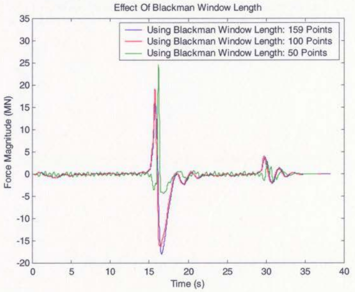

**Figure G.14 Effect of Blackman Window Length**

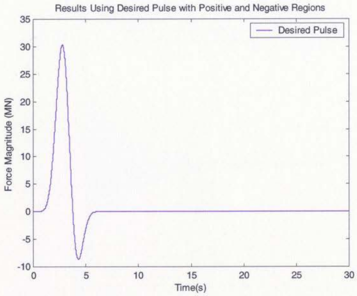

**Figure G.15 Desired Pulse with Positive and Negative Regions**

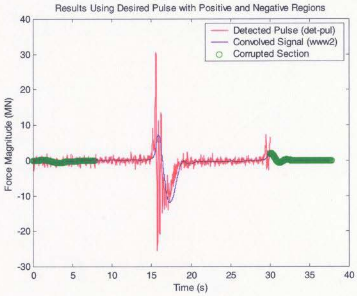

**Figure G.16 Effect of Desired Pulse with Positive and Negative Regions**

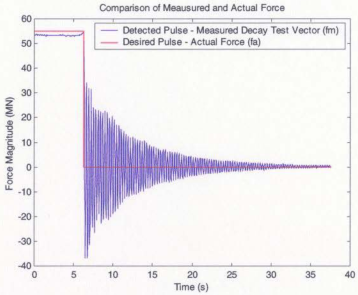

**Figure G.17 Desired Pulse for Measured Decay Force**

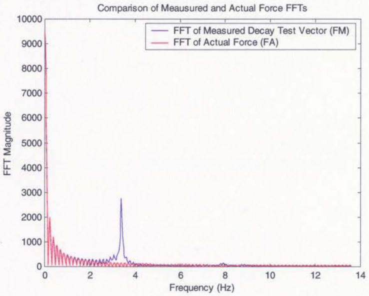

**Figure G.18 FFT of Desired Pulse for Measured Decay Force**

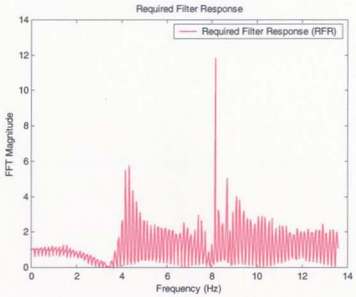

**Figure G.19 Required Filter Resp. for Desired Pulse for Decay Force**

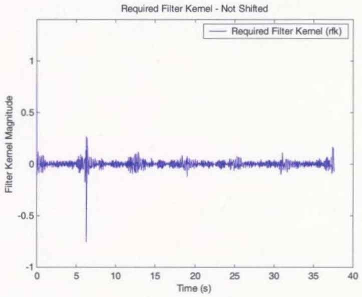

**Figure G.20 Required Filter Kernel from Desired Pulse for Decay Force**

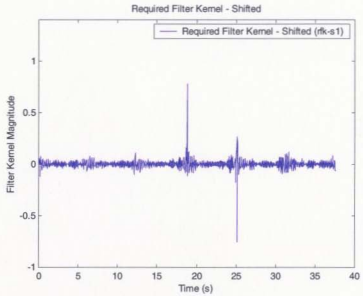

**Figure G.21 Required Filter Kernel - Shifted**

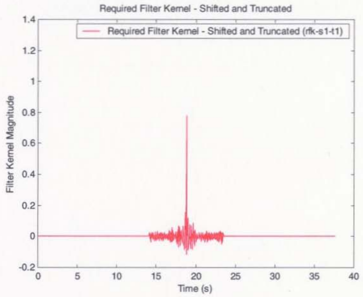

**Figure G.22 Required Filter Kernel –Truncated**

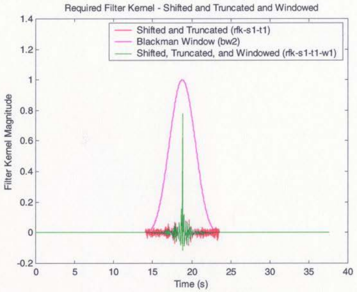

**Figure G.23 Required Filter Kernel –Windowed**

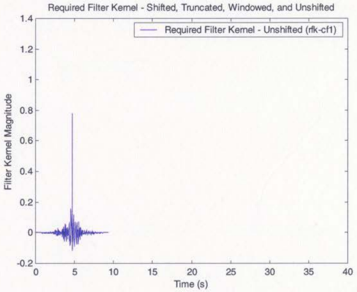

**Figure G.24 Required Filter Kernel – Unshifted**

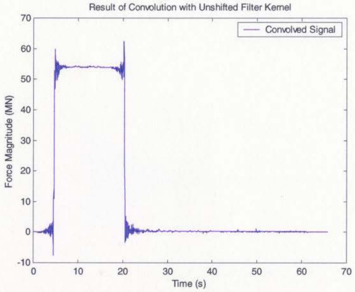

**Figure G.25 Resulted of Convolution with Unshifted Filter Kernel**

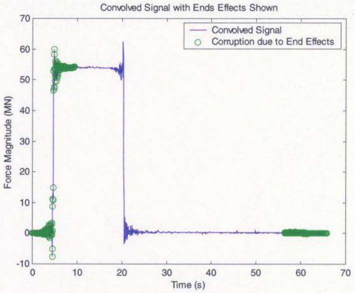

**Figure G.26 Convolved Signal With End Effects Shown**

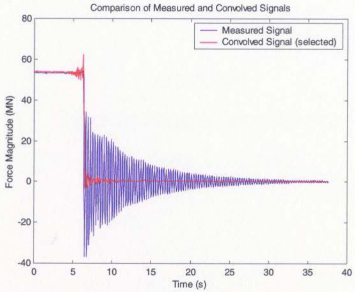

**Figure G.27 Comparison of Measured and Convolved Signals**

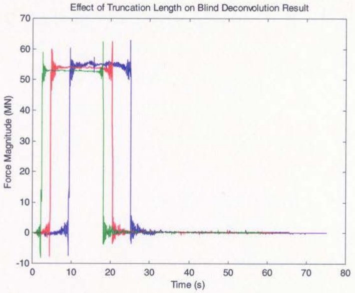

**Figure G.28 Effect of Truncation Length on Blind Deconvolution Result**

# Appendix H

## Discussion of Modifications Required to Facilitate Deconvolution

## H.1.0 Modifications to Facilitate Deconvolution

To illustrate the limitations of the convolve and deconvolve commands, they will be applied to some arbitrary simulated signals. Figure H.1 shows an 81 point sine wave, b[n], which will serve as the signal to be convolved. Figure H.2 shows an arbitrary 17 point signal, a[n], which will serve as the impulse response. Using the MATLAB command *"c=conv(a,b)"*, the convolved signal c[n] is produced, see Figure H.3. Note that this signal shows some resemblance to the original signal b[n], in that the characteristics of a sine wave are evident, but the first and last 17 points of the signal are corrupted. This is an expected result of the convolution process, known as end effects

(see Appendix G). The first and last *N* points (where *N* is the length of the "impulse response") of the convolved signal are unusable. The signal a[n] is deemed to be not fully immersed in the signal b[n] in those regions. If signal c[n] is now deconvolved with signal a[n], it should produce a signal that is identical to b[n]. Figure H.4 shows a comparison of q[n] and b[n] for this case. The two signals are identical, indicating that the convolution and deconvolution process was successful.

The entire process, though, is sensitive to the element values in the impulse response vector. If value of the first point of vector a[n] is changed to 0.0, the deconvolution procedure will not work. The error message produced indicates that the first coefficient of a[n] must be non-zero. If the value of the first point is iterated from 0.0, back towards the original value of 0.2, it is discovered that the convolution/deconvolution process is not correct until a value of at least 0.16 is reached. For example, using a value of a[1]=0.14, results in a q[n] signal that differs slightly from b[n] as shown in Figure H.5. Using a value of a[1]=0.10 results in the signal shown in Figure H.6.

The value of the first point of any vector a[n] can be altered to some threshold value where the deconvolve command (*deconv*) no longer works. In fact, this applies to all the points in the a[n] signal. To demonstrate this, an arbitrary a[n] 17 element signal composed entirely of points with value 0.1, was convolved and deconvolved with the 81point sine wave, shown in Figure H.1. Then the value of each point in a[n] was altered, iteratively, until the deconvolved signal no longer matched the original sine wave. This experiment indicates a trend towards larger magnitude changes (in the point value), for increasing vector element number. Figure H.7 illustrates this. The same trend can be generated for any arbitrary a[n] signal.

The only practical use for this knowledge is in a situation as described above, where the deconvolution of two measured signals produces a result that does not resemble that expected, as per Figure 5.17. A single point in the measured a[n] signal can be altered, until the deconvolution process works. The best point to select for alteration is the first since, in theory, it will require the smallest modification to facilitate the deconvolution.

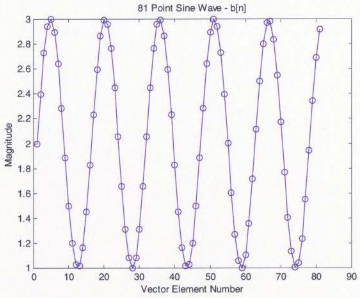

**Figure H.1 Arbitrary 81 Point Sine Wave**

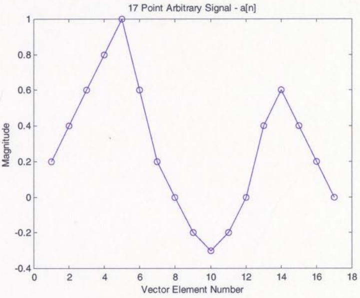

**Figure H.2 Arbitrary 17 Point Signal**

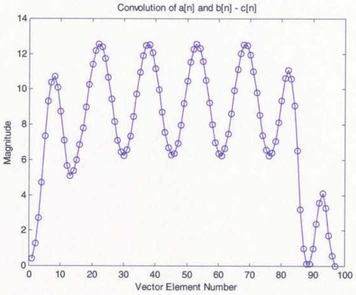

**Figure H.3 Convolution of 81 Point and 17 Point Signals**

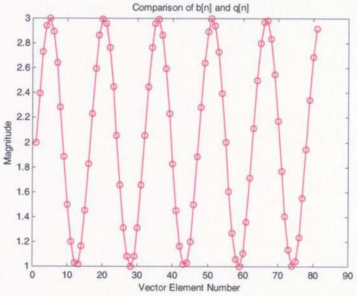

**Figure H.4 Comparison of Deconvolved and Original Signals**

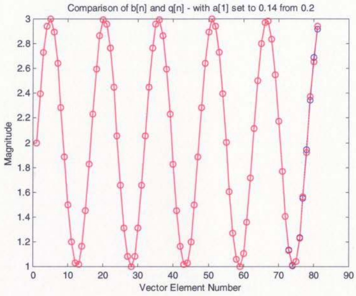

**Figure H.5 Effect of Inadequate Signal Modification**

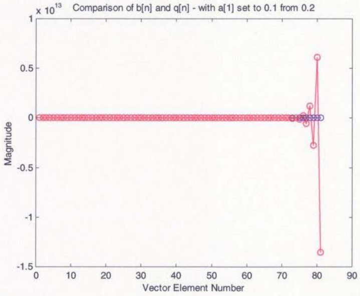

**Figure H.6 Effect of Inadequate Signal Modification (Example 2)**

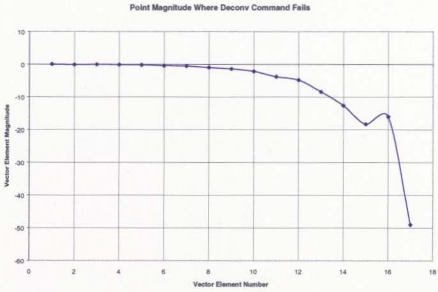

**Figure H.7 Effect of Vector Element Choice on Signal Modification**

# Appendix I

## Discussion of the Mitigation of Gibbs Effect

## I.1.0 Mitigation of Gibbs Effect

Gibbs effect is manifested as an overshoot or decaying oscillation observed at the discontinuity of a time series. It is caused by the truncation or removal of frequencies from a discrete signal in the frequency domain. Increasing the amount of truncation (i.e. removing more frequencies) causes the width of the oscillation to increase. The amplitude of the oscillation stays roughly the same, at approximately 9 percent.

Figures I.1 to I.5 illustrate the effect of this truncation. A time domain signal of a rectangular pulse is shown. In the frequency domain, this will be represented by a sinc function. After removing a range of frequencies, the time domain signal displays the overshoot and oscillation characteristic of Gibbs effect. This phenomenon occurs in low pass filtering, where a range high frequency components is eliminated. Similarly, in the inverse Fourier transform method, a band of frequencies in the resonance range is suppressed, leading to ringing at the discontinuities.

This phenomenon causes considerable confusion when examining wave impact signals that have been processed to remove inertial load either through filtering or use of the inverse Fourier transform method. It is extremely difficult, if not impossible, to say with certainty whether, or to what extent, the resulting signal has been corrupted by Gibbs effect.

The phenomenon is unavoidable. It is a fact of life when dealing with discrete signals. Several authors have discussed the use of windows and filters to suppress Gibbs effect commonly seen in truncated Fourier series. These will be mentioned here, but they will not be implemented or evaluated. This is a topic that could certainly use further examination.

Walker (1991) discusses the use of Cesaro and de la Vallee Poussin filters in the suppression of Gibbs phenomenon. Hamming (1983) recommends the use of the Lanczos window over the Cesaro, claiming it has a lesser effect on the rise time of the signal. Terrel (1980) and Smith (1999) recommend the use of windows such as Hamming, Blackman, and hanning to mitigate Gibbs effect, though this is at the expense of rise time.

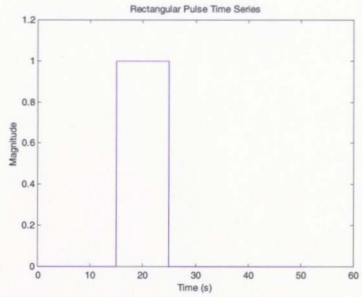

**Figure I.1 Rectangular Pulse Time Series**

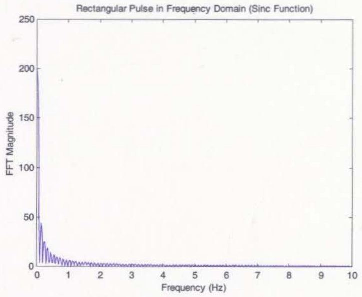

**Figure I.2 FFT of Rectangular Pulse**

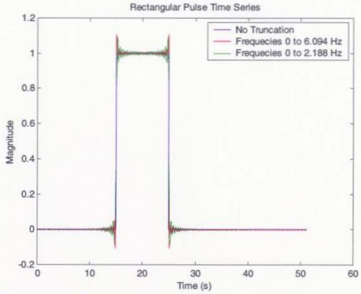

**Figure I.3 Effect of Pulse Truncation in Frequency Domain**

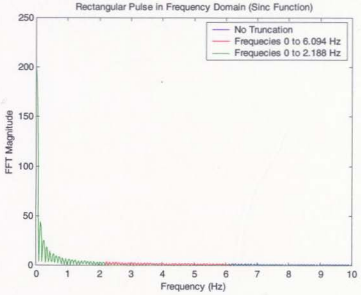

**Figure I.4 Truncated Pulses in Frequency Domain**

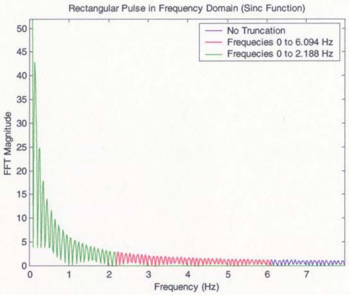

**Figure I.5 Truncated Pulses in Frequency Domain - Selected**

# Appendix J

# Generation of Simulated Decay Signals

## J.1.0 Generation of Simulated Decay Signals

To aid the evaluation of the normal mode approach in its ability to remove inertial forces from measured signals, a set of simulated decay time traces was generated. These signals were designed to represent measured decay test forces for a jacket structure model, such as that described in Section 3.0. It was assumed that input forces were applied one degree-of-freedom at a time, and that crosstalk did not occur. This is, if a force was applied in the X direction, then the resultant force would be measured only in the X direction.

The development of the simulated signals was based on the second-order differential equation of motion, as shown in Equation J.1.

$$[M]\{\ddot{x}(t)\}+[C]\{\dot{x}(t)\}+[K]\{x(t)\}=\{F(t)\}$$

where,

**J.1**

| | |
|---|---|
| $[M]$ | mass matrix, |
| $[C]$ | damping coefficient matrix, |
| $[K]$ | stiffness matrix, |
| $\{\ddot{x}(t)\}$ | acceleration vector, |
| $\{\dot{x}(t)\}$ | velocity vector, |
| $\{x(t)\}$ | displacement vector. |

This can be rearranged to the following form.

$$\{\ddot{x}(t)\} = -\frac{[C]}{[M]}\{\dot{x}(t)\} - \frac{[K]}{[M]}\{x(t)\} + \frac{1}{[M]}\{F(t)\} \qquad \textbf{J.2}$$

This differential equation can be solved using the Matlab solver *ODE45.m*. Implementation requires that Equation J.2 be expressed as a set of first order differential equations. For simplicity this will illustrated for a single degree-of-freedom system.

Let

$$\begin{aligned} u_1(t) &= x(t) \qquad \textbf{J.3} \\ u_2(t) &= \dot{x}(t) \\ \dot{u}_1(t) &= u_2(t) \\ \dot{u}_2(t) &= -\frac{C}{M}u_2(t) - \frac{K}{M}u_1(t) + \frac{1}{M}F(t) \end{aligned}$$

The output from ODE45 for this simplified case would be vectors for velocity, displacement, and time. For the more complicated six degree-of-freedom system, six velocity vectors, and six displacement vectors, would be produced along with the time vector. This has been implemented in Matlab scripts *run_dof6_sim3.m* and *dof6_sim3.m*, shown in Appendix K.

The inputs for this program include the mass, stiffness, and damping matrices. The stiffness matrix is a by-product of the normal mode method, implemented in script *normal_mode_v3.m* (which is a six degree-of-freedom version of *normal_mode_v3_3dof.m*). That program does not use the stiffness matrix explicitly, but can be generated using Equation J.4.

$$[K]=[F]_{\text{measured decay}}\left[\tilde{X}\right]^{-1} \qquad \textbf{J.4}$$

where,

| | |
|---|---|
| $[K]$ | stiffness matrix |
| $[F]_{\text{measured decay}}$ | force amplitudes from measured decay tests |
| $\left[\tilde{X}\right]$ | non-normalized eigenvectors |

The damping matrix is determined using the Rayleigh Damping method (see Appendix F), using the stiffness matrix as generated above.

## J.2.0 Sample Implementation

Script *run_dof6_sim3.m* requires the input of mass, stiffness, and damping matrices. Examples of these are shown in Table J.1. The value of the applied pulse force amplitude, its location and duration in the time series are input interactively. The system initial conditions are hard-coded in the script, and can be altered as appropriate. Figures J.1 to J.6 show sample output signals. These signals have been curve-fit using a cubic spline in order to obtain uniform time spacing. ODE45 produces signals with non-uniform time spacing. The curve fitting was done using script *spline_fit.m* shown in Appendix K.

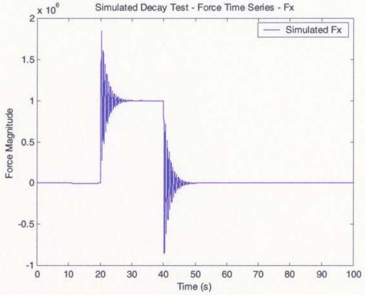

**Figure J.1 Simulated Decay Force Time Series – X Direction**

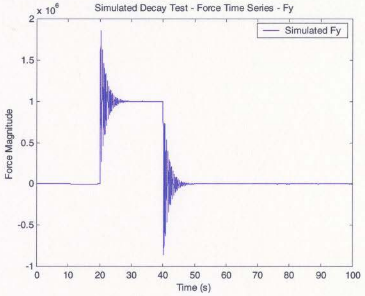

**Figure J.2 Simulated Decay Force Time Series – Y Direction**

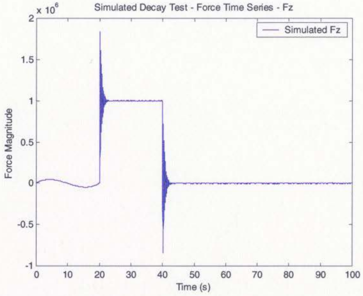

**Figure J.3 Simulated Decay Force Time Series – Z Direction**

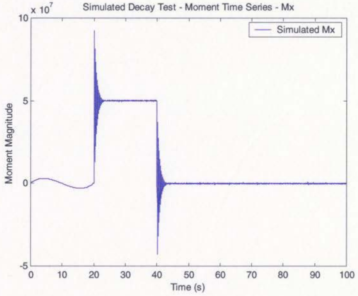

**Figure J.4 Simulated Decay Moment Time Series – About X**

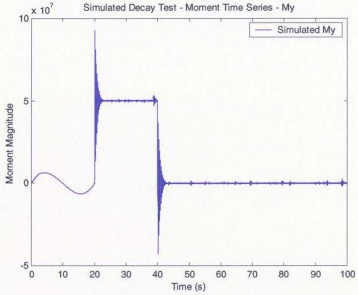

**Figure J.5 Simulated Decay Moment Time Series – About Y**

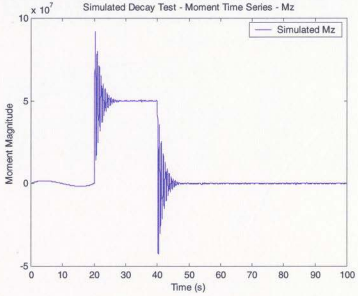

**Figure J.6 Simulated Decay Moment Time Series – About Z**

Table J.1 – Mass, Damping, Stiffness Matrices Used to Generate Simulated Decay Time Series

Mass Matrix

| | Mode 1 | Mode 2 | Mode 3 | Mode 4 | Mode 5 | Mode 6 |
|---|---|---|---|---|---|---|
| kg | 5.53E+06 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| kg | 0.00 | 5.53E+06 | 0.00 | 0.00 | 0.00 | 0.00 |
| kg | 0.00 | 0.00 | 5.53E+06 | 0.00 | 0.00 | 0.00 |
| kg -m$^2$ | 0.00 | 0.00 | 0.00 | 8.45E+08 | 0.00 | 0.00 |
| kg -m$^2$ | 0.00 | 0.00 | 0.00 | 0.00 | 1.34E+09 | 0.00 |
| kg -m$^2$ | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 2.18E+09 |

Damping Matrix

| | Mode 1 | Mode 2 | Mode 3 | Mode 4 | Mode 5 | Mode 6 |
|---|---|---|---|---|---|---|
| kg/s | 1.19E+07 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| kg/s | 0.00 | 1.25E+07 | 0.00 | 0.00 | 0.00 | 0.00 |
| kg/s | 0.00 | 0.00 | 3.64E+07 | 0.00 | 0.00 | 0.00 |
| kg-m/s | 0.00 | 0.00 | 0.00 | 4.26E+09 | 0.00 | 0.00 |
| kg-m/s | 0.00 | 0.00 | 0.00 | 0.00 | 7.02E+09 | 0.00 |
| kg-m/s | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 4.53E+09 |

Stiffness Matrix

| | Mode 1 | Mode 2 | Mode 3 | Mode 4 | Mode 5 | Mode 6 |
|---|---|---|---|---|---|---|
| N/m | 2.58E+09 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| N/m | 0.00 | 3.14E+09 | 0.00 | 0.00 | 0.00 | 0.00 |
| N/m | 0.00 | 0.00 | 2.42E+10 | 0.00 | 0.00 | 0.00 |
| N-m/m | 0.00 | 0.00 | 0.00 | 2.55E+12 | 0.00 | 0.00 |
| N-m/m | 0.00 | 0.00 | 0.00 | 0.00 | 4.27E+12 | 0.00 |
| N-m/m | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 8.80E+11 |

# Appendix K

## Matlab Scripts

```
%.........................................................................................
%
%
%       Program:            Normal_mode_v3_3dof.m
%       Developed by:       Fraser Winsor
%
%
%       Removes inertial force from the measured input signals. Damping
%       is based on Rayleigh damping method. This version considers only three
%       degrees-of-freedom.
%
%
%.........................................................................................
%
%
%       Input mass matrix, measured decay force vectors, and
%       the vector of natural frequencies.
%
%       m                   mass matrix
%       fmd                 force vector from the measured decay tests
%       w                   natural frequency vector value for the DOF of interest
%
%       h              time step (full scale)
%
%
[filename,pathname] = uigetfile('*.dat','Input Mass Matrix',50,50);
filename=lower(filename);
eval(['load ',[pathname,filename],';'])
f=findstr(filename,'.');
m=eval(filename(1:f(1)-1));
%
[filename,pathname] = uigetfile('*.dat','Input Measured Decay Force Vector',50,50);
filename=lower(filename);
eval(['load ',[pathname,filename],';'])
f=findstr(filename,'.');
fmd=eval(filename(1:f(1)-1));
%
[filename,pathname] = uigetfile('*.dat','Input Natural Frequency Vector',50,50);
filename=lower(filename);
eval(['load ',[pathname,filename],';'])
f=findstr(filename,'.');
w=eval(filename(1:f(1)-1));
%
```

```
h=input('Input time step (default = 0.036742)');
if isempty(h)
  h=0.036742;
end
%
%
%       Determine the non-normalized eigenvectors
%
%       phi_nn          non-normalized eigenvector
%
%
for j=1:3
    phi_nn(:,j)= (1/w(j)^2)*inv(m)*fmd(:,j);
end
phi_nn
%
%
%       Produce the coefficients that normalize the
%       eigenvectors. (non_norm_vector = c * norm_vector)
%
%       c(i)            scalar normalizing coefficient
%       phi_nn'transpose of non-normalized eigenvector
%
%
for i=1:3
  c(i)=sqrt(phi_nn(:,i)' * m * phi_nn(:,i));
end
%
%
%       Normalize eigenvectors.
%
%       phi                        normalized eigenvector
%
%
for i=1:3
  phi(:,i)=phi_nn(:,i)*(1/c(i));
end
%
%
%       Input the measured force vectors.
%
%       fm              Measured force vectors
%
%
```

```
[filename.pathname] = uigetfile('*.dat'.'Input Measured FX'.50.50);
filename=lower(filename);
eval(['load '.[pathname.filename].';'])
f=findstr(filename.'.');
fm(1.:)=eval(filename(1:f(1)-1))';
%
%
%       Evaluate the size of the input measured FX vector
%       and preallocate space for the entire vector - then
%       recalculate the first row values.
%
%
[fm_r_size.fm_c_size]=size(fm);
fm=zeros(3.fm_c_size);
fm(1.:)=eval(filename(1:f(1)-1))';
%
%
[filename.pathname] = uigetfile('*.dat'.'Input Measured FY'.50.50);
filename=lower(filename);
eval(['load '.[pathname.filename].';'])
f=findstr(filename.'.');
fm(2.:)=eval(filename(1:f(1)-1))';
%
[filename.pathname] = uigetfile('*.dat'.'Input Measured FZ'.50.50);
filename=lower(filename);
eval(['load '.[pathname.filename].';'])
f=findstr(filename.'.');
fm(3.:)=eval(filename(1:f(1)-1))';
%
%       Generates the displacements in the normalized coodinate system.
%
%       z               displacements in normalized coordivate system.
%
%
z=zeros(3.fm_c_size);
for t=1:fm_c_size
for i=1:3
  z(i.t)=(1/w(i)^2)*phi(:.i)'*fm(:.t);
end
end
%
%
%       Differentiates the diplacements in the normalized coordinate
%       system. producing accelerations in the normalized coordinate system.
```

```
%
%       The first and last points are calculated using the Forward-Difference,
%       and Backward Difference expressions, respectively. The remainder are
%       calculated using the Central-Difference expression. [Applied Numerical
%       Methods for Digital Computation   James, Smith, Wolford pp 343-348].
%
%       zdd     second derivative of the displacement in the normalized
%                       coordinate system (i.e. acceleration).
%
%
%
%
%       This section used to get velocity vectors needed for calculation of damping force.
%
%
zd=zeros(3,fm_c_size);
for i=1:3
  zd(i,1)=(-z(i,3)+4*z(i,2)-3*z(i,1))/(2*h);
  zd(i,2)=(-z(i,4)+4*z(i,3)-3*z(i,2))/(2*h);
%
        zd(i,fm_c_size)=(3*z(i,fm_c_size)-4*z(i,fm_c_size-1)+z(i,fm_c_size-2))/(2*h);
        zd(i,fm_c_size-1)=(3*z(i,fm_c_size-1)-4*z(i,fm_c_size-2)+z(i,fm_c_size-
3))/(2*h);
end
%
for t=3:(fm_c_size-2)
  for i=1:3
     zd(i,t)=(-z(i,t+2)+8*z(i,t+1)-8*z(i,t-1)+z(i,t-2))/(12*h);
  end
end
%
%
%       This section to get acceleration needed for inertial force calculation.
%
%
zdd=zeros(3,fm_c_size);
for i=1:3

  zdd(i,1)=(-z(i,4)+4*z(i,3)-5*z(i,2)+2*z(i,1))/(h^2);
  zdd(i,2)=(-z(i,5)+4*z(i,4)-5*z(i,3)+2*z(i,2))/(h^2);
  zdd(i,fm_c_size-1)=(2*z(i,fm_c_size-1)-5*z(i,fm_c_size-2)+4*z(i,fm_c_size-3)-
z(i,fm_c_size-4))/(h^2);
  zdd(i,fm_c_size)=(2*z(i,fm_c_size)-5*z(i,fm_c_size-1)+4*z(i,fm_c_size-2)-
z(i,fm_c_size-3))/(h^2);
```

```
end
for t=3:(fm_c_size-2)
  for i=1:3
   % zdd(i,t)=(z(i,t+1)-2*z(i,t)+z(i,t-1))/(h^2);
    zdd(i,t)=(-z(i,t+2)+16*z(i,t+1)-30*z(i,t)+16*z(i,t-1)-z(i,t-2))/(12*h^2);
  end
end
%
%
%      Converts the accelerations from the normalized coordinate
%      system to the physical coordinate system.
%
%      xdd     acceleration in the physical coordinate system.
%
%
%
xd=zeros(3,fm_c_size);
xd=phi*zd;
%
xdd=zeros(3,fm_c_size);
xdd=phi*zdd;
%
%
%
%      Determines the inertial force using the acceleration
%      in the physical coordinate system.
%
%      fi              inertial force vector
%
%
fi=zeros(3,fm_c_size);
fi=-1*m*xdd;
%
%
%      Generate stiffness matrix
%
%
k=fmd*inv(phi_nn);
%
%
%      Generate damping matrix using Rayleigh Method
%
%
damp_ratio=input(' Enter damping ratio (i.e. 0.05): ');
```

```
%
w  %type up frequencies for reference
%
omega_m=input(' Enter fundemental frequency (rad/s): ');
omega_n=input(' Enter high frequency (rad/s): ');
%
freq_vec = [(omega_m*omega_n); 1]; % vector containing frequencies
%
a = [(2*damp_ratio)/(omega_m + omega_n)]*freq_vec; % proportionality constants
%
C = a(1)*m + a(2)*k;
%
%
%       Determines the damping force using the velocity
%       in the physical coordinate system.
%
%       fd                  damping force vector
%
%
fd=zeros(3,fm_c_size);
fd=-1*C*xd;
%
%
%       Determine real force vectors
%
%       ft                  real force vector. (ft = fm - fi-fd)
%
%
ft = zeros(3,fm_c_size);
ft = fm-fi-fd;
%
%
%       Calculate time vector (be to used for plotting)
%
%
t_ft = zeros(1,fm_c_size);
t_ft=0:h:(h*(fm_c_size-1));
%
%
```

```
%.............................................................................................
%
%
%       Program:                Normal_mode_v3_3dof_nodamping.m
%       Developed by:           F. Winsor
%
%
%       Removes inertial force from the measured input signals. Damping
%       is assumed to be negligible. This version considers only three
%       degrees-of-freedom.
%
%
%.............................................................................................
%
%
%       Input mass matrix, measured decay force vectors, and
%       the vector of natural frequencies.
%
%       m                       mass matrix
%       fmd                     force vector from the measured decay tests
%       w                       natural frequency vector value for the DOF of interest
%
%       h               time step (full scale)
%
%
[filename,pathname] = uigetfile('*.dat','Input Mass Matrix',50,50);
filename=lower(filename);
eval(['load ',[pathname,filename],';'])
f=findstr(filename,'.');
m=eval(filename(1:f(1)-1));
%
[filename,pathname] = uigetfile('*.dat','Input Measured Decay Force Vector',50,50);
filename=lower(filename);
eval(['load ',[pathname,filename],';'])
f=findstr(filename,'.');
fmd=eval(filename(1:f(1)-1));
%
[filename,pathname] = uigetfile('*.dat','Input Natural Frequency Vector',50,50);
filename=lower(filename);
eval(['load ',[pathname,filename],';'])
f=findstr(filename,'.');
w=eval(filename(1:f(1)-1));
%
h=input('Input time step (default = 0.036742)');
```

```
if isempty(h)
  h=0.036742;
end
%
%
%       Determine the non-normalized eigenvectors
%
%       phi_nn          non-normalized eigenvector
%
%
for j=1:3
    phi_nn(:,j)= (1/w(j)^2)*inv(m)*fmd(:,j);
end
phi_nn
%
%
%       Produce the coefficients that normalize the
%       eigenvectors. (non_norm_vector = c * norm_vector)
%
%       c(i)            scalar normalizing coefficient
%       phi_nn'transpose of non-normalized eigenvector
%
%
for i=1:3
  c(i)=sqrt(phi_nn(:,i)' * m * phi_nn(:,i));
end
%
%
%       Normalize eigenvectors.
%
%       phi                     normalized eigenvector
%
%
for i=1:3
  phi(:,i)=phi_nn(:,i)*(1/c(i));
end
%
%
%       Input the measured force vectors.
%
%       fm              Measured force vectors
%
%
[filename,pathname] = uigetfile('*.dat','Input Measured FX',50,50);
```

```
filename=lower(filename);
eval(['load ',[pathname,filename],';'])
f=findstr(filename,'.');
fm(1,:)=eval(filename(1:f(1)-1))';
%
%
%       Evaluate the size of the input measured FX vector
%       and preallocate space for the entire vector - then
%       recalculate the first row values.
%
%
[fm_r_size,fm_c_size]=size(fm);
fm=zeros(3,fm_c_size);
fm(1,:)=eval(filename(1:f(1)-1))';
%
%
[filename,pathname] = uigetfile('*.dat','Input Measured FY',50,50);
filename=lower(filename);
eval(['load ',[pathname,filename],';'])
f=findstr(filename,'.');
fm(2,:)=eval(filename(1:f(1)-1))';
%
[filename,pathname] = uigetfile('*.dat','Input Measured FZ',50,50);
filename=lower(filename);
eval(['load ',[pathname,filename],';'])
f=findstr(filename,'.');
fm(3,:)=eval(filename(1:f(1)-1))';
%
%
%       Generates the displacements in the normalized coodinate system.
%
%       z               displacements in normalized coordivate system.
%
%
z=zeros(3,fm_c_size);
for t=1:fm_c_size
for i=1:3
  z(i,t)=(1/w(i)^2)*phi(:,i)'*fm(:,t);
end
end
%
%
%       Differentiates the diplacements in the normalized coordinate
%       system, producing accelerations in the normalized coordinate system.
```

```
%
%       The first and last points are calculated using the Forward-Difference,
%       and Backward Difference expressions, respectively. The remainder are
%       calculated using the Central-Difference expression. [Applied Numerical
%       Methods for Digital Computation  James, Smith, Wolford pp 343-348].
%
%       zdd     second derivative of the displacement in the normalized
%                       coordinate system (i.e. acceleration).
%
%
%
%       This section to get acceleration needed for inertial force calculation.
%
%
zdd=zeros(3,fm_c_size);
for i=1:3
  %
  zdd(i,1)=(-z(i,4)+4*z(i,3)-5*z(i,2)+2*z(i,1))/(h^2);
  zdd(i,2)=(-z(i,5)+4*z(i,4)-5*z(i,3)+2*z(i,2))/(h^2);
  %
  zdd(i,fm_c_size-1)=(2*z(i,fm_c_size-1)-5*z(i,fm_c_size-2)+4*z(i,fm_c_size-3)-
z(i,fm_c_size-4))/(h^2);
  zdd(i,fm_c_size)=(2*z(i,fm_c_size)-5*z(i,fm_c_size-1)+4*z(i,fm_c_size-2)-
z(i,fm_c_size-3))/(h^2);
end
for t=3:(fm_c_size-2)
  for i=1:3
    zdd(i,t)=(-z(i,t+2)+16*z(i,t+1)-30*z(i,t)+16*z(i,t-1)-z(i,t-2))/(12*h^2);
  end
end
%
%
%       Converts the accelerations from the normalized coordinate
%       system to the physical coordinate system.
%
%       xdd     acceleration in the physical coordinate system.
%
%
xdd=zeros(3,fm_c_size);
xdd=phi*zdd;
%
%
%       Determines the inertial force using the acceleration
%       in the physical coordinate system.
```

```
%
%       fi              inertial force vector
%
%
fi=zeros(3.fm_c_size);
fi=-1*m*xdd:
%
%
%       Determine real force vectors
%
%       ft              real force vector. (ft = fm - fi-fd)
%
%
ft = zeros(3.fm_c_size):
ft = fm-fi:
%
%
%       Calculate time vector (be to used for plotting)
%
%
t_ft = zeros(1.fm_c_size):
t_ft=0:h:(h*(fm_c_size-1)):
%
%
```

```
%...........................................................................................
%
%
%       Program:                Normal_mode_v2_remez.m
%       Developed by:           F. Winsor
%
%
%       Removes inertial force from the measured input signals. Damping
%       is assumed to be negligible. This version considers only three
%       degrees-of-freedom. The numerical differentiation is done using the
%       Matlab remez.m command.
%
%...........................................................................................
%
%
%       Input mass matrix, measured decay force vectors, and
%       the vector of natural frequencies.
%
%       m                       mass matrix
%       fmd                     force vector from the measured decay tests
%       w                       natural frequency vector value for the DOF of interest
%
%       h                       time step (full scale)
%
%
[filename,pathname] = uigetfile('*.dat','Input Mass Matrix',50,50);
filename=lower(filename);
eval(['load ',[pathname,filename],';'])
f=findstr(filename,'.');
m=eval(filename(1:f(1)-1));
%
[filename,pathname] = uigetfile('*.dat','Input Measured Decay Force Vector',50,50);
filename=lower(filename);
eval(['load ',[pathname,filename],';'])
f=findstr(filename,'.');
fmd=eval(filename(1:f(1)-1));
%
[filename,pathname] = uigetfile('*.dat','Input Natural Frequency Vector',50,50);
filename=lower(filename);
eval(['load ',[pathname,filename],';'])
f=findstr(filename,'.');
w=eval(filename(1:f(1)-1));
%
h=input('Input time step (default = 0.036742)');
```

```
if isempty(h)
  h=0.036742;
end
%
%
%       Determine the non-normalized eigenvectors
%
%       phi_nn          non-normalized eigenvector
%
%
for j=1:6
    phi_nn(:,j)= (1/w(j)^2)*inv(m)*fmd(:,j);
end
phi_nn
%
%
%       Produce the coefficients that normalize the
%       eigenvectors. (non_norm_vector = c * norm_vector)
%
%       c(i)            scalar normalizing coefficient
%       phi_nn'transpose of non-normalized eigenvector
%
%
for i=1:6
  c(i)=sqrt(phi_nn(:,i)' * m * phi_nn(:,i));
end
%
%
%       Normalize eigenvectors.
%
%       phi                     normalized eigenvector
%
%
for i=1:6
  phi(:,i)=phi_nn(:,i)*(1/c(i));
end
%
%
%       Input the measured force vectors.
%
%       fm              Measured force vectors
%
%
[filename,pathname] = uigetfile('*.dat','Input Measured FX',50,50);
```

```
filename=lower(filename);
eval(['load ',[pathname,filename],';'])
f=findstr(filename,'.');
fm(1,:)=eval(filename(1:f(1)-1))';
%
%
%       Evaluate the size of the input measured FX vector
%       and preallocate space for the entire vector - then
%       recalculate the first row values.
%
%
[fm_r_size,fm_c_size]=size(fm);
fm=zeros(6,fm_c_size);
fm(1,:)=eval(filename(1:f(1)-1))';
%
%
[filename,pathname] = uigetfile('*.dat','Input Measured FY',50,50);
filename=lower(filename);
eval(['load ',[pathname,filename],';'])
f=findstr(filename,'.');
fm(2,:)=eval(filename(1:f(1)-1))';
%
[filename,pathname] = uigetfile('*.dat','Input Measured FZ',50,50);
filename=lower(filename);
eval(['load ',[pathname,filename],';'])
f=findstr(filename,'.');
fm(3,:)=eval(filename(1:f(1)-1))';
%
[filename,pathname] = uigetfile('*.dat','Input Measured MX',50,50);
filename=lower(filename);
eval(['load ',[pathname,filename],';'])
f=findstr(filename,'.');
fm(4,:)=eval(filename(1:f(1)-1))';
%
[filename,pathname] = uigetfile('*.dat','Input Measured MY',50,50);
filename=lower(filename);
eval(['load ',[pathname,filename],';'])
f=findstr(filename,'.');
fm(5,:)=eval(filename(1:f(1)-1))';
%
[filename,pathname] = uigetfile('*.dat','Input Measured MZ',50,50);
filename=lower(filename);
eval(['load ',[pathname,filename],';'])
f=findstr(filename,'.');
```

```
fm(6,:)=eval(filename(1:f(1)-1))';
%
%
%       Generates the displacements in the normalized coodinate system.
%
%       z                    displacements in normalized coordinate system.
%
%
z=zeros(6,fm_c_size);
for t=1:fm_c_size
for i=1:6
  z(i,t)=(1/w(i)^2)*phi(:,i)'*fm(:,t);
end
end
%
%
%       zdd     second derivative of the displacement in the normalized
%               coordinate system (i.e. acceleration).
%
%
zd=zeros(6,fm_c_size);
zdd2=zeros(6,fm_c_size);
n=input('Input filter order : ');
samf=input('Input sample frequency : ');
b1=remez(n,[0 1],[0 pi*samf],'d')
for i=1:6
    zd(i,:)=filter(b1,1,z(i,:));
end
for i=1:6
  zdd2(i,:)=filter(b1,1,zd(i,:));
end
for i=1:6
  zdd(i,:)=zdd2(i,(n+1):fm_c_size);
end
[zdd_r,zdd_c]=size(zdd);
for i=1:6
  fm2(i,:)=fm(i,1:zdd_c); % truncate fm size
end
%
%       Converts the accelerations from the normalized coordinate
%       system to the physical coordinate system.
%
%       xdd     acceleration in the physical coordinate system.
%
```

```
%
xdd=zeros(6.fm_c_size);
xdd=phi*zdd;
%
%
%       Determines the inertial force using the acceleration
%       in the physical coordinate system.
%
%       fi                  inertial force vector
%
%
fi=zeros(6.fm_c_size);
fi=-1*m*xdd;
%
%
%       Determine real force vectors
%
%       ft                  real force vector. (ft = fm - fi)
%
%
ft = zeros(6.fm_c_size);
ft = fm2-fi;
%
%
% Calculate time vector (be to used for plotting)
%
%
t_ft = zeros(1.zdd_c);
t_ft=0:h:(h*(zdd_c-1));
%
%
```

```
%       --------------------------------------------------------------
%       Program:        FFT_matrix_attempt1b.m
%       --------------------------------------------------------------
%       Inverse Fourier Transform Method. Matrix Implementation One
%
%       Coded by:               F. Winsor
%
%       Version Date: Sept. 13. 1999
%
%       --------------------------------------------------------------
%       Load in original force data files. Three files for each
%       applied load direction. These are decay test results.
%
load global_fx.dat %
load global_fy.dat %  Load applied in X direction.
load global_fz.dat %
%
load global_y_fx.dat %
load global_y_fy.dat %          Load applied in Y direction.
load global_y_fz.dat %
%
load global_z_fx.dat %
load global_z_fy.dat %          Load applied in Z direction.
load global_z_fz.dat %
%
%       Select "measured force" signal from the original files.
%
fm_x_fx=global_fx(9001:10024); %
fm_x_fy=global_fy(9001:10024); %              X direction.
fm_x_fz=global_fz(9001:10024); %
%
fm_y_fx=global_y_fx(2501:3524); %
fm_y_fy=global_y_fy(2501:3524); %             Y direction.
fm_y_fz=global_y_fz(2501:3524); %
%
fm_z_fx=global_z_fx(1501:2524); %
fm_z_fy=global_z_fy(1501:2524); %             Z direction.
fm_z_fz=global_z_fz(1501:2524); %
%
%       Save these files in ascii format.
%
save fm_x_fx.dat fm_x_fx -ascii
save fm_x_fy.dat fm_x_fy -ascii
save fm_x_fz.dat fm_x_fz -ascii
save fm_y_fx.dat fm_y_fx -ascii
save fm_y_fy.dat fm_y_fy -ascii
save fm_y_fz.dat fm_y_fz -ascii
save fm_z_fx.dat fm_z_fx -ascii
save fm_z_fy.dat fm_z_fy -ascii
save fm_z_fz.dat fm_z_fz -ascii
%
%       Select step response from original signal.
%
step_x_fx=global_fx(9172:10195);
step_x_fy=global_fy(9172:10195);
step_x_fz=global_fz(9172:10195);
%
step_y_fx=global_y_fx(3040:4063);
step_y_fy=global_y_fy(3040:4063);
step_y_fz=global_y_fz(3040:4063);
%
step_z_fx=global_z_fx(1902:2925);
step_z_fy=global_z_fy(1902:2925);
step_z_fz=global_z_fz(1902:2925);
%
%       Save these files in ascii format.
%
save step_x_fx.dat step_x_fx -ascii
save step_x_fy.dat step_x_fy -ascii
save step_x_fz.dat step_x_fz -ascii
save step_y_fx.dat step_y_fx -ascii
save step_y_fy.dat step_y_fy -ascii
save step_y_fz.dat step_y_fz -ascii
save step_z_fx.dat step_z_fx -ascii
save step_z_fy.dat step_z_fy -ascii
save step_z_fz.dat step_z_fz -ascii
%
%       Call script First_deriv.m to differentiate
%       each step response to obtain the impulse
%       response. This must be done for the nine step
%       response files, separately.
%
%       For this program the step responses must be
%       differentiated in a certain order: step_x_fx,
%       step_x_fy, step_x_fz, etc.....
%
for diff_index=1:1:9
  first_deriv
  if diff_index==1
    imp_x_fx=zd';
  elseif diff_index==2
    imp_x_fy=zd';
  elseif diff_index==3
    imp_x_fz=zd';
        elseif diff_index==4
    imp_y_fx=zd';
        elseif diff_index==5
    imp_y_fy=zd';
        elseif diff_index==6
    imp_y_fz=zd';
        elseif diff_index==7
    imp_z_fx=zd';
        elseif diff_index==8
    imp_z_fy=zd';
        else diff_index==9
    imp_z_fz=zd';
  end
end
%
%
%       Take the fast Fourier transform (fft) of
%       the so-called measured force signals. Note
%       that a 1024 length fft is specified.
%
%
FM_x_fx=fft(fm_x_fx,1024);
FM_x_fy=fft(fm_x_fy,1024);
FM_x_fz=fft(fm_x_fz,1024);
FM_y_fx=fft(fm_y_fx,1024);
FM_y_fy=fft(fm_y_fy,1024);
FM_y_fz=fft(fm_y_fz,1024);
FM_z_fx=fft(fm_z_fx,1024);
FM_z_fy=fft(fm_z_fy,1024);
FM_z_fz=fft(fm_z_fz,1024);
%
%       Take the fast Fourier transform (fft) of the
%       impulse response functions.
%
H_x_fx=fft(imp_x_fx,1024);
H_x_fy=fft(imp_x_fy,1024);
H_x_fz=fft(imp_x_fz,1024);
H_y_fx=fft(imp_y_fx,1024);
```

```
H_y_fy=fft(imp_y_fy,1024);
H_y_fz=fft(imp_y_fz,1024);
H_z_fx=fft(imp_z_fx,1024);
H_z_fy=fft(imp_z_fy,1024);
H_z_fz=fft(imp_z_fz,1024);
%
%       Normalize the ffts of the impulse response
%       functions (frequency response functions. This ensures
%       that the diagonal terms in the "Hs" matrix have
%       values of 1.0.
%
%
Hs_x_fx=H_x_fx/H_x_fx(1);
Hs_x_fy=H_x_fy/H_x_fx(1);
Hs_x_fz=H_x_fz/H_x_fx(1);
Hs_y_fx=H_y_fx/H_y_fy(1);
Hs_y_fy=H_y_fy/H_y_fy(1);
Hs_y_fz=H_y_fz/H_y_fy(1);
Hs_z_fx=H_z_fx/H_z_fz(1);
Hs_z_fy=H_z_fy/H_z_fz(1);
Hs_z_fz=H_z_fz/H_z_fz(1);
%
%
%       Copy one of the sets of measured force signals
%       into a common matrix(size 3x1024). In this case,
%       the measured force from a test where the load
%       was applied in the X direction was selected.
%
for i=1:1024
  FM_all(1,i)=FM_x_fx(i,1);
  FM_all(2,i)=FM_x_fy(i,1);
  FM_all(3,i)=FM_x_fz(i,1);
end
%
%       Copy the normalized frequency response function
%       vectors into a common matrix. Then divide each
%       column vector(3x1)in the measured force matrix,
%       by the relevant frequency response matrix (3x3)
%       to obtain an actual force column vector which is
%       stored in an actual force matrix (3x1024).
%
%       Note that two commands are available to do the
%       division. One uses the Matlab operator known as
%       "backslash or left matrix divide". The other
%       inverts the frequency response  matrix (3x3) then
%       multiplies it by the measured force column vector.
%
for i=1:1024
  Hs_all(1,1)=Hs_x_fx(i,1);
  Hs_all(1,2)=Hs_y_fx(i,1);
  Hs_all(1,3)=Hs_z_fx(i,1);
  Hs_all(2,1)=Hs_x_fy(i,1);
  Hs_all(2,2)=Hs_y_fy(i,1);
  Hs_all(2,3)=Hs_z_fy(i,1);
  Hs_all(3,1)=Hs_x_fz(i,1);
  Hs_all(3,2)=Hs_y_fz(i,1);
  Hs_all(3,3)=Hs_z_fz(i,1);
  %
  %FA_all(:,i)=Hs_all\FM_all(:,i);
  FA_all(:,i)=inv(Hs_all)*FM_all(:,i);
end
%
%       Set the first column vector in the actual
%       force matrix equal to the first column vector
%       of the measured force matrix. This is to
%       avoid a divide by zero problem when computing
%       the inverse Fourier transforms.
%
FA_all(:,1)=FM_all(:,1);
%
%       Generate a vector of frequencies to be used
%       in plotting the results.
%
freq=27.2168*(0:511)/1024;
%
%       Get the inverse Fourier transform of each
%       row vector, to obtain the time
%       series of the actual forces.
%
fa_fx=ifft(FA_all(1,:),1024);
fa_fy=ifft(FA_all(2,:),1024);
fa_fz=ifft(FA_all(3,:),1024);
%
%       Plot comparison of measured and actual
%       force results.
%
plot (fm_x_fx)
hold on
```

```
plot(real(fa_fx).'r')
legend('fm x fx'.'fa fx')
title('impulse ffts normalized by initial value of diagonal terms')
pause
hold off
plot (fm_x_fy)
hold on
plot(real(fa_fy).'r')
legend('fm x fx'.'fa fx')
legend('fm x fy'.'fa fy')
title('impulse ffts normalized by initial value of diagonal terms')
pause
hold off
plot (fm_x_fz)
hold on
plot(real(fa_fz).'r')
legend('fm x fz'.'fa fz')
title('impulse ffts normalized by initial value of diagonal terms')
pause
hold off
```

```
% ------------------------------------------------------------
% Program:  FFT_matrix_deconv_a.m
% ------------------------------------------------------------
% Inverse Fourier Transform Method, Matrix Implementation
%
% Coded by:     F. Winsor
%
% Version Date: Jan 18, 2000
%
% ------------------------------------------------------------
% Load in original force data files. Three files for each
% applied load direction. These are decay test results.
%
load global_fx.dat %
load global_fy.dat %  Load applied in X direction.
load global_fz.dat %
%
load global_y_fx.dat %
load global_y_fy.dat %    Load applied in Y direction.
load global_y_fz.dat %
%
load global_z_fx.dat %
load global_z_fy.dat %    Load applied in Z direction.
load global_z_fz.dat %
%
% Select "measured force" signal from the original files.
%
fm_x_fx=global_fx(9001:10024); %
fm_x_fy=global_fy(9001:10024); %      X direction.
fm_x_fz=global_fz(9001:10024); %
%
fm_y_fx=global_y_fx(2501:3524); %
fm_y_fy=global_y_fy(2501:3524); %     Y direction.
fm_y_fz=global_y_fz(2501:3524); %
%
fm_z_fx=global_z_fx(1501:2524); %
fm_z_fy=global_z_fy(1501:2524); %     Z direction.
fm_z_fz=global_z_fz(1501:2524); %
%
% Save these files in ascii format.
%
save fm_x_fx.dat fm_x_fx -ascii
save fm_x_fy.dat fm_x_fy -ascii
save fm_x_fz.dat fm_x_fz -ascii
save fm_y_fx.dat fm_y_fx -ascii
save fm_y_fy.dat fm_y_fy -ascii
save fm_y_fz.dat fm_y_fz -ascii
save fm_z_fx.dat fm_z_fx -ascii
save fm_z_fy.dat fm_z_fy -ascii
save fm_z_fz.dat fm_z_fz -ascii
%
% Select step response from original signal.
%
step_x_fx=global_fx(9172:10195);
step_x_fy=global_fy(9172:10195);
step_x_fz=global_fz(9172:10195);
%
step_y_fx=global_y_fx(3040:4063);
step_y_fy=global_y_fy(3040:4063);
step_y_fz=global_y_fz(3040:4063);
%
step_z_fx=global_z_fx(1902:2925);
step_z_fy=global_z_fy(1902:2925);
step_z_fz=global_z_fz(1902:2925);
%
% Save these files in ascii format.
%
save step_x_fx.dat step_x_fx -ascii
save step_x_fy.dat step_x_fy -ascii
save step_x_fz.dat step_x_fz -ascii
save step_y_fx.dat step_y_fx -ascii
save step_y_fy.dat step_y_fy -ascii
save step_y_fz.dat step_y_fz -ascii
save step_z_fx.dat step_z_fx -ascii
save step_z_fy.dat step_z_fy -ascii
save step_z_fz.dat step_z_fz -ascii
%
% Call script First_deriv.m to differentiate
% each step response to obtain the impulse
% response. This must be done for the nine step
% response files. separately.
%
% For this program the step responses must be
% differentiated in a certain order: step_x_fx.
% step_x_fy. step_x_fz. etc.....
%
for diff_index=1:1:9
  first_deriv
  if diff_index==1
    imp_x_fx=zd';
  elseif diff_index==2
    imp_x_fy=zd';
  elseif diff_index==3
    imp_x_fz=zd';
    elseif diff_index==4
    imp_y_fx=zd';
    elseif diff_index==5
    imp_y_fy=zd';
    elseif diff_index==6
    imp_y_fz=zd';
    elseif diff_index==7
    imp_z_fx=zd';
    elseif diff_index==8
    imp_z_fy=zd';
    else diff_index==9
    imp_z_fz=zd';
  end
end
%
% Create normalized impulse response
% function vectors. They are normalized
% by their own mean values times the
% number of points in the vector.
%
%
h_x_fx=imp_x_fx/(-mean(imp_x_fx)*1024);
h_x_fy=imp_x_fy/(-mean(imp_x_fx)*1024);
h_x_fz=imp_x_fz/(-mean(imp_x_fx)*1024);
h_y_fx=imp_y_fx/(-mean(imp_y_fy)*1024);
h_y_fy=imp_y_fy/(-mean(imp_y_fy)*1024);
h_y_fz=imp_y_fz/(-mean(imp_y_fy)*1024);
h_z_fx=imp_z_fx/(-mean(imp_z_fz)*1024);
h_z_fy=imp_z_fy/(-mean(imp_z_fz)*1024);
h_z_fz=imp_z_fz/(-mean(imp_z_fz)*1024);
%
%
%
% Modify the first point of the impulse response signals.
%
%
%
%
```

```
if(sum(h_x_fx)<=0.0)
  h_x_fx(1)=h_x_fx(1)+abs(sum(h_x_fx)+sum(h_x_fx));
else
  h_x_fx(1)=h_x_fx(1)-abs(sum(h_x_fx)+sum(h_x_fx));
end
%%
%
if(sum(h_x_fy)<=0.0)
  h_x_fy(1)=h_x_fy(1)+abs(sum(h_x_fy)+sum(h_x_fy));
else
  h_x_fy(1)=h_x_fy(1)-abs(sum(h_x_fy)+sum(h_x_fy));
end
%%
%
if(sum(h_x_fz)<=0.0)
  h_x_fz(1)=h_x_fz(1)+abs(sum(h_x_fz)+sum(h_x_fz));
else
  h_x_fz(1)=h_x_fz(1)-abs(sum(h_x_fz)+sum(h_x_fz));
end
%%
%
%
if(sum(h_y_fx)<=0.0)
  h_y_fx(1)=h_y_fx(1)+abs(sum(h_y_fx)+sum(h_y_fx));
else
  h_y_fx(1)=h_y_fx(1)-abs(sum(h_y_fx)+sum(h_y_fx));
end
%%
%
if(sum(h_y_fy)<=0.0)
  h_y_fy(1)=h_y_fy(1)+abs(sum(h_y_fy)+sum(h_y_fy));
else
  h_y_fy(1)=h_y_fy(1)-abs(sum(h_y_fy)+sum(h_y_fy));
end
%%
%
if(sum(h_y_fz)<=0.0)
  h_y_fz(1)=h_y_fz(1)+abs(sum(h_y_fz)+sum(h_y_fz));
else
  h_y_fz(1)=h_y_fz(1)-abs(sum(h_y_fz)+sum(h_y_fz));
end
%%
%
%
```

```
if(sum(h_z_fx)<=0.0)
  h_z_fx(1)=h_z_fx(1)+abs(sum(h_z_fx)+sum(h_z_fx));
else
  h_z_fx(1)=h_z_fx(1)-abs(sum(h_z_fx)+sum(h_z_fx));
end
%%
%
if(sum(h_z_fy)<=0.0)
  h_z_fy(1)=h_z_fy(1)+abs(sum(h_z_fy)+sum(h_z_fy));
else
  h_z_fy(1)=h_z_fy(1)-abs(sum(h_z_fy)+sum(h_z_fy));
end
%%
%
if(sum(h_z_fz)<=0.0)
  h_z_fz(1)=h_z_fz(1)+abs(sum(h_z_fz)+sum(h_z_fz));
else
  h_z_fz(1)=h_z_fz(1)-abs(sum(h_z_fz)+sum(h_z_fz));
end
%%
%
%
% Select the signals to be used as measured input signals.
%
%
%
fm_x=fm_x_fx;
fm_y=fm_x_fy;
fm_z=fm_x_fz;
%
%
%
% Pad measured signals with zeros to 2047 points.
%
%
%
fm_x_pad=zeros(2047,1);
fm_y_pad=zeros(2047,1);
fm_z_pad=zeros(2047,1);
fm_x_pad(1:1024)=fm_x;
fm_y_pad(1:1024)=fm_y;
fm_z_pad(1:1024)=fm_z;
%
%
```

```
%
% Perform the deconvolution
%
%
%
[q_x1,r_x1]=deconv(fm_x_pad,h_x_fx);
[q_x2,r_x2]=deconv(fm_x_pad,h_y_fx);
[q_x3,r_x3]=deconv(fm_x_pad,h_z_fx);
[q_y1,r_y1]=deconv(fm_y_pad,h_x_fy);
[q_y2,r_y2]=deconv(fm_y_pad,h_y_fy);
[q_y3,r_y3]=deconv(fm_y_pad,h_z_fy);
[q_z1,r_z1]=deconv(fm_z_pad,h_x_fz);
[q_z2,r_z2]=deconv(fm_z_pad,h_y_fz);
[q_z3,r_z3]=deconv(fm_z_pad,h_z_fz);
%
%
% Plot the results
%
%
%
hold off
plot (fm_x)
hold on
plot (q_x1,'r')
title ('fm-x vs q-x1 from h-x-fx')
legend('measured','actual')
xlabel ('Vector Element Number')
ylabel ('Force (MN)')
pause

hold off
plot (fm_x)
hold on
plot (q_x2,'r')
title ('fm-x vs q-x2 from h-y-fx')
legend('measured','actual')
xlabel ('Vector Element Number')
ylabel ('Force (MN)')
pause

hold off
plot (fm_x)
hold on
plot (q_x3,'r')
```

```
title ('fm-x vs q-x3 from h-z-fx')
legend('measured','actual')
xlabel ('Vector Element Number')
ylabel ('Force (MN)')
pause

hold off
plot (fm_y)
hold on
plot (q_y1,'r')
title ('fm-y vs q-y1 from h-x-fy')
legend('measured','actual')
xlabel ('Vector Element Number')
ylabel ('Force (MN)')
pause

hold off
plot (fm_y)
hold on
plot (q_y2,'r')
title ('fm-y vs q-y2 from h-y-fy')
legend('measured','actual')
xlabel ('Vector Element Number')
ylabel ('Force (MN)')
pause

hold off
plot (fm_y)
hold on
plot (q_y3,'r')
title ('fm-y vs q-y3 from h-z-fy')
legend('measured','actual')
xlabel ('Vector Element Number')
ylabel ('Force (MN)')
pause

hold off
plot (fm_z)
hold on
plot (q_z1,'r')
title ('fm-z vs q-z1 from h-x-fz')
legend('measured','actual')
xlabel ('Vector Element Number')
ylabel ('Force (MN)')
```

```
pause

hold off
plot (fm_z)
hold on
plot (q_z2.'r')
title ('fm-z vs q-z2 from h-y-fz - modification h-z-fz(1)= 1.133')
legend('measured'.'actual')
xlabel ('Vector Element Number')
ylabel ('Force (MN)')
pause

hold off
plot (fm_z)
hold on
plot (q_z3.'r')
title ('fm-z vs q-z3 from h-z-fz')
legend('measured'.'actual')
xlabel ('Vector Element Number')
ylabel ('Force (MN)')
pause
```

```
% ----------------------------------------------------------
% Program:  ift_wave_sdof.m
% ----------------------------------------------------------
% Single degree of freedom implementation of inverse Fourier
% transform method applied to measured wave impact signals
%
% Coded by:      F. Winsor
%
% Version Date: Jan 18. 2000
%
% ----------------------------------------------------------
%
%       Input wave input signals
%
%
%
[filename.pathname] = uigetfile('*.dat'.'Wave Impact Signal - X Direction: '.50.50):
filename=lower(filename):
eval(['load '.[pathname.filename].':'])
f=findstr(filename.'.'):
fm_x(1.:)=eval(filename(1:f(1)-1))':
%
%
[filename.pathname] = uigetfile('*.dat'.'Wave Impact Signal - Y Direction: '.50.50):
filename=lower(filename):
eval(['load '.[pathname.filename].':'])
f=findstr(filename.'.'):
fm_y(1.:)=eval(filename(1:f(1)-1))':
%
%
[filename.pathname] = uigetfile('*.dat'.'Wave Impact Signal - Z Direction: '.50.50):
filename=lower(filename):
eval(['load '.[pathname.filename].':'])
f=findstr(filename.'.'):
fm_z(1.:)=eval(filename(1:f(1)-1))':
%
%
%       Input signal sample rate. Use to determine plotting
%       vectors for time and frequency scales.
%
%
%
fs=input('Input Sample rate (default = 27.2168)'):
if isempty(fs)
```

```
  fs=27.2168;
end
%
%
time=(0:(1/fs):(1023*(1/fs)));
%
freq=fs*(0:511)/1024;
%
%
%       Plot the measured signal time traces.
%
%
%
hold off
plot(time,fm_x)
title('Wave Impact Signal - X direction')
legend('fm-x')
xlabel('Time (s)')
ylabel('Force (MN)')
pause
%
%
hold off
plot(time,fm_y)
title('Wave Impact Signal - Y direction')
legend('fm-y')
xlabel('Time (s)')
ylabel('Force (MN)')
pause
%
hold off
plot(time,fm_z)
title('Wave Impact Signal - Z direction')
legend('fm-z')
xlabel('Time (s)')
ylabel('Force (MN)')
pause
%
%
%       Determine the FFTs of measured signals, impulse response
%       signals and normalized frequency responses.
%
%
first_deriv % input step_x_fx.dat
```

```
%
hx=zd;
%
FM_x=fft(fm_x,1024);
Hx=fft(hx,1024);
Hx1=Hx/Hx(1);
%
first_deriv % input step_y_fy.dat
%
hy=zd;
%
FM_y=fft(fm_y,1024);
Hy=fft(hy,1024);
Hy1=Hy/Hy(1);
%
first_deriv % input step_z_fz.dat
%
hz=zd;
%
FM_z=fft(fm_z,1024);
Hz=fft(hz,1024);
Hz1=Hz/Hz(1);
%
%
%
%       Plot FFTs of measured signals and
%       normalized frequency responses
%
%
%
%
hold off
plot(freq,abs(FM_x(1:512)))
title('FFT of measured signal of X Direction')
legend('FM-x')
xlabel('Frequency (Hz)')
ylabel('FFT Magnitude')
pause
%
%
%
hold off
plot(freq,abs(FM_y(1:512)))
title('FFT of measured signal of Y Direction')
```

```
legend('FM-y')
xlabel('Frequency (Hz)')
ylabel('FFT Magnitude')
pause
%
%
%
hold off
plot(freq,abs(FM_z(1:512)))
title('FFT of measured signal of Z Direction')
legend('FM-z')
xlabel('Frequency (Hz)')
ylabel('FFT Magnitude')
pause
%
%
%
hold off
plot(freq,abs(Hx1(1:512)))
title ('Normalized Frequency Response - X Direction')
legend('Hx1')
xlabel('Frequency (Hz)')
ylabel('FFT Magnitude')
pause
%
hold off
plot(freq,abs(Hy1(1:512)))
title ('Normalized Frequency Response - Y Direction')
legend('Hy1')
xlabel('Frequency (Hz)')
ylabel('FFT Magnitude')
pause
%
hold off
plot(freq,abs(Hz1(1:512)))
title ('Normalized Frequency Response - Z Direction')
legend('Hz1')
xlabel('Frequency (Hz)')
ylabel('FFT Magnitude')
pause
%
%
%
%       Determine the FFT of the actual force signals
```

```
%
%
%
%
%
FA_x=FM_x./Hx1;
FA_y=FM_y./Hy1;
FA_z=FM_z./Hz1;
%
%
%       Plot comparisons of the measured and actual force FFTs
%
%
%
%
hold off
plot(freq,abs(FM_x(1:512)))
hold on
plot(freq,abs(FA_x(1:512)),'r')
title ('FFTs of measured and actual force signals - X Direction')
legend('FM-x','FA-x')
xlabel('Frequency (Hz)')
ylabel('FFT Magnitude')
pause
%
hold off
plot(freq,abs(FM_y(1:512)))
hold on
plot(freq,abs(FA_y(1:512)),'r')
title ('FFTs of measured and actual force signals - Y Direction')
legend('FM-y','FA-y')
xlabel('Frequency (Hz)')
ylabel('FFT Magnitude')
pause
%
hold off
plot(freq,abs(FM_z(1:512)))
hold on
plot(freq,abs(FA_z(1:512)),'r')
title ('FFTs of measured and actual force signals - Z Direction')
legend('FM-z','FA-z')
xlabel('Frequency (Hz)')
ylabel('FFT Magnitude')
pause
```

```
%
%
%
%       Determine the actual force time series signals
%
%
%
%
%
fa_x=ifft(FA_x,1024);
fa_y=ifft(FA_y,1024);
fa_z=ifft(FA_z,1024);
%
%
%
%       Plot the measured and actual force time series signals
%
%
%
%
hold off
plot(time,real(fm_x))
hold on
plot(time,real(fa_x),'r')
title('measured and actual force signals')
legend('fm-x  - measured signal','fa-x  - actual signal')
xlabel('Time (s)')
ylabel('Force (MN)')
pause
%
hold off
plot(time,real(fm_y))
hold on
plot(time,real(fa_y),'r')
title('measured and actual force signals')
legend('fm-y  - measured signal','fa-y  - actual signal')
xlabel('Time (s)')
ylabel('Force (MN)')
pause
%
hold off
plot(time,real(fm_z))
hold on
plot(time,real(fa_z),'r')
```

```
title('measured and actual force signals')
legend('fm-z  - measured signal','fa-z  - actual signal')
xlabel('Time (s)')
ylabel('Force (MN)')
pause
%
%
%       Plot comparisons of the measured force FFT and
%       the frequency response. This can illustrate the
%       shift in the resonant peak caused by added mass.
%
%
%
%       Note scale factor applied to (some of the) normalized
%       frequency response for plotting purposes.
%
%
hold off
plot(freq,abs(FM_x(1:512)))
hold on
plot(freq,abs(Hx1(1:512)*1),'r')
title ('FFTs of Measured Signal and Normalized Frequency Response - X Direction')
legend('FM-x','Hx1')
xlabel('Frequency (Hz)')
ylabel('FFT Magnitude')
pause
%
%
hold off
plot(freq,abs(FM_y(1:512)))
hold on
plot(freq,abs(Hy1(1:512)*1),'r')
title ('FFTs of Measured Signal and Normalized Frequency Response - Y Direction')
legend('FM-y','Hy1')
xlabel('Frequency (Hz)')
ylabel('FFT Magnitude')
pause
%
%
hold off
plot(freq,abs(FM_z(1:512)))
hold on
plot(freq,abs(Hz1(1:512)*44.7),'r')
title ('FFTs of Measured Signal and Normalized Frequency Response - Z Direction')
```

```
legend('FM-z','Hz1')
xlabel('Frequency (Hz)')
ylabel('FFT Magnitude')
pause
%
```

```
%       -------------------------------------------------------------
%       Program:        FFT_matrix_wave.m
%       -------------------------------------------------------------
%       Inverse Fourier Transform Method, Matrix Implementation One
%
%       Coded by:               F. Winsor
%
%       Version Date: Sept. 13, 1999
%
%       -------------------------------------------------------------
%
%
%       Input wave input signals
%
%
[filename,pathname] = uigetfile('*.dat','Wave Impact Signal - X Direction: ',50,50);
filename=lower(filename);
eval(['load ',[pathname,filename],';'])
f=findstr(filename,'.');
fm_x(1,:)=eval(filename(1:f(1)-1))';
%
%
[filename,pathname] = uigetfile('*.dat','Wave Impact Signal - Y Direction: ',50,50);
filename=lower(filename);
eval(['load ',[pathname,filename],';'])
f=findstr(filename,'.');
fm_y(1,:)=eval(filename(1:f(1)-1))';
%
%
[filename,pathname] = uigetfile('*.dat','Wave Impact Signal - Z Direction: ',50,50);
filename=lower(filename);
eval(['load ',[pathname,filename],';'])
f=findstr(filename,'.');
fm_z(1,:)=eval(filename(1:f(1)-1))';
%
%
%
%       Input signal sample rate. Use to determine plotting
%       vectors for time and frequency scales.
%
%
fs=input('Input Sample rate (default = 27.2168)');
if isempty(fs)
  fs=27.2168;
```

```
end
%
%
time=(0:(1/fs):(1023*(1/fs)));
%
freq=fs*(0:511)/1024;
%
%       Call script First_deriv.m to differentiate
%       each step response to obtain the impulse
%       response. This must be done for the nine step
%       response files, separately.
%
%       For this program the step responses must be
%       differentiated in a certain order: step_x_fx,
%       step_x_fy, step_x_fz, etc.....
%
for diff_index=1:1:9
  first_deriv
  if diff_index==1
    imp_x_fx=zd';
  elseif diff_index==2
    imp_x_fy=zd';
  elseif diff_index==3
    imp_x_fz=zd';
        elseif diff_index==4
    imp_y_fx=zd';
        elseif diff_index==5
    imp_y_fy=zd';
        elseif diff_index==6
    imp_y_fz=zd';
        elseif diff_index==7
    imp_z_fx=zd';
        elseif diff_index==8
    imp_z_fy=zd';
        else diff_index==9
    imp_z_fz=zd';
  end
end
%
%
%       Take the fast Fourier transform (fft) of
%       the so-called measured force signals. Note
%       that a 1024 length fft is specified.
%
```

```
%
FM_x=fft(fm_x,1024);
FM_y=fft(fm_y,1024);
FM_z=fft(fm_z,1024);
%
%       Take the fast Fourier transform (fft) of the
%       impulse response functions.
%
H_x_fx=fft(imp_x_fx,1024);
H_x_fy=fft(imp_x_fy,1024);
H_x_fz=fft(imp_x_fz,1024);
H_y_fx=fft(imp_y_fx,1024);
H_y_fy=fft(imp_y_fy,1024);
H_y_fz=fft(imp_y_fz,1024);
H_z_fx=fft(imp_z_fx,1024);
H_z_fy=fft(imp_z_fy,1024);
H_z_fz=fft(imp_z_fz,1024);
%
%       Normalize the ffts of the impulse response
%       functions (frequency response functions with
%       the first point in each signal. This ensures
%       that the first point in the "Hs" signal has
%       a value of 1.0.
%
Hs_x_fx=H_x_fx/H_x_fx(1);
Hs_x_fy=H_x_fy/H_x_fx(1);
Hs_x_fz=H_x_fz/H_x_fx(1);
Hs_y_fx=H_y_fx/H_y_fy(1);
Hs_y_fy=H_y_fy/H_y_fy(1);
Hs_y_fz=H_y_fz/H_y_fy(1);
Hs_z_fx=H_z_fx/H_z_fz(1);
Hs_z_fy=H_z_fy/H_z_fz(1);
Hs_z_fz=H_z_fz/H_z_fz(1);
%
%
%       Copy one of the sets of measured force signals
%       into a common matrix(size 3x1024). In this case,
%       the measured force from a test where the load
%       was applied in the X direction was selected.
%
%
%
%
for i=1:1024
```

```
  FM_all(1,i)=FM_x(1,i);
  FM_all(2,i)=FM_y(1,i);
  FM_all(3,i)=FM_z(1,i);
end
%
%       Copy the normalized frequency response function
%       vectors into a common matrix. Then divide each
%       column vector(3x1)in the measured force matrix.
%       by the relevant frequency response matrix (3x3)
%       to obtain an actual force column vector which is
%       stored in an actual force matrix (3x1024).
%
%       Note that two commands are available to do the
%       division. One uses the Matlab operator known as
%       "backslash or left matrix divide". The other
%       inverts the frequency response  matrix (3x3) then
%       multiplies it by the measured force column vector.
%
for i=1:1024
  Hs_all(1,1)=Hs_x_fx(i,1);
  Hs_all(1,2)=Hs_y_fx(i,1);
  Hs_all(1,3)=Hs_z_fx(i,1);
  Hs_all(2,1)=Hs_x_fy(i,1);
  Hs_all(2,2)=Hs_y_fy(i,1);
  Hs_all(2,3)=Hs_z_fy(i,1);
  Hs_all(3,1)=Hs_x_fz(i,1);
  Hs_all(3,2)=Hs_y_fz(i,1);
  Hs_all(3,3)=Hs_z_fz(i,1);
  %
  %FA_all(:,i)=Hs_all\FM_all(:,i);
  FA_all(:,i)=inv(Hs_all)*FM_all(:,i);
end
%
%       Set the first column vector in the actual
%       force matrix equal to the first column vector
%       of the measured force matrix. This is to
%       avoid a divide by zero problem when computing
%       the inverse Fourier transforms.
%
FA_all(:,1)=FM_all(:,1);
%
%       Generate a vector of frequencies to be used
%       in plotting the results.
%
```

```
%
%       Get the inverse Fourier transform of each
%       row vector, to obtain the time
%       series of the actual forces.
%
fa_fx=ifft(FA_all(1,:),1024);
fa_fy=ifft(FA_all(2,:),1024);
fa_fz=ifft(FA_all(3,:),1024);
%
%       Plot comparison of measured and actual
%       force results.
%
plot (time,fm_x)
hold on
plot(time,real(fa_fx),'r')
legend('measured X force','actual x force')
title('MDOF - measured and actual force signals')
xlabel('Time (s)')
ylabel('Force (MN)')
%print
pause
hold off
plot (time,fm_y)
hold on
plot(time,real(fa_fy),'r')
legend('measured Y force','actual Y force')
title('MDOF - measured and actual force signals')
xlabel('Time (s)')
ylabel('Force (MN)')
%print
pause
hold off
plot (time,fm_z)
hold on
plot(time,real(fa_fz),'r')
legend('measured Z force','actual Z force')
title('MDOF - measured and actual force signals')
xlabel('Time (s)')
ylabel('Force (MN)')
%print
pause
hold off
```

```
%.........................................................................................
%
%      Program:      fft_modulus_general.m
%
%      Coded by:     Fraser Winsor
%
%
%      This script calculates the fft modulus and phase. The modulus or magnitude
%      is scaled. that is. a sinusoidal signal of arbitrary amplitude will produce
%      a fft magnitude peak at the same value.
%
%      Information on the scaling method can be found on page 6-117 of the "Matlab
%      Signal Processing Toolbox. User's Guide. Version 4"
%
%.........................................................................................
%
%      Input time series
%
%
[filename.pathname] = uigetfile('*.dat'.'FX Time Series'.50.50):
filename=lower(filename):
eval(['load '.[pathname.filename].':'])
f=findstr(filename.'.'):
fx=eval(filename(1:f(1)-1)):
%
[filename.pathname] = uigetfile('*.dat'.'FY Time Series'.50.50):
filename=lower(filename):
eval(['load '.[pathname.filename].':'])
f=findstr(filename.'.'):
fy=eval(filename(1:f(1)-1)):
%
%
[filename.pathname] = uigetfile('*.dat'.'FZ Time Series'.50.50):
filename=lower(filename):
eval(['load '.[pathname.filename].':'])
f=findstr(filename.'.'):
fz=eval(filename(1:f(1)-1)):
%
[filename.pathname] = uigetfile('*.dat'.'MX Time Series'.50.50):
filename=lower(filename):
eval(['load '.[pathname.filename].':'])
f=findstr(filename.'.'):
mx=eval(filename(1:f(1)-1)):
%
```

```
[filename,pathname] = uigetfile('*.dat','MY Time Series',50,50);
filename=lower(filename);
eval(['load ',[pathname,filename],';'])
f=findstr(filename,'.');
my=eval(filename(1:f(1)-1));
%
%
[filename,pathname] = uigetfile('*.dat','MZ Time Series',50,50);
filename=lower(filename);
eval(['load ',[pathname,filename],';'])
f=findstr(filename,'.');
mz=eval(filename(1:f(1)-1));
%
%
%       Choose the degree-of-freedom being analyzed. The time series will be plotted up
%       and the starting point can be selected graphically.
%
%
prin_dof=input('DOF of Interest (1=FX, 2=FY, etc) ');
%
if prin_dof==1
  plot (fx)
  title('FX')
  zoom
  pause
  [xxx,yyy] = ginput(1)
  t1=round(xxx)
  zoom
elseif prin_dof==2
  plot (fy)
  title('FY')
  zoom
  pause
  [xxx,yyy] = ginput(1)
  t1=round(xxx)
  zoom
elseif prin_dof==3
  plot (fz)
  title('FZ')
  zoom
  pause
  [xxx,yyy] = ginput(1)
  t1=round(xxx)
  zoom
```

```
elseif prin_dof==4
  plot (mx)
  title('MX')
  zoom
  pause
  [xxx,yyy] = ginput(1)
  t1=round(xxx)
  zoom
elseif prin_dof==5
  plot (my)
  title('MY')
  zoom
  pause
  [xxx,yyy] = ginput(1)
  t1=round(xxx)
  zoom
elseif prin_dof==6
  plot (mz)
  title('MZ')
  zoom
  pause
  [xxx,yyy] = ginput(1)
  t1=round(xxx)
  zoom
end
%
%
%       Input the length of the fft, and the time spacing
%
%
l_fft=input('Input FFT Length:  ');
t2=t1+l_fft;
time_step=input('Input time step DT:  ');
%
%
%       Evaluate the fft modulus and phase for each dof, and plot the results
%
%
x_fx=fx(t1:t2);
x_fx_fft=fft(x_fx,l_fft);
pn_x_fx=abs(x_fx_fft)*2/length(x_fx);
frq=(1/time_step)*(0:((l_fft/2)-1))/l_fft;
phase_fx=angle(x_fx_fft);
phase_fx_deg=phase_fx.*180/pi;
```

```
subplot (2,1,1),plot (frq,pn_x_fx(1:l_fft/2))
title ('fft fx   modulus')
subplot (2,1,2),plot (frq,phase_fx_deg(1:(l_fft/2)))
title ('fft fx   phase')
zoom
pause
%
%
x_fy=fy(t1:t2);
x_fy_fft=fft(x_fy,l_fft);
pn_x_fy=abs(x_fy_fft)*2/length(x_fy);
phase_fy=angle(x_fy_fft);
phase_fy_deg=phase_fy.*180/pi;
subplot (2,1,1),plot (frq,pn_x_fy(1:l_fft/2))
title ('fft fy   modulus')
subplot (2,1,2),plot (frq,phase_fy_deg(1:(l_fft/2)))
title ('fft fy   phase')
pause
%
%
%
x_fz=fz(t1:t2);
x_fz_fft=fft(x_fz,l_fft);
pn_x_fz=abs(x_fz_fft)*2/length(x_fz);
phase_fz=angle(x_fz_fft);
phase_fz_deg=phase_fz.*180/pi;
subplot (2,1,1),plot (frq,pn_x_fz(1:l_fft/2))
title ('fft fz   modulus')
subplot (2,1,2),plot (frq,phase_fz_deg(1:(l_fft/2)))
title ('fft fz   phase')
pause
%
%
%
x_mx=mx(t1:t2);
x_mx_fft=fft(x_mx,l_fft);
pn_x_mx=abs(x_mx_fft)*2/length(x_mx);
phase_mx=angle(x_mx_fft);
phase_mx_deg=phase_mx.*180/pi;
subplot (2,1,1),plot (frq,pn_x_mx(1:l_fft/2))
title ('fft mx  modulus')
subplot (2,1,2),plot (frq,phase_mx_deg(1:(l_fft/2)))
title ('fft mx   phase')
pause
```

```
%
%
x_my=my(t1:t2);
x_my_fft=fft(x_my,l_fft);
pn_x_my=abs(x_my_fft)*2/length(x_my);
phase_my=angle(x_my_fft);
phase_my_deg=phase_my.*180/pi;
subplot (2,1,1),plot (frq,pn_x_my(1:l_fft/2))
title ('fft my    modulus')
subplot (2,1,2),plot (frq,phase_my_deg(1:(l_fft/2)))
title ('fft my   phase')
pause
%
%
x_mz=mz(t1:t2);
x_mz_fft=fft(x_mz,l_fft);
pn_x_mz=abs(x_mz_fft)*2/length(x_mz);
phase_mz=angle(x_mz_fft);
phase_mz_deg=phase_mz.*180/pi;
subplot (2,1,1),plot (frq,pn_x_mz(1:l_fft/2))
title ('fft mz   modulus')
subplot (2,1,2),plot (frq,phase_mz_deg(1:(l_fft/2)))
title ('fft mz    phase')
pause
%
```

```
%.............................................................................................
%
%
%     Program:                  Run_dof6_sim3.m
%
%     Coded by:     Fraser Winsor
%
%
%     Script to generate displacement and velocity vectors in 6 degree-of-freedom
%     based on the equation of motion.
%
%     Inputs: Mass matrix (6x6)
%                               Stiffness matrix (6x6). see normal mode method
%                               Damping matrix (6x6). see Rayleigh damping
%                               Initial conditions (displacement and velocity)
%                               Initial force value
%                               Pulse start time and duration
%                               Signal length or time span
%
%     Outputs:      6 force and moment vectors. plus a time vector as ascii files
%
%     Note:                     (1)    Calls script dof6_sim3.m
%                               (2)    Check initial conditions
%                               (3)    Check applied force in script dof6_sim3.m
%
%
%.............................................................................................
%
%
%
%     Define global variables
%
%
global M K C X Y Z t0 tf per_cr t_start t_duration f_init
%
%
%     Input mass. stiffness. and damping coefficient matrices
%
%
[filename.pathname] = uigetfile('*.dat'.'Input Mass Matrix'.50.50);
filename=lower(filename);
eval(['load '.[pathname.filename].';'])
f=findstr(filename.'.');
M=eval(filename(1:f(1)-1));
```

```
%
[filename,pathname] = uigetfile('*.dat','Input Stiffness Matrix',50,50);
filename=lower(filename);
eval(['load ',[pathname,filename],';'])
f=findstr(filename,'.');
K=eval(filename(1:f(1)-1));
%
[filename,pathname] = uigetfile('*.dat','Input Damping Matrix',50,50);
filename=lower(filename);
eval(['load ',[pathname,filename],';'])
f=findstr(filename,'.');
C=eval(filename(1:f(1)-1));
%
%
%       Input initial force value, pulse start time, pulse duration
%       (specific to the pulse type force used in dof6_sim3.m)
%
%
f_init=input(' Enter initial force value: ');
t_start=input(' Enter step start time: ');
t_duration=input(' Enter step duration time: ');
%
%
%       Set up simplfied matrices for input into solver
%
%
X=C/M;
Y=K/M;
Z=inv(M);
%
%
%       Define initial conditions
%
%
t0=input(' Enter t0: ');
tf=input(' Enter tf: ');
u0 = [0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0];
%
%
%       Use ODE45 to solve differential equations
%
%
options=odeset('MaxStep',0.05); %This command sets the maximum step size
[t,u] = ode45('dof6_sim3',[t0 tf],u0,options);
```

```
%
%
%       Generate force signals by [K]*{d}, where {d} represents displacement vector.
%       Output signals to ascii files.
%
%
d(1,:)=u(:,1)';
d(2,:)=u(:,3)';
d(3,:)=u(:,5)';
d(4,:)=u(:,7)';
d(5,:)=u(:,9)';
d(6,:)=u(:,11)';
%
N = length(u)
for i=1:N,
  F_out(:,i)=K*d(:,i);
end
%
FX = F_out(1,:)';
FY = F_out(2,:)';
FZ = F_out(3,:)';
MX = F_out(4,:)';
MY = F_out(5,:)';
MZ = F_out(6,:)';
%
[filename,pathname] = uiputfile('*.dat','Output Force File - FX',50,50);
filename=lower(filename);
eval(['save ',[pathname,filename],' FX',' -ascii'])
%
[filename,pathname] = uiputfile('*.dat','Output Force File - FY',50,50);
filename=lower(filename);
eval(['save ',[pathname,filename],' FY',' -ascii'])
%
[filename,pathname] = uiputfile('*.dat','Output Force File - FZ',50,50);
filename=lower(filename);
eval(['save ',[pathname,filename],' FZ',' -ascii'])
%
[filename,pathname] = uiputfile('*.dat','Output Force File - MX',50,50);
filename=lower(filename);
eval(['save ',[pathname,filename],' MX',' -ascii'])
%
[filename,pathname] = uiputfile('*.dat','Output Force File - MY',50,50);
filename=lower(filename);
eval(['save ',[pathname,filename],' MY',' -ascii'])
```

```
%
[filename,pathname] = uiputfile('*.dat','Output Force File - MZ',50,50);
filename=lower(filename);
eval(['save ',[pathname,filename],' MZ',' -ascii'])
%
%
[filename,pathname] = uiputfile('*.dat','Output time vector File - t',50,50);
filename=lower(filename);
eval(['save ',[pathname,filename],' t',' -ascii'])
%
%
```

```
%.................................................................................................
%
%
%       Program:        Dof6_sim3.m
%
%       Coded by:       Fraser Winsor
%
%
%       Script called by Run_dof6_sim3.m to generate displacement
%       and velocity vectors in 6 degree-of-freedom based on the equation of motion.
%
%.................................................................................................
%
%
%       Define function name
%
%
function udot=dof6_sim3(t,u)
%
%
%       Define global variables
%
%
global M K C X Y Z t0 tf per_cr t_start t_duration f_init
%
%
%       Define applied force. In this case a step is applied. Remember to adjust
%       the location of f_init for each degree-of-freedom. F(1) = Fx, F(2)=Fy, F(3)=Fz,
%       F(4)= Mx, F(5)=My, F(6)=Mz.
%
%
if t <= t_start
  F(1) = 0;
  F(2) = 0;
  F(3) = 0;
  F(4) = 0;
  F(5) = 0;
  F(6) = 0;
elseif t <= (t_start + t_duration)
  F(1) = f_init;
  F(2) = 0;
  F(3) = 0;
  F(4) = 0;
  F(5) = 0;
  F(6) = 0;
else
  F(1) = 0;
  F(2) = 0;
  F(3) = 0;
  F(4) = 0;
  F(5) = 0;
  F(6) = 0;
end
%
%
%       Set up 2nd order differential equations as a set of first order
%       differential equations. Here the odd udot elements (i.e. udot(1,:),
%       udot(3,:), etc.) represent the displacement vectors, while the even
%       udot elements represent velocity vectors.
%
%
udot(1,:)=u(2);
udot(3,:)=u(4);
udot(5,:)=u(6);
udot(7,:)=u(8);
udot(9,:)=u(10);
udot(11,:)=u(12);
%
udot(2,:)=-
(X(1,1)*u(2)+X(1,2)*u(4)+X(1,3)*u(6)+X(1,4)*u(8)+X(1,5)*u(10)+X(1,6)*u(12))-...
  (Y(1,1)*u(1)+Y(1,2)*u(3)+Y(1,3)*u(5)+Y(1,4)*u(7)+Y(1,5)*u(9)+Y(1,6)*u(11))+...
  (Z(1,1)*F(1)+Z(1,2)*F(2)+Z(1,3)*F(3)+Z(1,4)*F(4)+Z(1,5)*F(5)+Z(1,6)*F(6));
%
udot(4,:)=-
(X(2,1)*u(2)+X(2,2)*u(4)+X(2,3)*u(6)+X(2,4)*u(8)+X(2,5)*u(10)+X(2,6)*u(12))-...
  (Y(2,1)*u(1)+Y(2,2)*u(3)+Y(2,3)*u(5)+Y(2,4)*u(7)+Y(2,5)*u(9)+Y(2,6)*u(11))+...
  (Z(2,1)*F(1)+Z(2,2)*F(2)+Z(2,3)*F(3)+Z(2,4)*F(4)+Z(2,5)*F(5)+Z(2,6)*F(6));
%
udot(6,:)=-
(X(3,1)*u(2)+X(3,2)*u(4)+X(3,3)*u(6)+X(3,4)*u(8)+X(3,5)*u(10)+X(3,6)*u(12))-...
  (Y(3,1)*u(1)+Y(3,2)*u(3)+Y(3,3)*u(5)+Y(3,4)*u(7)+Y(3,5)*u(9)+Y(3,6)*u(11))+...
  (Z(3,1)*F(1)+Z(3,2)*F(2)+Z(3,3)*F(3)+Z(3,4)*F(4)+Z(3,5)*F(5)+Z(3,6)*F(6));
%
udot(8,:)=-
(X(4,1)*u(2)+X(4,2)*u(4)+X(4,3)*u(6)+X(4,4)*u(8)+X(4,5)*u(10)+X(4,6)*u(12))-...
  (Y(4,1)*u(1)+Y(4,2)*u(3)+Y(4,3)*u(5)+Y(4,4)*u(7)+Y(4,5)*u(9)+Y(4,6)*u(11))+...
  (Z(4,1)*F(1)+Z(4,2)*F(2)+Z(4,3)*F(3)+Z(4,4)*F(4)+Z(4,5)*F(5)+Z(4,6)*F(6));
%
```

```
udot(10,:)=-
(X(5,1)*u(2)+X(5,2)*u(4)+X(5,3)*u(6)+X(5,4)*u(8)+X(5,5)*u(10)+X(5,6)*u(12))-...
  (Y(5,1)*u(1)+Y(5,2)*u(3)+Y(5,3)*u(5)+Y(5,4)*u(7)+Y(5,5)*u(9)+Y(5,6)*u(11))+...
  (Z(5,1)*F(1)+Z(5,2)*F(2)+Z(5,3)*F(3)+Z(5,4)*F(4)+Z(5,5)*F(5)+Z(5,6)*F(6));
%
udot(12,:)=-
(X(6,1)*u(2)+X(6,2)*u(4)+X(6,3)*u(6)+X(6,4)*u(8)+X(6,5)*u(10)+X(6,6)*u(12))-...
  (Y(6,1)*u(1)+Y(6,2)*u(3)+Y(6,3)*u(5)+Y(6,4)*u(7)+Y(6,5)*u(9)+Y(6,6)*u(11))+...
  (Z(6,1)*F(1)+Z(6,2)*F(2)+Z(6,3)*F(3)+Z(6,4)*F(4)+Z(6,5)*F(5)+Z(6,6)*F(6));
%
```

```
%.............................................................................................
%
%
%       Program:                First_deriv.m
%       Coded by:               F. Winsor
%
%
%
%       Script to calculate the first derivative of a signal (fx)
%       using the central-difference expression with error of order h^4. The first
%       two and last two point are calculated using forward-difference and backward
%       differnce expressions with error of order h^2. These expression can be found
%       in "Applied Numerical Methods for Digital Computation" (2nd ed.) by James,
%       Smith, Wolford.
%
%
%.............................................................................................
%
%
h=input('Input time step (default = 0.036742)');
if isempty(h)
  h=0.036742;
end
%
%
%
[filename,pathname] = uigetfile('*.dat','Input signal to be differentiated: ',50,50);
filename=lower(filename);
eval(['load ',[pathname,filename],';'])
f=findstr(filename,'.');
fx(1,:)=eval(filename(1:f(1)-1))';
%
[fm_r_size,fm_c_size]=size(fx);
%
%
z=fx;
%
zd=zeros(fm_r_size,1);
%
  zd( 1)=(-z( 3)+4*z( 2)-3*z( 1))/(2*h);
  zd( 2)=(-z( 4)+4*z( 3)-3*z( 2))/(2*h);
%
        zd( fm_r_size)=(3*z( fm_r_size)-4*z( fm_r_size-1)+z( fm_r_size-2))/(2*h);
        zd( fm_r_size-1)=(3*z( fm_r_size-1)-4*z( fm_r_size-2)+z( fm_r_size-3))/(2*h);
```

```
%
for t=3:(fm_r_size-2)
        zd( t)=(-z( t+2)+8*z( t+1)-8*z( t-1)+z( t-2))/(12*h);
end
```

```
%........................................................................................................
%
%
%       Program:                Spline_fit.m
%
%       Coded by:       Fraser Winsor
%
%
%
%       Script to fit a cubic spline (with uniform point spacing) through
%       time series with unequal point spacing. This was written to spline fit the
%       results of run_dof6_sim3.m. which produces simulated decay signal with unequal
%       point spacing.
%
%........................................................................................................
%
%       Load time series vectors
%
%
[filename.pathname] = uigetfile('*.dat'.'Signal Vector FX'.50.50):
filename=lower(filename):
eval(['load '.[pathname.filename].':'])
f=findstr(filename.'.'):
FX=eval(filename(1:f(1)-1)):
%
%
[filename.pathname] = uigetfile('*.dat'.'Signal Vector FY'.50.50):
filename=lower(filename):
eval(['load '.[pathname.filename].':'])
f=findstr(filename.'.'):
FY=eval(filename(1:f(1)-1)):
%
%
[filename.pathname] = uigetfile('*.dat'.'Signal Vector FZ'.50.50):
filename=lower(filename);
eval(['load '.[pathname.filename].':'])
f=findstr(filename.'.'):
FZ=eval(filename(1:f(1)-1)):
%
%
[filename.pathname] = uigetfile('*.dat'.'Signal Vector MX'.50.50):
filename=lower(filename);
eval(['load '.[pathname.filename].':'])
f=findstr(filename.'.'):
```

```
MX=eval(filename(1:f(1)-1));
%
%
[filename.pathname] = uigetfile('*.dat'.'Signal Vector MY'.50.50);
filename=lower(filename);
eval(['load '.[pathname.filename].';'])
f=findstr(filename.'.');
MY=eval(filename(1:f(1)-1));
%
%
[filename.pathname] = uigetfile('*.dat'.'Signal Vector MZ'.50.50);
filename=lower(filename);
eval(['load '.[pathname.filename].';'])
f=findstr(filename.'.');
MZ=eval(filename(1:f(1)-1));
%
%
[filename.pathname] = uigetfile('*.dat'.'Time Vector '.50.50);
filename=lower(filename);
eval(['load '.[pathname.filename].';'])
f=findstr(filename.'.');
TV=eval(filename(1:f(1)-1));
%
%
%       Select point spacing. and length of signal
%
%
TVi=0:0.01:50;
%
%
%       Use Spline.m to fit cubic splines
%
%
FXi=spline(TV.FX.TVi);
FYi=spline(TV.FY.TVi);
FZi=spline(TV.FZ.TVi);
MXi=spline(TV.MX.TVi);
MYi=spline(TV.MY.TVi);
MZi=spline(TV.MZ.TVi);
%
%
%       Transpose signals for output
%
%
```

```
FX2=FXi';
FY2=FYi';
FZ2=FZi';
MX2=MXi';
MY2=MYi';
MZ2=MZi';
TV2=TVi';
%
%
%       Output spline fit signals as ascii files
%
%
[filename,pathname] = uiputfile('*.dat','Output Interpolated Force File - FX',50,50);
filename=lower(filename);
eval(['save ',[pathname,filename],' FX2',' -ascii'])
%
%
[filename,pathname] = uiputfile('*.dat','Output Interpolated Force File - FY',50,50);
filename=lower(filename);
eval(['save ',[pathname,filename],' FY2',' -ascii'])
%
%
[filename,pathname] = uiputfile('*.dat','Output Interpolated Force File - FZ',50,50);
filename=lower(filename);
eval(['save ',[pathname,filename],' FZ2',' -ascii'])
%
%
[filename,pathname] = uiputfile('*.dat','Output Interpolated Force File - MX',50,50);
filename=lower(filename);
eval(['save ',[pathname,filename],' MX2',' -ascii'])
%
%
[filename,pathname] = uiputfile('*.dat','Output Interpolated Force File - MY',50,50);
filename=lower(filename);
eval(['save ',[pathname,filename],' MY2',' -ascii'])
%
%
[filename,pathname] = uiputfile('*.dat','Output Interpolated Force File - MZ',50,50);
filename=lower(filename);
eval(['save ',[pathname,filename],' MZ2',' -ascii'])
%
%
[filename,pathname] = uiputfile('*.dat','Output Time Series',50,50);
filename=lower(filename);
```

```
eval(['save ',[pathname,filename],' TV2',' -ascii'])
%
```

```
%...................................................................
%
%       Program:       mov_avg_v2.m
%
%       Coded by:      F. Winsor
%
%       Based on Algorithms provided in:
%
%       "The Scientist and Engineer's Guide to
%        Digital Signal Processing - Second Edition",
%        Steven W. Smith, California Technical Publishing,
%        1997-1999.
%
%
%       Several versions of the moving average filter are
%       implemented, and compared to results obtained
%       by convolution with various windows.
%
%...................................................................
%
%       Input signal to be filtered
%
[filename,pathname] = uigetfile('*.dat','Input Signal',50,50);
filename=lower(filename);
eval(['load ',[pathname,filename],';'])
f=findstr(filename,'.');
x=eval(filename(1:f(1)-1));
%
%       Input filter length
%
M=input('Moving Average Filter length (default=11) must be ODD:');
if isempty(M)
  M=11.0;
end

%
%
%       Input sample rate to calculate frequency
%       and time scales for plotting.
%
%
%
fs=input('Input Sample rate (default = 27.2168)');
if isempty(fs)
```

```
  fs=27.2168;
end
%
%
f_axis=fs*(0:511)/1024;
t_axis=(0:(1/fs):(1023*(1/fs)));
%
%
%       Moving Average Filter - One Sided
%

len_x=length(x);

y=zeros(len_x,1);

for i=1:(len_x-(M-1))
  for j=0:(M-1)
    y(i)=y(i)+x(i+j);
  end
  y(i)=y(i)/M;
end

%
%       Moving Average Filter - Symmetrical
%
len_x=length(x);

y2=zeros(len_x,1);

for i=((M-1)/2)+1:(len_x-(M-1)/2)
  for j=(-(M-1)/2):((M-1)/2)
    y2(i)=y2(i)+x(i+j);
  end
  y2(i)=y2(i)/M;
end

%
%       Moving Average Filter - Recursive
%

len_x=length(x);

y3=zeros(len_x,1);
```

```
y3((M-1)/2)=(sum(x(1:M)))/M;
y3(((M-1)/2)+1)=(sum(x(2:(M+1))))/M;

for i=(((M-1)/2)+2):len_x-(((M-1)/2)+1)
  y3(i)=((M*y3(i-1))+x(i+((M-1)/2))-x(i-(((M-1)/2)+1)))/M;
end

%
%       Convolution using M length rectangular pulse
%

h=zeros(M,1);
h(1:M)=1/M;


y4=conv(x,h);

%
%       Convolution using M length Blackman window
%

bk1=Blackman(M);
bk2=bk1/(trapz(bk1));
y5=conv(x,bk2);

%
%       Convolution using M length Gaussian
%

h2=conv(h,h);
gauss=conv(h2,h2);


gauss2=resample(gauss,M,((M+M-1)+(M+M-1)-1));
gauss3=gauss2/trapz(gauss2);

y6=conv(x,gauss3);
%
%
%       Compare results
%
t_axis2=(0:(1/fs):((length(y4)-1)*(1/fs)));
%
plot(t_axis,x,'b')
```

```
hold on
plot(t_axis.y.'r')
plot(t_axis.y2.'g')
plot(t_axis.y3.'c')
plot(t_axis2.y4.'m')
plot(t_axis2.y5.'k')
plot(t_axis2.y6.'y')

legend('input'.'output - one sided'.'output - symmetrical'.'output - recursive'.'output - conv
rec pulse'.'output - conv Blackman'.'output - conv Gaussian')
hold off
pause

%
%       Step response
%

s_h=cumtrapz(h); % step response of rectangular pulse

s_bk2=cumtrapz(bk2); %step response Blackman window

s_gauss3=cumtrapz(gauss3); %step response Blackman window

plot(s_h.'m')
hold on
plot(s_bk2.'k')
plot(s_gauss3.'y')
legend ('step response - rec pulse'.'step response - Blackman'.'step response - Gaussian')
pause
hold off

%
%       Freq. response
%

h_1024=zeros(1024.1);
h_1024(1:M)=h;
H_1024=fft(h_1024);

bk2_1024=zeros(1024.1);
bk2_1024(1:M)=bk2;
BK2_1024=fft(bk2_1024);

gauss3_1024=zeros(1024.1);
```

```
gauss3_1024(1:M)=gauss3;
Gauss3_1024=fft(gauss3_1024);

plot(f_axis.abs(H_1024(1:len_x/2)).'m')
hold on
plot(f_axis.abs(BK2_1024(1:len_x/2)).'k')
plot(f_axis.abs(Gauss3_1024(1:len_x/2)).'y')
legend ('freq response - rec pulse'.'freq response - Blackman'.'freq response - Gaussian')

%
%       Moving Average Filter - Symmetrical
%       Applied to delta function to obtain the
%       impulse response.
%

load delta2.dat

len_delta2=length(delta2);

imp=zeros(len_delta2.1);

for i=((M-1)/2)+1:(len_delta2-(M-1)/2)
  for j=(-(M-1)/2):((M-1)/2)
    imp(i)=imp(i)+delta2(i+j);
  end
  imp(i)=imp(i)/M;
end

freq=fft(imp.len_delta2);
step=cumtrapz(imp);

X=fft(x);
Y2=fft(y2);

hold off
plot(t_axis.x.'b')
hold on
plot(t_axis.y2.'r')
legend('x - input'.'y - output')
title('Moving Average Filter - Symmetrical')
pause

hold off
plot(f_axis.abs(X(1:(len_x/2))/(X(1))).'b')
```

```
hold on
plot(f_axis,abs(Y2(1:(len_x/2))/(Y2(1))),'r')
plot(f_axis,abs(freq(1:(len_x/2))),'m')
legend('X - fft of input','Y - fft of output','frequency response')
title('Moving Average Filter - Symmetrical')
%gtext(strcat('filter length = ',num2str(M)))
pause

hold off
plot(imp)
legend('impulse response')
title('Moving Average Filter - Symmetrical')
pause

hold off
plot(step)
legend('step response')
title('Moving Average Filter - Symmetrical')
pause

hold off
plot(f_axis,abs(freq(1:(len_x/2))))
legend('frequency response')
title('Moving Average Filter - Symmetrical')
pause

abs_freq=abs(freq);
abs_X=abs(X);
abs_Y2=abs(Y2);

break

%
%       Save files in ascii format.
%
eval(['save ' strcat('mov_avg_h_',num2str(M),'.dat') ' imp -ascii'])
eval(['save ' strcat('mov_avg_step_',num2str(M),'.dat') ' step -ascii'])
eval(['save ' strcat('mov_avg_freq_',num2str(M),'.dat') ' abs_freq -ascii'])
eval(['save ' strcat('mov_avg_input_',num2str(M),'.dat') ' x -ascii'])
eval(['save ' strcat('mov_avg_output_',num2str(M),'.dat') ' y2 -ascii'])
eval(['save ' strcat('mov_avg_input_fft_',num2str(M),'.dat') ' abs_X -ascii'])
eval(['save ' strcat('mov_avg_output_fft_',num2str(M),'.dat') ' abs_Y2 -ascii'])
```

```
%.....................................................................
%
%       Program:        single_pole_v2.m
%
%       Coded by:       F. Winsor
%
%       Based on Algorithms provided in:
%
%       "The Scientist and Engineer's Guide to
%        Digital Signal Processing - Second Edition".
%        Steven W. Smith. California Technical Publishing.
%        1997-1999.
%
%
%       Implement single pole low pass filter in
%       single stage and four stage form.
%
%
%........................ ..........................................
%
%       Input signal to be filtered
%
%
[filename,pathname] = uigetfile('*.dat','Input Signal',50,50);
filename=lower(filename);
eval(['load ',[pathname,filename],';'])
f=findstr(filename,'.');
x=eval(filename(1:f(1)-1));
%
%
FC=input('Cutoff Frequency - between 0.0 and 0.5 (default 0.5) :');
if isempty(FC)
  FC=0.5;
end
%
%
%       Input sample rate to calculate frequency
%       and time scales for plotting.
%
%
%
fs=input('Input Sample rate (default = 27.2168)');
if isempty(fs)
  fs=27.2168;
```

```
end
%
%
f_axis=fs*(0:511)/1024;
t_axis=(0:(1/fs):(1023*(1/fs)));
%
%
x_decay=exp(-2*pi*FC);
%
%       Single pole recursive low pass
%
a0=(1-x_decay);
b1=x_decay;
%
len_x=length(x);

y=zeros(len_x,1);

y(1)=a0*x(1);

for i=2:(len_x)
  y(i)=a0*x(i)+b1*y(i-1);
end
%
%
plot(t_axis,x)
hold on
plot(t_axis,y,'r')
legend('input','output - single pole recursive low pass')
hold off
pause
%
%       Four stage low pass
%
%
x_decay2=exp(-14.445*FC);
%
a0=(1-x_decay2)^4;
b1=4*x_decay2;
b2=-6*x_decay2^2;
b3=4*x_decay2^3;
b4=-x_decay2^4;
%
len_x=length(x);
```

```
y2=zeros(len_x,1);

y2(1)=a0*x(1);
y2(2)=a0*x(2)+b1*y2(1);
y2(3)=a0*x(3)+b1*y2(2)+b2*y2(1);


for i=4:(len_x)
  y2(i)=a0*x(i)+b1*y2(i-1)+b2*y2(i-2)+b3*y2(i-3);
end
%
%
hold off
plot(t_axis,x)
hold on
plot(t_axis,y,'r')
plot(t_axis,y2,'g')
legend('input','output - single pole recursive low pass','output - four stage low pass')
hold off
pause
%

%
X=fft(x,len_x);
Y=fft(y,len_x);
Y2=fft(y2,len_x);
%
hold off
plot(f_axis,abs(X(1:len_x/2)),'b')
hold on
plot(f_axis,abs(Y(1:len_x/2)),'r')
plot(f_axis,abs(Y2(1:len_x/2)),'m')
legend('input fft','output fft - single pole recursive low pass','output fft- four stage low
pass')
hold off
pause
%
%
%      Repeat procedures from above using a delta function
%      to obtain the impulse responses.
%
load delta2.dat
%
```

```
%
x_decay=exp(-2*pi*FC);
%
%       Single pole recursive low pass (for impulse response).
%
a0=(1-x_decay);
b1=x_decay;
%
len_delta2=length(delta2);

h1=zeros(len_delta2,1);

h1(1)=a0*delta2(1);

for i=2:(len_delta2)
  h1(i)=a0*delta2(i)+b1*h1(i-1);
end
%
%
%       Four stage low pass (for impulse response).
%
%
x_decay2=exp(-14.445*FC);
%
a0=(1-x_decay2)^4;
b1=4*x_decay2;
b2=-6*x_decay2^2;
b3=4*x_decay2^3;
b4=-x_decay2^4;
%
len_delta2=length(delta2);

h2=zeros(len_delta2,1);

h2(1)=a0*delta2(1);
h2(2)=a0*delta2(2)+b1*h2(1);
h2(3)=a0*delta2(3)+b1*h2(2)+b2*h2(1);


for i=4:(len_delta2)
  h2(i)=a0*delta2(i)+b1*h2(i-1)+b2*h2(i-2)+b3*h2(i-3);
end
%
%
```

```
H1=fft(h1,len_delta2);
H2=fft(h2,len_delta2);
%
step1=cumtrapz(h1);
step2=cumtrapz(h2);
%



abs_H1=abs(H1);
abs_H2=abs(H2);

abs_X=abs(X);
abs_Y=abs(Y);
abs_Y2=abs(Y2);

break

save sin_pole_h1.dat h1 -ascii
save sin_pole_step1.dat step1 -ascii
save sin_pole_freq1.dat abs_H1 -ascii

save sin_pole_h2.dat h2 -ascii
save sin_pole_step2.dat step2 -ascii
save sin_pole_freq2.dat abs_H2 -ascii

save sin_pole_input.dat x -ascii
save sin_pole_output1.dat y -ascii
save sin_pole_output2.dat y2 -ascii

save sin_pole_input_fft.dat abs_X -ascii
save sin_pole_output1_fft.dat abs_Y -ascii
save sin_pole_output2_fft.dat abs_Y2 -ascii
```

```
%.................................................................
%
%       Program:      win_sinc_v2.m
%
%       Coded by:     F. Winsor
%
%       Based on Algorithms provided in:
%
%       "The Scientist and Engineer's Guide to
%        Digital Signal Processing - Second Edition",
%        Steven W. Smith, California Technical Publishing,
%        1997-1999.
%
%
%       Windowed sinc filter is implemented. Filter kernel is generated
%       in two halves, about the point of symmetry, and then normalized.
%
%.................................................................
%
%       Input signal to be filtered
%
%
[filename,pathname] = uigetfile('*.dat','Input Signal',50,50);
filename=lower(filename);
eval(['load ',[pathname,filename],';'])
f=findstr(filename,'.');
x=eval(filename(1:f(1)-1));
%
M=input('Filter length - must be EVEN (default=10) :');
if isempty(M)
  M=10.0;
end
%
%
%       Input sample rate to calculate frequency
%       and time scales for plotting.
%
%
%
fs=input('Input Sample rate (default = 27.2168)');
if isempty(fs)
  fs=27.2168;
end
%
```

```
%
f_axis=fs*(0:511)/1024;
t_axis=(0:(1/fs):(1023*(1/fs)));
%
%
%
%
hold off
len_x=length(x);
X=fft(x,len_x);
plot(f_axis,abs(X(1:len_x/2)))
pause
%
%
%
FC=input('Cutoff Frequency - between 0.0 and 0.5 (default 0.5) :');
if isempty(FC)
  FC=0.5;
end
%
%       Calculate filter kernel
%
for i=1:((M/2))
  h(i)=((sin(2*pi*FC*(i-((M/2)+1))))/(i-((M/2)+1)))*(0.42 - 0.5*cos(2*pi*i/(M+1)) +
0.08*cos(4*pi*i/(M+1)));
end

h((M/2)+1)=2*pi*FC;

for i=((M/2)+2):M+1
  h(i)=((sin(2*pi*FC*(i-((M/2)+1))))/(i-((M/2)+1)))*(0.42 - 0.5*cos(2*pi*i/(M+1)) +
0.08*cos(4*pi*i/(M+1)));
end
%
%       Normalize filter kernel
%
h_n=zeros(1,len_x);
h_n(1:M+1)=h/sum(h);
%
% Convolution to produce filtered output
%
y=conv(h_n,x);
%
%
```

```
t_axis2=(0:(1/fs):((length(y)-1)*(1/fs)));

%
hold off
plot(t_axis.x)
hold on
plot(t_axis2.y.'r')
legend('x - input signal'.'y - conv using windowed sinc')
hold off
pause
%
% Determine step response
%
s_h_n=cumtrapz(h_n);
plot(s_h_n)
legend('step response')
pause
%
% Determine frequency response (padded to length of x)
%
Y=fft(y.length(x));
H_n=fft(h_n.length(x));
plot(f_axis.abs(X(1:len_x/2)/X(1)))
hold on
plot(f_axis.abs(H_n(1:(length(x)/2))).'r')
plot(f_axis.abs(Y(1:len_x/2)/Y(1)).'m')
legend('input fft'.'frequency response'.'output fft')
pause

hold off

abs_H_n=abs(H_n);
abs_X=abs(X);
abs_Y=abs(Y);

break

eval(['save ' strcat('win_sinc_h_'.num2str(M).'.dat') ' h_n -ascii'])
eval(['save ' strcat('win_sinc_step_'.num2str(M).'.dat') ' s_h_n -ascii'])
eval(['save ' strcat('win_sinc_freq_'.num2str(M).'.dat') ' abs_H_n -ascii'])
eval(['save ' strcat('win_sinc_input_'.num2str(M).'.dat') ' x -ascii'])
eval(['save ' strcat('win_sinc_output_'.num2str(M).'.dat') ' y -ascii'])
eval(['save ' strcat('win_sinc_input_fft_'.num2str(M).'.dat') ' abs_X -ascii'])
eval(['save ' strcat('win_sinc_output_fft_'.num2str(M).'.dat') ' abs_Y -ascii'])
```

```
%..................................................................
%
%       Program:        kaiser_lp_v3.m
%
%       Coded by:       F. Winsor
%
%
%
%       Use MATLAB commands kaiserord, fir1, and filter
%       to generate a kaiser window.
%
%
%
%..................................................................
%
%       Input signal to be filtered
%
%
[filename,pathname] = uigetfile('*.dat','Input Signal',50,50);
filename=lower(filename);
eval(['load ',[pathname,filename],';'])
f=findstr(filename,'.');
x=eval(filename(1:f(1)-1));
%
%
%
%
%       Input sample rate to calculate frequency
%       and time scales for plotting.
%
%
%
fs=input('Input Sample rate (default = 27.2168)');
if isempty(fs)
  fs=27.2168;
end
%
%
f_axis=fs*(0:511)/1024;
t_axis=(0:(1/fs):(1023*(1/fs)));
%
%
hold off
```

```
len_x=length(x);
X=fft(x,len_x);
plot(f_axis,abs(X(1:len_x/2)))
pause
%
%
fsamp=input('Sample Rate (default=1) :');% sampling rate
if isempty(fsamp)
  fsamp=1.0;
end
%
%
%       Defines start and stop of transition band
%
%
fcuts_1=input('Passband Edge  :');fcuts_2=input('Stopband Edge  :');
fcuts=[fcuts_1 fcuts_2];
%
fc1i=fcuts_1*1000;
fc2i=fcuts_2*1000;
%
%
%       Set desired amplitude of pass and stop bands
%
%
mags_1=input('Passband Amplitude (default=1.0)  :');
if isempty(mags_1)
  mags_1=1.0;
end
%
mags_2=input('Stopband Amplitude (default=0.0)  :');
if isempty(mags_2)
  mags_2=0.0;
end

mags=[mags_1 mags_2];
%
%
%       Deviation allowed in pass and stop band
%       two values are required but only the
%       lowest is used.
%
%
devs_pr=input('Percent Band Ripple (default=1)  :');
```

```
if isempty(devs_pr)

  devs_pr=1;

end

devs=[devs_pr/100 devs_pr/100];
%
%
[n,Wn,beta,ftype]=kaiserord(fcuts,mags,devs,fsamp);
%
%       n = filter order
%       Wn = normalized freq band edges
%       beta = Kaiser window parameter
%
%       Use MATLAB command fir1.m to determine
%       the impulse response.
%
hh=fir1(n,Wn,ftype,kaiser(n+1,beta));
%
%
%       Determine the filtered output signal.
%
%
y2=filter(hh,1,x);
%
%
%       Plot results.
%
%
plot(t_axis,x,'b')
hold on
plot(t_axis,y2,'r')
legend('input','output')
title('low pass filter - kaiser window')
pause

X=fft(x,length(x));
Y2=fft(y2,length(x));
HH=fft(hh,length(x));

hold off
plot(f_axis,abs(X(1:length(x)/2)/X(1)))
hold on
```

```
plot(f_axis,abs(Y2(1:length(x)/2))/Y2(1),'r')
plot(f_axis,abs(HH(1:length(x)/2)),'g')

legend('input fft','output fft', 'freq resp')
title('low pass filter - kaiser window')
pause

step=cumtrapz(hh);

hold off
plot(hh)
legend('hh - impulse response')
pause

hold off
plot(abs(HH(1:length(x)/2)),'b')
legend('HH - freq response')
pause

hold off
plot(step)
legend('step response')
pause

abs_HH=abs(HH);
abs_X=abs(X);
abs_Y2=abs(Y2);

break

eval(['save ' strcat('ksr_h_p',num2str(fc1i),'_s',num2str(fc2i),'_pr',num2str(devs_pr),'.dat')
' hh -ascii'])
eval(['save '
strcat('ksr_step_p',num2str(fc1i),'_s',num2str(fc2i),'_pr',num2str(devs_pr),'.dat') ' step -
ascii'])
eval(['save '
strcat('ksr_freq_p',num2str(fc1i),'_s',num2str(fc2i),'_pr',num2str(devs_pr),'.dat') ' abs_HH
-ascii'])
eval(['save '
strcat('ksr_input_p',num2str(fc1i),'_s',num2str(fc2i),'_pr',num2str(devs_pr),'.dat') ' x -
ascii'])
eval(['save '
strcat('ksr_output_p',num2str(fc1i),'_s',num2str(fc2i),'_pr',num2str(devs_pr),'.dat') ' y2 -
ascii'])
```

```
eval(['save '
strcat('ksr_input_fft_p',num2str(fc1i),'_s',num2str(fc2i),'_pr',num2str(devs_pr),'.dat') '
abs_X -ascii'])
eval(['save '
strcat('ksr_output_fft_p',num2str(fc1i),'_s',num2str(fc2i),'_pr',num2str(devs_pr),'.dat') '
abs_Y2 -ascii'])
```

```
%..............................................................
%
%       Program:       cheb_v3.m
%
%       Coded by:      F. Winsor
%
%       Based on Algorithms provided in:
%
%       "The Scientist and Engineer's Guide to
%        Digital Signal Processing - Second Edition".
%        Steven W. Smith. California Technical Publishing.
%        1997-1999.
%
%
%       Generates a Chebychev type 1 filter. Calls script
%       sub_cheb_v1.m
%
%
%
%
%..............................................................
%
%       Input signal to be filtered
%
%
%
[filename,pathname] = uigetfile('*.dat','Input Signal',50,50);
filename=lower(filename);
eval(['load ',[pathname,filename],';'])
f=findstr(filename,'.');
x=eval(filename(1:f(1)-1));
%
%
%
%
%
%
%       Input sample rate to calculate frequency
%       and time scales for plotting.
%
%
%
fs=input('Input Sample rate (default = 27.2168)');
if isempty(fs)
```

```
  fs=27.2168;
end
%
%
f_axis=fs*(0:511)/1024;
t_axis=(0:(1/fs):(1023*(1/fs)));
%
%
%
global FC LH PR NP k
global A0 A1 A2 B1 B2
%
%
for i=1:23
  A(i)=0;
  B(i)=0;
end
%
%
A(3)=1;
B(3)=1;
%
%
%       Input parameters
%
%
%
FC=input('Cutoff Freq (0 to 0.5)  :');
LH=input('Enter 0 for LP, 1 for HP  :');
PR=input('Percent ripple (0 to 29)  :');
NP=input('Number of poles (2,4,...20)  :');
%
%
%       Call subroutine sub_cheb_v1.m
%
%
for k=1:NP/2
  %
  %
  sub_cheb_v2
  %
  %
  for j=1:23
    TA(j)=A(j);
    TB(j)=B(j);
  end
  %
  %
  for j=3:23
    A(j)=A0*TA(j)+A1*TA(j-1)+A2*TA(j-2);
    B(j)=TB(j)-B1*TB(j-1)-B2*TB(j-2);
  end
  %
end
%
%
%       Combine the coefficients.
%
%
B(3)=0;
for i=1:21
  A(i)=A(i+2);
  B(i)=-B(i+2);
end
%
%
%       Normalize the gain.
%
%
%
SA=0;
SB=0;
for i=1:21
  if LH==0
    SA=SA+A(i);
  end

  if LH==0
    SB=SB+B(i);
  end

  if LH==1
    SA=SA+A(i)*(-1)^i;
  end

  if LH==1
    SB=SB+B(i)*(-1)^i;
  end
```

```
end
%
GAIN=SA/(1-SB);
%
%
for i=1:21
  A(i)=A(i)/GAIN;
end
%
%
%
%
len_x=length(x);

y=zeros(len_x,1);

%
%
%
A0_F=A(1);
A1_F=A(2);
A2_F=A(3);
A3_F=A(4);
A4_F=A(5);
A5_F=A(6);
A6_F=A(7);
A7_F=A(8);
A8_F=A(9);
A9_F=A(10);
A10_F=A(11);
A11_F=A(12);
A12_F=A(13);
A13_F=A(14);
A14_F=A(15);
A15_F=A(16);
A16_F=A(17);
A17_F=A(18);
A18_F=A(19);
A19_F=A(20);
A20_F=A(21);
A21_F=A(22);
A22_F=A(23);
%
B0_F=B(1);
```

```
B1_F=B(2);
B2_F=B(3);
B3_F=B(4);
B4_F=B(5);
B5_F=B(6);
B6_F=B(7);
B7_F=B(8);
B8_F=B(9);
B9_F=B(10);
B10_F=B(11);
B11_F=B(12);
B12_F=B(13);
B13_F=B(14);
B14_F=B(15);
B15_F=B(16);
B16_F=B(17);
B17_F=B(18);
B18_F=B(19);
B19_F=B(20);
B20_F=B(21);
B21_F=B(22);
B22_F=B(23);
%
%
%       Apply the recursion coefficients depending on the
%       number of poles selected.
%
%
%
if NP==2
        y(1)=A0_F*x(1);
        y(2)=A0_F*x(2)+A1_F*x(1)+B1_F*y(1);

        for i=3:(len_x)
         y(i)=A0_F*x(i)+A1_F*x(i-1)+A2_F*x(i-2)+B1_F*y(i-1)+B2_F*y(i-2);
        end
end
%
%
%
if NP==4
        y(1)=A0_F*x(1);
  y(2)=A0_F*x(2)+A1_F*x(1)+B1_F*y(1);
  y(3)=A0_F*x(3)+A1_F*x(2)+A2_F*x(1)+B1_F*y(2)+B2_F*y(1);
```

```
y(4)=A0_F*x(4)+A1_F*x(3)+A2_F*x(2)+A3_F*x(1)+B1_F*y(3)+B2_F*y(2)+B3_F*y(
1);

        for i=5:(len_x)
         y(i)=A0_F*x(i)+A1_F*x(i-1)+A2_F*x(i-2)+A3_F*x(i-3)+A4_F*x(i-
4)+B1_F*y(i-1)+B2_F*y(i-2)+B3_F*y(i-3)+B4_F*y(i-4);
        end
end
%
%
%
%
%
%
if NP==6
        y(1)=A0_F*x(1);
  y(2)=A0_F*x(2)+A1_F*x(1)+B1_F*y(1);
  y(3)=A0_F*x(3)+A1_F*x(2)+A2_F*x(1)+B1_F*y(2)+B2_F*y(1);

y(4)=A0_F*x(4)+A1_F*x(3)+A2_F*x(2)+A3_F*x(1)+B1_F*y(3)+B2_F*y(2)+B3_F*y(
1);

y(5)=A0_F*x(5)+A1_F*x(4)+A2_F*x(3)+A3_F*x(2)+A4_F*x(1)+B1_F*y(4)+B2_F*y(
3)+B3_F*y(2)+B4_F*y(1);
        y(6)=A0_F*x(6)+A1_F*x(5)+A2_F*x(4)+A3_F*x(3)+A4_F*x(2)+A5_F*x(1)+B
1_F*y(5)+B2_F*y(4)+B3_F*y(3)+B4_F*y(2)+B5_F*y(1);

        for i=7:(len_x)
         y(i)=A0_F*x(i)+A1_F*x(i-1)+A2_F*x(i-2)+A3_F*x(i-3)+A4_F*x(i-
4)+A5_F*x(i-5)+A6_F*x(i-6)+B1_F*y(i-1)+B2_F*y(i-2)+B3_F*y(i-3)+B4_F*y(i-
4)+B5_F*y(i-5)+B6_F*y(i-6);
        end
end
%
%
%
%
if NP==8
        y(1)=A0_F*x(1);
  y(2)=A0_F*x(2)+A1_F*x(1)+B1_F*y(1);
  y(3)=A0_F*x(3)+A1_F*x(2)+A2_F*x(1)+B1_F*y(2)+B2_F*y(1);
```

```
y(4)=A0_F*x(4)+A1_F*x(3)+A2_F*x(2)+A3_F*x(1)+B1_F*y(3)+B2_F*y(2)+B3_F*y(
1);

y(5)=A0_F*x(5)+A1_F*x(4)+A2_F*x(3)+A3_F*x(2)+A4_F*x(1)+B1_F*y(4)+B2_F*y(
3)+B3_F*y(2)+B4_F*y(1);
        y(6)=A0_F*x(6)+A1_F*x(5)+A2_F*x(4)+A3_F*x(3)+A4_F*x(2)+A5_F*x(1)+B
1_F*y(5)+B2_F*y(4)+B3_F*y(3)+B4_F*y(2)+B5_F*y(1);
        y(7)=A0_F*x(7)+A1_F*x(6)+A2_F*x(5)+A3_F*x(4)+A4_F*x(3)+A5_F*x(2)+
A6_F*x(1)+B1_F*y(6)+B2_F*y(5)+B3_F*y(4)+B4_F*y(3)+B5_F*y(2)+B6_F*y(1);
        y(8)=A0_F*x(8)+A1_F*x(7)+A2_F*x(6)+A3_F*x(5)+A4_F*x(4)+A5_F*x(3)+
A6_F*x(2)+A7_F*x(1)+B1_F*y(7)+B2_F*y(6)+B3_F*y(5)+B4_F*y(4)+B5_F*y(3)+B6
_F*y(2)+B7_F*y(1);

  for i=9:(len_x)
         y(i)=A0_F*x(i)+A1_F*x(i-1)+A2_F*x(i-2)+A3_F*x(i-3)+A4_F*x(i-
4)+A5_F*x(i-5)+A6_F*x(i-6)+A7_F*x(i-7)+A8_F*x(i-8)+B1_F*y(i-1)+B2_F*y(i-
2)+B3_F*y(i-3)+B4_F*y(i-4)+B5_F*y(i-5)+B6_F*y(i-6)+B7_F*y(i-7)+B8_F*y(i-8);
        end
end
%
%
if NP==10
        y(1)=A0_F*x(1);
  y(2)=A0_F*x(2)+A1_F*x(1)+B1_F*y(1);
  y(3)=A0_F*x(3)+A1_F*x(2)+A2_F*x(1)+B1_F*y(2)+B2_F*y(1);

y(4)=A0_F*x(4)+A1_F*x(3)+A2_F*x(2)+A3_F*x(1)+B1_F*y(3)+B2_F*y(2)+B3_F*y(
1);

y(5)=A0_F*x(5)+A1_F*x(4)+A2_F*x(3)+A3_F*x(2)+A4_F*x(1)+B1_F*y(4)+B2_F*y(
3)+B3_F*y(2)+B4_F*y(1);
        y(6)=A0_F*x(6)+A1_F*x(5)+A2_F*x(4)+A3_F*x(3)+A4_F*x(2)+A5_F*x(1)+B
1_F*y(5)+B2_F*y(4)+B3_F*y(3)+B4_F*y(2)+B5_F*y(1);
        y(7)=A0_F*x(7)+A1_F*x(6)+A2_F*x(5)+A3_F*x(4)+A4_F*x(3)+A5_F*x(2)+
A6_F*x(1)+B1_F*y(6)+B2_F*y(5)+B3_F*y(4)+B4_F*y(3)+B5_F*y(2)+B6_F*y(1);
        y(8)=A0_F*x(8)+A1_F*x(7)+A2_F*x(6)+A3_F*x(5)+A4_F*x(4)+A5_F*x(3)+
A6_F*x(2)+A7_F*x(1)+B1_F*y(7)+B2_F*y(6)+B3_F*y(5)+B4_F*y(4)+B5_F*y(3)+B6
_F*y(2)+B7_F*y(1);

y(9)=A0_F*x(9)+A1_F*x(8)+A2_F*x(7)+A3_F*x(6)+A4_F*x(5)+A5_F*x(4)+A6_F*x(
3)+A7_F*x(2)+A8_F*x(1)+B1_F*y(8)+B2_F*y(7)+B3_F*y(6)+B4_F*y(5)+B5_F*y(4)
+B6_F*y(3)+B7_F*y(2)+B8_F*y(1);
```

```
y(10)=A0_F*x(10)+A1_F*x(9)+A2_F*x(8)+A3_F*x(7)+A4_F*x(6)+A5_F*x(5)+A6_F
*x(4)+A7_F*x(3)+A8_F*x(2)+A9_F*x(1)+B1_F*y(9)+B2_F*y(8)+B3_F*y(7)+B4_F*y
(6)+B5_F*y(5)+B6_F*y(4)+B7_F*y(3)+B8_F*y(2)+B9_F*y(1);

  for i=11:(len_x)
          y(i)=A0_F*x(i)+A1_F*x(i-1)+A2_F*x(i-2)+A3_F*x(i-3)+A4_F*x(i-
4)+A5_F*x(i-5)+A6_F*x(i-6)+A7_F*x(i-7)+A8_F*x(i-8)+A9_F*x(i-9)+A10_F*x(i-
10)+B1_F*y(i-1)+B2_F*y(i-2)+B3_F*y(i-3)+B4_F*y(i-4)+B5_F*y(i-5)+B6_F*y(i-
6)+B7_F*y(i-7)+B8_F*y(i-8)+B9_F*y(i-9)+B10_F*y(i-10);
        end
end
%
%
if NP==12
        y(1)=A0_F*x(1);
  y(2)=A0_F*x(2)+A1_F*x(1)+B1_F*y(1);
  y(3)=A0_F*x(3)+A1_F*x(2)+A2_F*x(1)+B1_F*y(2)+B2_F*y(1);

y(4)=A0_F*x(4)+A1_F*x(3)+A2_F*x(2)+A3_F*x(1)+B1_F*y(3)+B2_F*y(2)+B3_F*y(
1);

y(5)=A0_F*x(5)+A1_F*x(4)+A2_F*x(3)+A3_F*x(2)+A4_F*x(1)+B1_F*y(4)+B2_F*y(
3)+B3_F*y(2)+B4_F*y(1);
        y(6)=A0_F*x(6)+A1_F*x(5)+A2_F*x(4)+A3_F*x(3)+A4_F*x(2)+A5_F*x(1)+B
1_F*y(5)+B2_F*y(4)+B3_F*y(3)+B4_F*y(2)+B5_F*y(1);
        y(7)=A0_F*x(7)+A1_F*x(6)+A2_F*x(5)+A3_F*x(4)+A4_F*x(3)+A5_F*x(2)+
A6_F*x(1)+B1_F*y(6)+B2_F*y(5)+B3_F*y(4)+B4_F*y(3)+B5_F*y(2)+B6_F*y(1);
        y(8)=A0_F*x(8)+A1_F*x(7)+A2_F*x(6)+A3_F*x(5)+A4_F*x(4)+A5_F*x(3)+
A6_F*x(2)+A7_F*x(1)+B1_F*y(7)+B2_F*y(6)+B3_F*y(5)+B4_F*y(4)+B5_F*y(3)+B6
_F*y(2)+B7_F*y(1);

y(9)=A0_F*x(9)+A1_F*x(8)+A2_F*x(7)+A3_F*x(6)+A4_F*x(5)+A5_F*x(4)+A6_F*x(
3)+A7_F*x(2)+A8_F*x(1)+B1_F*y(8)+B2_F*y(7)+B3_F*y(6)+B4_F*y(5)+B5_F*y(4)
+B6_F*y(3)+B7_F*y(2)+B8_F*y(1);

y(10)=A0_F*x(10)+A1_F*x(9)+A2_F*x(8)+A3_F*x(7)+A4_F*x(6)+A5_F*x(5)+A6_F
*x(4)+A7_F*x(3)+A8_F*x(2)+A9_F*x(1)+B1_F*y(9)+B2_F*y(8)+B3_F*y(7)+B4_F*y
(6)+B5_F*y(5)+B6_F*y(4)+B7_F*y(3)+B8_F*y(2)+B9_F*y(1);

y(11)=A0_F*x(11)+A1_F*x(10)+A2_F*x(9)+A3_F*x(8)+A4_F*x(7)+A5_F*x(6)+A6_
F*x(5)+A7_F*x(4)+A8_F*x(3)+A9_F*x(2)+A10_F*x(1)+B1_F*y(10)+B2_F*y(9)+B3_
F*y(8)+B4_F*y(7)+B5_F*y(6)+B6_F*y(5)+B7_F*y(4)+B8_F*y(3)+B9_F*y(2)+B10_F
*y(1);
```

```
        y(12)=A0_F*x(12)+A1_F*x(11)+A2_F*x(10)+A3_F*x(9)+A4_F*x(8)+A5_F*x(
7)+A6_F*x(6)+A7_F*x(5)+A8_F*x(4)+A9_F*x(3)+A10_F*x(2)+A11_F*x(1)+B1_F*y
(11)+B2_F*y(10)+B3_F*y(9)+B4_F*y(8)+B5_F*y(7)+B6_F*y(6)+B7_F*y(5)+B8_F*y
(4)+B9_F*y(3)+B10_F*y(2)+B11_F*y(1);

  for i=13:(len_x)
          y(i)=A0_F*x(i)+A1_F*x(i-1)+A2_F*x(i-2)+A3_F*x(i-3)+A4_F*x(i-
4)+A5_F*x(i-5)+A6_F*x(i-6)+A7_F*x(i-7)+A8_F*x(i-8)+A9_F*x(i-9)+A10_F*x(i-
10)+A11_F*x(i-11)+A12_F*x(i-12)+B1_F*y(i-1)+B2_F*y(i-2)+B3_F*y(i-3)+B4_F*y(i-
4)+B5_F*y(i-5)+B6_F*y(i-6)+B7_F*y(i-7)+B8_F*y(i-8)+B9_F*y(i-9)+B10_F*y(i-
10)+B11_F*y(i-11)+B12_F*y(i-12);
        end
end
%
%
if NP==14
        y(1)=A0_F*x(1);
  y(2)=A0_F*x(2)+A1_F*x(1)+B1_F*y(1);
  y(3)=A0_F*x(3)+A1_F*x(2)+A2_F*x(1)+B1_F*y(2)+B2_F*y(1);

y(4)=A0_F*x(4)+A1_F*x(3)+A2_F*x(2)+A3_F*x(1)+B1_F*y(3)+B2_F*y(2)+B3_F*y(
1);

y(5)=A0_F*x(5)+A1_F*x(4)+A2_F*x(3)+A3_F*x(2)+A4_F*x(1)+B1_F*y(4)+B2_F*y(
3)+B3_F*y(2)+B4_F*y(1);
        y(6)=A0_F*x(6)+A1_F*x(5)+A2_F*x(4)+A3_F*x(3)+A4_F*x(2)+A5_F*x(1)+B
1_F*y(5)+B2_F*y(4)+B3_F*y(3)+B4_F*y(2)+B5_F*y(1);
        y(7)=A0_F*x(7)+A1_F*x(6)+A2_F*x(5)+A3_F*x(4)+A4_F*x(3)+A5_F*x(2)+
A6_F*x(1)+B1_F*y(6)+B2_F*y(5)+B3_F*y(4)+B4_F*y(3)+B5_F*y(2)+B6_F*y(1);
        y(8)=A0_F*x(8)+A1_F*x(7)+A2_F*x(6)+A3_F*x(5)+A4_F*x(4)+A5_F*x(3)+
A6_F*x(2)+A7_F*x(1)+B1_F*y(7)+B2_F*y(6)+B3_F*y(5)+B4_F*y(4)+B5_F*y(3)+B6
_F*y(2)+B7_F*y(1);

y(9)=A0_F*x(9)+A1_F*x(8)+A2_F*x(7)+A3_F*x(6)+A4_F*x(5)+A5_F*x(4)+A6_F*x(
3)+A7_F*x(2)+A8_F*x(1)+B1_F*y(8)+B2_F*y(7)+B3_F*y(6)+B4_F*y(5)+B5_F*y(4)
+B6_F*y(3)+B7_F*y(2)+B8_F*y(1);

y(10)=A0_F*x(10)+A1_F*x(9)+A2_F*x(8)+A3_F*x(7)+A4_F*x(6)+A5_F*x(5)+A6_F
*x(4)+A7_F*x(3)+A8_F*x(2)+A9_F*x(1)+B1_F*y(9)+B2_F*y(8)+B3_F*y(7)+B4_F*y
(6)+B5_F*y(5)+B6_F*y(4)+B7_F*y(3)+B8_F*y(2)+B9_F*y(1);

y(11)=A0_F*x(11)+A1_F*x(10)+A2_F*x(9)+A3_F*x(8)+A4_F*x(7)+A5_F*x(6)+A6_
F*x(5)+A7_F*x(4)+A8_F*x(3)+A9_F*x(2)+A10_F*x(1)+B1_F*y(10)+B2_F*y(9)+B3_
```

```
F*y(8)+B4_F*y(7)+B5_F*y(6)+B6_F*y(5)+B7_F*y(4)+B8_F*y(3)+B9_F*y(2)+B10_F
*y(1);
        y(12)=A0_F*x(12)+A1_F*x(11)+A2_F*x(10)+A3_F*x(9)+A4_F*x(8)+A5_F*x(
7)+A6_F*x(6)+A7_F*x(5)+A8_F*x(4)+A9_F*x(3)+A10_F*x(2)+A11_F*x(1)+B1_F*y
(11)+B2_F*y(10)+B3_F*y(9)+B4_F*y(8)+B5_F*y(7)+B6_F*y(6)+B7_F*y(5)+B8_F*y
(4)+B9_F*y(3)+B10_F*y(2)+B11_F*y(1);
        y(13)=A0_F*x(13)+A1_F*x(12)+A2_F*x(11)+A3_F*x(10)+A4_F*x(9)+A5_F*
x(8)+A6_F*x(7)+A7_F*x(6)+A8_F*x(5)+A9_F*x(4)+A10_F*x(3)+A11_F*x(2)+A12_
F*x(1)+B1_F*y(12)+B2_F*y(11)+B3_F*y(10)+B4_F*y(9)+B5_F*y(8)+B6_F*y(7)+B7
_F*y(6)+B8_F*y(5)+B9_F*y(4)+B10_F*y(3)+B11_F*y(2)+B12_F*y(1);
        y(14)=A0_F*x(14)+A1_F*x(13)+A2_F*x(12)+A3_F*x(11)+A4_F*x(10)+A5_F
*x(9)+A6_F*x(8)+A7_F*x(7)+A8_F*x(6)+A9_F*x(5)+A10_F*x(4)+A11_F*x(3)+A12
_F*x(2)+A13_F*x(1)+B1_F*y(13)+B2_F*y(12)+B3_F*y(11)+B4_F*y(10)+B5_F*y(9)
+B6_F*y(8)+B7_F*y(7)+B8_F*y(6)+B9_F*y(5)+B10_F*y(4)+B11_F*y(3)+B12_F*y(2
)+B13_F*y(1);

  for i=15:(len_x)
         y(i)=A0_F*x(i)+A1_F*x(i-1)+A2_F*x(i-2)+A3_F*x(i-3)+A4_F*x(i-
4)+A5_F*x(i-5)+A6_F*x(i-6)+A7_F*x(i-7)+A8_F*x(i-8)+A9_F*x(i-9)+A10_F*x(i-
10)+A11_F*x(i-11)+A12_F*x(i-12)+A13_F*x(i-13)+A14_F*x(i-14)+B1_F*y(i-
1)+B2_F*y(i-2)+B3_F*y(i-3)+B4_F*y(i-4)+B5_F*y(i-5)+B6_F*y(i-6)+B7_F*y(i-
7)+B8_F*y(i-8)+B9_F*y(i-9)+B10_F*y(i-10)+B11_F*y(i-11)+B12_F*y(i-
12)+B13_F*y(i-13)+B14_F*y(i-14);
        end
end
%
%
if NP==16
        y(1)=A0_F*x(1);
  y(2)=A0_F*x(2)+A1_F*x(1)+B1_F*y(1);
  y(3)=A0_F*x(3)+A1_F*x(2)+A2_F*x(1)+B1_F*y(2)+B2_F*y(1);

y(4)=A0_F*x(4)+A1_F*x(3)+A2_F*x(2)+A3_F*x(1)+B1_F*y(3)+B2_F*y(2)+B3_F*y(
1);

y(5)=A0_F*x(5)+A1_F*x(4)+A2_F*x(3)+A3_F*x(2)+A4_F*x(1)+B1_F*y(4)+B2_F*y(
3)+B3_F*y(2)+B4_F*y(1);
        y(6)=A0_F*x(6)+A1_F*x(5)+A2_F*x(4)+A3_F*x(3)+A4_F*x(2)+A5_F*x(1)+B
1_F*y(5)+B2_F*y(4)+B3_F*y(3)+B4_F*y(2)+B5_F*y(1);
        y(7)=A0_F*x(7)+A1_F*x(6)+A2_F*x(5)+A3_F*x(4)+A4_F*x(3)+A5_F*x(2)+
A6_F*x(1)+B1_F*y(6)+B2_F*y(5)+B3_F*y(4)+B4_F*y(3)+B5_F*y(2)+B6_F*y(1);
        y(8)=A0_F*x(8)+A1_F*x(7)+A2_F*x(6)+A3_F*x(5)+A4_F*x(4)+A5_F*x(3)+
A6_F*x(2)+A7_F*x(1)+B1_F*y(7)+B2_F*y(6)+B3_F*y(5)+B4_F*y(4)+B5_F*y(3)+B6
_F*y(2)+B7_F*y(1);
```

```
y(9)=A0_F*x(9)+A1_F*x(8)+A2_F*x(7)+A3_F*x(6)+A4_F*x(5)+A5_F*x(4)+A6_F*x(
3)+A7_F*x(2)+A8_F*x(1)+B1_F*y(8)+B2_F*y(7)+B3_F*y(6)+B4_F*y(5)+B5_F*y(4)
+B6_F*y(3)+B7_F*y(2)+B8_F*y(1);

y(10)=A0_F*x(10)+A1_F*x(9)+A2_F*x(8)+A3_F*x(7)+A4_F*x(6)+A5_F*x(5)+A6_F
*x(4)+A7_F*x(3)+A8_F*x(2)+A9_F*x(1)+B1_F*y(9)+B2_F*y(8)+B3_F*y(7)+B4_F*y
(6)+B5_F*y(5)+B6_F*y(4)+B7_F*y(3)+B8_F*y(2)+B9_F*y(1);

y(11)=A0_F*x(11)+A1_F*x(10)+A2_F*x(9)+A3_F*x(8)+A4_F*x(7)+A5_F*x(6)+A6_
F*x(5)+A7_F*x(4)+A8_F*x(3)+A9_F*x(2)+A10_F*x(1)+B1_F*y(10)+B2_F*y(9)+B3_
F*y(8)+B4_F*y(7)+B5_F*y(6)+B6_F*y(5)+B7_F*y(4)+B8_F*y(3)+B9_F*y(2)+B10_F
*y(1);
      y(12)=A0_F*x(12)+A1_F*x(11)+A2_F*x(10)+A3_F*x(9)+A4_F*x(8)+A5_F*x(
7)+A6_F*x(6)+A7_F*x(5)+A8_F*x(4)+A9_F*x(3)+A10_F*x(2)+A11_F*x(1)+B1_F*y
(11)+B2_F*y(10)+B3_F*y(9)+B4_F*y(8)+B5_F*y(7)+B6_F*y(6)+B7_F*y(5)+B8_F*y
(4)+B9_F*y(3)+B10_F*y(2)+B11_F*y(1);
      y(13)=A0_F*x(13)+A1_F*x(12)+A2_F*x(11)+A3_F*x(10)+A4_F*x(9)+A5_F*
x(8)+A6_F*x(7)+A7_F*x(6)+A8_F*x(5)+A9_F*x(4)+A10_F*x(3)+A11_F*x(2)+A12_
F*x(1)+B1_F*y(12)+B2_F*y(11)+B3_F*y(10)+B4_F*y(9)+B5_F*y(8)+B6_F*y(7)+B7
_F*y(6)+B8_F*y(5)+B9_F*y(4)+B10_F*y(3)+B11_F*y(2)+B12_F*y(1);
      y(14)=A0_F*x(14)+A1_F*x(13)+A2_F*x(12)+A3_F*x(11)+A4_F*x(10)+A5_F
*x(9)+A6_F*x(8)+A7_F*x(7)+A8_F*x(6)+A9_F*x(5)+A10_F*x(4)+A11_F*x(3)+A12
_F*x(2)+A13_F*x(1)+B1_F*y(13)+B2_F*y(12)+B3_F*y(11)+B4_F*y(10)+B5_F*y(9)
+B6_F*y(8)+B7_F*y(7)+B8_F*y(6)+B9_F*y(5)+B10_F*y(4)+B11_F*y(3)+B12_F*y(2
)+B13_F*y(1);
      y(15)=A0_F*x(15)+A1_F*x(14)+A2_F*x(13)+A3_F*x(12)+A4_F*x(11)+A5_F
*x(10)+A6_F*x(9)+A7_F*x(8)+A8_F*x(7)+A9_F*x(6)+A10_F*x(5)+A11_F*x(4)+A1
2_F*x(3)+A13_F*x(2)+A14_F*x(1)+B1_F*y(14)+B2_F*y(13)+B3_F*y(12)+B4_F*y(1
1)+B5_F*y(10)+B6_F*y(9)+B7_F*y(8)+B8_F*y(7)+B9_F*y(6)+B10_F*y(5)+B11_F*y
(4)+B12_F*y(3)+B13_F*y(2)+B14_F*y(1);
      y(16)=A0_F*x(16)+A1_F*x(15)+A2_F*x(14)+A3_F*x(13)+A4_F*x(12)+A5_F
*x(11)+A6_F*x(10)+A7_F*x(9)+A8_F*x(8)+A9_F*x(7)+A10_F*x(6)+A11_F*x(5)+A
12_F*x(4)+A13_F*x(3)+A14_F*x(2)+A15_F*x(1)+B1_F*y(15)+B2_F*y(14)+B3_F*y(
13)+B4_F*y(12)+B5_F*y(11)+B6_F*y(10)+B7_F*y(9)+B8_F*y(8)+B9_F*y(7)+B10_F
*y(6)+B11_F*y(5)+B12_F*y(4)+B13_F*y(3)+B14_F*y(2)+B15_F*y(1);

 for i=17:(len_x)
       y(i)=A0_F*x(i)+A1_F*x(i-1)+A2_F*x(i-2)+A3_F*x(i-3)+A4_F*x(i-
4)+A5_F*x(i-5)+A6_F*x(i-6)+A7_F*x(i-7)+A8_F*x(i-8)+A9_F*x(i-9)+A10_F*x(i-
10)+A11_F*x(i-11)+A12_F*x(i-12)+A13_F*x(i-13)+A14_F*x(i-14)+A15_F*x(i-
15)+A16_F*x(i-16)+B1_F*y(i-1)+B2_F*y(i-2)+B3_F*y(i-3)+B4_F*y(i-4)+B5_F*y(i-
5)+B6_F*y(i-6)+B7_F*y(i-7)+B8_F*y(i-8)+B9_F*y(i-9)+B10_F*y(i-10)+B11_F*y(i-
11)+B12_F*y(i-12)+B13_F*y(i-13)+B14_F*y(i-14)+B15_F*y(i-15)+B16_F*y(i-16);
```

```
        end
end
%
%
if NP==18
        y(1)=A0_F*x(1);
  y(2)=A0_F*x(2)+A1_F*x(1)+B1_F*y(1);
  y(3)=A0_F*x(3)+A1_F*x(2)+A2_F*x(1)+B1_F*y(2)+B2_F*y(1);

y(4)=A0_F*x(4)+A1_F*x(3)+A2_F*x(2)+A3_F*x(1)+B1_F*y(3)+B2_F*y(2)+B3_F*y(
1);

y(5)=A0_F*x(5)+A1_F*x(4)+A2_F*x(3)+A3_F*x(2)+A4_F*x(1)+B1_F*y(4)+B2_F*y(
3)+B3_F*y(2)+B4_F*y(1);
        y(6)=A0_F*x(6)+A1_F*x(5)+A2_F*x(4)+A3_F*x(3)+A4_F*x(2)+A5_F*x(1)+B
1_F*y(5)+B2_F*y(4)+B3_F*y(3)+B4_F*y(2)+B5_F*y(1);
        y(7)=A0_F*x(7)+A1_F*x(6)+A2_F*x(5)+A3_F*x(4)+A4_F*x(3)+A5_F*x(2)+
A6_F*x(1)+B1_F*y(6)+B2_F*y(5)+B3_F*y(4)+B4_F*y(3)+B5_F*y(2)+B6_F*y(1);
        y(8)=A0_F*x(8)+A1_F*x(7)+A2_F*x(6)+A3_F*x(5)+A4_F*x(4)+A5_F*x(3)+
A6_F*x(2)+A7_F*x(1)+B1_F*y(7)+B2_F*y(6)+B3_F*y(5)+B4_F*y(4)+B5_F*y(3)+B6
_F*y(2)+B7_F*y(1);

y(9)=A0_F*x(9)+A1_F*x(8)+A2_F*x(7)+A3_F*x(6)+A4_F*x(5)+A5_F*x(4)+A6_F*x(
3)+A7_F*x(2)+A8_F*x(1)+B1_F*y(8)+B2_F*y(7)+B3_F*y(6)+B4_F*y(5)+B5_F*y(4)
+B6_F*y(3)+B7_F*y(2)+B8_F*y(1);

y(10)=A0_F*x(10)+A1_F*x(9)+A2_F*x(8)+A3_F*x(7)+A4_F*x(6)+A5_F*x(5)+A6_F
*x(4)+A7_F*x(3)+A8_F*x(2)+A9_F*x(1)+B1_F*y(9)+B2_F*y(8)+B3_F*y(7)+B4_F*y
(6)+B5_F*y(5)+B6_F*y(4)+B7_F*y(3)+B8_F*y(2)+B9_F*y(1);

y(11)=A0_F*x(11)+A1_F*x(10)+A2_F*x(9)+A3_F*x(8)+A4_F*x(7)+A5_F*x(6)+A6_
F*x(5)+A7_F*x(4)+A8_F*x(3)+A9_F*x(2)+A10_F*x(1)+B1_F*y(10)+B2_F*y(9)+B3_
F*y(8)+B4_F*y(7)+B5_F*y(6)+B6_F*y(5)+B7_F*y(4)+B8_F*y(3)+B9_F*y(2)+B10_F
*y(1);
        y(12)=A0_F*x(12)+A1_F*x(11)+A2_F*x(10)+A3_F*x(9)+A4_F*x(8)+A5_F*x(
7)+A6_F*x(6)+A7_F*x(5)+A8_F*x(4)+A9_F*x(3)+A10_F*x(2)+A11_F*x(1)+B1_F*y
(11)+B2_F*y(10)+B3_F*y(9)+B4_F*y(8)+B5_F*y(7)+B6_F*y(6)+B7_F*y(5)+B8_F*y
(4)+B9_F*y(3)+B10_F*y(2)+B11_F*y(1);
        y(13)=A0_F*x(13)+A1_F*x(12)+A2_F*x(11)+A3_F*x(10)+A4_F*x(9)+A5_F*
x(8)+A6_F*x(7)+A7_F*x(6)+A8_F*x(5)+A9_F*x(4)+A10_F*x(3)+A11_F*x(2)+A12_
F*x(1)+B1_F*y(12)+B2_F*y(11)+B3_F*y(10)+B4_F*y(9)+B5_F*y(8)+B6_F*y(7)+B7
_F*y(6)+B8_F*y(5)+B9_F*y(4)+B10_F*y(3)+B11_F*y(2)+B12_F*y(1);
        y(14)=A0_F*x(14)+A1_F*x(13)+A2_F*x(12)+A3_F*x(11)+A4_F*x(10)+A5_F
*x(9)+A6_F*x(8)+A7_F*x(7)+A8_F*x(6)+A9_F*x(5)+A10_F*x(4)+A11_F*x(3)+A12
```

```
_F*x(2)+A13_F*x(1)+B1_F*y(13)+B2_F*y(12)+B3_F*y(11)+B4_F*y(10)+B5_F*y(9)
+B6_F*y(8)+B7_F*y(7)+B8_F*y(6)+B9_F*y(5)+B10_F*y(4)+B11_F*y(3)+B12_F*y(2
)+B13_F*y(1);
        y(15)=A0_F*x(15)+A1_F*x(14)+A2_F*x(13)+A3_F*x(12)+A4_F*x(11)+A5_F
*x(10)+A6_F*x(9)+A7_F*x(8)+A8_F*x(7)+A9_F*x(6)+A10_F*x(5)+A11_F*x(4)+A1
2_F*x(3)+A13_F*x(2)+A14_F*x(1)+B1_F*y(14)+B2_F*y(13)+B3_F*y(12)+B4_F*y(1
1)+B5_F*y(10)+B6_F*y(9)+B7_F*y(8)+B8_F*y(7)+B9_F*y(6)+B10_F*y(5)+B11_F*y
(4)+B12_F*y(3)+B13_F*y(2)+B14_F*y(1);
        y(16)=A0_F*x(16)+A1_F*x(15)+A2_F*x(14)+A3_F*x(13)+A4_F*x(12)+A5_F
*x(11)+A6_F*x(10)+A7_F*x(9)+A8_F*x(8)+A9_F*x(7)+A10_F*x(6)+A11_F*x(5)+A
12_F*x(4)+A13_F*x(3)+A14_F*x(2)+A15_F*x(1)+B1_F*y(15)+B2_F*y(14)+B3_F*y(
13)+B4_F*y(12)+B5_F*y(11)+B6_F*y(10)+B7_F*y(9)+B8_F*y(8)+B9_F*y(7)+B10_F
*y(6)+B11_F*y(5)+B12_F*y(4)+B13_F*y(3)+B14_F*y(2)+B15_F*y(1);
        y(17)=A0_F*x(17)+A1_F*x(16)+A2_F*x(15)+A3_F*x(14)+A4_F*x(13)+A5_F
*x(12)+A6_F*x(11)+A7_F*x(10)+A8_F*x(9)+A9_F*x(8)+A10_F*x(7)+A11_F*x(6)+
A12_F*x(5)+A13_F*x(4)+A14_F*x(3)+A15_F*x(2)+A16_F*x(1)+B1_F*y(16)+B2_F*
y(15)+B3_F*y(14)+B4_F*y(13)+B5_F*y(12)+B6_F*y(11)+B7_F*y(10)+B8_F*y(9)+B
9_F*y(8)+B10_F*y(7)+B11_F*y(6)+B12_F*y(5)+B13_F*y(4)+B14_F*y(3)+B15_F*y(
2)+B16_F*y(1);
        y(18)=A0_F*x(18)+A1_F*x(17)+A2_F*x(16)+A3_F*x(15)+A4_F*x(14)+A5_F
*x(13)+A6_F*x(12)+A7_F*x(11)+A8_F*x(10)+A9_F*x(9)+A10_F*x(8)+A11_F*x(7)+
A12_F*x(6)+A13_F*x(5)+A14_F*x(4)+A15_F*x(3)+A16_F*x(2)+A17_F*x(1)+B1_F*
y(17)+B2_F*y(16)+B3_F*y(15)+B4_F*y(14)+B5_F*y(13)+B6_F*y(12)+B7_F*y(11)+
B8_F*y(10)+B9_F*y(9)+B10_F*y(8)+B11_F*y(7)+B12_F*y(6)+B13_F*y(5)+B14_F*y
(4)+B15_F*y(3)+B16_F*y(2)+B17_F*y(1);

  for i=19:(len_x)
         y(i)=A0_F*x(i)+A1_F*x(i-1)+A2_F*x(i-2)+A3_F*x(i-3)+A4_F*x(i-
4)+A5_F*x(i-5)+A6_F*x(i-6)+A7_F*x(i-7)+A8_F*x(i-8)+A9_F*x(i-9)+A10_F*x(i-
10)+A11_F*x(i-11)+A12_F*x(i-12)+A13_F*x(i-13)+A14_F*x(i-14)+A15_F*x(i-
15)+A16_F*x(i-16)+A17_F*x(i-17)+A18_F*x(i-18)+B1_F*y(i-1)+B2_F*y(i-
2)+B3_F*y(i-3)+B4_F*y(i-4)+B5_F*y(i-5)+B6_F*y(i-6)+B7_F*y(i-7)+B8_F*y(i-
8)+B9_F*y(i-9)+B10_F*y(i-10)+B11_F*y(i-11)+B12_F*y(i-12)+B13_F*y(i-
13)+B14_F*y(i-14)+B15_F*y(i-15)+B16_F*y(i-16)+B17_F*y(i-17)+B18_F*y(i-18);
        end
end
%
%
if NP==20
  y(1)=A0_F*x(1);
  y(2)=A0_F*x(2)+A1_F*x(1)+B1_F*y(1);
  y(3)=A0_F*x(3)+A1_F*x(2)+A2_F*x(1)+B1_F*y(2)+B2_F*y(1);
```

```
y(4)=A0_F*x(4)+A1_F*x(3)+A2_F*x(2)+A3_F*x(1)+B1_F*y(3)+B2_F*y(2)+B3_F*y(
1);

y(5)=A0_F*x(5)+A1_F*x(4)+A2_F*x(3)+A3_F*x(2)+A4_F*x(1)+B1_F*y(4)+B2_F*y(
3)+B3_F*y(2)+B4_F*y(1);
        y(6)=A0_F*x(6)+A1_F*x(5)+A2_F*x(4)+A3_F*x(3)+A4_F*x(2)+A5_F*x(1)+B
1_F*y(5)+B2_F*y(4)+B3_F*y(3)+B4_F*y(2)+B5_F*y(1);
        y(7)=A0_F*x(7)+A1_F*x(6)+A2_F*x(5)+A3_F*x(4)+A4_F*x(3)+A5_F*x(2)+
A6_F*x(1)+B1_F*y(6)+B2_F*y(5)+B3_F*y(4)+B4_F*y(3)+B5_F*y(2)+B6_F*y(1);
        y(8)=A0_F*x(8)+A1_F*x(7)+A2_F*x(6)+A3_F*x(5)+A4_F*x(4)+A5_F*x(3)+
A6_F*x(2)+A7_F*x(1)+B1_F*y(7)+B2_F*y(6)+B3_F*y(5)+B4_F*y(4)+B5_F*y(3)+B6
_F*y(2)+B7_F*y(1);

y(9)=A0_F*x(9)+A1_F*x(8)+A2_F*x(7)+A3_F*x(6)+A4_F*x(5)+A5_F*x(4)+A6_F*x(
3)+A7_F*x(2)+A8_F*x(1)+B1_F*y(8)+B2_F*y(7)+B3_F*y(6)+B4_F*y(5)+B5_F*y(4)
+B6_F*y(3)+B7_F*y(2)+B8_F*y(1);

y(10)=A0_F*x(10)+A1_F*x(9)+A2_F*x(8)+A3_F*x(7)+A4_F*x(6)+A5_F*x(5)+A6_F
*x(4)+A7_F*x(3)+A8_F*x(2)+A9_F*x(1)+B1_F*y(9)+B2_F*y(8)+B3_F*y(7)+B4_F*y
(6)+B5_F*y(5)+B6_F*y(4)+B7_F*y(3)+B8_F*y(2)+B9_F*y(1);

y(11)=A0_F*x(11)+A1_F*x(10)+A2_F*x(9)+A3_F*x(8)+A4_F*x(7)+A5_F*x(6)+A6_
F*x(5)+A7_F*x(4)+A8_F*x(3)+A9_F*x(2)+A10_F*x(1)+B1_F*y(10)+B2_F*y(9)+B3_
F*y(8)+B4_F*y(7)+B5_F*y(6)+B6_F*y(5)+B7_F*y(4)+B8_F*y(3)+B9_F*y(2)+B10_F
*y(1);
        y(12)=A0_F*x(12)+A1_F*x(11)+A2_F*x(10)+A3_F*x(9)+A4_F*x(8)+A5_F*x(
7)+A6_F*x(6)+A7_F*x(5)+A8_F*x(4)+A9_F*x(3)+A10_F*x(2)+A11_F*x(1)+B1_F*y
(11)+B2_F*y(10)+B3_F*y(9)+B4_F*y(8)+B5_F*y(7)+B6_F*y(6)+B7_F*y(5)+B8_F*y
(4)+B9_F*y(3)+B10_F*y(2)+B11_F*y(1);
        y(13)=A0_F*x(13)+A1_F*x(12)+A2_F*x(11)+A3_F*x(10)+A4_F*x(9)+A5_F*
x(8)+A6_F*x(7)+A7_F*x(6)+A8_F*x(5)+A9_F*x(4)+A10_F*x(3)+A11_F*x(2)+A12_
F*x(1)+B1_F*y(12)+B2_F*y(11)+B3_F*y(10)+B4_F*y(9)+B5_F*y(8)+B6_F*y(7)+B7
_F*y(6)+B8_F*y(5)+B9_F*y(4)+B10_F*y(3)+B11_F*y(2)+B12_F*y(1);
        y(14)=A0_F*x(14)+A1_F*x(13)+A2_F*x(12)+A3_F*x(11)+A4_F*x(10)+A5_F
*x(9)+A6_F*x(8)+A7_F*x(7)+A8_F*x(6)+A9_F*x(5)+A10_F*x(4)+A11_F*x(3)+A12
_F*x(2)+A13_F*x(1)+B1_F*y(13)+B2_F*y(12)+B3_F*y(11)+B4_F*y(10)+B5_F*y(9)
+B6_F*y(8)+B7_F*y(7)+B8_F*y(6)+B9_F*y(5)+B10_F*y(4)+B11_F*y(3)+B12_F*y(2
)+B13_F*y(1);
        y(15)=A0_F*x(15)+A1_F*x(14)+A2_F*x(13)+A3_F*x(12)+A4_F*x(11)+A5_F
*x(10)+A6_F*x(9)+A7_F*x(8)+A8_F*x(7)+A9_F*x(6)+A10_F*x(5)+A11_F*x(4)+A1
2_F*x(3)+A13_F*x(2)+A14_F*x(1)+B1_F*y(14)+B2_F*y(13)+B3_F*y(12)+B4_F*y(1
1)+B5_F*y(10)+B6_F*y(9)+B7_F*y(8)+B8_F*y(7)+B9_F*y(6)+B10_F*y(5)+B11_F*y
(4)+B12_F*y(3)+B13_F*y(2)+B14_F*y(1);
```

```
        y(16)=A0_F*x(16)+A1_F*x(15)+A2_F*x(14)+A3_F*x(13)+A4_F*x(12)+A5_F
*x(11)+A6_F*x(10)+A7_F*x(9)+A8_F*x(8)+A9_F*x(7)+A10_F*x(6)+A11_F*x(5)+A
12_F*x(4)+A13_F*x(3)+A14_F*x(2)+A15_F*x(1)+B1_F*y(15)+B2_F*y(14)+B3_F*y(
13)+B4_F*y(12)+B5_F*y(11)+B6_F*y(10)+B7_F*y(9)+B8_F*y(8)+B9_F*y(7)+B10_F
*y(6)+B11_F*y(5)+B12_F*y(4)+B13_F*y(3)+B14_F*y(2)+B15_F*y(1);
        y(17)=A0_F*x(17)+A1_F*x(16)+A2_F*x(15)+A3_F*x(14)+A4_F*x(13)+A5_F
*x(12)+A6_F*x(11)+A7_F*x(10)+A8_F*x(9)+A9_F*x(8)+A10_F*x(7)+A11_F*x(6)+
A12_F*x(5)+A13_F*x(4)+A14_F*x(3)+A15_F*x(2)+A16_F*x(1)+B1_F*y(16)+B2_F*
y(15)+B3_F*y(14)+B4_F*y(13)+B5_F*y(12)+B6_F*y(11)+B7_F*y(10)+B8_F*y(9)+B
9_F*y(8)+B10_F*y(7)+B11_F*y(6)+B12_F*y(5)+B13_F*y(4)+B14_F*y(3)+B15_F*y(
2)+B16_F*y(1);
        y(18)=A0_F*x(18)+A1_F*x(17)+A2_F*x(16)+A3_F*x(15)+A4_F*x(14)+A5_F
*x(13)+A6_F*x(12)+A7_F*x(11)+A8_F*x(10)+A9_F*x(9)+A10_F*x(8)+A11_F*x(7)+
A12_F*x(6)+A13_F*x(5)+A14_F*x(4)+A15_F*x(3)+A16_F*x(2)+A17_F*x(1)+B1_F*
y(17)+B2_F*y(16)+B3_F*y(15)+B4_F*y(14)+B5_F*y(13)+B6_F*y(12)+B7_F*y(11)+
B8_F*y(10)+B9_F*y(9)+B10_F*y(8)+B11_F*y(7)+B12_F*y(6)+B13_F*y(5)+B14_F*y
(4)+B15_F*y(3)+B16_F*y(2)+B17_F*y(1);
        y(19)=A0_F*x(19)+A1_F*x(18)+A2_F*x(17)+A3_F*x(16)+A4_F*x(15)+A5_F
*x(14)+A6_F*x(13)+A7_F*x(12)+A8_F*x(11)+A9_F*x(10)+A10_F*x(9)+A11_F*x(8)
+A12_F*x(7)+A13_F*x(6)+A14_F*x(5)+A15_F*x(4)+A16_F*x(3)+A17_F*x(2)+A18_
F*x(1)+B1_F*y(18)+B2_F*y(17)+B3_F*y(16)+B4_F*y(15)+B5_F*y(14)+B6_F*y(13)
+B7_F*y(12)+B8_F*y(11)+B9_F*y(10)+B10_F*y(9)+B11_F*y(8)+B12_F*y(7)+B13_
F*y(6)+B14_F*y(5)+B15_F*y(4)+B16_F*y(3)+B17_F*y(2)+B18_F*y(1);
        y(20)=A0_F*x(20)+A1_F*x(19)+A2_F*x(18)+A3_F*x(17)+A4_F*x(16)+A5_F
*x(15)+A6_F*x(14)+A7_F*x(13)+A8_F*x(12)+A9_F*x(11)+A10_F*x(10)+A11_F*x(9
)+A12_F*x(8)+A13_F*x(7)+A14_F*x(6)+A15_F*x(5)+A16_F*x(4)+A17_F*x(3)+A18
_F*x(2)+A19_F*x(1)+B1_F*y(19)+B2_F*y(18)+B3_F*y(17)+B4_F*y(16)+B5_F*y(15)
+B6_F*y(14)+B7_F*y(13)+B8_F*y(12)+B9_F*y(11)+B10_F*y(10)+B11_F*y(9)+B12_
F*y(8)+B13_F*y(7)+B14_F*y(6)+B15_F*y(5)+B16_F*y(4)+B17_F*y(3)+B18_F*y(2)
+B19_F*y(1);

  for i=21:(len_x)
        y(i)=A0_F*x(i)+A1_F*x(i-1)+A2_F*x(i-2)+A3_F*x(i-3)+A4_F*x(i-
4)+A5_F*x(i-5)+A6_F*x(i-6)+A7_F*x(i-7)+A8_F*x(i-8)+A9_F*x(i-9)+A10_F*x(i-
10)+A11_F*x(i-11)+A12_F*x(i-12)+A13_F*x(i-13)+A14_F*x(i-14)+A15_F*x(i-
15)+A16_F*x(i-16)+A17_F*x(i-17)+A18_F*x(i-18)+A19_F*x(i-19)+A20_F*x(i-
20)+B1_F*y(i-1)+B2_F*y(i-2)+B3_F*y(i-3)+B4_F*y(i-4)+B5_F*y(i-5)+B6_F*y(i-
6)+B7_F*y(i-7)+B8_F*y(i-8)+B9_F*y(i-9)+B10_F*y(i-10)+B11_F*y(i-11)+B12_F*y(i-
12)+B13_F*y(i-13)+B14_F*y(i-14)+B15_F*y(i-15)+B16_F*y(i-16)+B17_F*y(i-
17)+B18_F*y(i-18)+B19_F*y(i-19)+B20_F*y(i-20);
        end
end
%
%
```

```
%
%
%
%
hold off
plot(t_axis.x)
hold on
plot(t_axis.y.'r')
legend('input'.'output - Chebychev')
hold off
pause
%
X=fft(x.len_x);
Y=fft(y.len_x);
hold off
plot(f_axis.abs(X(1:len_x/2)).'b')
hold on
plot(f_axis.abs(Y(1:len_x/2)).'r')
legend('input fft'.'output fft- Chebychev')
pause
%
%
%       Repeat procedure for delta function
%
%
%
%
load delta2.dat
%
%
len_x=length(x);

h=zeros(len_x.1);
%
%
%
if NP==2
        h(1)=A0_F*delta2(1);
        h(2)=A0_F*delta2(2)+A1_F*delta2(1)+B1_F*h(1);

        for i=3:(len_x)
         h(i)=A0_F*delta2(i)+A1_F*delta2(i-1)+A2_F*delta2(i-2)+B1_F*h(i-
1)+B2_F*h(i-2);
        end
```

```
end
%
%
%
if NP==4
        h(1)=A0_F*delta2(1);
  h(2)=A0_F*delta2(2)+A1_F*delta2(1)+B1_F*h(1);
  h(3)=A0_F*delta2(3)+A1_F*delta2(2)+A2_F*delta2(1)+B1_F*h(2)+B2_F*h(1);

h(4)=A0_F*delta2(4)+A1_F*delta2(3)+A2_F*delta2(2)+A3_F*delta2(1)+B1_F*h(3)+B
2_F*h(2)+B3_F*h(1);

        for i=5:(len_x)
         h(i)=A0_F*delta2(i)+A1_F*delta2(i-1)+A2_F*delta2(i-2)+A3_F*delta2(i-
3)+A4_F*delta2(i-4)+B1_F*h(i-1)+B2_F*h(i-2)+B3_F*h(i-3)+B4_F*h(i-4);
        end
end
%
%
%
%
%
%
if NP==6
        h(1)=A0_F*delta2(1);
  h(2)=A0_F*delta2(2)+A1_F*delta2(1)+B1_F*h(1);
  h(3)=A0_F*delta2(3)+A1_F*delta2(2)+A2_F*delta2(1)+B1_F*h(2)+B2_F*h(1);

h(4)=A0_F*delta2(4)+A1_F*delta2(3)+A2_F*delta2(2)+A3_F*delta2(1)+B1_F*h(3)+B
2_F*h(2)+B3_F*h(1);

h(5)=A0_F*delta2(5)+A1_F*delta2(4)+A2_F*delta2(3)+A3_F*delta2(2)+A4_F*delta2(
1)+B1_F*h(4)+B2_F*h(3)+B3_F*h(2)+B4_F*h(1);
        h(6)=A0_F*delta2(6)+A1_F*delta2(5)+A2_F*delta2(4)+A3_F*delta2(3)+A4_F*
delta2(2)+A5_F*delta2(1)+B1_F*h(5)+B2_F*h(4)+B3_F*h(3)+B4_F*h(2)+B5_F*h(1);

        for i=7:(len_x)
         h(i)=A0_F*delta2(i)+A1_F*delta2(i-1)+A2_F*delta2(i-2)+A3_F*delta2(i-
3)+A4_F*delta2(i-4)+A5_F*delta2(i-5)+A6_F*delta2(i-6)+B1_F*h(i-1)+B2_F*h(i-
2)+B3_F*h(i-3)+B4_F*h(i-4)+B5_F*h(i-5)+B6_F*h(i-6);
        end
end
%
%
```

```
%
if NP==8
        h(1)=A0_F*delta2(1);
  h(2)=A0_F*delta2(2)+A1_F*delta2(1)+B1_F*h(1);
  h(3)=A0_F*delta2(3)+A1_F*delta2(2)+A2_F*delta2(1)+B1_F*h(2)+B2_F*h(1);

h(4)=A0_F*delta2(4)+A1_F*delta2(3)+A2_F*delta2(2)+A3_F*delta2(1)+B1_F*h(3)+B
2_F*h(2)+B3_F*h(1);

h(5)=A0_F*delta2(5)+A1_F*delta2(4)+A2_F*delta2(3)+A3_F*delta2(2)+A4_F*delta2(
1)+B1_F*h(4)+B2_F*h(3)+B3_F*h(2)+B4_F*h(1);
        h(6)=A0_F*delta2(6)+A1_F*delta2(5)+A2_F*delta2(4)+A3_F*delta2(3)+A4_F*
delta2(2)+A5_F*delta2(1)+B1_F*h(5)+B2_F*h(4)+B3_F*h(3)+B4_F*h(2)+B5_F*h(1);
        h(7)=A0_F*delta2(7)+A1_F*delta2(6)+A2_F*delta2(5)+A3_F*delta2(4)+A4_F*
delta2(3)+A5_F*delta2(2)+A6_F*delta2(1)+B1_F*h(6)+B2_F*h(5)+B3_F*h(4)+B4_F*
h(3)+B5_F*h(2)+B6_F*h(1);
        h(8)=A0_F*delta2(8)+A1_F*delta2(7)+A2_F*delta2(6)+A3_F*delta2(5)+A4_F*
delta2(4)+A5_F*delta2(3)+A6_F*delta2(2)+A7_F*delta2(1)+B1_F*h(7)+B2_F*h(6)+B
3_F*h(5)+B4_F*h(4)+B5_F*h(3)+B6_F*h(2)+B7_F*h(1);

  for i=9:(len_x)
         h(i)=A0_F*delta2(i)+A1_F*delta2(i-1)+A2_F*delta2(i-2)+A3_F*delta2(i-
3)+A4_F*delta2(i-4)+A5_F*delta2(i-5)+A6_F*delta2(i-6)+A7_F*delta2(i-
7)+A8_F*delta2(i-8)+B1_F*h(i-1)+B2_F*h(i-2)+B3_F*h(i-3)+B4_F*h(i-4)+B5_F*h(i-
5)+B6_F*h(i-6)+B7_F*h(i-7)+B8_F*h(i-8);
        end
end
%
%
if NP==10
        h(1)=A0_F*delta2(1);
  h(2)=A0_F*delta2(2)+A1_F*delta2(1)+B1_F*h(1);
  h(3)=A0_F*delta2(3)+A1_F*delta2(2)+A2_F*delta2(1)+B1_F*h(2)+B2_F*h(1);

h(4)=A0_F*delta2(4)+A1_F*delta2(3)+A2_F*delta2(2)+A3_F*delta2(1)+B1_F*h(3)+B
2_F*h(2)+B3_F*h(1);

h(5)=A0_F*delta2(5)+A1_F*delta2(4)+A2_F*delta2(3)+A3_F*delta2(2)+A4_F*delta2(
1)+B1_F*h(4)+B2_F*h(3)+B3_F*h(2)+B4_F*h(1);
        h(6)=A0_F*delta2(6)+A1_F*delta2(5)+A2_F*delta2(4)+A3_F*delta2(3)+A4_F*
delta2(2)+A5_F*delta2(1)+B1_F*h(5)+B2_F*h(4)+B3_F*h(3)+B4_F*h(2)+B5_F*h(1);
        h(7)=A0_F*delta2(7)+A1_F*delta2(6)+A2_F*delta2(5)+A3_F*delta2(4)+A4_F*
delta2(3)+A5_F*delta2(2)+A6_F*delta2(1)+B1_F*h(6)+B2_F*h(5)+B3_F*h(4)+B4_F*
h(3)+B5_F*h(2)+B6_F*h(1);
```

```
        h(8)=A0_F*delta2(8)+A1_F*delta2(7)+A2_F*delta2(6)+A3_F*delta2(5)+A4_F*
delta2(4)+A5_F*delta2(3)+A6_F*delta2(2)+A7_F*delta2(1)+B1_F*h(7)+B2_F*h(6)+B
3_F*h(5)+B4_F*h(4)+B5_F*h(3)+B6_F*h(2)+B7_F*h(1);

h(9)=A0_F*delta2(9)+A1_F*delta2(8)+A2_F*delta2(7)+A3_F*delta2(6)+A4_F*delta2(
5)+A5_F*delta2(4)+A6_F*delta2(3)+A7_F*delta2(2)+A8_F*delta2(1)+B1_F*h(8)+B2_
F*h(7)+B3_F*h(6)+B4_F*h(5)+B5_F*h(4)+B6_F*h(3)+B7_F*h(2)+B8_F*h(1);

h(10)=A0_F*delta2(10)+A1_F*delta2(9)+A2_F*delta2(8)+A3_F*delta2(7)+A4_F*delta
2(6)+A5_F*delta2(5)+A6_F*delta2(4)+A7_F*delta2(3)+A8_F*delta2(2)+A9_F*delta2(
1)+B1_F*h(9)+B2_F*h(8)+B3_F*h(7)+B4_F*h(6)+B5_F*h(5)+B6_F*h(4)+B7_F*h(3)
+B8_F*h(2)+B9_F*h(1);

  for i=11:(len_x)
          h(i)=A0_F*delta2(i)+A1_F*delta2(i-1)+A2_F*delta2(i-2)+A3_F*delta2(i-
3)+A4_F*delta2(i-4)+A5_F*delta2(i-5)+A6_F*delta2(i-6)+A7_F*delta2(i-
7)+A8_F*delta2(i-8)+A9_F*delta2(i-9)+A10_F*delta2(i-10)+B1_F*h(i-1)+B2_F*h(i-
2)+B3_F*h(i-3)+B4_F*h(i-4)+B5_F*h(i-5)+B6_F*h(i-6)+B7_F*h(i-7)+B8_F*h(i-
8)+B9_F*h(i-9)+B10_F*h(i-10);
        end
end
%
%
if NP==12
        h(1)=A0_F*delta2(1);
  h(2)=A0_F*delta2(2)+A1_F*delta2(1)+B1_F*h(1);
  h(3)=A0_F*delta2(3)+A1_F*delta2(2)+A2_F*delta2(1)+B1_F*h(2)+B2_F*h(1);

h(4)=A0_F*delta2(4)+A1_F*delta2(3)+A2_F*delta2(2)+A3_F*delta2(1)+B1_F*h(3)+B
2_F*h(2)+B3_F*h(1);

h(5)=A0_F*delta2(5)+A1_F*delta2(4)+A2_F*delta2(3)+A3_F*delta2(2)+A4_F*delta2(
1)+B1_F*h(4)+B2_F*h(3)+B3_F*h(2)+B4_F*h(1);
        h(6)=A0_F*delta2(6)+A1_F*delta2(5)+A2_F*delta2(4)+A3_F*delta2(3)+A4_F*
delta2(2)+A5_F*delta2(1)+B1_F*h(5)+B2_F*h(4)+B3_F*h(3)+B4_F*h(2)+B5_F*h(1);
        h(7)=A0_F*delta2(7)+A1_F*delta2(6)+A2_F*delta2(5)+A3_F*delta2(4)+A4_F*
delta2(3)+A5_F*delta2(2)+A6_F*delta2(1)+B1_F*h(6)+B2_F*h(5)+B3_F*h(4)+B4_F*
h(3)+B5_F*h(2)+B6_F*h(1);
        h(8)=A0_F*delta2(8)+A1_F*delta2(7)+A2_F*delta2(6)+A3_F*delta2(5)+A4_F*
delta2(4)+A5_F*delta2(3)+A6_F*delta2(2)+A7_F*delta2(1)+B1_F*h(7)+B2_F*h(6)+B
3_F*h(5)+B4_F*h(4)+B5_F*h(3)+B6_F*h(2)+B7_F*h(1);

h(9)=A0_F*delta2(9)+A1_F*delta2(8)+A2_F*delta2(7)+A3_F*delta2(6)+A4_F*delta2(
5)+A5_F*delta2(4)+A6_F*delta2(3)+A7_F*delta2(2)+A8_F*delta2(1)+B1_F*h(8)+B2_
F*h(7)+B3_F*h(6)+B4_F*h(5)+B5_F*h(4)+B6_F*h(3)+B7_F*h(2)+B8_F*h(1);

h(10)=A0_F*delta2(10)+A1_F*delta2(9)+A2_F*delta2(8)+A3_F*delta2(7)+A4_F*delta
2(6)+A5_F*delta2(5)+A6_F*delta2(4)+A7_F*delta2(3)+A8_F*delta2(2)+A9_F*delta2(
1)+B1_F*h(9)+B2_F*h(8)+B3_F*h(7)+B4_F*h(6)+B5_F*h(5)+B6_F*h(4)+B7_F*h(3)
+B8_F*h(2)+B9_F*h(1);

h(11)=A0_F*delta2(11)+A1_F*delta2(10)+A2_F*delta2(9)+A3_F*delta2(8)+A4_F*delt
a2(7)+A5_F*delta2(6)+A6_F*delta2(5)+A7_F*delta2(4)+A8_F*delta2(3)+A9_F*delta2
(2)+A10_F*delta2(1)+B1_F*h(10)+B2_F*h(9)+B3_F*h(8)+B4_F*h(7)+B5_F*h(6)+B6
_F*h(5)+B7_F*h(4)+B8_F*h(3)+B9_F*h(2)+B10_F*h(1);
        h(12)=A0_F*delta2(12)+A1_F*delta2(11)+A2_F*delta2(10)+A3_F*delta2(9)+A
4_F*delta2(8)+A5_F*delta2(7)+A6_F*delta2(6)+A7_F*delta2(5)+A8_F*delta2(4)+A9_
F*delta2(3)+A10_F*delta2(2)+A11_F*delta2(1)+B1_F*h(11)+B2_F*h(10)+B3_F*h(9)+
B4_F*h(8)+B5_F*h(7)+B6_F*h(6)+B7_F*h(5)+B8_F*h(4)+B9_F*h(3)+B10_F*h(2)+B
11_F*h(1);

  for i=13:(len_x)
          h(i)=A0_F*delta2(i)+A1_F*delta2(i-1)+A2_F*delta2(i-2)+A3_F*delta2(i-
3)+A4_F*delta2(i-4)+A5_F*delta2(i-5)+A6_F*delta2(i-6)+A7_F*delta2(i-
7)+A8_F*delta2(i-8)+A9_F*delta2(i-9)+A10_F*delta2(i-10)+A11_F*delta2(i-
11)+A12_F*delta2(i-12)+B1_F*h(i-1)+B2_F*h(i-2)+B3_F*h(i-3)+B4_F*h(i-
4)+B5_F*h(i-5)+B6_F*h(i-6)+B7_F*h(i-7)+B8_F*h(i-8)+B9_F*h(i-9)+B10_F*h(i-
10)+B11_F*h(i-11)+B12_F*h(i-12);
        end
end
%
%
if NP==14
        h(1)=A0_F*delta2(1);
  h(2)=A0_F*delta2(2)+A1_F*delta2(1)+B1_F*h(1);
  h(3)=A0_F*delta2(3)+A1_F*delta2(2)+A2_F*delta2(1)+B1_F*h(2)+B2_F*h(1);

h(4)=A0_F*delta2(4)+A1_F*delta2(3)+A2_F*delta2(2)+A3_F*delta2(1)+B1_F*h(3)+B
2_F*h(2)+B3_F*h(1);

h(5)=A0_F*delta2(5)+A1_F*delta2(4)+A2_F*delta2(3)+A3_F*delta2(2)+A4_F*delta2(
1)+B1_F*h(4)+B2_F*h(3)+B3_F*h(2)+B4_F*h(1);
        h(6)=A0_F*delta2(6)+A1_F*delta2(5)+A2_F*delta2(4)+A3_F*delta2(3)+A4_F*
delta2(2)+A5_F*delta2(1)+B1_F*h(5)+B2_F*h(4)+B3_F*h(3)+B4_F*h(2)+B5_F*h(1);
        h(7)=A0_F*delta2(7)+A1_F*delta2(6)+A2_F*delta2(5)+A3_F*delta2(4)+A4_F*
delta2(3)+A5_F*delta2(2)+A6_F*delta2(1)+B1_F*h(6)+B2_F*h(5)+B3_F*h(4)+B4_F*
h(3)+B5_F*h(2)+B6_F*h(1);
```

```
        h(8)=A0_F*delta2(8)+A1_F*delta2(7)+A2_F*delta2(6)+A3_F*delta2(5)+A4_F*
delta2(4)+A5_F*delta2(3)+A6_F*delta2(2)+A7_F*delta2(1)+B1_F*h(7)+B2_F*h(6)+B
3_F*h(5)+B4_F*h(4)+B5_F*h(3)+B6_F*h(2)+B7_F*h(1);

h(9)=A0_F*delta2(9)+A1_F*delta2(8)+A2_F*delta2(7)+A3_F*delta2(6)+A4_F*delta2(
5)+A5_F*delta2(4)+A6_F*delta2(3)+A7_F*delta2(2)+A8_F*delta2(1)+B1_F*h(8)+B2_
F*h(7)+B3_F*h(6)+B4_F*h(5)+B5_F*h(4)+B6_F*h(3)+B7_F*h(2)+B8_F*h(1);

h(10)=A0_F*delta2(10)+A1_F*delta2(9)+A2_F*delta2(8)+A3_F*delta2(7)+A4_F*delta
2(6)+A5_F*delta2(5)+A6_F*delta2(4)+A7_F*delta2(3)+A8_F*delta2(2)+A9_F*delta2(
1)+B1_F*h(9)+B2_F*h(8)+B3_F*h(7)+B4_F*h(6)+B5_F*h(5)+B6_F*h(4)+B7_F*h(3)
+B8_F*h(2)+B9_F*h(1);

h(11)=A0_F*delta2(11)+A1_F*delta2(10)+A2_F*delta2(9)+A3_F*delta2(8)+A4_F*delt
a2(7)+A5_F*delta2(6)+A6_F*delta2(5)+A7_F*delta2(4)+A8_F*delta2(3)+A9_F*delta2
(2)+A10_F*delta2(1)+B1_F*h(10)+B2_F*h(9)+B3_F*h(8)+B4_F*h(7)+B5_F*h(6)+B6
_F*h(5)+B7_F*h(4)+B8_F*h(3)+B9_F*h(2)+B10_F*h(1);
        h(12)=A0_F*delta2(12)+A1_F*delta2(11)+A2_F*delta2(10)+A3_F*delta2(9)+A
4_F*delta2(8)+A5_F*delta2(7)+A6_F*delta2(6)+A7_F*delta2(5)+A8_F*delta2(4)+A9_
F*delta2(3)+A10_F*delta2(2)+A11_F*delta2(1)+B1_F*h(11)+B2_F*h(10)+B3_F*h(9)+
B4_F*h(8)+B5_F*h(7)+B6_F*h(6)+B7_F*h(5)+B8_F*h(4)+B9_F*h(3)+B10_F*h(2)+B
11_F*h(1);
        h(13)=A0_F*delta2(13)+A1_F*delta2(12)+A2_F*delta2(11)+A3_F*delta2(10)+
A4_F*delta2(9)+A5_F*delta2(8)+A6_F*delta2(7)+A7_F*delta2(6)+A8_F*delta2(5)+A9
_F*delta2(4)+A10_F*delta2(3)+A11_F*delta2(2)+A12_F*delta2(1)+B1_F*h(12)+B2_F
*h(11)+B3_F*h(10)+B4_F*h(9)+B5_F*h(8)+B6_F*h(7)+B7_F*h(6)+B8_F*h(5)+B9_F
*h(4)+B10_F*h(3)+B11_F*h(2)+B12_F*h(1);
        h(14)=A0_F*delta2(14)+A1_F*delta2(13)+A2_F*delta2(12)+A3_F*delta2(11)+
A4_F*delta2(10)+A5_F*delta2(9)+A6_F*delta2(8)+A7_F*delta2(7)+A8_F*delta2(6)+A
9_F*delta2(5)+A10_F*delta2(4)+A11_F*delta2(3)+A12_F*delta2(2)+A13_F*delta2(1)
+B1_F*h(13)+B2_F*h(12)+B3_F*h(11)+B4_F*h(10)+B5_F*h(9)+B6_F*h(8)+B7_F*h(
7)+B8_F*h(6)+B9_F*h(5)+B10_F*h(4)+B11_F*h(3)+B12_F*h(2)+B13_F*h(1);

  for i=15:(len_x)
         h(i)=A0_F*delta2(i)+A1_F*delta2(i-1)+A2_F*delta2(i-2)+A3_F*delta2(i-
3)+A4_F*delta2(i-4)+A5_F*delta2(i-5)+A6_F*delta2(i-6)+A7_F*delta2(i-
7)+A8_F*delta2(i-8)+A9_F*delta2(i-9)+A10_F*delta2(i-10)+A11_F*delta2(i-
11)+A12_F*delta2(i-12)+A13_F*delta2(i-13)+A14_F*delta2(i-14)+B1_F*h(i-
1)+B2_F*h(i-2)+B3_F*h(i-3)+B4_F*h(i-4)+B5_F*h(i-5)+B6_F*h(i-6)+B7_F*h(i-
7)+B8_F*h(i-8)+B9_F*h(i-9)+B10_F*h(i-10)+B11_F*h(i-11)+B12_F*h(i-
12)+B13_F*h(i-13)+B14_F*h(i-14);
        end
end
%
```

```
%
if NP==16
        h(1)=A0_F*delta2(1);
  h(2)=A0_F*delta2(2)+A1_F*delta2(1)+B1_F*h(1);
  h(3)=A0_F*delta2(3)+A1_F*delta2(2)+A2_F*delta2(1)+B1_F*h(2)+B2_F*h(1);

h(4)=A0_F*delta2(4)+A1_F*delta2(3)+A2_F*delta2(2)+A3_F*delta2(1)+B1_F*h(3)+B
2_F*h(2)+B3_F*h(1);

h(5)=A0_F*delta2(5)+A1_F*delta2(4)+A2_F*delta2(3)+A3_F*delta2(2)+A4_F*delta2(
1)+B1_F*h(4)+B2_F*h(3)+B3_F*h(2)+B4_F*h(1);
        h(6)=A0_F*delta2(6)+A1_F*delta2(5)+A2_F*delta2(4)+A3_F*delta2(3)+A4_F*
delta2(2)+A5_F*delta2(1)+B1_F*h(5)+B2_F*h(4)+B3_F*h(3)+B4_F*h(2)+B5_F*h(1);
        h(7)=A0_F*delta2(7)+A1_F*delta2(6)+A2_F*delta2(5)+A3_F*delta2(4)+A4_F*
delta2(3)+A5_F*delta2(2)+A6_F*delta2(1)+B1_F*h(6)+B2_F*h(5)+B3_F*h(4)+B4_F*
h(3)+B5_F*h(2)+B6_F*h(1);
        h(8)=A0_F*delta2(8)+A1_F*delta2(7)+A2_F*delta2(6)+A3_F*delta2(5)+A4_F*
delta2(4)+A5_F*delta2(3)+A6_F*delta2(2)+A7_F*delta2(1)+B1_F*h(7)+B2_F*h(6)+B
3_F*h(5)+B4_F*h(4)+B5_F*h(3)+B6_F*h(2)+B7_F*h(1);

h(9)=A0_F*delta2(9)+A1_F*delta2(8)+A2_F*delta2(7)+A3_F*delta2(6)+A4_F*delta2(
5)+A5_F*delta2(4)+A6_F*delta2(3)+A7_F*delta2(2)+A8_F*delta2(1)+B1_F*h(8)+B2_
F*h(7)+B3_F*h(6)+B4_F*h(5)+B5_F*h(4)+B6_F*h(3)+B7_F*h(2)+B8_F*h(1);

h(10)=A0_F*delta2(10)+A1_F*delta2(9)+A2_F*delta2(8)+A3_F*delta2(7)+A4_F*delta
2(6)+A5_F*delta2(5)+A6_F*delta2(4)+A7_F*delta2(3)+A8_F*delta2(2)+A9_F*delta2(
1)+B1_F*h(9)+B2_F*h(8)+B3_F*h(7)+B4_F*h(6)+B5_F*h(5)+B6_F*h(4)+B7_F*h(3)
+B8_F*h(2)+B9_F*h(1);

h(11)=A0_F*delta2(11)+A1_F*delta2(10)+A2_F*delta2(9)+A3_F*delta2(8)+A4_F*delt
a2(7)+A5_F*delta2(6)+A6_F*delta2(5)+A7_F*delta2(4)+A8_F*delta2(3)+A9_F*delta2
(2)+A10_F*delta2(1)+B1_F*h(10)+B2_F*h(9)+B3_F*h(8)+B4_F*h(7)+B5_F*h(6)+B6
_F*h(5)+B7_F*h(4)+B8_F*h(3)+B9_F*h(2)+B10_F*h(1);
        h(12)=A0_F*delta2(12)+A1_F*delta2(11)+A2_F*delta2(10)+A3_F*delta2(9)+A
4_F*delta2(8)+A5_F*delta2(7)+A6_F*delta2(6)+A7_F*delta2(5)+A8_F*delta2(4)+A9_
F*delta2(3)+A10_F*delta2(2)+A11_F*delta2(1)+B1_F*h(11)+B2_F*h(10)+B3_F*h(9)+
B4_F*h(8)+B5_F*h(7)+B6_F*h(6)+B7_F*h(5)+B8_F*h(4)+B9_F*h(3)+B10_F*h(2)+B
11_F*h(1);
        h(13)=A0_F*delta2(13)+A1_F*delta2(12)+A2_F*delta2(11)+A3_F*delta2(10)+
A4_F*delta2(9)+A5_F*delta2(8)+A6_F*delta2(7)+A7_F*delta2(6)+A8_F*delta2(5)+A9
_F*delta2(4)+A10_F*delta2(3)+A11_F*delta2(2)+A12_F*delta2(1)+B1_F*h(12)+B2_F
*h(11)+B3_F*h(10)+B4_F*h(9)+B5_F*h(8)+B6_F*h(7)+B7_F*h(6)+B8_F*h(5)+B9_F
*h(4)+B10_F*h(3)+B11_F*h(2)+B12_F*h(1);
```

```
        h(14)=A0_F*delta2(14)+A1_F*delta2(13)+A2_F*delta2(12)+A3_F*delta2(11)+
A4_F*delta2(10)+A5_F*delta2(9)+A6_F*delta2(8)+A7_F*delta2(7)+A8_F*delta2(6)+A
9_F*delta2(5)+A10_F*delta2(4)+A11_F*delta2(3)+A12_F*delta2(2)+A13_F*delta2(1)
+B1_F*h(13)+B2_F*h(12)+B3_F*h(11)+B4_F*h(10)+B5_F*h(9)+B6_F*h(8)+B7_F*h(
7)+B8_F*h(6)+B9_F*h(5)+B10_F*h(4)+B11_F*h(3)+B12_F*h(2)+B13_F*h(1);
        h(15)=A0_F*delta2(15)+A1_F*delta2(14)+A2_F*delta2(13)+A3_F*delta2(12)+
A4_F*delta2(11)+A5_F*delta2(10)+A6_F*delta2(9)+A7_F*delta2(8)+A8_F*delta2(7)+
A9_F*delta2(6)+A10_F*delta2(5)+A11_F*delta2(4)+A12_F*delta2(3)+A13_F*delta2(2
)+A14_F*delta2(1)+B1_F*h(14)+B2_F*h(13)+B3_F*h(12)+B4_F*h(11)+B5_F*h(10)+
B6_F*h(9)+B7_F*h(8)+B8_F*h(7)+B9_F*h(6)+B10_F*h(5)+B11_F*h(4)+B12_F*h(3)
+B13_F*h(2)+B14_F*h(1);
        h(16)=A0_F*delta2(16)+A1_F*delta2(15)+A2_F*delta2(14)+A3_F*delta2(13)+
A4_F*delta2(12)+A5_F*delta2(11)+A6_F*delta2(10)+A7_F*delta2(9)+A8_F*delta2(8)
+A9_F*delta2(7)+A10_F*delta2(6)+A11_F*delta2(5)+A12_F*delta2(4)+A13_F*delta2(
3)+A14_F*delta2(2)+A15_F*delta2(1)+B1_F*h(15)+B2_F*h(14)+B3_F*h(13)+B4_F*h
(12)+B5_F*h(11)+B6_F*h(10)+B7_F*h(9)+B8_F*h(8)+B9_F*h(7)+B10_F*h(6)+B11_
F*h(5)+B12_F*h(4)+B13_F*h(3)+B14_F*h(2)+B15_F*h(1);

  for i=17:(len_x)
          h(i)=A0_F*delta2(i)+A1_F*delta2(i-1)+A2_F*delta2(i-2)+A3_F*delta2(i-
3)+A4_F*delta2(i-4)+A5_F*delta2(i-5)+A6_F*delta2(i-6)+A7_F*delta2(i-
7)+A8_F*delta2(i-8)+A9_F*delta2(i-9)+A10_F*delta2(i-10)+A11_F*delta2(i-
11)+A12_F*delta2(i-12)+A13_F*delta2(i-13)+A14_F*delta2(i-14)+A15_F*delta2(i-
15)+A16_F*delta2(i-16)+B1_F*h(i-1)+B2_F*h(i-2)+B3_F*h(i-3)+B4_F*h(i-
4)+B5_F*h(i-5)+B6_F*h(i-6)+B7_F*h(i-7)+B8_F*h(i-8)+B9_F*h(i-9)+B10_F*h(i-
10)+B11_F*h(i-11)+B12_F*h(i-12)+B13_F*h(i-13)+B14_F*h(i-14)+B15_F*h(i-
15)+B16_F*h(i-16);
        end
end
%
%
if NP==18
        h(1)=A0_F*delta2(1);
  h(2)=A0_F*delta2(2)+A1_F*delta2(1)+B1_F*h(1);
  h(3)=A0_F*delta2(3)+A1_F*delta2(2)+A2_F*delta2(1)+B1_F*h(2)+B2_F*h(1);

h(4)=A0_F*delta2(4)+A1_F*delta2(3)+A2_F*delta2(2)+A3_F*delta2(1)+B1_F*h(3)+B
2_F*h(2)+B3_F*h(1);

h(5)=A0_F*delta2(5)+A1_F*delta2(4)+A2_F*delta2(3)+A3_F*delta2(2)+A4_F*delta2(
1)+B1_F*h(4)+B2_F*h(3)+B3_F*h(2)+B4_F*h(1);
        h(6)=A0_F*delta2(6)+A1_F*delta2(5)+A2_F*delta2(4)+A3_F*delta2(3)+A4_F*
delta2(2)+A5_F*delta2(1)+B1_F*h(5)+B2_F*h(4)+B3_F*h(3)+B4_F*h(2)+B5_F*h(1);
```

```
        h(7)=A0_F*delta2(7)+A1_F*delta2(6)+A2_F*delta2(5)+A3_F*delta2(4)+A4_F*
delta2(3)+A5_F*delta2(2)+A6_F*delta2(1)+B1_F*h(6)+B2_F*h(5)+B3_F*h(4)+B4_F*
h(3)+B5_F*h(2)+B6_F*h(1);
        h(8)=A0_F*delta2(8)+A1_F*delta2(7)+A2_F*delta2(6)+A3_F*delta2(5)+A4_F*
delta2(4)+A5_F*delta2(3)+A6_F*delta2(2)+A7_F*delta2(1)+B1_F*h(7)+B2_F*h(6)+B
3_F*h(5)+B4_F*h(4)+B5_F*h(3)+B6_F*h(2)+B7_F*h(1);

h(9)=A0_F*delta2(9)+A1_F*delta2(8)+A2_F*delta2(7)+A3_F*delta2(6)+A4_F*delta2(
5)+A5_F*delta2(4)+A6_F*delta2(3)+A7_F*delta2(2)+A8_F*delta2(1)+B1_F*h(8)+B2_
F*h(7)+B3_F*h(6)+B4_F*h(5)+B5_F*h(4)+B6_F*h(3)+B7_F*h(2)+B8_F*h(1);

h(10)=A0_F*delta2(10)+A1_F*delta2(9)+A2_F*delta2(8)+A3_F*delta2(7)+A4_F*delta
2(6)+A5_F*delta2(5)+A6_F*delta2(4)+A7_F*delta2(3)+A8_F*delta2(2)+A9_F*delta2(
1)+B1_F*h(9)+B2_F*h(8)+B3_F*h(7)+B4_F*h(6)+B5_F*h(5)+B6_F*h(4)+B7_F*h(3)
+B8_F*h(2)+B9_F*h(1);

h(11)=A0_F*delta2(11)+A1_F*delta2(10)+A2_F*delta2(9)+A3_F*delta2(8)+A4_F*delt
a2(7)+A5_F*delta2(6)+A6_F*delta2(5)+A7_F*delta2(4)+A8_F*delta2(3)+A9_F*delta2
(2)+A10_F*delta2(1)+B1_F*h(10)+B2_F*h(9)+B3_F*h(8)+B4_F*h(7)+B5_F*h(6)+B6
_F*h(5)+B7_F*h(4)+B8_F*h(3)+B9_F*h(2)+B10_F*h(1);
        h(12)=A0_F*delta2(12)+A1_F*delta2(11)+A2_F*delta2(10)+A3_F*delta2(9)+A
4_F*delta2(8)+A5_F*delta2(7)+A6_F*delta2(6)+A7_F*delta2(5)+A8_F*delta2(4)+A9_
F*delta2(3)+A10_F*delta2(2)+A11_F*delta2(1)+B1_F*h(11)+B2_F*h(10)+B3_F*h(9)+
B4_F*h(8)+B5_F*h(7)+B6_F*h(6)+B7_F*h(5)+B8_F*h(4)+B9_F*h(3)+B10_F*h(2)+B
11_F*h(1);
        h(13)=A0_F*delta2(13)+A1_F*delta2(12)+A2_F*delta2(11)+A3_F*delta2(10)+
A4_F*delta2(9)+A5_F*delta2(8)+A6_F*delta2(7)+A7_F*delta2(6)+A8_F*delta2(5)+A9
_F*delta2(4)+A10_F*delta2(3)+A11_F*delta2(2)+A12_F*delta2(1)+B1_F*h(12)+B2_F
*h(11)+B3_F*h(10)+B4_F*h(9)+B5_F*h(8)+B6_F*h(7)+B7_F*h(6)+B8_F*h(5)+B9_F
*h(4)+B10_F*h(3)+B11_F*h(2)+B12_F*h(1);
        h(14)=A0_F*delta2(14)+A1_F*delta2(13)+A2_F*delta2(12)+A3_F*delta2(11)+
A4_F*delta2(10)+A5_F*delta2(9)+A6_F*delta2(8)+A7_F*delta2(7)+A8_F*delta2(6)+A
9_F*delta2(5)+A10_F*delta2(4)+A11_F*delta2(3)+A12_F*delta2(2)+A13_F*delta2(1)
+B1_F*h(13)+B2_F*h(12)+B3_F*h(11)+B4_F*h(10)+B5_F*h(9)+B6_F*h(8)+B7_F*h(
7)+B8_F*h(6)+B9_F*h(5)+B10_F*h(4)+B11_F*h(3)+B12_F*h(2)+B13_F*h(1);
        h(15)=A0_F*delta2(15)+A1_F*delta2(14)+A2_F*delta2(13)+A3_F*delta2(12)+
A4_F*delta2(11)+A5_F*delta2(10)+A6_F*delta2(9)+A7_F*delta2(8)+A8_F*delta2(7)+
A9_F*delta2(6)+A10_F*delta2(5)+A11_F*delta2(4)+A12_F*delta2(3)+A13_F*delta2(2
)+A14_F*delta2(1)+B1_F*h(14)+B2_F*h(13)+B3_F*h(12)+B4_F*h(11)+B5_F*h(10)+
B6_F*h(9)+B7_F*h(8)+B8_F*h(7)+B9_F*h(6)+B10_F*h(5)+B11_F*h(4)+B12_F*h(3)
+B13_F*h(2)+B14_F*h(1);
        h(16)=A0_F*delta2(16)+A1_F*delta2(15)+A2_F*delta2(14)+A3_F*delta2(13)+
A4_F*delta2(12)+A5_F*delta2(11)+A6_F*delta2(10)+A7_F*delta2(9)+A8_F*delta2(8)
+A9_F*delta2(7)+A10_F*delta2(6)+A11_F*delta2(5)+A12_F*delta2(4)+A13_F*delta2(
3)+A14_F*delta2(2)+A15_F*delta2(1)+B1_F*h(15)+B2_F*h(14)+B3_F*h(13)+B4_F*h
(12)+B5_F*h(11)+B6_F*h(10)+B7_F*h(9)+B8_F*h(8)+B9_F*h(7)+B10_F*h(6)+B11_
F*h(5)+B12_F*h(4)+B13_F*h(3)+B14_F*h(2)+B15_F*h(1);
        h(17)=A0_F*delta2(17)+A1_F*delta2(16)+A2_F*delta2(15)+A3_F*delta2(14)+
A4_F*delta2(13)+A5_F*delta2(12)+A6_F*delta2(11)+A7_F*delta2(10)+A8_F*delta2(9
)+A9_F*delta2(8)+A10_F*delta2(7)+A11_F*delta2(6)+A12_F*delta2(5)+A13_F*delta2
(4)+A14_F*delta2(3)+A15_F*delta2(2)+A16_F*delta2(1)+B1_F*h(16)+B2_F*h(15)+B
3_F*h(14)+B4_F*h(13)+B5_F*h(12)+B6_F*h(11)+B7_F*h(10)+B8_F*h(9)+B9_F*h(8)
+B10_F*h(7)+B11_F*h(6)+B12_F*h(5)+B13_F*h(4)+B14_F*h(3)+B15_F*h(2)+B16_
F*h(1);
        h(18)=A0_F*delta2(18)+A1_F*delta2(17)+A2_F*delta2(16)+A3_F*delta2(15)+
A4_F*delta2(14)+A5_F*delta2(13)+A6_F*delta2(12)+A7_F*delta2(11)+A8_F*delta2(1
0)+A9_F*delta2(9)+A10_F*delta2(8)+A11_F*delta2(7)+A12_F*delta2(6)+A13_F*delta
2(5)+A14_F*delta2(4)+A15_F*delta2(3)+A16_F*delta2(2)+A17_F*delta2(1)+B1_F*h(
17)+B2_F*h(16)+B3_F*h(15)+B4_F*h(14)+B5_F*h(13)+B6_F*h(12)+B7_F*h(11)+B8
_F*h(10)+B9_F*h(9)+B10_F*h(8)+B11_F*h(7)+B12_F*h(6)+B13_F*h(5)+B14_F*h(4)
+B15_F*h(3)+B16_F*h(2)+B17_F*h(1);

  for i=19:(len_x)
          h(i)=A0_F*delta2(i)+A1_F*delta2(i-1)+A2_F*delta2(i-2)+A3_F*delta2(i-
3)+A4_F*delta2(i-4)+A5_F*delta2(i-5)+A6_F*delta2(i-6)+A7_F*delta2(i-
7)+A8_F*delta2(i-8)+A9_F*delta2(i-9)+A10_F*delta2(i-10)+A11_F*delta2(i-
11)+A12_F*delta2(i-12)+A13_F*delta2(i-13)+A14_F*delta2(i-14)+A15_F*delta2(i-
15)+A16_F*delta2(i-16)+A17_F*delta2(i-17)+A18_F*delta2(i-18)+B1_F*h(i-
1)+B2_F*h(i-2)+B3_F*h(i-3)+B4_F*h(i-4)+B5_F*h(i-5)+B6_F*h(i-6)+B7_F*h(i-
7)+B8_F*h(i-8)+B9_F*h(i-9)+B10_F*h(i-10)+B11_F*h(i-11)+B12_F*h(i-
12)+B13_F*h(i-13)+B14_F*h(i-14)+B15_F*h(i-15)+B16_F*h(i-16)+B17_F*h(i-
17)+B18_F*h(i-18);
        end
end
%
%
if NP==20
  h(1)=A0_F*delta2(1);
  h(2)=A0_F*delta2(2)+A1_F*delta2(1)+B1_F*h(1);
  h(3)=A0_F*delta2(3)+A1_F*delta2(2)+A2_F*delta2(1)+B1_F*h(2)+B2_F*h(1);

h(4)=A0_F*delta2(4)+A1_F*delta2(3)+A2_F*delta2(2)+A3_F*delta2(1)+B1_F*h(3)+B
2_F*h(2)+B3_F*h(1);

h(5)=A0_F*delta2(5)+A1_F*delta2(4)+A2_F*delta2(3)+A3_F*delta2(2)+A4_F*delta2(
1)+B1_F*h(4)+B2_F*h(3)+B3_F*h(2)+B4_F*h(1);
        h(6)=A0_F*delta2(6)+A1_F*delta2(5)+A2_F*delta2(4)+A3_F*delta2(3)+A4_F*
delta2(2)+A5_F*delta2(1)+B1_F*h(5)+B2_F*h(4)+B3_F*h(3)+B4_F*h(2)+B5_F*h(1);
```

```
        h(7)=A0_F*delta2(7)+A1_F*delta2(6)+A2_F*delta2(5)+A3_F*delta2(4)+A4_F*
delta2(3)+A5_F*delta2(2)+A6_F*delta2(1)+B1_F*h(6)+B2_F*h(5)+B3_F*h(4)+B4_F*
h(3)+B5_F*h(2)+B6_F*h(1);
        h(8)=A0_F*delta2(8)+A1_F*delta2(7)+A2_F*delta2(6)+A3_F*delta2(5)+A4_F*
delta2(4)+A5_F*delta2(3)+A6_F*delta2(2)+A7_F*delta2(1)+B1_F*h(7)+B2_F*h(6)+B
3_F*h(5)+B4_F*h(4)+B5_F*h(3)+B6_F*h(2)+B7_F*h(1);

h(9)=A0_F*delta2(9)+A1_F*delta2(8)+A2_F*delta2(7)+A3_F*delta2(6)+A4_F*delta2(
5)+A5_F*delta2(4)+A6_F*delta2(3)+A7_F*delta2(2)+A8_F*delta2(1)+B1_F*h(8)+B2_
F*h(7)+B3_F*h(6)+B4_F*h(5)+B5_F*h(4)+B6_F*h(3)+B7_F*h(2)+B8_F*h(1);

h(10)=A0_F*delta2(10)+A1_F*delta2(9)+A2_F*delta2(8)+A3_F*delta2(7)+A4_F*delta
2(6)+A5_F*delta2(5)+A6_F*delta2(4)+A7_F*delta2(3)+A8_F*delta2(2)+A9_F*delta2(
1)+B1_F*h(9)+B2_F*h(8)+B3_F*h(7)+B4_F*h(6)+B5_F*h(5)+B6_F*h(4)+B7_F*h(3)
+B8_F*h(2)+B9_F*h(1);

h(11)=A0_F*delta2(11)+A1_F*delta2(10)+A2_F*delta2(9)+A3_F*delta2(8)+A4_F*delt
a2(7)+A5_F*delta2(6)+A6_F*delta2(5)+A7_F*delta2(4)+A8_F*delta2(3)+A9_F*delta2
(2)+A10_F*delta2(1)+B1_F*h(10)+B2_F*h(9)+B3_F*h(8)+B4_F*h(7)+B5_F*h(6)+B6
_F*h(5)+B7_F*h(4)+B8_F*h(3)+B9_F*h(2)+B10_F*h(1);
        h(12)=A0_F*delta2(12)+A1_F*delta2(11)+A2_F*delta2(10)+A3_F*delta2(9)+A
4_F*delta2(8)+A5_F*delta2(7)+A6_F*delta2(6)+A7_F*delta2(5)+A8_F*delta2(4)+A9_
F*delta2(3)+A10_F*delta2(2)+A11_F*delta2(1)+B1_F*h(11)+B2_F*h(10)+B3_F*h(9)+
B4_F*h(8)+B5_F*h(7)+B6_F*h(6)+B7_F*h(5)+B8_F*h(4)+B9_F*h(3)+B10_F*h(2)+B
11_F*h(1);
        h(13)=A0_F*delta2(13)+A1_F*delta2(12)+A2_F*delta2(11)+A3_F*delta2(10)+
A4_F*delta2(9)+A5_F*delta2(8)+A6_F*delta2(7)+A7_F*delta2(6)+A8_F*delta2(5)+A9
_F*delta2(4)+A10_F*delta2(3)+A11_F*delta2(2)+A12_F*delta2(1)+B1_F*h(12)+B2_F
*h(11)+B3_F*h(10)+B4_F*h(9)+B5_F*h(8)+B6_F*h(7)+B7_F*h(6)+B8_F*h(5)+B9_F
*h(4)+B10_F*h(3)+B11_F*h(2)+B12_F*h(1);
        h(14)=A0_F*delta2(14)+A1_F*delta2(13)+A2_F*delta2(12)+A3_F*delta2(11)+
A4_F*delta2(10)+A5_F*delta2(9)+A6_F*delta2(8)+A7_F*delta2(7)+A8_F*delta2(6)+A
9_F*delta2(5)+A10_F*delta2(4)+A11_F*delta2(3)+A12_F*delta2(2)+A13_F*delta2(1)
+B1_F*h(13)+B2_F*h(12)+B3_F*h(11)+B4_F*h(10)+B5_F*h(9)+B6_F*h(8)+B7_F*h(
7)+B8_F*h(6)+B9_F*h(5)+B10_F*h(4)+B11_F*h(3)+B12_F*h(2)+B13_F*h(1);
        h(15)=A0_F*delta2(15)+A1_F*delta2(14)+A2_F*delta2(13)+A3_F*delta2(12)+
A4_F*delta2(11)+A5_F*delta2(10)+A6_F*delta2(9)+A7_F*delta2(8)+A8_F*delta2(7)+
A9_F*delta2(6)+A10_F*delta2(5)+A11_F*delta2(4)+A12_F*delta2(3)+A13_F*delta2(2
)+A14_F*delta2(1)+B1_F*h(14)+B2_F*h(13)+B3_F*h(12)+B4_F*h(11)+B5_F*h(10)+
B6_F*h(9)+B7_F*h(8)+B8_F*h(7)+B9_F*h(6)+B10_F*h(5)+B11_F*h(4)+B12_F*h(3)
+B13_F*h(2)+B14_F*h(1);
        h(16)=A0_F*delta2(16)+A1_F*delta2(15)+A2_F*delta2(14)+A3_F*delta2(13)+
A4_F*delta2(12)+A5_F*delta2(11)+A6_F*delta2(10)+A7_F*delta2(9)+A8_F*delta2(8)
+A9_F*delta2(7)+A10_F*delta2(6)+A11_F*delta2(5)+A12_F*delta2(4)+A13_F*delta2(
3)+A14_F*delta2(2)+A15_F*delta2(1)+B1_F*h(15)+B2_F*h(14)+B3_F*h(13)+B4_F*h
(12)+B5_F*h(11)+B6_F*h(10)+B7_F*h(9)+B8_F*h(8)+B9_F*h(7)+B10_F*h(6)+B11_
F*h(5)+B12_F*h(4)+B13_F*h(3)+B14_F*h(2)+B15_F*h(1);
        h(17)=A0_F*delta2(17)+A1_F*delta2(16)+A2_F*delta2(15)+A3_F*delta2(14)+
A4_F*delta2(13)+A5_F*delta2(12)+A6_F*delta2(11)+A7_F*delta2(10)+A8_F*delta2(9
)+A9_F*delta2(8)+A10_F*delta2(7)+A11_F*delta2(6)+A12_F*delta2(5)+A13_F*delta2
(4)+A14_F*delta2(3)+A15_F*delta2(2)+A16_F*delta2(1)+B1_F*h(16)+B2_F*h(15)+B
3_F*h(14)+B4_F*h(13)+B5_F*h(12)+B6_F*h(11)+B7_F*h(10)+B8_F*h(9)+B9_F*h(8)
+B10_F*h(7)+B11_F*h(6)+B12_F*h(5)+B13_F*h(4)+B14_F*h(3)+B15_F*h(2)+B16_
F*h(1);
        h(18)=A0_F*delta2(18)+A1_F*delta2(17)+A2_F*delta2(16)+A3_F*delta2(15)+
A4_F*delta2(14)+A5_F*delta2(13)+A6_F*delta2(12)+A7_F*delta2(11)+A8_F*delta2(1
0)+A9_F*delta2(9)+A10_F*delta2(8)+A11_F*delta2(7)+A12_F*delta2(6)+A13_F*delta
2(5)+A14_F*delta2(4)+A15_F*delta2(3)+A16_F*delta2(2)+A17_F*delta2(1)+B1_F*h(
17)+B2_F*h(16)+B3_F*h(15)+B4_F*h(14)+B5_F*h(13)+B6_F*h(12)+B7_F*h(11)+B8
_F*h(10)+B9_F*h(9)+B10_F*h(8)+B11_F*h(7)+B12_F*h(6)+B13_F*h(5)+B14_F*h(4)
+B15_F*h(3)+B16_F*h(2)+B17_F*h(1);
        h(19)=A0_F*delta2(19)+A1_F*delta2(18)+A2_F*delta2(17)+A3_F*delta2(16)+
A4_F*delta2(15)+A5_F*delta2(14)+A6_F*delta2(13)+A7_F*delta2(12)+A8_F*delta2(1
1)+A9_F*delta2(10)+A10_F*delta2(9)+A11_F*delta2(8)+A12_F*delta2(7)+A13_F*delt
a2(6)+A14_F*delta2(5)+A15_F*delta2(4)+A16_F*delta2(3)+A17_F*delta2(2)+A18_F*
delta2(1)+B1_F*h(18)+B2_F*h(17)+B3_F*h(16)+B4_F*h(15)+B5_F*h(14)+B6_F*h(13
)+B7_F*h(12)+B8_F*h(11)+B9_F*h(10)+B10_F*h(9)+B11_F*h(8)+B12_F*h(7)+B13_
F*h(6)+B14_F*h(5)+B15_F*h(4)+B16_F*h(3)+B17_F*h(2)+B18_F*h(1);
        h(20)=A0_F*delta2(20)+A1_F*delta2(19)+A2_F*delta2(18)+A3_F*delta2(17)+
A4_F*delta2(16)+A5_F*delta2(15)+A6_F*delta2(14)+A7_F*delta2(13)+A8_F*delta2(1
2)+A9_F*delta2(11)+A10_F*delta2(10)+A11_F*delta2(9)+A12_F*delta2(8)+A13_F*de
lta2(7)+A14_F*delta2(6)+A15_F*delta2(5)+A16_F*delta2(4)+A17_F*delta2(3)+A18_F
*delta2(2)+A19_F*delta2(1)+B1_F*h(19)+B2_F*h(18)+B3_F*h(17)+B4_F*h(16)+B5_
F*h(15)+B6_F*h(14)+B7_F*h(13)+B8_F*h(12)+B9_F*h(11)+B10_F*h(10)+B11_F*h(
9)+B12_F*h(8)+B13_F*h(7)+B14_F*h(6)+B15_F*h(5)+B16_F*h(4)+B17_F*h(3)+B18
_F*h(2)+B19_F*h(1);

  for i=21:(len_x)
         h(i)=A0_F*delta2(i)+A1_F*delta2(i-1)+A2_F*delta2(i-2)+A3_F*delta2(i-
3)+A4_F*delta2(i-4)+A5_F*delta2(i-5)+A6_F*delta2(i-6)+A7_F*delta2(i-
7)+A8_F*delta2(i-8)+A9_F*delta2(i-9)+A10_F*delta2(i-10)+A11_F*delta2(i-
11)+A12_F*delta2(i-12)+A13_F*delta2(i-13)+A14_F*delta2(i-14)+A15_F*delta2(i-
15)+A16_F*delta2(i-16)+A17_F*delta2(i-17)+A18_F*delta2(i-18)+A19_F*delta2(i-
19)+A20_F*delta2(i-20)+B1_F*h(i-1)+B2_F*h(i-2)+B3_F*h(i-3)+B4_F*h(i-
4)+B5_F*h(i-5)+B6_F*h(i-6)+B7_F*h(i-7)+B8_F*h(i-8)+B9_F*h(i-9)+B10_F*h(i-
10)+B11_F*h(i-11)+B12_F*h(i-12)+B13_F*h(i-13)+B14_F*h(i-14)+B15_F*h(i-
15)+B16_F*h(i-16)+B17_F*h(i-17)+B18_F*h(i-18)+B19_F*h(i-19)+B20_F*h(i-20);
      end
```

```
end
%
%       Plot results
%
%
H=fft(h.len_x);
step=cumtrapz(h);
%
%
hold off
plot(f_axis.abs(X(1:(len_x/2))/(X(1))).'b')
hold on
plot(f_axis.abs(Y(1:(len_x/2))/(Y(1))).'r')
plot(f_axis.abs(H(1:(len_x/2))).'m')
legend('X - fft of input'.'Y - fft of output'.'frequency response')
title('Chebychev Filter')
pause

hold off
plot(h)
legend('impulse response')
title('Chebychev Filter')
pause

hold off
plot(step)
legend('step response')
title('Chebychev Filter')
pause


abs_H=abs(H);
abs_X=abs(X);
abs_Y=abs(Y);

break

eval(['save ' strcat('cheb_h_PR'.num2str(PR).'_NP'.num2str(NP).'.dat') ' h -ascii'])
eval(['save ' strcat('cheb_step_PR'.num2str(PR).'_NP'.num2str(NP).'.dat') ' step -ascii'])
eval(['save ' strcat('cheb_freq_PR'.num2str(PR).'_NP'.num2str(NP).'.dat') ' abs_H -ascii'])
eval(['save ' strcat('cheb_input_PR'.num2str(PR).'_NP'.num2str(NP).'.dat') ' x -ascii'])
eval(['save ' strcat('cheb_output_PR'.num2str(PR).'_NP'.num2str(NP).'.dat') ' y -ascii'])
eval(['save ' strcat('cheb_input_fft_PR'.num2str(PR).'_NP'.num2str(NP).'.dat') ' abs_X -
ascii'])
```

```
eval(['save ' strcat('cheb_output_fft_PR',num2str(PR),'_NP',num2str(NP),'.dat') ' abs_Y -
ascii'])
```

```
%...................................................................
%
%       Program:        sub_cheb_v2.m
%
%       Coded by:       F. Winsor
%
%       Based on Algorithms provided in:
%
%       "The Scientist and Engineer's Guide to
%        Digital Signal Processing - Second Edition".
%        Steven W. Smith. California Technical Publishing.
%        1997-1999.
%
%
%       Subroutine called by Cheb_v3.m
%
%
%
%
%
%...................................................................
%
%
global FC LH PR NP k
global A0 A1 A2 B1 B2
%
%       Calculate the pole location on the unit circle.
%
%
RP=-cos(pi/(NP*2)+(k-1)*pi/NP);
IP=sin(pi/(NP*2)+(k-1)*pi/NP);
%
%       Warp froma circle to an ellipse.
%
if PR~=0
  ES=sqrt((100/(100-PR))^2-1);
  VX=(1/NP)*log((1/ES)+sqrt((1/ES^2)+1));
  KX=(1/NP)*log((1/ES)+sqrt((1/ES^2)-1));
  KX=(exp(KX)+exp(-KX))/2;
  RP=RP*((exp(VX)-exp(-VX))/2)/KX;
  IP=IP*((exp(VX)+exp(-VX))/2)/KX;
end
%
%       s domain to z domain conversion.
```

```
%
%
  T=2*tan(1/2);
  W=2*pi*FC;
  M=RP^2+IP^2;
  D=4-4*RP*T+M*T^2;
  X0=T^2/D;
  X1=2*T^2/D;
  X2=T^2/D;
  Y1=(8-2*M*T^2)/D;
  Y2=(-4-4*RP*T-M*T^2)/D;

  %
  %     lp tp lp. or lp to hp transform.
  %
  %
        if LH==1
    K=-cos(W/2+1/2)/cos(W/2-1/2);
  elseif LH==0
    K=sin(1/2-W/2)/sin(1/2+W/2);
  end

    D=1+Y1*K-Y2*K^2;
    A0=(X0-X1*K+X2*K^2)/D;
    A1=(-2*X0*K+X1+X1*K^2-2*X2*K)/D;
    A2=(X0*K^2-X1*K+X2)/D;
    B1=(2*K+Y1+Y1*K^2-2*Y2*K)/D;
    B2=(-(K^2)-Y1*K+Y2)/D;
    if LH==1
      A1=-A1;
      B1=-B1;
    end
```

```
%......................................................................
%
%       Program:        Band_reject_recur_v2.m
%
%       Coded by:       F. Winsor
%
%       Based on Algorithms provided in:
%
%       "The Scientist and Engineer's Guide to
%        Digital Signal Processing - Second Edition".
%        Steven W. Smith, California Technical Publishing.
%        1997-1999.
%
%
%       Generates a band reject filter using a recursive algorithm.
%
%
%
%
%......................................................................
%
%       Input signal to be filtered
%
%
%
%
[filename,pathname] = uigetfile('*.dat','Input Signal',50,50);
filename=lower(filename);
eval(['load ',[pathname,filename],';'])
f=findstr(filename,'.');
x=eval(filename(1:f(1)-1));
%
%
%
%       Input sample rate to calculate frequency
%       and time scales for plotting.
%
%
%
fs=input('Input Sample rate (default = 27.2168)');
if isempty(fs)
  fs=27.2168;
end
%
```

```
%
f_axis=fs*(0:511)/1024;
t_axis=(0:(1/fs):(1023*(1/fs)));
%
%
X=fft(x.length(x));
plot(f_axis.abs(X(1:(length(x)/2))))
legend('input signal fft')
pause
hold off
%
%
%       Input cutoff frequency and transition bandwidth.
%
%
%
F=input('Centre freq (0 to 0.5)  :');
BW=input('Transition bandwidth (0 to 0.5) :');
%
%
%       Determine recursion coefficients.
%
%
%
R=1-3*BW:
K=(1-2*R*cos(2*pi*F)+R^2)/(2-2*cos(2*pi*F)):
%
A0=K:
A1=-2*K*cos(2*pi*F):
A2=K:
B1=2*R*cos(2*pi*F);
B2=-R^2:
%
%
%       Apply recursion.
%
%
len_x=length(x):

y=zeros(len_x.1):

y(1)=A0*x(1):
y(2)=A0*x(2)+A1*x(1)+B1*y(1):
```

```
for i=3:(len_x)
  y(i)=A0*x(i)+A1*x(i-1)+A2*x(i-2)+B1*y(i-1)+B2*y(i-2);
end
%
%
%       Plot results
%
%
%
plot(t_axis,x)
hold on
plot(t_axis,y,'r')
legend('input signal','output signal')
title('Band Reject - Recursive')
pause
hold off
X=fft(x);
Y=fft(y);
plot(f_axis,abs(X(1:512))/X(1))
hold on
plot(f_axis,abs(Y(1:512))/Y(1),'r')
legend('input signal fft (scaled for plot)','output signal fft (scaled for plot)')
title('Band Reject - Recursive')
pause
hold off
%
%
%       Repeat procedure using a delta function to
%       obtain the impulse response.
%
%
delta=zeros(len_x,1);
%delta(1)=1.0;
delta(len_x/2)=1.0;
%
len_x=length(x);

imp=zeros(len_x,1);

imp(1)=A0*delta(1);
imp(2)=A0*delta(2)+A1*delta(1)+B1*imp(1);
```

```
for i=3:(len_x)
  imp(i)=A0*delta(i)+A1*delta(i-1)+A2*delta(i-2)+B1*imp(i-1)+B2*imp(i-2);
end
%
% Determine step response
%
step=cumtrapz(imp);
plot(step)
legend('step response')
title('Band Reject - Recursive')
pause
%
%
FR=fft(imp,len_x);
%
plot(f_axis,abs(X(1:512))/X(1))
hold on
plot(f_axis,abs(Y(1:512))/Y(1),'r')
plot(f_axis,abs(FR(1:512)),'m')
legend('input signal fft (scaled for plot)','output signal fft (scaled for plot)','frequency
response')
title('Band Reject - Recursive')
pause
hold off
%

break

abs_FR=abs(FR);
abs_X=abs(X);
abs_Y=abs(Y);

F_I=F*10000;
BW_I=BW*10000;

eval(['save ' strcat('br_rec_h_F',int2str(F_I),'_BW',int2str(BW_I),'.dat') ' imp -ascii'])
eval(['save ' strcat('br_rec_step_F',int2str(F_I),'_BW',int2str(BW_I),'.dat') ' step -ascii'])
eval(['save ' strcat('br_rec_freq_F',int2str(F_I),'_BW',int2str(BW_I),'.dat') ' abs_FR -
ascii'])
eval(['save ' strcat('br_rec_input_F',int2str(F_I),'_BW',int2str(BW_I),'.dat') ' x -ascii'])
eval(['save ' strcat('br_rec_output_F',int2str(F_I),'_BW',int2str(BW_I),'.dat') ' y -ascii'])
eval(['save ' strcat('br_rec_input_fft_F',int2str(F_I),'_BW',int2str(BW_I),'.dat') ' abs_X -
ascii'])
```

```
eval(['save ' strcat('br_rec_output_fft_F',int2str(F_I),'_BW',int2str(BW_I),'.dat') ' abs_Y -
ascii'])
```

```
%................................................................
%
%       Program:      Band_reject_win_sinc_v2.m
%
%       Coded by:     F. Winsor
%
%       Based on Algorithms provided in:
%
%       "The Scientist and Engineer's Guide to
%        Digital Signal Processing - Second Edition".
%        Steven W. Smith. California Technical Publishing.
%        1997-1999.
%
%
%       Generates a band reject filter using the algoritm
%       for the windowed sinc filter. using spectral inversion.
%
%
%
%................................................................
%
%       Input signal to be filtered
%
%
%
%
%
[filename.pathname] = uigetfile('*.dat'.'Input Signal'.50.50):
filename=lower(filename):
eval(['load '.[pathname.filename].';'])
f=findstr(filename.'.'):
x=eval(filename(1:f(1)-1)):
%
%
%
%       Input sample rate to calculate frequency
%       and time scales for plotting.
%
%
%
fs=input('Input Sample rate (default = 27.2168)'):
if isempty(fs)
  fs=27.2168:
end
```

```
%
%
f_axis=fs*(0:511)/1024;
t_axis=(0:(1/fs):(1023*(1/fs)));
%
%
X=fft(x.length(x));
plot(f_axis.abs(X(1:(length(x)/2))))
legend('input signal fft')
pause
hold off
%
%
%       Input filter length
%
%
%
M=input('Filter length - must be EVEN (default=10) :');
if isempty(M)
  M=10.0;
end
%
%
%       Input low cutoff frequency (recall Nyquist=0.5)
%
%
FCL=input('Low Frequency Cutoff - between 0.0 and 0.5 (default 0.5) :');
if isempty(FCL)
  FCL=0.5;
end
%
%       Calculate 'low frequency' filter kernel
%
for i=1:((M/2))
  hl(i)=((sin(2*pi*FCL*(i-((M/2)+1))))/(i-((M/2)+1)))*(0.42 - 0.5*cos(2*pi*i/(M+1)) +
0.08*cos(4*pi*i/(M+1)));
end

hl((M/2)+1)=2*pi*FCL;

for i=((M/2)+2):M+1
  hl(i)=((sin(2*pi*FCL*(i-((M/2)+1))))/(i-((M/2)+1)))*(0.42 - 0.5*cos(2*pi*i/(M+1)) +
0.08*cos(4*pi*i/(M+1)));
end
```

```
%
%       Normalize 'low frequency' filter kernel
%
hl_n=hl/sum(hl);
%
%
%       Input low cutoff frequency (recall Nyquist=0.5)
%
%
FCH=input('High Frequency Cutoff - between 0.0 and 0.5 (default 0.5) :');
if isempty(FCH)
  FCH=0.5;
end
%
%       Calculate 'high frequency' filter kernel
%
for i=1:((M/2))
  hh(i)=((sin(2*pi*FCH*(i-((M/2)+1))))/(i-((M/2)+1)))*(0.42 - 0.5*cos(2*pi*i/(M+1)) +
0.08*cos(4*pi*i/(M+1)));
end

hh((M/2)+1)=2*pi*FCH;

for i=((M/2)+2):M+1
  hh(i)=((sin(2*pi*FCH*(i-((M/2)+1))))/(i-((M/2)+1)))*(0.42 - 0.5*cos(2*pi*i/(M+1)) +
0.08*cos(4*pi*i/(M+1)));
end
%
%       Normalize 'high frequency' filter kernel
%
hh_n=hh/sum(hh);
%
%
%       Use spectral inversion to obtain actual high freq filter kernel
%
%
hh_n_si=hh_n*-1.0;
hh_n_si(M/2+1)=hh_n_si(M/2+1)+1.0;
%
%       Add filter kernels to get band reject kernel
%
h_n_br=hl_n+hh_n_si;
%
%
```

```
% Convolution to produce filtered output
%
y=conv(h_n_br,x);
%
%
t_axis2=(0:(1/fs):((length(y)-1)*(1/fs)));
%
plot(t_axis,x)
hold on
plot(t_axis2,y,'r')
legend('x - input signal','y - output signal')
title('Band Reject - Windowed Sinc using Spectral Inversion')
hold off
pause
%
% Determine step response
%
s_h_n_br=cumtrapz(h_n_br);
plot(s_h_n_br)
legend('step response')
title('Band Reject - Windowed Sinc using Spectral Inversion')
pause
%
% Determine frequency response (padded to length of x)
%
H_n_br=fft(h_n_br,length(x));
Y=fft(y,length(x));
plot(f_axis,abs(H_n_br(1:(length(x)/2))))
hold on
plot(f_axis,abs(X(1:(length(x)/2))/X(1)),'r')
plot(f_axis,abs(Y(1:(length(x)/2))/Y(1)),'m')
legend('frequency response','input signal fft (scaled for plot)','output signal fft (scaled for
plot)')
title('Band Reject - Windowed Sinc using Spectral Inversion')
pause
hold off
%
%
%

abs_H_n_br=abs(H_n_br);
abs_X=abs(X);
abs_Y=abs(Y);
```

```
FCL_I=FCL*10000;
FCH_I=FCH*10000;

break

eval(['save ' strcat('brw_h_M',int2str(M),'_L',int2str(FCL_I),'_H',int2str(FCH_I),'.dat') '
h_n_br -ascii'])
eval(['save ' strcat('brw_step_M',int2str(M),'_L',int2str(FCL_I),'_H',int2str(FCH_I),'.dat')
' s_h_n_br -ascii'])
eval(['save ' strcat('brw_freq_M',int2str(M),'_L',int2str(FCL_I),'_H',int2str(FCH_I),'.dat')
' abs_H_n_br -ascii'])
eval(['save '
strcat('brw_input_M',int2str(M),'_L',int2str(FCL_I),'_H',int2str(FCH_I),'.dat') ' x -ascii'])
eval(['save '
strcat('brw_output_M',int2str(M),'_L',int2str(FCL_I),'_H',int2str(FCH_I),'.dat') ' y -
ascii'])
eval(['save '
strcat('brw_input_fft_M',int2str(M),'_L',int2str(FCL_I),'_H',int2str(FCH_I),'.dat') ' abs_X
-ascii'])
eval(['save '
strcat('brw_output_fft_M',int2str(M),'_L',int2str(FCL_I),'_H',int2str(FCH_I),'.dat') '
abs_Y -ascii'])
```