

Interactive Educational Models for Structural Dynamics

by

Gilberto Mosqueda

B.S., Civil and Environmental Engineering
University of California at Irvine, 1996

Submitted to the Department of Civil and Environmental Engineering
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Civil and Environmental Engineering

at the

Massachusetts Institute of Technology

June 1998

© 1998 Massachusetts Institute of Technology
All rights reserved

Signature of Author

Department of Civil and Environmental Engineering

May 8, 1998

Certified by

Eduardo Kausel

Professor of Civil and Environmental Engineering

Thesis Supervisor

Accepted by.....

Joseph M. Sussman

Chairman, Departmental Committee on Graduate Studies

JUN 02 1998

LIBRARIES

Eng.

Interactive Educational Models for Structural Dynamics

by

Gilberto Mosqueda

Submitted to the Department of Civil and Environmental Engineering
on May 8, 1998 in Partial Fulfillment of the
Requirements for the Degree of
Masters of Science in Civil and Environmental Engineering.

ABSTRACT

An interactive program developed for the MATLAB environment is presented to assist students in learning structural dynamics. The program is intended to supplement lectures and textbooks by providing a tool for students to observe the effect of parameters on dynamic systems. The functions of the program include plotting the response of a single degree of freedom system subjected to a variety of dynamics loads. The forcing functions can be applied directly on the mass or as support motion. In addition other insightful plots can easily be accessed through the program interface, such as the transfer function and Fourier Transform of the excitation force. The graphical interface is designed to be intuitive and emphasizes simplicity in using the program, obtaining plots and more importantly, comparing results.

In analyzing the response of structures due to support excitation, the program makes use of a unique relationship that exists between the support motion and response type. A formulation is derived for the support impulse response function that, when convolved with the excitation, results in function that relates the input type to the output type. Inputting displacement results in the displacement response while an input acceleration record will produce the acceleration response.

Thesis Supervisor: Eduardo Kausel
Title: Professor of Civil and Environmental Engineering

Acknowledgements

I would like to thank Professor Eduardo Kausel for giving me the opportunity to work on this project and guiding me from start to finish. His insights were key to the success of this report

This work would not have been possible without the contribution of NSF. This project was funded by the Mid-America Earthquake Center for the purpose of innovation in undergraduate teaching methods.

I would also like to thank Olga, whose visits gave me something to look forward to and for her support in achieving my educational goals. Though school separated us for some time, the compensation will come when we're back together again.

To my brother Eddie who accompanied me during my first year in Boston. I'm also in debt to him for his consulting on the pedagogy of technology and providing valuable input into this report.

El apoyo y el cariño de mi familia durante mis años de estudio, han echo todo esto posible. Por eso, le quiero dar las gracias a mis padres, Pedro y Alicia, y a mi hermana Patricia por siempre ayudarme en todas las maneras posibles. A mis hermanitos, Pedro Jr. y Jose, que algun dia llegen a cumplir mas que yo, se que si pueden.

Graduate school was not even an option until Professor Villaverde gave me the opportunity to get involved in research at UCI and develop an interest in academia.

I would not have been able to survive in Boston without certain individuals who gave me some memorable times here. For this I would like to thank Jose y Lorena for getting me out of MIT every now and then and especially for proofreading this report; My roommates Carlos and Hector, we got to do Puerto Rico again; Abel for his advice on computers when they were driving me crazy; My classmates: Casey, Emma, Kathy, Mia, Patricia, David, Gerry, Larry, Mark, and Salvatore.

En memoria de mi Abuelita, Soledad Cruz.

TABLE OF CONTENTS

LIST OF FIGURES	10
LIST OF TABLES	11
CHAPTER 1	13
INTRODUCTION	
CHAPTER 2	17
USING TECHNOLOGY TO ENHANCE EDUCATION	
2.1 LEARNING ENVIRONMENT	19
2.2 INTERFACE DESIGN	22
2.3 EDUCATIONAL SOFTWARE	24
CHAPTER 3	27
MATLAB IN EDUCATION	
3.1 CREATING EDUCATIONAL MODELS	28
3.2 DISTRIBUTING MATLAB SOFTWARE.....	29
CHAPTER 4	31
SDOF SYSTEM	
4.1 EQUATIONS OF MOTION	31
4.2 FREE VIBRATION	33
4.3 FORCE ON MASS.....	34
4.3.1 <i>Harmonic Loading</i>	34
4.3.2 <i>Random Loading</i>	35
4.3.2.1 State-Space Solution.....	35

4.4	SUPPORT MOTION	37
4.4.1	<i>Harmonic Excitation</i>	38
4.4.2	<i>Random Excitation</i>	38
4.4.2.1	Convolution.....	39
4.4.2.2	Input-Output relation for support motion	41
4.5	PERIODIC LOADING– INITIAL CONDITIONS.....	44
4.5.1	<i>Force On Mass</i>	44
4.5.2	<i>Support Motion</i>	46
4.6	FOURIER TRANSFORM	46
CHAPTER 5	47

PROGRAM DESCRIPTION

5.1	INPUT OPTIONS.....	48
5.1.1	<i>System Properties</i>	51
5.1.2	<i>Initial Conditions</i>	51
5.1.3	<i>Display Parameters</i>	52
5.2	FORCING FUNCTION	53
5.2.1	<i>Free Vibration</i>	54
5.2.2	<i>Sinusoidal</i>	55
5.2.3	<i>Point Data</i>	56
5.2.4	<i>File</i>	58
5.2.5	<i>Transient/Periodic Loading</i>	60
5.3	OUTPUT OPTIONS	61
5.3.1	<i>Displacement Response</i>	61
5.3.2	<i>Plot Options Menu</i>	61
5.3.2.1	Forcing Function	62
5.3.2.2	Transfer Function and Phase Angle.....	63
5.3.2.3	Load Fourier Transform	64
5.3.2.4	Response Fourier Transform	65
5.4	PROGRAM STRUCTURE	66

CHAPTER 6	69
APPLYING THE SOFTWARE	
6.1 PROGRAM GOALS	70
6.2 EXAMPLES OF USE	72
6.3 PROBLEM SET	73
6.3.1 <i>Free Vibration</i>	74
6.3.2 <i>Harmonic Excitation</i>	75
6.3.3 <i>Earthquake Engineering</i>	78
6.4 REMARKS	82
CHAPTER 7	83
CONCLUSION	
BIBLIOGRAPHY	85
APENDIX.....	87

LIST OF FIGURES

Figure 1. Cart model for single degree of freedom system.....	32
Figure 2. Free body diagram for force on mass.	32
Figure 3. Free body diagram for support motion.	33
Figure 4. Screen snapshot of main window.	49
Figure 5. System properties input options.	51
Figure 6. Initial Conditions input options.	52
Figure 7. Display Parameters input options.	53
Figure 8. Sinusoidal Loading Options	55
Figure 9. Force Time History produced from settings in Figure 8.	56
Figure 10. Point Data input options	57
Figure 11. Excitation created from input in Figure 10.....	57
Figure 12. File selection input options.....	59
Figure 13. Forcing function created with <i>data.in</i> input file.	59
Figure 14. Periodic loading with period of 2 seconds.....	60
Figure 15. Excitation Plot corresponding to response in Figure 4.....	62
Figure 16. Transfer Function and Phase Angle corresponding to Figure 4.	63
Figure 17. FFT of both forcing functions shown in Figure 16.	64
Figure 18. FFT of response plots displayed in Figure 4.	65
Figure 19. Flow chart of SDOF PLOT program (Part 1).....	67
Figure 20. Flow Chart of SDOF PLOT program (Part 2).....	68
Figure 21. Display from Free Vibration problem.	74
Figure 22. Response for case a.....	76
Figure 23. Response for case b.	76
Figure 24. Response for case c.....	77
Figure 25. Transfer function for 1% and 5% damping.	78
Figure 26. First 20 seconds of the acceleration record for El Centro.....	79

Figure 27. FFT of El Centro acceleration record show in Figure 26.	80
Figure 28. Response to El Centro with 1% and 5% damping.	81
Figure 29. FFT for Response plots shown in Figure 28.	81

LIST OF TABLES

Table 1. Unit combinations for Free Vibration and Force on Mass.	50
Table 2. Units' combinations for Support Motion.	50

CHAPTER 1

INTRODUCTION

The purpose of this project is to develop a tutorial software package to assist in the teaching of structural dynamics. In dynamics, the complex equations and variable motion of bodies under dynamic loads make it difficult for many students to visualize the problems they are challenged with. Although it is the conceptual understanding of theories and equations that are essential for problem solving, a firm visual representation could simplify the learning process for students that may experience difficulties. This program aims to overcome some of the limits of traditional teaching tools by making use of modern technology to incorporate interactive figures into the structural dynamics curriculum.

Since the phenomenon under study is time dependent, teaching and learning in dynamics has great potential for improvement with the use of visual tools provided by computer graphics. Computers offer the ability to create animations and interactive figures, which are not possible with current teaching tools such as textbooks and chalkboards. The advantage of using computer graphics is that they are able to represent the dynamic phenomena adequately and allow the user to experiment with the effect of parameters (Chabay and Sherwood, 1992).

These new educational technologies are not intended to replace existing teaching methods, but rather to increase the availability of options for instructors to present material to students (Bracket, 1998). The use of computers as teaching tools is relatively new, thus the effect they will have on improving the learning process is not precisely known. In order to tap the potentials that exist for technology in education, there is a need for educators to introduce innovative teaching tools into the field. Moreover, it seems logical that the use of computers in education would have the greatest impact in a course such as structural dynamics where pictures in books cannot display the subject matter under consideration.

In an effort to improve the teaching of structural dynamics, an interactive tutorial program was developed for students enrolled in a course at the undergraduate level. The program allows the user to quickly obtain the graphical response of a single degree of freedom (s dof) system subjected to a variety of dynamic loads. The input parameter for the program could be varied, including mass, stiffness, damping ratio, and applied dynamic loads.

The tutorial program is designed with student-oriented features in mind. The program is user friendly making it simple to use and understand. It was designed to simplify changing system parameters, create forcing functions, and graph the response to these dynamic loads. Within minutes, the user of the program can begin to create plots and observe the role each variable plays in controlling the behavior of a vibrating system. Curious students could quickly obtain answers to “what if?” scenarios by modifying input variables and pressing a button to plot the response. In addition to understanding behavior,

recognizing these same variables within equations will simplify the mathematical expressions for the student and build on their problem solving skills.

The SDOF PLOT program was created as a function for the MATLAB (MATrix LABoratory) computing environment. Starting the function from the MATLAB command prompt will produce the user-friendly interface. Students could use this program on their own time to review dynamics or to assist them in completing homework assignments. Students' personal use of the program will provide an inexpensive option for tutorials that can be accessed at any time without the limitations of tutors whom are, unfortunately, scarcely available. The program can also be used as a tool to supplement lectures. The display can be projected onscreen to provide students with visual aids that can lead to a deeper understanding of the concepts being presented to them. It is precisely these enhancements that technology can add to a teaching environment that will be discuss next.

CHAPTER 2

USING TECHNOLOGY TO ENHANCE

EDUCATION

Perhaps the largest difference between computers and existing educational media is the two-way, dynamic capabilities of interactive programs (Chabay and Sherwood, 1992). Interactive programs require active input from the user and are able to modify the display according to the user's response. In fact, some programs require interaction from the user in order for anything to happen at all. This works as an advantage for computer tutorials since they move at the pace of the user.

Other existing educational media are primarily passive and of a one-way nature. Such limited student interaction with learning material relegates students to the role of passive observers whom are expected to absorb new information as it is passed on from instructors and textbooks. A cautionary note about lectures is that they tend to move rapidly and at a pace that may be comprehensible to only a small fraction of the students present. Appropriate technologies can eliminate such constraints through software tutorials that require interaction and move at the pace of the user. Means and Olsen (1995) assert that advanced levels of comprehension and conceptual understanding take place not by the

passive reception of facts but by the active processing of information. This is in agreement with the constructivist theory on learning behavior. “Central to the vision of constructivism is the notion of the organism as ‘active’ – not just responding to stimuli, as in the behaviorist rubric, but engaging, grappling, and seeking to make sense of things.” (Perkins, 1992)

Because of its uniqueness from traditional teaching methods, technology in the classroom offers the potential to better reach current educational goals, and may allow educators to reach previously unobtainable goals (Brackett, 1998). The expansion of teaching methods allows students to learn in a variety of environments, offering more options from which to acquire their knowledge. It has been the author’s experience as a learner and teaching assistant in a course on structural dynamics that students sometime need theories explained from different perspectives before they are able to fully understand concepts. Sometimes an explanation in the lecture might suffice, at other times the book will function best, or even an explanation from another student could be much better in getting the point across to the learner.

Computers simply provide additional means for students to access information that may lead to a clearer understanding of concepts presented through lectures or textbooks. Computers might work well in some areas and might only work for some students. Overall, technology could supplement the learning process, and possibly speed it up, by providing alternate teaching and learning tools. As Alfred Bork (1987) states, the main advantage of the computer as a teaching tool is it allows educators to make learning

interactive for all students. The needs of each student can be given more attention by individualizing the learning experience.

While technology can help create powerful educational opportunities for students, technology in and of itself is not enough to sufficiently enhance the process of teaching. Salomon and Perkins (1996) assert that in order for computer technology to profoundly affect learning in a drastic manner, the whole culture of the learning environment must be considered.

2.1 Learning Environment

The potential that exists for computers in education can be better understood by looking at the different components of an educational environment. David Perkins (1992) describes the five facets of a learning environment as: information banks, symbol pads, construction kits, phenomenaria, and task managers. Furthermore, he explains how each facet could be improved amid new information processing technology.

1. **Information banks** consist primarily of textbooks and other references such as dictionaries and encyclopedias. Lectures also fall into this category. Technology expands the amount of information available, as well as shorten the access path. An example of this is the World Wide Web (WWW) with the use of hypertext links providing access to information all over the world.
2. **Symbol Pads** provide surfaces for the construction and manipulation of symbols primarily for the purpose of supporting the learner's short-term memories. In addition to notepads, technology has introduced word processor and drawing applications featuring easy editing and rearranging of large portions of symbols.

3. **Construction Kits** include laboratory apparatus that allow students to assemble objects and simulations; a classical example being Lego building blocks. This area has been dramatically expanded by computers, allowing for abstract entities such as commands in programming language, creatures in a simulated ecology, or equations in a mathematics program.
4. **Phenomenaria** serve the specific purpose of presenting phenomena and making them accessible to scrutiny and manipulation. Information technology provides flexible resources for creating complex phenomenaria such as simulations.
5. **Task Managers** provide a means of setting tasks, guides, and sometimes help in the execution of these tasks. In this aspect, information-processing technologies bring the possibility of electronic task managers.

Perkins' facets seem to focus on K-12 or pre-college teaching, though they could arguably apply to a learning environment in higher education, the focus of this paper. Moreover, the high availability of computers in universities, unlike many preparatory schools, allows for an even better implementation of computers into the learning environment.

A typical classroom setting consist mainly of "information banks" (teacher and text), "symbol pads" (notebooks, scratch paper, worksheets), and "task managers" (the teacher, written instructions) (Perkins, 1992). Through laboratory exercises, some classes do include "phenomenaria" (physics demonstrations, strength testing of materials) and "construction kits" (building circuits, programming). Most of these laboratory settings are only available for a limited time since they usually require the instructor or an assistant to

be present. In addition, expensive equipment is put in jeopardy by allowing novice users to experiment freely. To avoid this risk, the lab supervisor performs the experiment while the students simply observe without much interactive input into the exercises.

Using technology, present components of the learning environment could be enhanced and those that are non-existent could be introduced to build a more effective environment. For example, the recent introduction of the COMMAND (Course Management and Delivery System) system at MIT provides improved information banks and task managers by placing courses virtually on-line (Sanchez, 1998). The course homepage contains lecture notes, homework assignments and a timetable displaying due dates for upcoming assignments. When applicable, students could submit their homework online as well. With most of the related course information on the web, not only can students access documents faster, but they can also access them from any computer online. The COMMAND environment could be further enhanced to include animations and interactive figures by incorporating a web-based learning environment such as *Momentous*, currently under development (Shepherdson, 1998).

Computers have potential to enhance each aspect of an educational environment, however, most work is needed in construction kits and phenomenaria since they appear to be the most deficient in current classrooms. It is particularly these two areas that this project targets to improve. With respect to phenomenaria, computers displaying simulations could be projected in the classroom as a visualization tool to enhance the lecture. Virtual laboratories could also be created that will allow the user to build an apparatus and simulate its behavior. An electronic learning package, including simulations

and interactive programs would make phenomenaria and construction kits available to the students at all times at which computers are available. In addition, software will allow students to experiment freely with simulations at their own pace. The MATLAB/Simulink package provides an excellent example of an interactive electronic environment for system dynamics (Tomovic, 1996). Another example is the Working Model™ package that simulates the physical behavior of dynamic systems including kinematics and mechanical vibrations (Interactive Revolution).

One of the truly great advantages of infusing technology into educational practices is that computer simulations can encourage students to think about problems in different ways. While simulations could probably be better demonstrated without a computer, physical experiments are also very expensive to arrange and present (Kerr, 1996: p. 11).

2.2 Interface Design

Developing successful educational software, especially at the college level, requires three main pools of knowledge for the designer. One area of expertise is, naturally, a solid background on the subject for which the program is intended. Developing software also requires programming skills, particularly in the area of building a user-interface. The other, which increases program usability, is a level of knowledge as to what works in terms of computer aided education. A highly sophisticated program with exceptional content will do very little for users if these contents are not easily accessible. A user-friendly interface with clear graphics is almost as important as the contents of the program.

According to Chabay and Sherwood (1992), when designing an interface “a major challenge is to produce graphics and text displays that make intuitively obvious what next moves are possible”. An intuitive program is particularly important in getting students to actually use the program. It is questionable as to how much time students will spend using educational software if it requires time to learn how to use. One might think that on-screen instructions are sufficient to insure ease of use, however, even experienced users do not read online instructions.

In order to make the program as user-friendly as possible, it is essential that the display is made central to designing the application. The screen should not be cluttered with information since this could distract the user’s attention from significant objects. Placing emphasis on dominant graphics by highlighting, enlarging, or bolding will attract attention to items of importance. Also, only necessary graphics such as those that are currently active should remain on the screen. Other irrelevant graphics need to be removed or shown on separate slides. A clear and concise screen is important to keep the student focused on what is important and to avoid confusing the user with more options than are necessary.

To take full advantage of technology, an interface should allow for direct manipulation of parameters and conditions from the user. Without this, the contents will be presented in a fashion similar to that of a book with preset pictures. Controls to manipulate input variables could be simplified by restricting user actions to only one input device, the mouse or keyboard. If the keyboard is required, it is best to limit its use, especially to type in long commands. Also, due to the random possibilities of keyboard

input, it needs to be checked for validity with feedback provided to the user on mistakes or program limitations.

It is important that the user be aware of what actions are possible at all times. The user should feel a sense of control over the program to avoid frustration, otherwise the program will become a hindrance rather than a tool. In order to insure its use among students, the program needs to simplify learning and not provide more material to learn. Metaxas (1996) states, "Keeping it simple pays off not only in terms of production cost and robustness, but it also... increases usability". The above suggestions in design, could improve the use and effectiveness of educational programs among students.

2.3 Educational Software

Within the last several years, engineering educators have been highly active in developing educational software, though most work has been concentrated in electrical engineering education. Only a few developers have published results from their experience using technology in the classroom, and for this reason, not enough extensive studies have been conducted to determine the success computer tutorials may have at the college level. In Civil Engineering, very little has been done in terms of educational software. The few existing programs dealing with dynamics are focused more on kinematics. Software on mechanical vibrations with applications to structural dynamics is almost non-existent. Following is a list of software programs developed and used for engineering education.

At Rose-Hulman Institute of Technology, the program "Working Model" is being used to help students visualize problems and develop their dynamic intuition (Cornwell,

1996). Results of surveys given to students show that this simulations program was rated as the best teaching tool in terms of visualization and intuition. Students also suggested that the program was not as effective in teaching “problem solving skills” and “learning/comprehension”. Lectures and homework were found to be the best tool for this purpose.

Another program “BEST” (Basic Engineering Software for Teaching Dynamics), which also focuses on kinematics, was developed to improve teaching at the university of Missouri—Rolla. Here it was found that students “gained some visualization ability but they did not, ... gain a significantly greater understanding of the conceptual elegance of dynamics or problem solving ability” from the program (Flori, 1994). The instructor found the most effective use of the program was to run simulations on screen during lectures then turn to the blackboards to write the governing equations. This procedure proved to be successful in helping students to better understand the equations and concepts.

Observations from use of the MATLAB/Simulink package at Purdue University reveal that visual software helps students spend more time on analyzing engineering problems and optimizing the design instead of spending time number crunching and learning the mathematics (Tomovic, 1996). Here, the software was applied to a system dynamics course for mechanical engineers. Additionally, Tomovic mentions that the success of applying MATLAB is partly due to the ease of learning how to use the software.

From the reviewed programs, the use of technology in the classroom, particularly in dynamics, is assisting students in their visualization skills. It also appears as though some programs did not work as well as intended. In order to have a more successful

implementation of software tutorials, project goals need to be clearly defined. In addition instructors need to adopt new pedagogical techniques, which are supportive of technology and help meet the set goals. Simply distributing software among students will not assure its proper use. Homework assignments need to be modified to include technology in the curriculum. Problem sets could ask more advanced questions that can be solved by applying the software to assure students properly use the software.

CHAPTER 3

MATLAB IN EDUCATION

This SDOF PLOT application was developed using the MATLAB environment. MATLAB has become the premium technical computing environment in engineering and science both in industry and government (Math Works, 1995). The professional version of the MATLAB software is quite expensive. However, *The Student Edition of MATLAB*, makes it affordable for students with only insignificant limitations. These limitations are mainly in the size of matrices that can be used (maximum of 8192 elements).

The extremely high level language of MATLAB performs vast numerical, as well as symbolic, operations from the command line or from a script program file. The ability to create objects such as matrices and to perform operations on the fly makes MATLAB a good example of a construction kit. Its powerful graphics library also gives MATLAB the capabilities to display phenomenaria. Indeed, it has had great success in academia as it has been used “not simply to get the answers, but rather to understand how to get the answers” (Math Works, 1995).

The MATLAB package comes complete with demonstrations and sample programs, some of which contain great educational value. The DEMO library helps in two aspects, by: 1) by explaining mathematical concepts with the use of graphics; and 2)

teaching how to use the actual MATLAB software for various purposes. Many of the demonstration programs include a slide show explaining the different steps to solve the stipulated problem, concurrently showing the MATLAB commands used to execute each step of the analysis.

3.1 Creating Educational Models

MATLAB offers a variety of mathematical functions, as well as a graphics library, making it ideal for creating interactive educational modules. The developer needs not worry about creating plotting functions, scaling the axis, or writing complicated mathematical algorithms such as convolution to get a program running. MATLAB functions provide for the creation of interactive figures in the minimal required programming time, allowing the developer to concentrate on the application's effectiveness. Information on applying MATLAB functions used within the proposed program is presented in Chapter 4.

The SDOF PLOT program was created as a function for the MATLAB computing environment, requiring the user to have MATLAB installed in their computer to run the application. Initially, the program was developed using MATLAB version 4.3 in the UNIX environment. With the release of version 5.1 within the final stages of development, the program was upgraded to remove functions that will be eliminated in future versions. Basically, the program was modified to remove errors or warning given by both versions used. In the final stages, the program was tested using version 5.1 on both UNIX and PC environments as well as the UNIX version 4.3. Both versions used in UNIX consisted of

the professional edition of MATLAB. These same versions are available to all MIT students through the ATHENA network. On the PC, the professional and the student versions 5.1 were used for testing and the program appeared to functioned well on both editions. A technical note on using PCs, the main window appears as transparent when first called from the MATAB prompt on some computers. Minimizing the window to the toolbar and, then again maximizing, appeared to get the window back to normal view. Other PC test also resulted in different color schemes such as the screen producing a quick flash in a color similar to the plotting color. Besides the color coordination, the calculations and plots worked well and where clearly visible.

3.2 Distributing MATLAB Software

Custom MATLAB functions consists of an ASCII text file that the software compiles at execution time. Keeping the program files in this format gives MATLAB some platform independence. This program, for example, was developed using a UNIX environment and was transferred directly to the PC without making any additional changes.

In a computer environment such as the one established at MIT, students will be able to access the script file containing the MATLAB program from a course locker and run MATLAB from the MIT-ATHENA network. The UNIX network is available through computer clusters campus-wide providing convenient access for students on campus.

A disadvantage in using MATLAB to develop educational models is that the user is required to have access to MATLAB in order to execute the program. In addition, the user must also have access to the script files containing the program. This could provide access

problems for students wishing to study off-campus or using this type of program at schools which are not as conveniently wired as MIT. Another option is for students to purchase *The Student Edition of MATLAB* that is comparable in cost to course textbooks.

However, MATLAB functions are not only restricted to the MATLAB environment. An additional compiler is available which transform MATLAB code into C or C++ code, enabling the program to run independent of MATLAB.

CHAPTER 4

SDOF SYSTEM

This chapter explains the methods used to solve for the response of a sdof system as illustrated in the SDOF PLOT program. A brief overview of the equations and functions used within the program will be given. Different methods were used to solve for the displacement response of the sdof model, depending on the forcing selected. The response resulting from free vibration and harmonic excitation is solved from an analytical expression while the rest of the problems required a numerical technique. The numerical methods used for arbitrary forcing functions are the convolution integral and a solution in the state-space form using a MATLAB function.

4.1 Equations of Motion

This program computes the response of a single degree of freedom system that can be familiarly modeled as a rolling mass attached by a spring and viscous damper (Figure 1). Within the program, the vibrating system can be excited in three different ways: 1) *Initial Conditions*; 2) *Force on Mass*; and 3) *Ground Motion*. Disturbances due to *Initial Conditions* can be combined with the other two loading conditions, but, *Force on Mass* and *Ground Motion* are mutually exclusive. Using the rolling cart model for a sdof, the

equations of motion are obtained for the different forcing conditions. The model consists of a mass, m , a spring with linear stiffness, k , and a viscous damper with a constant, c .

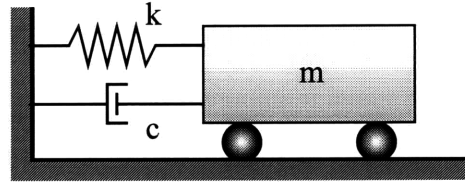


Figure 1. Cart model for single degree of freedom system.

In the case that a time variant force is applied on the mass, the equation of motion in terms of displacement, v , can be obtained from the free body diagram in Figure 2.

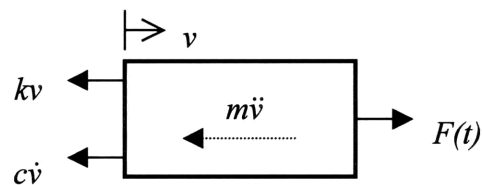


Figure 2. Free body diagram for force on mass.

$$m\ddot{v} + c\dot{v} + kv = F(t) \quad (1)$$

For support motion, v_g , the equation of motion in terms of absolute response is given by equation (2). An equivalent equation can also be expressed in terms of the relative response y , where $y = v - v_g$.

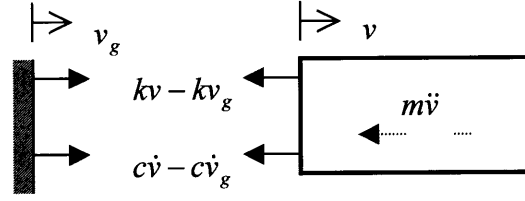


Figure 3. Free body diagram for support motion.

$$m\ddot{v} + c\dot{v} + kv = c\dot{v}_g + kv_g \quad (2)$$

$$m\ddot{y} + c\dot{y} + ky = -m\ddot{v}_g \quad (3)$$

4.2 Free Vibration

In the event that the system is subjected to some initial disturbance, and left to vibrate freely with no other interaction on the system, then the forcing function becomes zero. Using equation (1), the resulting equilibrium equation is a second order homogenous differential equation with a known analytical solution. For free vibration, given initial velocity, \dot{v}_0 , and displacement, v_0 , the program uses equation (5) to solve for the displacement, $v(t)$. In equation (5), the natural frequency $\omega = \sqrt{k/m}$, the damped frequency $\omega_d = \omega\sqrt{1-\xi^2}$, and the damping ratio is given by $\xi = \frac{c}{2\sqrt{km}}$.

$$m\ddot{v} + c\dot{v} + kv = 0 \quad (4)$$

$$v(t) = e^{-\xi\omega t} \left(v_0 \cos(\omega_d t) + \frac{\dot{v}_0 + \xi\omega v_0}{\omega_d} \sin(\omega_d t) \right) \quad (5)$$

4.3 Force on Mass

In the case that the force is applied on the mass, then the following mathematical solutions are implemented in the program to solve for the response.

4.3.1 Harmonic Loading

Free vibration apart, the simplest response to obtain from a dynamic system is that due to harmonic loading. Applying a simple harmonic forcing function on the mass also has a known analytical expression for the response. The solution for the displacement given by equation (7) considers only the steady state response of the vibrating system due to a harmonic force of magnitude P_0 .

$$m\ddot{v} + c\dot{v} + kv = P_0 \sin(\omega t) \quad (6)$$

$$v(t) = H(r) \frac{P_0}{k} \sin(\omega t - \phi) \quad (7)$$

In equation (7) the transfer function $H(r)$ and the phase angle ϕ are give by:

$$H(r) = \left[\frac{1}{(1-r^2)^2 + (2\xi r)^2} \right]^{1/2} \quad (8)$$

$$\phi = \tan^{-1} \left(\frac{2\xi r}{1-r^2} \right) \quad (9)$$

Consequently, these same equations are used to calculate the graphs of the transfer function and phase angle, which can be displayed by the program. For plotting purposes, the value of the frequency ratio, r , is given a range from 0 to 3.

4.3.2 Random Loading

Since an exact analytical solution cannot be derived for all possible cases, solving the response to any arbitrary loading requires the use of numerical techniques. One of the simplest tools to use in MATLAB is the state-space solution, which works effectively for any arbitrary load vector. In addition, this function can also account for the effects of initial conditions.

4.3.2.1 State-Space Solution

MATLAB has a variety of functions available for solving differential equations in the state-space form. For this reason it can be convenient to express a structural system subjected to dynamic loading in terms of state-space equations. The transformation is quite simple and can be done in terms of the mass, stiffness, and damping matrices, even when dealing with multi-degree of freedom systems. These matrices are essential to solving the equations of motion in any form. The transformation from the equilibrium equation of motion to the state-space form will be shown for a sdof system. Once the state-space equations are known, MATLAB provides a function to solve for the response of the system given any arbitrary loading vector and initial conditions.

Equations in state-space form consist of a set of first order differential equations. In the case of a single degree of freedom system, two first order equations are used to represent the second order equation of motion. The state-space form for use in MATLAB is as follows:

$$\begin{aligned}\dot{x} &= Ax + Bu \\ y &= Cx + Du\end{aligned}\tag{10}$$

The goal here is to represent the equations of motion in the form of equation (10).
Recalling the equation of motion for a single degree of freedom system and solving for the acceleration, \ddot{v} , equation (11) is obtained.

$$\begin{aligned}m\ddot{v} + c\dot{v} + kv &= F(t) \\ \ddot{v} &= -\frac{c}{m}\dot{v} - \frac{k}{m}v + \frac{1}{m}F(t)\end{aligned}\tag{11}$$

Next provide the following substitutions:

$$\begin{aligned}x_1 &= v \\ x_2 &= \dot{v} \\ u &= F(t)\end{aligned}\tag{12}$$

to obtain two equivalent first order equations.

$$\begin{aligned}\dot{x}_1 &= 0x_1 + x_2 + 0u \\ \dot{x}_2 &= -\frac{k}{m}x_1 - \frac{c}{m}x_2 + \frac{1}{m}u\end{aligned}\tag{13}$$

Letting, the value of $y = x_1 = v$, the state space equations can be expressed in matrix form as:

$$\begin{aligned}\begin{Bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{Bmatrix} &= \begin{bmatrix} 0 & 1 \\ -k/m & -c/m \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix} + \begin{Bmatrix} 0 \\ 1/m \end{Bmatrix} u \\ \{y\} &= [1 \quad 0] \begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix} + \{0\}u\end{aligned}\tag{14}$$

In MATLAB, equations in state-space form with continuous time and arbitrary input can be solved using the LSIM function. The command for executing this function contains the following syntax:

$$[x, y] = \text{LSIM}(A, B, C, D, t, u, y0, v0)$$

The required input matrices A, B, C, and D used in equation (10) are given for a sdof system in equation (14). The inputs t and u are both vectors that must be of the same length. Vector t contains continuous time points and vector u contains the forcing function corresponding to the time vector. The time points in vector t must be equally spaced with all points having the same increment. The last two optional input variables specify the systems initial conditions, namely displacement ($y0$) and velocity ($v0$). Finally the output x and y give the displacement and velocity response in vector form. These output vectors are also similar in size to t and correspond in time with the input time vector. For an input time vector of size $1 \times n$, then x is a $2 \times n$ matrix consisting of both displacement and velocity, one in each row of the matrix. The vector y of size $1 \times n$ contains repetitive data for the displacement.

4.4 *Support Motion*

In addition to applying the force on the mass, this program also allows for analysis of the response of structures due to *Support Motion*. The ground excitation functions that can be created for this selection are identical to the options available for *Force on Mass*.

4.4.1 Harmonic Excitation

When the support excitation is harmonic, an analytical solution also exists for the steady state response. Using the equation for relative motion, the equilibrium equation can be expressed by equation (15). The response is then given by equation (16).

$$m\ddot{v} + c\dot{v} + kv = v_{g0} \sin(\omega t) + \omega v_{g0} \cos(\omega t) \quad (15)$$

$$v(t) = H_g(r) v_{g0} \sin(\omega t - \phi_g) \quad (16)$$

In the above equation, the support transfer function, $H_g(r)$, and the phase angle, ϕ_g , for the response due to harmonic support excitation are given by:

$$H_g(r) = \left[\frac{1 + (2\xi r)^2}{(1 - r^2)^2 + (2\xi r)^2} \right]^{1/2} \quad (17)$$

$$\phi_g = \tan^{-1} \left(\frac{(2\xi r)^3}{(1 - r^2) + (2\xi r)^2} \right) \quad (18)$$

4.4.2 Random Excitation

In the case of support motion a different solution technique is used to solve for the response to arbitrary loading. The method used in this case is convolution, primarily because of the convenient input-output relationship that results from applying this method. First, the method of convolution will be summarized, then an extremely flexible solution to support motion based on the convolution integral will be given.

4.4.2.1 Convolution

The convolution integral provides a general solution to arbitrary excitation, which can be easily implemented in computer applications. To solve for the response of a sdof system using convolution, it is necessary to first obtain Duhamel's Integral.

A unit impulse applied to a sdof system at time $t = \tau$ is transferred to the system in the form of momentum, i.e.

$$F\Delta t = m\dot{v} = 1$$

After the impulse, the system absorbs the momentum resulting in some displacement and velocity of the system. Since no loading occurs thereafter, the system is left to vibrate freely with the initial conditions induced by the impulse. These initial conditions due to a unit impulse are:

$$v = 0, \quad \dot{v} = \frac{1}{m}$$

The total response for a unit impulse applied at time $t = \tau$ could then be expressed as the free vibration response given by equation (5) with the above initial conditions.

$$\begin{aligned} h(t - \tau) &= e^{-\xi\omega t} \left(0 \cdot \cos(\omega_d t) + \frac{1/m + \xi\omega \cdot 0}{\omega_d} \sin(\omega_d t) \right) \\ &= e^{-\xi\omega t} \left(\frac{1}{m\omega_d} \sin(\omega_d t) \right) \end{aligned} \quad (19)$$

The resulting expression for $h(t - \tau)$ is the impulse response function due to a unit impulse applied at time τ .

Now suppose we have an arbitrary load $F(t)$. Any load can be divided along the time axis into segments of width dt and be considered as the sum of several impulses. The system response to this arbitrary load is then the sum or superposition of the response due to each individual impulse. Symbolically this can be written as:

$$v(t) = \sum_{i=0}^n h(t - \tau_i)F(\tau) \quad (20)$$

As the time step dt becomes infinitesimally small, the summation described in equation (39) can be written in integral form as Duhamel's Integral.

$$v(t) = \int_0^t h(t - \tau)F(\tau)d\tau \quad (21)$$

This integral can be more conveniently expressed as the convolution of the two integrated functions. The displacement response of a sdof system subjected to an arbitrary load, $F(t)$, can be expressed as:

$$v(t) = h(t) * F(t) \quad (22)$$

MATLAB also provides a function for execution of the convolution integral. The solution for the displacement using convolution can be carried out in MATLAB code as:

$$v = \text{conv}(h, F) * dt.$$

The input vectors, h and F are the two time functions to be convoluted, namely the impulse response function and the forcing function. Multiplying the result by the time-step (dt) used in both input vectors is required to obtain the correct magnitude of the response since MATLAB assumes a time step of one.

4.4.2.2 Input-Output relation for support motion

This program makes use of a special relationship that exists between support excitation and absolute response type. The formulation is based on a support impulse response that functions regardless of the input record selected, whether it is acceleration, velocity, or displacement. The response type or output displayed by the program is consistent with the input selected.

In terms of the convolution integral, the solution for relative displacement given ground acceleration, as expressed in equation (3), is given by equation (23). The effective force, $F(t)$, is substituted by the right hand side of equation (3).

$$y(t) = -m\ddot{v}_g * h \quad (23)$$

The relative velocity and acceleration of the system can also be found by differentiating the above equation to obtain:

$$\dot{y}(t) = -m\dot{v}_g * \dot{h} \quad (24)$$

$$\begin{aligned} \ddot{y}(t) &= \frac{-m\ddot{v}_g}{m} - m\dot{v}_g * \ddot{h} \\ &= -\ddot{v}_g - m\dot{v}_g * \ddot{h} \end{aligned} \quad (25)$$

Equation (25) for relative acceleration can be rewritten in terms of the absolute acceleration by substituting $\ddot{v} = \ddot{y} + \ddot{v}_g$.

$$\ddot{y}(t) + \ddot{v}_g = -m\dot{v}_g * \ddot{h} \quad (26)$$

$$\ddot{v} = -m\dot{h} * \ddot{v}_g \quad (27)$$

The absolute acceleration can now be obtained given a ground acceleration record and the second derivative of the impulse response function. The latter function can be obtained by taking the second derivative with respect to time of the impulse response function given by equation (19).

$$\dot{h}(t) = \frac{e^{-\xi\omega t}}{m\omega_d} (\omega_d \cos(\omega_d t) - \xi\omega \sin(\omega_d t)) \quad (28)$$

$$\ddot{h}(t) = \frac{e^{-\xi\omega t}}{m\omega_d} (2\xi\omega\omega_d \cos(\omega_d t) + \omega^2(1 - 2\xi^2) \sin(\omega_d t)) \quad (29)$$

From equation (27), the expression $-m\dot{h}$ can then be thought of as the impulse response function for absolute acceleration given ground acceleration.

Turning now to equation (13) for absolute displacement response, its solution can also be obtained in terms of convolution. The solution to this equation consists again of the convolution between the impulse response function and effective loading:

$$v(t) = h * (c\dot{v}_g + kv_g) \quad (30)$$

If both the ground velocity and ground displacements are known, then equation (30) can be solved directly. However, only one, velocity or displacement is usually known and obtaining the other by numerical methods can lead to inaccurate results. Therefore, it is advantageous to express the solution for $v(t)$ in terms of only one ground motion variable.

Displacement is a convenient choice since it will provide a function that relates ground displacement to absolute displacement.

$$\begin{aligned}
 v(t) &= c(\dot{h} * \dot{v}_g) + k(h * v_g) & (31) \\
 &= c(\dot{h} * v_g) + k(h * v_g) \\
 &= (c\dot{h} + kh) * v_g
 \end{aligned}$$

Substituting in the appropriate expression for the impulse response function and its derivative, the expression for absolute displacement becomes:

$$\begin{aligned}
 v(t) &= \left(\frac{ce^{-\xi\omega t}}{m\omega_d} (\omega_d \cos(\omega_d t) - \xi\omega \sin(\omega_d t)) + \frac{ke^{-\xi\omega t}}{m\omega_d} \sin(\omega_d t) \right) * v_g & (32) \\
 &= e^{-\xi\omega t} (2\xi\omega_d \cos(\omega_d t) + \omega(1 - 2\xi^2) \sin(\omega_d t)) * v_g
 \end{aligned}$$

The $-m$ factor aside, the rest of convolution term to the left is simply \ddot{h} .

Alternatively, the $-m\ddot{h}$ term can also be obtained directly from equation (31) since the impulse response function must satisfy the equation for free vibration, $m\ddot{h} + c\dot{h} + kh = 0$.

It is particularly interesting to note that the solution to obtain the absolute displacement from the ground displacement is similar to that of obtaining the absolute acceleration from the ground acceleration. In fact, the two procedures are identical and provide a convenient function that relates the input type to the output type. From both cases, the term $-m\ddot{h}$ can be generalized as the ground impulse response function that relates input type to output type. This expression is particularly useful for computer applications since one function can provide a relationship between ground displacement to absolute displacement or ground acceleration to absolute acceleration.

$$\begin{aligned} v &= -m\ddot{h} * v \\ \ddot{v} &= -m\ddot{h} * \ddot{v}_g \end{aligned} \quad (33)$$

This technique does impose some limitations on the use of initial conditions. The use of initial conditions should be limited only when finding the response in terms of absolute displacement. The complete solution to displacement response given a ground displacement record and initial conditions can be obtained by the superposition of equations (5 & 32).

$$v(t) = e^{-\xi\omega t} \left(v_0 \cos(\omega_d t) + \frac{\dot{v}_0 + \xi\omega v_0}{\omega_d} \sin(\omega_d t) \right) + -m\ddot{h} * v$$

4.5 Periodic Loading– Initial Conditions

When loading the structure with an arbitrary periodic load, and the steady state periodic response is desired, then the true initial condition must be found such that the responses at the beginning of each period are equivalent. In other words, the initial conditions must match the final conditions at the end of a period. The procedure to obtain the true initial conditions is quite similar, but different calculations are required, in considering the two different loading conditions being considered.

4.5.1 Force On Mass

Within each period, the complete response is given by the sum of the free vibration due to the true initial conditions plus the response due to the forcing. The response due to

forcing can be expressed as the convolution of the forcing function for the period and the impulse response function. Taking advantage of the periodicity of the response, the following two equations must be satisfied for a period of length t_p .

$$v_0 = v(t_p) = e^{-\xi\omega_p} \left(v_0 \cos(\omega_d t_p) + \frac{\dot{v}_0 + \xi\omega v_0}{\omega_d} \sin(\omega_d t_p) \right) + F(t) * h(t) |_{t_p} \quad (34)$$

$$\dot{v}_0 = \dot{v}(t_p) = e^{-\xi\omega_p} \left(\dot{v}_0 \cos(\omega_d t_p) - \frac{\omega(\xi\dot{v}_0 + \omega v_0)}{\omega_d} \sin(\omega_d t_p) \right) + F(t) * \dot{h}(t) |_{t_p} \quad (35)$$

To simplify the equations, the following substitutions will be made:

$$C = e^{-\xi\omega_p} \cos(\omega_d t_p)$$

$$S = e^{-\xi\omega_p} \sin(\omega_d t_p)$$

Equations (34 & 35) can now be written more conveniently as:

$$v_0 = \left(C + \xi S \frac{\omega}{\omega_d} \right) v_0 + \frac{S}{\omega_d} \dot{v}_0 + F(t) * h(t) |_{t_p} \quad (36)$$

$$\dot{v}_0 = -\frac{\omega^2}{\omega_d} S v_0 + \left(C - \xi S \frac{\omega}{\omega_d} \right) \dot{v}_0 + F(t) * \dot{h}(t) |_{t_p} \quad (37)$$

These two equations can be solved simultaneously in matrix form for the unknown initial displacement and velocity.

$$\begin{Bmatrix} v_0 \\ \dot{v}_0 \end{Bmatrix} = \begin{bmatrix} 1 - C - \xi S \frac{\omega}{\omega_d} & -\frac{S}{\omega_d} \\ \frac{\omega^2 S}{\omega_d} & 1 - C + \xi S \frac{\omega}{\omega_d} \end{bmatrix}^{-1} \begin{Bmatrix} F(t) * h(t) |_{t_p} \\ F(t) * \dot{h}(t) |_{t_p} \end{Bmatrix} \quad (38)$$

4.5.2 Support Motion

For support motion the same procedure is carried out as above. The main difference in this case is in obtaining the response due to the support excitation. The resulting solution for the initial conditions is given in the following equation.

$$\begin{Bmatrix} v_0 \\ \dot{v}_0 \end{Bmatrix} = \begin{bmatrix} 1 - C - \xi S \frac{\omega}{\omega_d} & \frac{-S}{\omega_d} \\ \frac{\omega^2 S}{\omega_d} & 1 - C + \xi S \frac{\omega}{\omega_d} \end{bmatrix}^{-1} \begin{Bmatrix} F(t) * \ddot{h}(t)|_{t_p} \\ F(t) * \ddot{h}(t)|_{t_p} \end{Bmatrix} \quad (39)$$

4.6 Fourier Transform

In obtaining the Fourier Transform of the response or excitation, the data vector is first resized using linearly interpolation to obtain a vector of n points where n is a power of two. Having a vector of this size will enable the application of the Fast Fourier Transform to the data allowing for a much faster analysis. Once the vector is resized, a MATLAB function is executed to apply the Fast Fourier Transform. In MATLAB the FFT (`v_fft`) of a vector v is obtained as follows:

$$v_fft = fft(v)$$

The FFT function returns an array of complex numbers, twice the length of the input vector. To obtain the absolute magnitude of the part that is of interest, the `abs` function (absolute value) is used on only the first half of the vector. For a vector with n points, the MATLAB command would be as follows:

$$V_fft = abs(v_fft(1:n/2)) * dt$$

CHAPTER 5

PROGRAM DESCRIPTION

An interactive program was developed to aid students in the process of learning structural dynamics (Figure 4). This program aims to visually reinforce the fundamental concepts of dynamics by graphically demonstrating the response of structures under dynamic loads. The program is simple to use and allows for the comparison in response between different systems and excitation. The interface is developed to be user-friendly and intuitive so as to minimize time spent learning how to use the program. Although there are more powerful analysis tools, which also include animations, most of these require days to learn, and hours just to set up a model. This application is designed to get the student started quickly and manage the completion of a meaningful exercise in a limited amount of time.

The program accepts user input through buttons, pull-down menus, and editable text boxes. It also provides a graphical output window that displays the response of a single degree of freedom system subjected to the selected loading conditions. In addition to the response, other graphs are also available for display, such as the transfer function and the Fourier Transform of the loading. The available output, as well as input options, featured in the program will be discussed in greater detail in the sections that follow.

The program display is organized by grouping common objects together in separate panels. The top half of the main window is used to display the plot of the response. The two main control buttons are located in the center of the display to emphasize their importance. These buttons are used for adding or removing plots from the graphics area. The remaining lower half of the window contains the interface input options. These options are also grouped in distinctive panels and are titled for identification. The different sections and their individual components can be viewed in Figure 4. Although much emphasis has been placed on cluttered screens, at the same time it is important to keep all of the available options visible to the student at all times. This insures awareness of possible moves, giving the student a sense of control over the program. To simplify the interface, the panel section for loading criteria adjusts to provide options corresponding only to the current conditions.

5.1 Input Options

This program makes use of the MATLAB graphics and user-interface libraries to form an intuitive display that provides user-friendly interaction. Numerical input data, such as mass, is entered through editable text boxes. Pull-down menus provide for the selection of different loading conditions and types available. Buttons maintain the overall control of the program actions like plotting or exiting the program. Top-level menu items are also included to provide for the option of various plots that may be of interest to the student.

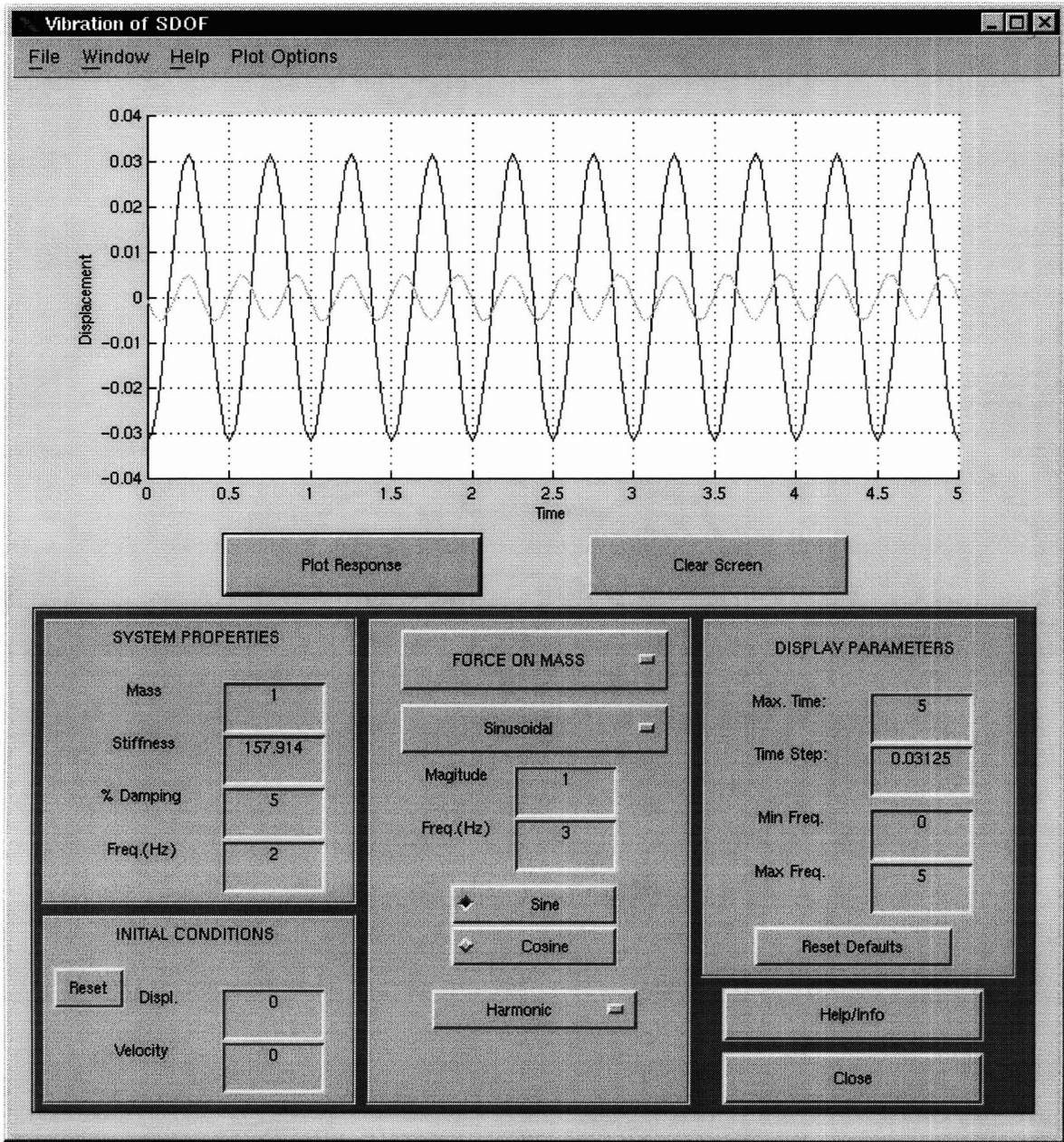


Figure 4. Screen snapshot of main window.

Unit options are not included within the program. Rather than numerical values, the user should concentrate on relative magnitudes between the different examples under comparison. If exact values are desired, then the graphs do contain a numerical scale and the units are consistent with those selected for the input. Examples of the input unit combinations that can be used are shown in Tables 1 and 2.

Table 1. Unit combinations for Free Vibration and Force on Mass.

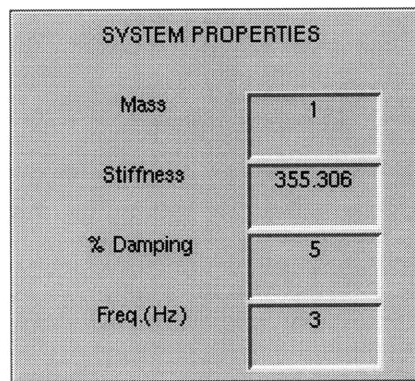
Mass	Stiffness	Init. Vel.	Init. Disp	Force	Output
Kg	N/m	m/s	m	N	m
lb*s ² /in.	lb./in	in/sec	in	lb.	in.
lb*s ² /ft (slug).	lb./ft	ft/sec	ft	Lb	ft

Table 2. Units' combinations for Support Motion.

Mass	Stiffness	Init. Vel.	Init. Disp	Excitation	Output
Kg	N/m	m/s	m	m	m
Kg	N/m	-	-	m/s ²	m/s ²
lb*s ² /in.	lb/in	in/sec	in	in	in.
lb*s ² /ft (slug).	lb/ft	ft/sec	ft	ft	ft

5.1.1 System Properties

The SDOF PLOT application allows the user to define a system consisting of a mass, spring, and dashpot as discussed in Chapter 3. The *SYSTEM PROPERTIES* input provides a means of specifying the system to be studied. The *Mass*, *% Damping*, *Stiffness*, and *Freq. (Hz.)* (frequency) are numerically entered into the program through the section of the user interface shown in Figure 5.2. The *Mass* and *Stiffness* selected will automatically update the frequency input option. However, if a certain frequency is desired, entering a value for *Freq. (Hz.)* will automatically update the *Stiffness* of the system while keeping the *Mass* value constant. The damping value entered is in terms of *% Damping* or the damping ratio X 100.



SYSTEM PROPERTIES	
Mass	1
Stiffness	355.306
% Damping	5
Freq.(Hz)	3

Figure 5. System properties input options.

5.1.2 Initial Conditions

In addition to the different forcing functions available, the oscillator can also be disturbed with initial conditions. The *INITIAL CONDITIONS* parameters allow for specifying the initial displacement (*Displ.*) and *Velocity*. A *Reset* button is provided to set

both of these values back to zero. Care should be taken to set these values back to zero when they are no longer desired, especially when they are automatically updated while using periodic loading.

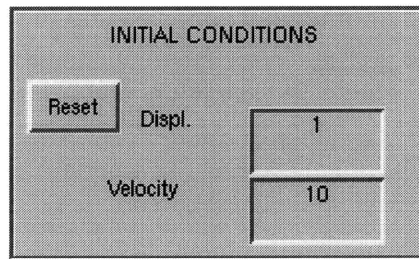


Figure 6. Initial Conditions input options.

5.1.3 Display Parameters

This section allows for the control and scaling of display properties for the output plots. The *Max. Time* input specifies the length of time for which to plot the response. The forcing function is also plotted for this same time length. The *Time Step* feature allows the user to select a time step discretization for the time scale used when calculating the response and displaying plots. A default time step of 16 steps per natural period is provided as the default and is automatically updated when there is a change in frequency. The *Time Step* input can be altered if one wished to explore the effects it has on calculations or plot resolution as long as its set after the *System Properties*.

The *Min. Freq.* and *Max Freq.* inputs provide a means of controlling the scale of plots in the frequency domain. Both of these values, specified in Hertz (Hz.) set the range for which to plot the FFT functions. The *Reset Defaults* button provides an easy way to

reset the minimum frequency to zero and the maximum to the nyquist frequency corresponding to the current time step. Pressing this button will also reset the *Time Step* input to one-sixteenth of the natural period of the system. It should be noted that the displayed time step is usually not used when calculating the FFT. The time step is changed to resize the data vector in order to obtain the number of sampling points equal to a power of two, thus resulting in a different nyquist frequency.

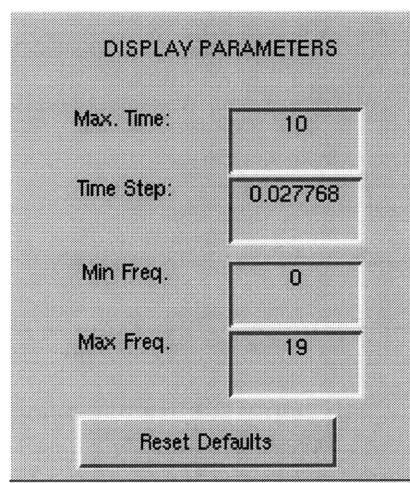


Figure 7. Display Parameters input options.

5.2 Forcing Function

SDOF Plot offers three different loading conditions, which can be applied on the single degree of freedom system. These include *Free Vibration*, *Force on Mass*, and *Support Motion*. *Free Vibration* allows for the input of initial conditions with the corresponding output of displacement in similar units. In the case that *Force on Mass* is selected, the input loading function consists of a force applied on the mass. The response

output is the displacement with units consistent with the input loading and system properties selected. The remaining selection, *Support Motion*, will set the input forcing function as ground motion. For a given ground displacement, the graphed output is also the displacement response of the mass. A ground acceleration input record will provide the acceleration response output. This input output relationship was discussed in section 4.4.2.2.

Two pull-down menus control the loading condition and type. The pull-down menu located on the top portion of the loading panel allows the user to choose between the three different loading conditions described above. The lower menu selects the type of loading, whether it is *Sinusoidal*, from *Point Data*, or a *File*. These three options are available for both, *Force on Mass* and *Support Motion*. Selecting between these options updates the loading section of the interface to include only the relevant inputs. Proper manipulation of these options will allow the user to create classical loading conditions. Simple forcing functions can be created on command via the interface using *Sinusoidal* and *Point Data*, or more complicated scenarios can be entered via a text-input file requiring a specific format.

5.2.1 Free Vibration

Selecting the *Free Vibration* (default) option simply allows the user to observe the free vibration behavior of the single degree of freedom system subjected to initial conditions. This selection displays a blank space in the center interface area corresponding to the input criteria for loading. The remaining sections remain visible for manipulation of

initial conditions and system properties. This analysis introduces the student to damping, frequency, effects of initial conditions, and other basic principles of vibrating systems. By modifying the system parameters and pressing the plot button, the program will quickly demonstrate to the student how each property affects the natural behavior of the vibrating system.

5.2.2 Sinusoidal

The *Sinusoidal* function allows the user to create sinusoidal-based forcing functions. For this purpose, the interface displays input option for: 1) *Magnitude*, 2) Frequency (*Freq.(Hz)*), 3) Initial Time (*To*), and 4) Final Time (*Tf*). The output signal will begin from zero at the selected time and oscillate at the specified frequency until reaching the final time. For example, the settings in Figure 8 will produce the forcing function displayed in Figure 9.

FORCE ON MASS	<input type="checkbox"/>
Sinusoidal	<input type="checkbox"/>
Magitude	1
Freq.(Hz)	2
To	1
Tf	5
Transient	<input type="checkbox"/>

Figure 8. Sinusoidal Loading Options

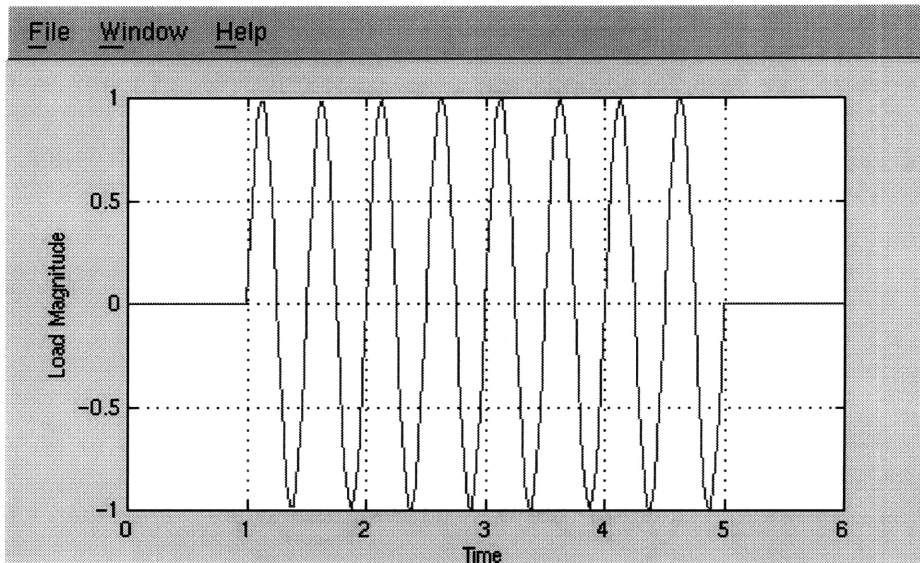


Figure 9. Force Time History produced from settings in Figure 8.

In addition to the *Transient/Periodic* loading options, which will be discussed later, selecting *Sinusoidal* also adds another option for *Harmonic* loading. This creates a simpler input option for the user, requiring only magnitude and frequency. In addition, for harmonic loading, only the steady state response is included in the output. Another feature is two radio buttons to choose between sine or cosine functions for different phase lags in the loading. The *Sinusoidal-Harmonic* option is selected in Figure 4.

5.2.3 Point Data

This option allows the user to enter three data points to form a forcing function. For each point, the time and magnitude of the excitation can be entered. The point (0,0) is used as default to allow up to four points, but specifying a new magnitude for this time can overwrite the zero value. Using the selected time step, linear interpolation between the

specified points creates the forcing function shown in Figure 11 for the input values displayed in Figure 10.

	Time	Mag.
1	1	1
2	3	-1
3	5	1

Figure 10. Point Data input options

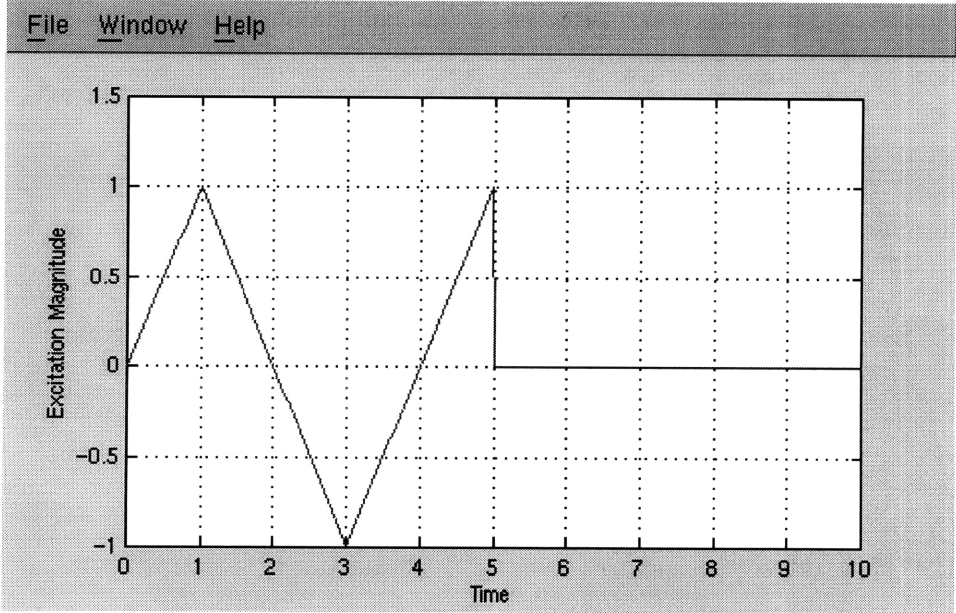


Figure 11. Excitation created from input in Figure 10.

5.2.4 File

The previously mentioned loading features allow users of the program to create various forcing functions in seconds, although the options are limited. The *File* option gives more flexibility to users who wish to pursue the response of a specific forcing function that cannot be created with the previously mentioned tools. The advantage of the *File* option is that any forcing function can be applied and analyzed using this program, even an earthquake. However, this option is more time consuming since a text file needs to be created. To file needs a specific format, with the number of points to be considered in the first line of text and the following lines should contain the time and magnitude for each point. Once the file is completed, the name of the file can be typed in the *FILE NAME* dialogue box only if it is located within the MATLAB path. Otherwise, the file can be found using the *Open File* button that opens a familiar dialogue window for opening files similar to those of many windows programs. The contents of a sample input file named *data.in* is shown below as an example. Entering the file into the program immediately plots the data in the load window. For more help on creating input files, the user can obtain specific instructions on the required format online using the *Format Help* button.

```
7
0.0    0.0
1.0    1.0
2.0   -1.0
3.0    1.0
4.0    2.0
5.0   -1.0
7.0    0.0
```

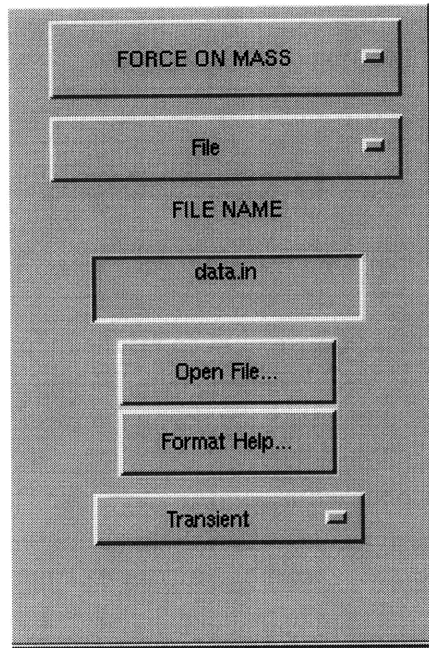


Figure 12. File selection input options.

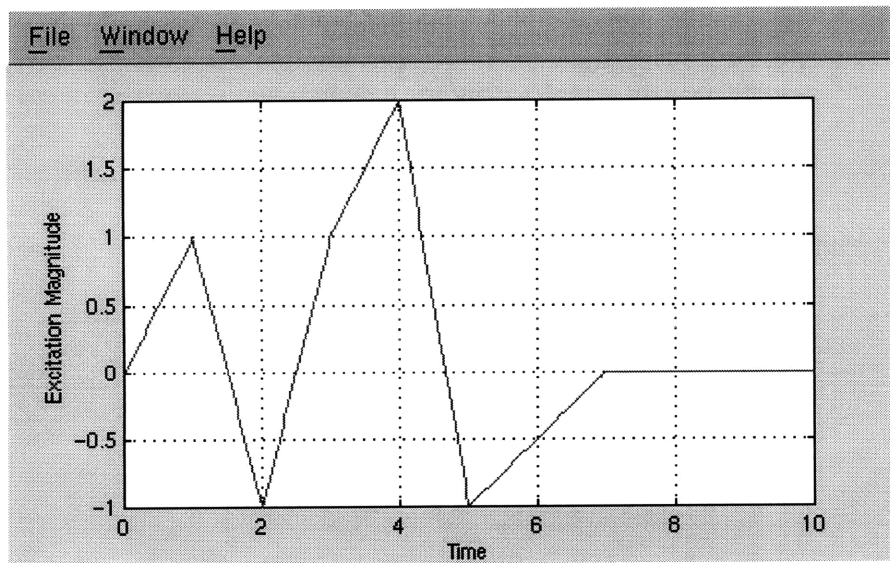


Figure 13. Forcing function created with *data.in* input file.

5.2.5 Transient/Periodic Loading

Forcing functions and the resulting response conditions can be treated as either transient or periodic. As mentioned earlier, *Sinusoidal* also has another option for *Harmonic*. These options are given by the pull down menu located towards the bottom of the loading selection panel. The transient response will produce a forcing function as specified by the input controls and solve for the response of the system subjected to the loading and initial conditions specified in the interface. The previous examples all had the transient option selected. *Periodic* loading, on the other hand, provides another input option to enter a time period for which to cycle the loading. The selected loading is then taken from zero to the length of the period and cycled throughout the display time. To provide a true periodic response, the proper initial conditions are calculated and displayed in the appropriate section of the interface. As an example of the features added by the *Periodic* selection, consider the input in Figure 8 with a period of two seconds. Notice how the forcing function in Figure 14 differs from the equivalent transient loading in Figure 9. The function is continuously repeated after the selected period of 2 seconds.

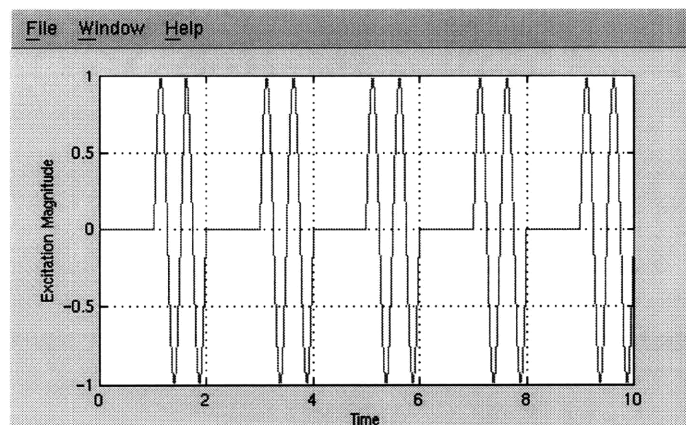


Figure 14. Periodic loading with period of 2 seconds.

5.3 *Output Options*

SDOF PLOT provides output in the form of graphics consisting of insightful plots. These plots are intended to give a visual representation of functions that may be helpful to students in understanding the behavior of vibrating systems.

5.3.1 Displacement Response

The main graphics window displays the displacement response of the sdof system subjected to the loading and initial conditions input. This graphics area is located in the upper half of the main window as shown in Figure 4. Plots in this axis are continuously added to the screen in different colors until the CLEAR button is pressed.

5.3.2 Plot Options Menu

SDOF PLOT can also display other plots, which are accessible through the *Plot_Options* menu. These graphs, once selected by the user, appear in a separate window and display plots related to the latest data displayed in the main window. They are continuously updated along with the displacement plot each time the *Plot* button is pressed. If the screen becomes too cluttered, these extra graphics windows can be closed and re-selected as needed.

5.3.2.1 Forcing Function

One of the extra plots available from the menu is the graph of the excitation. This plot ensures that the loading is in understanding with the student by providing a visual representation of the load in the time domain. It also provides the user with an input-output correlation on the screen. For example, the response displayed in Figure 4 resulted from the excitation show in Figure 15. Other examples of forcing function plots were shown previously in Figure 9, Figure 11, and Figure 13.

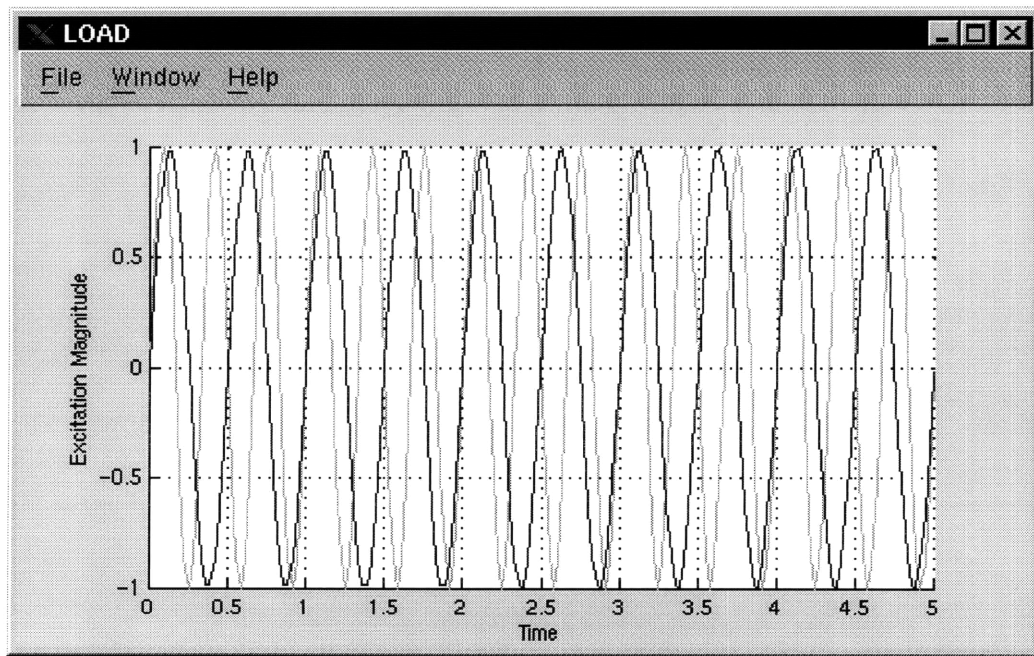


Figure 15. Excitation Plot corresponding to response in Figure 4.

5.3.2.2 Transfer Function and Phase Angle

In order to introduce students to transfer function and phase angle, a plot can be created alongside the response so that students can compare both plots and how they vary with frequency ratio and damping. Similar to the response plot, previous transfer functions remain on the plot until the *Clear Screen* button is pressed. In addition, these plots are coordinated with the same color as the displacement response in order to identify which function corresponds with which. A marker is provided for *Sinusoidal* excitation denoting the current position of the frequency ratio to show the student exactly where the current system lies within the transfer function and the phase angle. This visual aid will allow the user to predict the effect of variable excitation frequency and the phenomena of resonance.

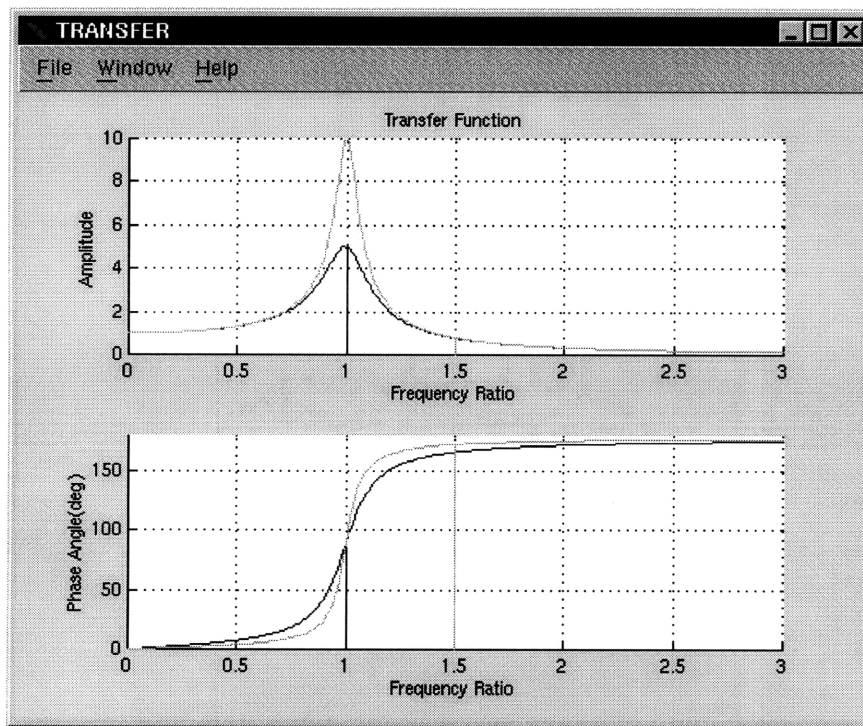


Figure 16. Transfer Function and Phase Angle corresponding to Figure 4.

5.3.2.3 Load Fourier Transform

Once students begin to feel comfortable with plots in the time domain, then it would be helpful to begin to visualize the frequency content of the excitation. This data becomes particularly important when dealing with periodic loading. Visualizing the frequency content of the loading with the Load FFT plot along with the transfer function will expose students to the important role the loading frequency plays in the resulting behavior of a vibrating system. Obtaining the FFT for a simple sinusoidal function as seen in Figure 17, could also assist the student in clarifying the meaning of the FFT.

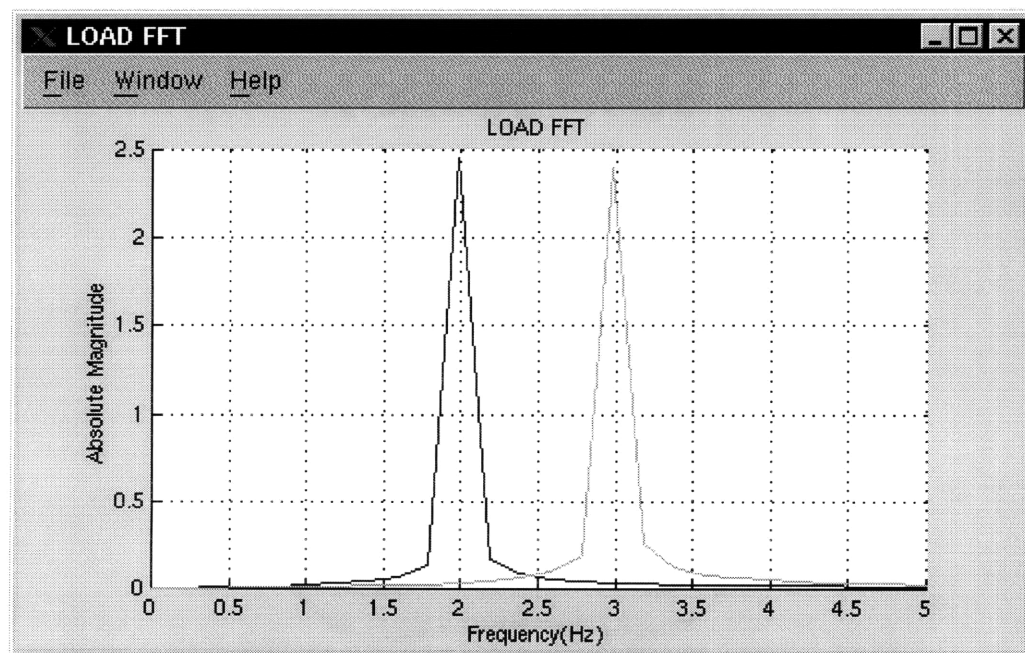


Figure 17. FFT of both forcing functions shown in Figure 16.

5.3.2.4 Response Fourier Transform

The Fast Fourier Transform option is also available for the response data. Observing the frequency content of the response alongside the FFT of the excitation will also help reinforce the dominance of the excitation frequency and/or the natural frequency present within the response.

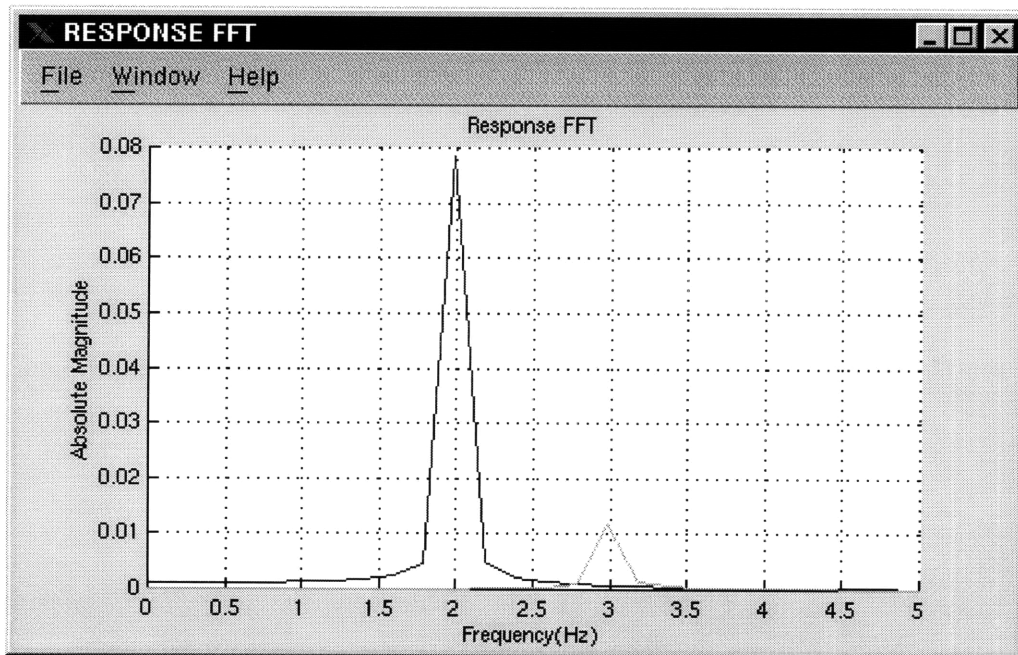


Figure 18. FFT of response plots displayed in Figure 4.

5.4 *Program Structure*

This program is written as on long recursive function to allow the complete program to fit within one file. Keeping the program to one file will help resolve some of the portability issues involved in distributing the software. All that will be required to run the program will be the MATLAB software and the single script file.

MATLAB identifies the available functions by file name, meaning that each function must be placed within a separate file in order to be recognized as a valid function within the MATLAB search path. Though more than one function can be placed in one file as a sub-function, these functions can only be called by other functions within the same file. Several functions within one file were considered as an option for organizing this program, but this option had some limitations in using the user-interface tools. The main problem encountered was that the user-interface tools “Callback” feature could not call sub-functions eventhough they co-existed in the same file. Only one sub-function was created within the file. This sub-function is responsible for resizing the vectors used for FFT analysis by linear interpolation.

Instead of functions, the program is divided by the use of “*if- elseif- end*” statements. The program basically consists of one of “*if- elseif- end*” statement organized by the “option” argument of the function. Recursively calling the function with the proper “option” string argument will direct the function call to the proper starting place. A framework of the program structure is given in the flowchart in Figures 19 and 20.

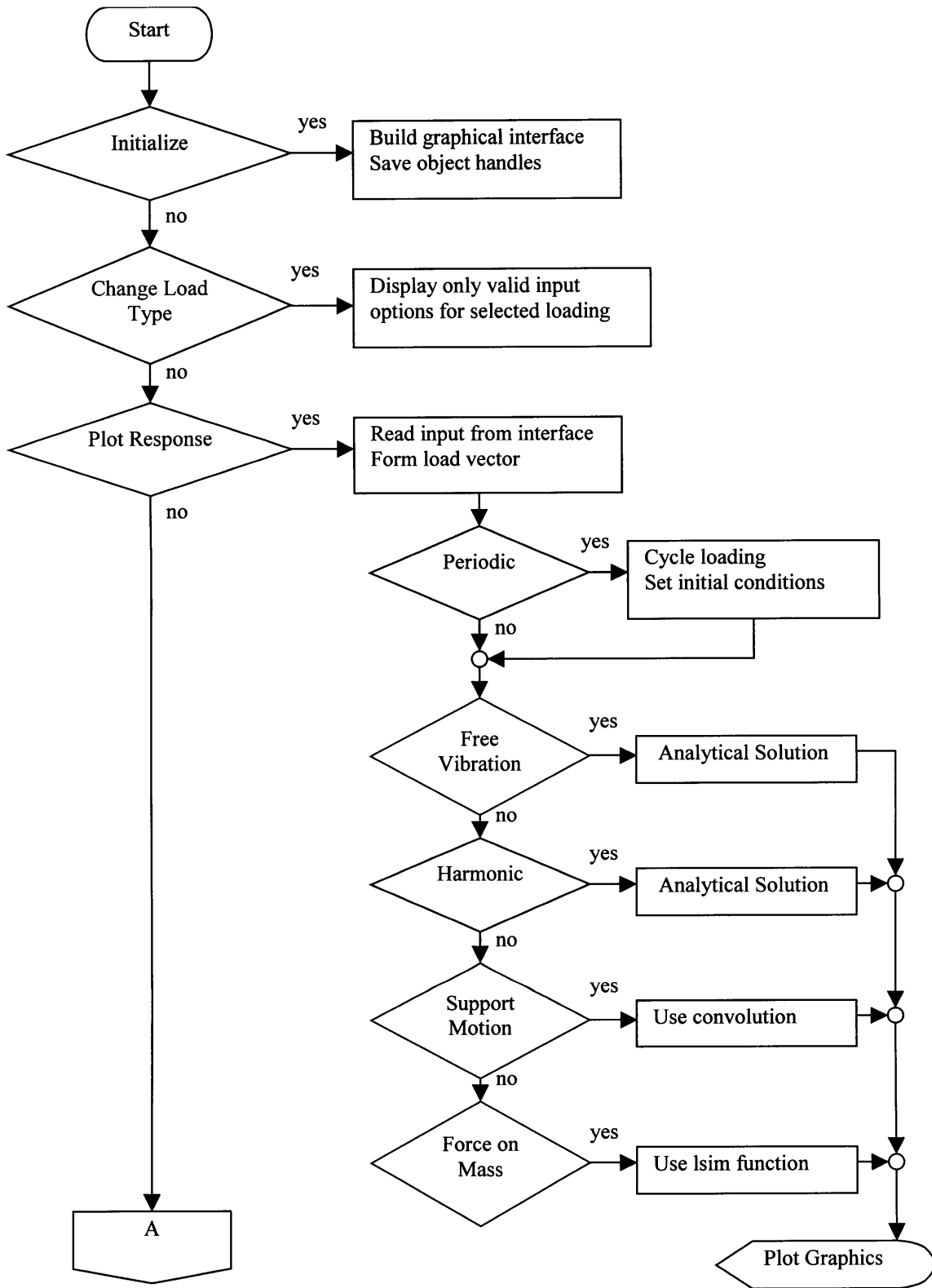


Figure 19. Flow chart of SDOF PLOT program (Part 1).

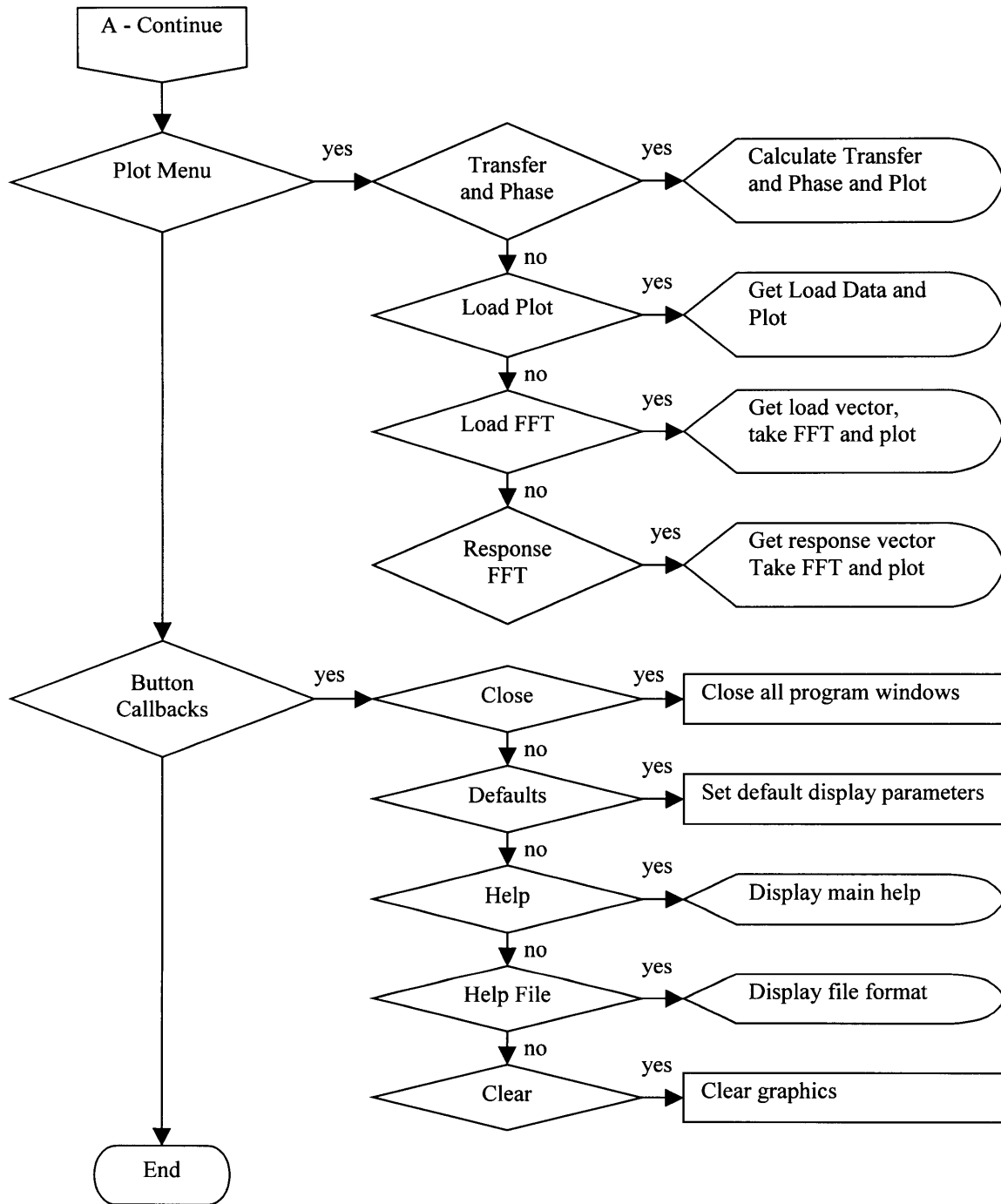


Figure 20. Flow Chart of SDOF PLOT program (Part 2).

CHAPTER 6

APPLYING THE SOFTWARE

This program only provides the results or output to a specified input, working in some ways like a “black box”. The program does not display the methods or the calculations being used to solve the problem under study. This is a disadvantage for students seeking to learn theory and mathematics from the software program. The program does not include lecture material explaining derivations of equations. However, students will still maintain other resources such as lectures and textbooks providing access these essential skills required for success in a structural dynamics course.

One clear advantage in using the software is that students are allowed to concentrate on the physics of the system and predict how physical changes, such as in mass, affects the response under different excitations. Particularly when getting involved with more complicated systems, the mathematics could become the main obstacle in understanding the dynamics equations. Many engineering students lack the mathematical skills to successfully understand the derivations of equations and how to solve them (Tomovic, 1996). These difficulties could hinder students and distract them from concentrating on and understanding the dynamics.

Instead, students may begin by observing the response of several systems, then be able to interpret for themselves how a change in stiffness, damping, or mass will affect the overall response of the vibrating mass. Once the physical behavior is understood, then it will be much easier for students to go back and review textbooks on the subject. With the gained insight of the effect of parameters, students will be able to physically make sense of the mathematical formulations to explain the results obtained while using the program. This reverse approach might produce some motivation for students since they will now be aware of physical applications of the equations. In traditional teaching methods, first the derivation of equations are presented, then applications to physical problems are presented.

6.1 Program Goals

Before proposing examples on how to use this program, it is important to discuss the goals of the program in order to understand the program design. This MATLAB application provides an excellent tool for plotting the response of a single degree of freedom system to variable loading conditions. Observing the change in response, as the different inputs are modified, allows the user to understand both, how a change in the system properties or a change in loading affects the overall behavior of the vibrating system. The intention of this program is for the user to quickly obtain a graphical solution to their problem given the system properties and excitation. It could also function as a valuable tool used to compare the responses under different situations, as the system itself, or the loading is modified. To fulfill these task, the user interface offers ease of use in modifying the system properties, loading conditions, and displaying the results

The main objective of the program design is for it to be simple to use. Once the desired system properties and loading are selected, pressing the *Plot_Response* button instantly creates the selected plots. Modifying the inputs and again pressing *Plot_Response* updates the graphs by providing the new response in the main window, while still maintaining the previous ones. This makes it very simple for the user to compare the responses. New graphs continue to be added to the screen until the *Clear Screen* button is pressed.

In addition to the displacement response of the system, the program can also display other graphs that provide insight into dynamic behavior. These additional plots are available in the *Plot_Options* menu. The forcing function is plotted to provide the time variance of the loading and verify that the program correctly understood the user input. Also, the plot of the load alongside the response will provide a visual input-output relationship on the screen. When selected by the user, the transfer function and phase angle are plotted as a function of the frequency ratio ($\omega_{loading} / \omega_{natural}$), assuming harmonic loading. If the loading selected is sinusoidal, a marker appears at the location of the current frequency ratio. Subsequent plots of the transfer function and phase angle, along with the frequency marker, are plotted with colors corresponding to those of the response plot. Especially in the case of periodic loading, the Fast Fourier Transform of the load could be of interest so that students may visualize and begin to give thought to the frequency content of the signal. The program provides options for plotting the FFT of both the loading and the response.

6.2 *Examples of Use*

The students could use the program on their own, allowing them to have full control and experiment with different options at their own rate. “Playing” with the program could serve as most beneficial to students. This exercise will allow students to construct variable conditions and observe how a sdof would react under different circumstances. The probability of this situation occurring is questionable since it is uncertain how much time students will invest using educational software. It is more likely that students will only go through educational tasks only for the sake of completing homework and will likely try to complete them in the quickest possible manner (Flori, 1994).

To assure students use the program, and especially those who most need to reap the benefits, homework assignments should require students to use the software. Such assignments could ask fundamental questions regarding the response of structures and the parameters that control the response. For example, displaying the free vibration response of a sdof system and asking the student how a change in damping affects the response graph would insure that students understand the role of damping. The student could then be asked to draw the new response alongside and explain why such effect takes place. Answers could include the location of the variables within the governing equations and its mathematical implications on the outcome. In this example, the negative damping ratio factor within the exponential term of the free vibration solution mathematically signifies the rate of decay.

Students with sufficient knowledge could answer some proposed questions without any additional tools. Students without a full understanding could quickly find their results by applying the computer program. This would allow advanced students to get by without using a program that is below their level of comprehension. On the other hand, students in need will be required to go through the complete exercises with the software and it will assure its use were it is needed most.

A program such as this one also provides a tool to supplement lectures with interactive graphics. With the use of a computer projector, the screen could be displayed to the entire class. The professor could talk about the effects of certain variables, go over the governing equations involved, and then display the program to show the results. This will be particularly useful since the professor will not have to spend valuable lecture time drawing different graphs on the board, but rather be able to display graphs within seconds. In addition, the computer provides more presentable graphics making it an ideal choice for this purpose.

6.3 *Problem Set*

As a demonstration, several exercises are proposed with the program being used to obtain the result. Depending on the level of understanding of the user, some of these problems can be solved without the need of software. Students deficient in the topic of dynamics will be able to benefit from the software by using it to complete the assigned tasks. Rather than a straightforward approach, which may be unknown to inexperienced students, a visual procedure simplified by the program could be used. The program allows

students to vary parameters until they come to an understanding of the behavior, enabling them to answer the proposed question.

6.3.1 Free Vibration

A mass of 10 kg is supported by a linear spring and a viscous damper. Using a stiffness of 1000 N/m. plot the displacement response due to an initial displacement of 1 m.

- Compare the response for both 1% and 10% damping. How does an increase in damping affect the response?
- Using 1% damping and keeping the mass constant, change the natural frequency to 10 Hz. How does this response differ from the previous cases?

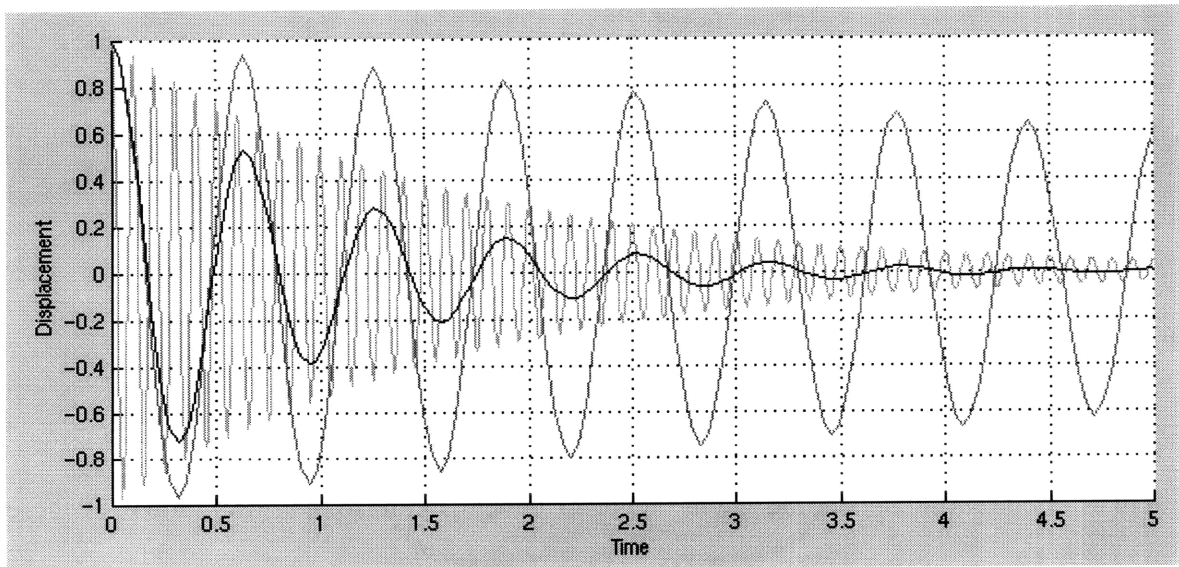


Figure 21. Display from Free Vibration problem.

From Figure 21, it can be seen that an increase in damping reduces the amplitude of the response at a much faster rate. An increase in frequency, in addition to increasing the oscillation rate, also increases the rate of decay of the response amplitude. A mathematical

explanation to this result can be seen from the mathematical formulation, mainly from the negative factors of frequency and damping in the exponential decay term of the response:

$$v(t) = e^{-\xi\omega t} \left(v_o \cos(\omega_d t) + \frac{\dot{v}_o + \xi\omega v_o}{\omega_d} \sin(\omega_d t) \right)$$

Exponential
Decay

Oscillation
Term

6.3.2 Harmonic Excitation

For the next exercise, the following options are selected for the loading conditions: *Force on Mass-Sinusoidal-Harmonic*.

In the previous example we saw that damping assisted in decaying the response at a faster rate and minimized the vibration of the system. How does damping affect the response under dynamic forcing conditions? Plot the response for each of the following cases and determine the effects of damping under harmonic loading:

Use Mass = 1, Natural Freq. = 3 Hz., Load Magnitude $P_o = 1$.

- a) Load Freq. = 2 Hz., % Damping = 1 and % Damping = 5
- b) Load Freq. = 3 Hz., % Damping = 1 and % Damping = 5
- c) Load Freq. = 4 Hz., % Damping = 1 and % Damping = 5

Plots obtained from the three cases are as follows:

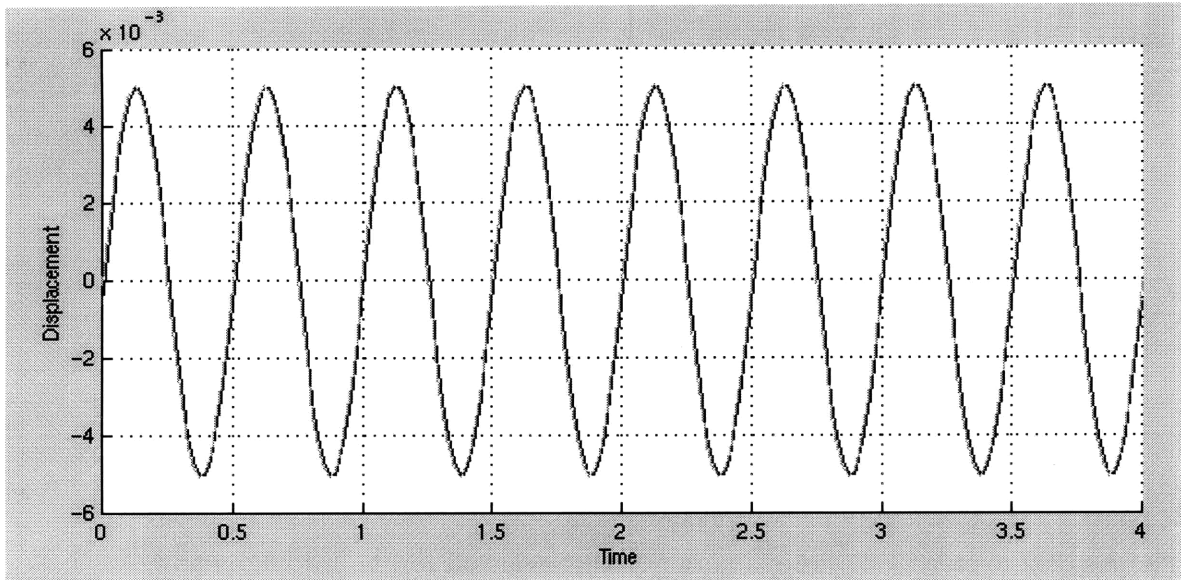


Figure 22. Response for case a.

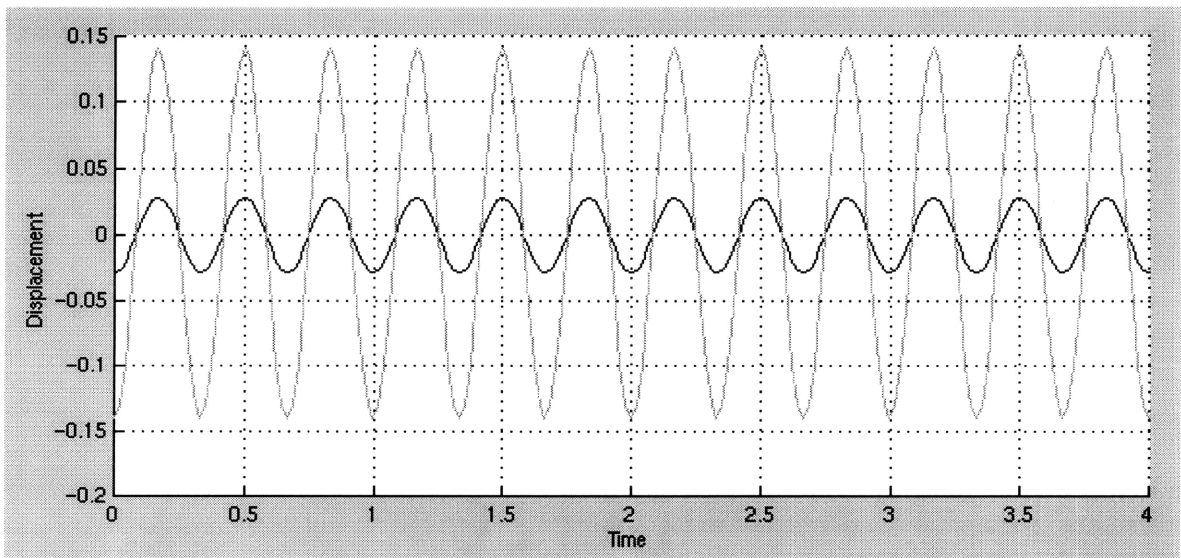


Figure 23. Response for case b.

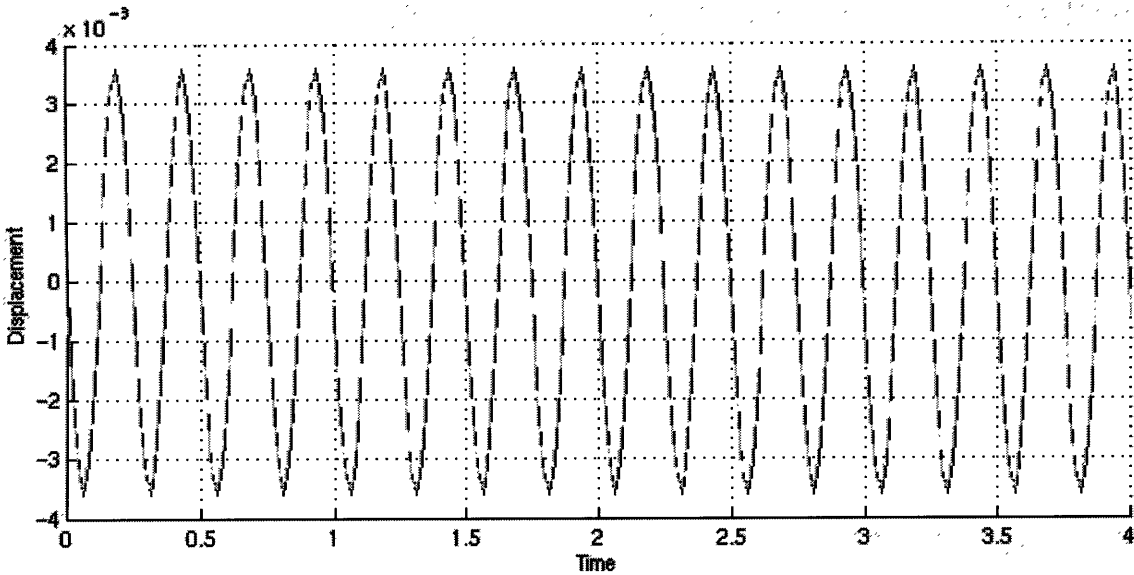


Figure 24. Response for case c.

In both cases *a* and *c*, the response was hardly affected by the increase in damping. In fact, both cases appear identical, making it difficult to distinguish between both graphs.

Case *b* on the other hand had a significant reduction in response by the increase in damping. Increasing the damping ratio from 1% to 5%, the maximum amplitude was reduced to almost 20% of the original. From this observation, we can conclude that under harmonic loading, damping has a significant affect in the response only when the loading frequency is near the natural frequency of the system. When the natural frequency is equivalent to the forcing frequency, as in this case, the structure is said to be in resonance with the loading. To better understand this behavior, let's take a look at the Transfer Function, which provides a plot of the amplitude of the response as a function of the frequency ratio (loading frequency over the natural frequency).

From this transfer function, it is clear that the plotted curves for different damping ratios differ only in the vicinity where the frequency ratio is one. This area can be thought of as the damping controlled region since this parameter has such a great effect on the response.

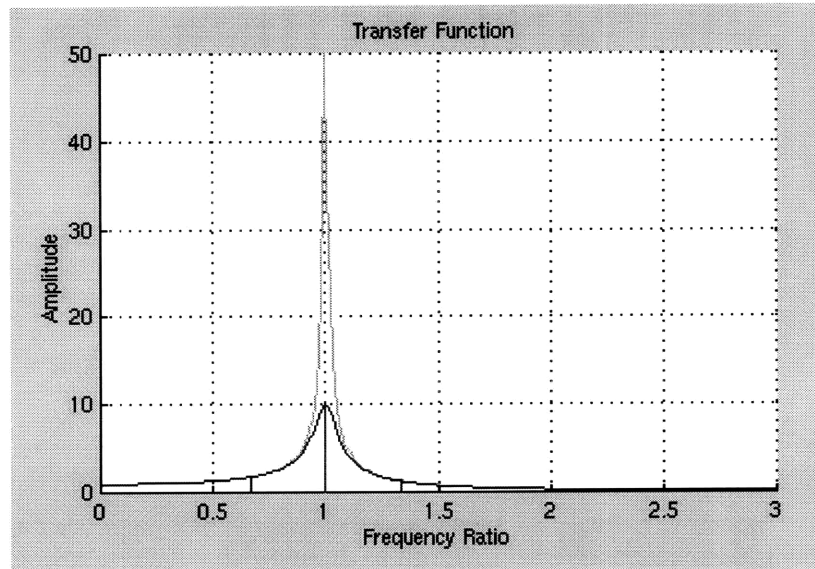


Figure 25. Transfer function for 1% and 5% damping.

6.3.3 Earthquake Engineering

Using the *Support Motion – File* option, earthquake analysis for a sdof can be accomplished with this program.

Begin by properly formatting the file and opening it with the program. In this case, the El Centro acceleration record is used for the analysis. The file consists of 3001 points with a time step of 0.02 seconds resulting in 60 seconds of data. Only the first 20 seconds will be

considered by setting the maximum display time to 20 seconds and the time step to 0.02 seconds.

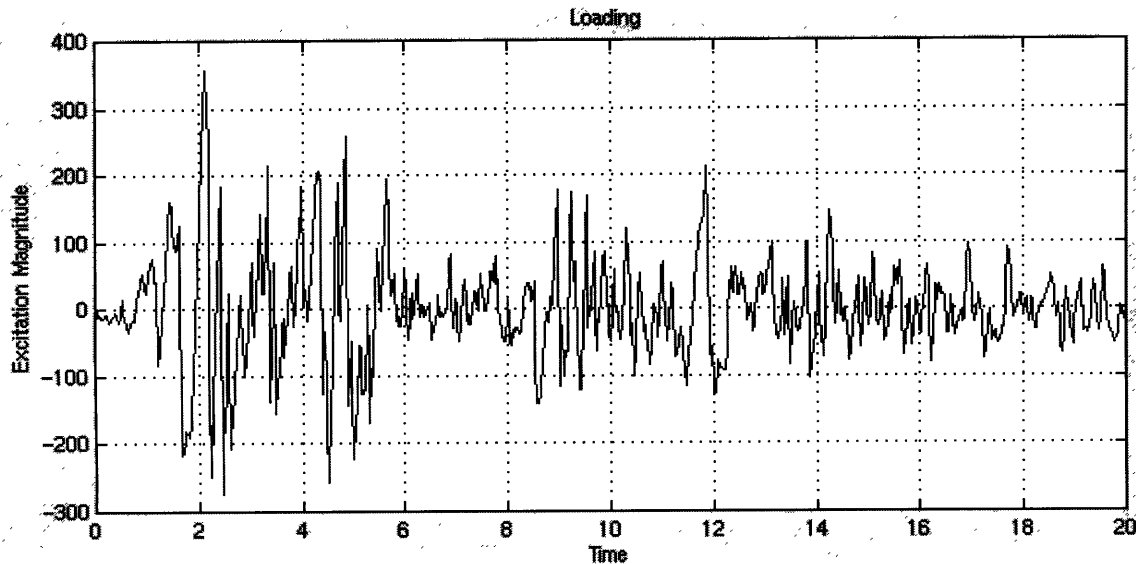


Figure 26. First 20 seconds of the acceleration record for El Centro.

Unlike harmonic loading, it is much more difficult to predict how a dynamic system will react to random excitation such as an earthquake. We can, however, apply some of what we have learned from the previous case, mainly that a sdof is excited the most under resonant conditions. To learn a bit about the frequency content of random signals, students can look at the Fourier decomposition of the excitation signal or the plot in the frequency domain. This plot can be easily obtained from the *Plot_Options* menu. This plot shows the contribution of each individual frequency within the random earthquake signal. A higher value obtained for a particular frequency substantiates the strong presence of that frequency. For this earthquake, the dominant frequencies in the El Centro earthquake are between 1 and 2 Hz. Therefore, applying the lessons learned from harmonic loading, we

can expect sdof systems with a natural frequency in this range to be excited the most by this acceleration record. Try different frequencies and see if this is true.

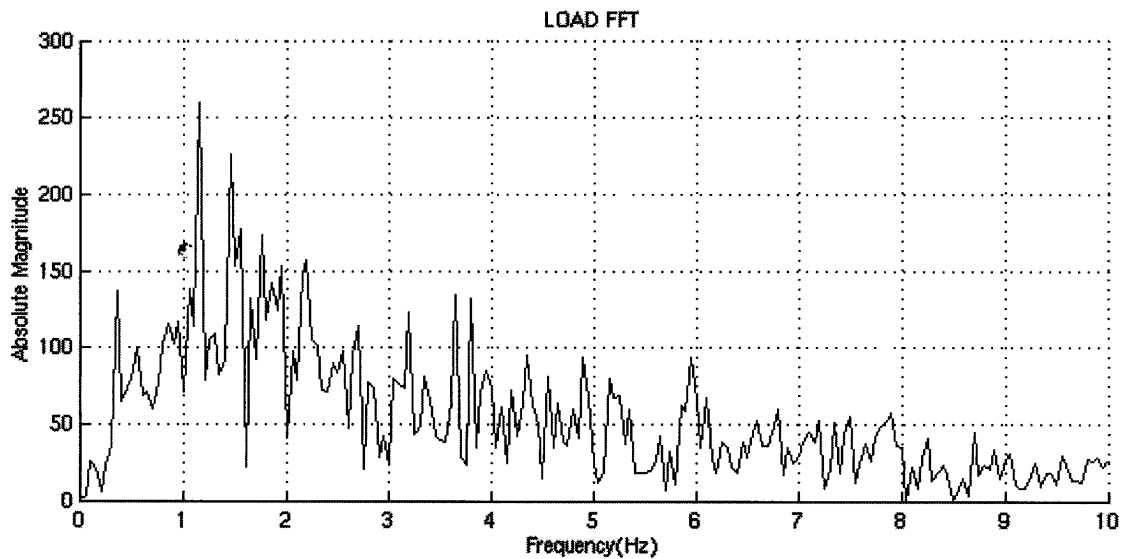


Figure 27. FFT of El Centro acceleration record show in Figure 26.

Usually structures subjected to resonant loading are the most in danger of damage during an earthquake. Under harmonic loading, the parameter that best controlled the response under resonance was the damping ratio. Does this rule also apply to random loading as well?

Input a sdof system with a mass of 1 kg, and a natural frequency of 2 Hz. Plot the response for 1% and 5% damping. Also, obtain the *Response FFT* for both cases.

An increase in damping did, in fact, cause a reduction in response. For this case the peak response was reduced to about 80% with increased damping. From the frequency content of the response signals, we can conclude that the dominant frequency present in the response (approximately the natural frequency) was significantly reduced.

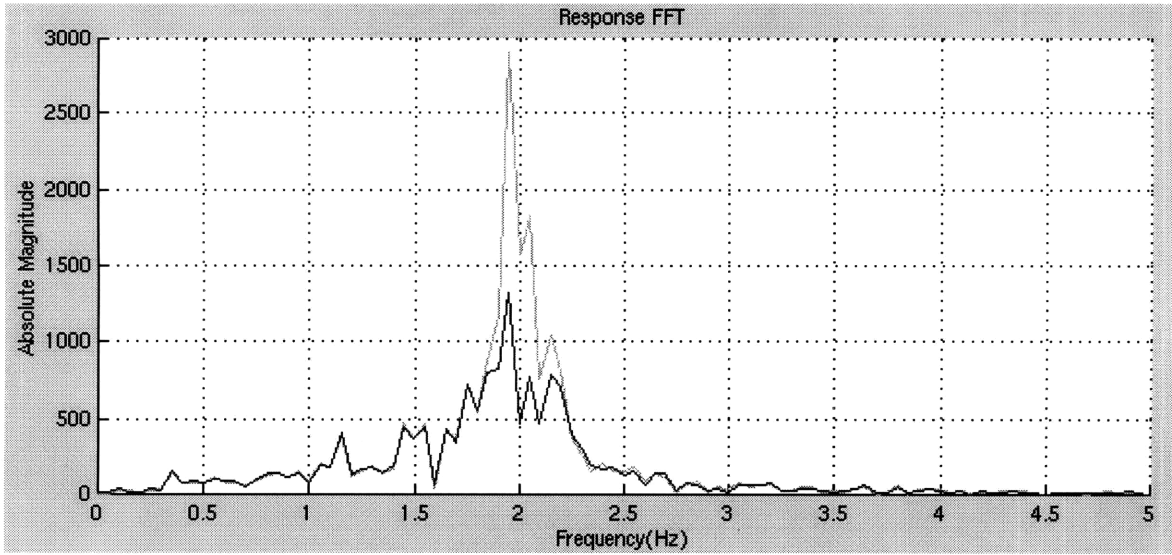


Figure 28. Response to El Centro with 1% and 5% damping.

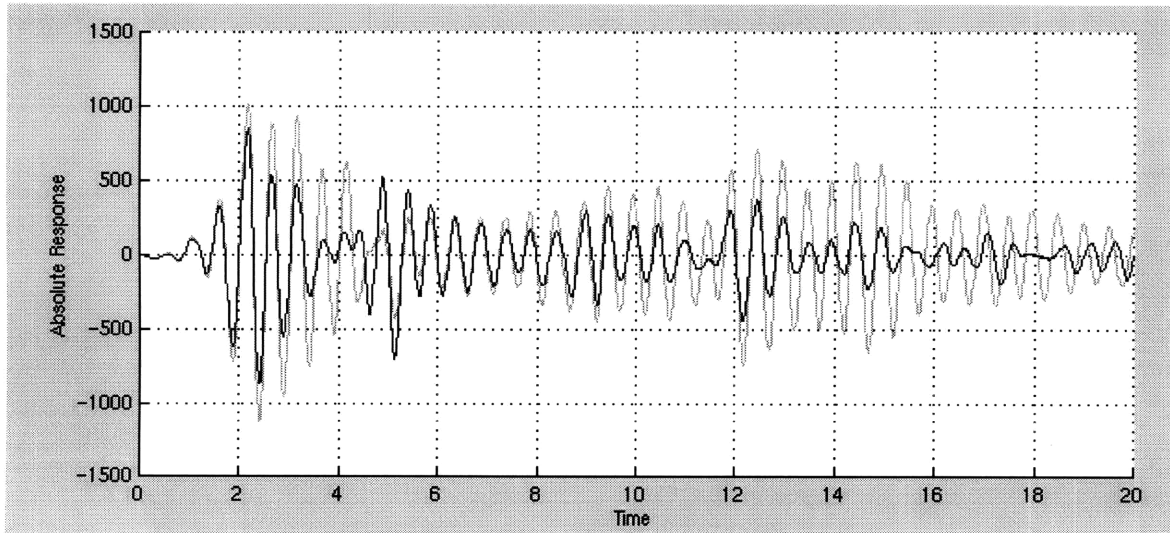


Figure 29. FFT for Response plots shown in Figure 28.

6.4 *Remarks*

From the reviewed problems, simple lessons can be learned from observing the response of structures without much knowledge of the mathematics. Providing simple exercises that incorporate the use of technology allows students to experiment with software and gain insight into the role each parameter plays in controlling the response. Reviewing the mathematics with this gained knowledge simplifies the complicated mathematical expressions for students. Students will be able to identify variables in the equations with some intuition as to how they affect the overall outcome. Usually, it is not until all the parameters are understood that equations begin to make sense.

CHAPTER 7

CONCLUSION

Information technology offers the potential to improve educational environments by overcoming some of the limitations of current educational media and traditional classroom pedagogical techniques. Books are usually limited to one or two plots as examples with the text. Lectures and interaction with the professor is also limited by time constraints. SDOF PLOT goes beyond current educational media by providing a tool that will engage students' curiosity with its interactive graphics and simplicity. An interactive program such as this one will allow students to obtain plots with the parameters of their choice. In addition, they will be able to observe how each plot varies as the input parameters are modified. Computers accomplish what textbooks and chalkboards simply cannot, especially for a subject where the phenomenon is time-dependant such as structural dynamics.

One of the main features of this program is its simplicity. The input options are minimized without much sacrifice with respect to content. Automation in updating some parameters is incorporated within the program to further limit actions required by the user. Simplicity was central to the design in order to increase its usability among students.

It is very easy for designers to get wrapped up in content and overcrowd the program with material. Complexity in educational programs will not simplify the learning experience but rather supplement the task of learning. Educational models are not intended to compete with professional programs such as a Finite Element Analysis package. The key is to allow a student to quickly apply an analysis and obtain the results.

BIBLIOGRAPHY

- [1] Bork, A. (1987). "The Potential for Interactive Technology". Byte (p. 201). Cited in Hativa, N., and Lesgold, A. (1996). "Situational Effects in Classroom Technology Implementations: Unfulfilled Expectations and Unexpected Outcomes". In Kerr, S. T. (Eds.). *Technology and the Future of Schooling*. The University of Chicago Press, IL.
- [2] Brackett, G. (1998). "Lecture Notes - Spring 1998". *Course T-525 Designing Educational Experiences Using Networks and Webs*. Harvard Graduate School of Education. Cambridge, MA.
- [3] Chabay, R.W. and Sherwood, B. A. (1992). "A Practical Guide for the Creation of Educational Software". In Larkin, J. H. and Chabay, R. W. (Eds.) *Computer-Assisted Instruction and Intelligent Tutoring Systems: Shared Issues and Complementary Approaches*. Lawrence Erlbaum Associates, Hillsdale, NJ.
- [4] Cornwell P. J. (1996). "Teaching Dynamics Using Modern Tools". *Computers in Education Journal*, vol. VI n. 4, pp 18-24.
- [5] Flori, R. E. (1994). "Computer-Aided Instruction in Dynamics: Does it Improve Learning?". *Proceedings of 1994 Frontiers in Education Conference*. San Jose, CA, Nov. 2-6, 1994. IEEE, New York, NY.
- [6] Kerr, S. T. (1996). "Visions of Sugarplums: The Future of Technology, Education and the Schools". In Kerr, S. T. (Eds.). *Technology and the Future of Schooling*. The University of Chicago Press, IL.
- [7] Interactive Revolution (1998). *Company Web Page*. <http://www.krev.com/>.
- [8] MathWorks, Inc. (1995). *The Student Edition of MATLAB: Version 4 User's Guide*. Prentice Hall, Englewood Cliffs, NJ
- [9] MathWorks, Inc. (1993). *MATLAB, High-Performance Numeric Computation and Visualization Software: Building a Graphical User Interface*. MathWorks, Natick, MA.

- [10] Means, B., and Olson, K. (1995). *Technology's Role in Education Reform: Findings from a National Study of Innovating Schools*. U.S. Department of Education, Washington D.C.
- [11] Metaxas P. T. (1996). "On User Interfaces for Educational Multimedia Applications". *Proceedings 1996 IEEE International Conference on Multi Media Engineering Education*. Melbourne, Vic, Australia, July 3-5, 1996. IEEE, New York, NY.
- [12] Paz, M. (1991). *Structural Dynamics: Theory and Computation, 3rd Edition*. Van Nostrand Reinhold, New York, NY.
- [13] Perkins, D. N. (1992). "Technology Meets Constructivism: Do They Make a Marriage?". In Duffy, T. M. and Jonassen, D. H. (Eds.), *Constructivism and the Technology of Instruction: A Conservation*, Lawrence Erlbaum Associates, Hillsdale, NJ.
- [14] Salomon, G., and Perkins, D. (1996). "Learning in Wonderland: What Do Computers Really Offer Education". In Kerr, S. T. (Eds.). *Technology and the Future of Schooling*. The University of Chicago Press, IL.
- [15] Sanchez, A. (1998). *Web Based Course Management and Delivery System (COMMAND)*, Thesis (S. M.). Massachusetts Institute of Technology, Department of Civil and Environmental Engineering. Cambridge, MA.
- [16] Shepherdson, E. (1998). *Teaching Structural Behavior through an Interactive and Complete Learning Environments*, Thesis (S. M.). Massachusetts Institute of Technology, Department of Civil and Environmental Engineering. Cambridge, MA.
- [17] Tomovic, M. M. (1996). "Simulink Helps Students Focus on the Physics". *Computers in Education Journal*, vol. VI n. 4, pp 14-17.

APPENDIX

PROGRAM CODE


```

function sdf(option, arg1, arg2)

%*****
%*
%*      STRUCTURAL DYNAMICS TUTORIAL - FREE/FORCED VIBRATION OF SDOF *
%*      ----- *
%*      by Gilberto Mosqueda *
%*      Last Edited: April 4, 1998 *
%* *
%*      This program written for use in MATLAB will graphically *
%*      demonstrate the free and forced vibrations of a single *
%*      degree of freedom systems. The user can set the properties *
%*      of the system (mass, damping, stiffness) or the loading and *
%*      instantly obtain the response. Previous plots remain on *
%*      the screen until the CLEAR button is pressed. Other plots *
%*      such as the transfer function, the impulse response function, *
%*      and the fft of periodic loading are also displayed. *
%* *
%*      To Start the program, type ">sdf" at the MATLAB prompt. *
%* *
%*****

global PROP_HAND LOAD_HAND PCOLOR; % Decalre some global variables

if nargin<1 %Set to initialize for no option
    option = 'initialize';
end;

clr_plts = ['r'; 'b'; 'm'; 'c'; 'g'; 'y']; %Plotting colors (in order)

%*****
%*      INITIALIZE: CREATE WINDOW WITH GUI TOOLS *
%*****

if strcmp(option, 'initialize')
    % Create window and format with plot and controls

    trans_fig=findobj('type', 'figure', 'Name', 'TRANSFER');
    if ~isempty(trans_fig)
        close(trans_fig)
    end
    dyna_fig = figure( ...
        'Name', 'TRANSFER', ...
        'Unit', 'pixels' ,...
        'UserData', 0 ,...
        'Position', [725 25 300 400] ,...
        'NumberTitle', 'off', ...
        'Colormap', []);
    load_fig=findobj('type', 'figure', 'Name', 'LOAD');
    if ~isempty(load_fig)
        close(load_fig)
    end
    load_fig = figure( ...
        'Name', 'LOAD', ...
        'Unit', 'pixels' ,...
        'Position', [725 450 300 275] ,...
        'NumberTitle', 'off', ...
        'Colormap', []);
    data_fig = figure( ...
        'Name', 'Vibration of SDOF', ...
        'Unit', 'pixels' ,...

```

```

        'Position', [25 25 690 700] ,...
        'UserData', [load_fig dyna_fig],...
        'NumberTitle', 'off', ...
        'Colormap', []);
set(gca, 'Position', [0.13 0.62 0.775 0.343902])
back_frm = uicontrol(data_fig, ...
    'Style', 'frame', ...
    'Units', 'normalized',...
    'BackgroundColor', [.4 .4 .4],...
    'Position', [ .02 .02 .96 .48 ]);
%subplot(2, 1, 1);
grid;, hold on;
xlabel('Time');
ylabel('Displacement');
% Initialize variables
M = 5;, K = 1000;, D = 5;
Y0 = 0;, V0 = 0;, P0=0;, OM=0;
TIME = 10;, PCOLOR = 1;
F=sqrt(K/M)/(2*pi);
delT= 1/(F*16);
nyquist = ceil(1/(2*delT));

% *****CREATE MENUS FOR PLOTTING *****

option_menu = uimenu(data_fig,...
    'Position', 1,...
    'Label', 'Plot Options');
load_fun = uimenu(option_menu,...
    'Label', 'Forcing Function',...
    'Callback', 'sdof(''plotting'', ''load'')');
tran_fun = uimenu(option_menu,...
    'Label', 'Transfer and Phase ',...
    'Callback', 'sdof(''plotting'', ''transfer'')');
resp_fft = uimenu(option_menu,...
    'Label', 'Response FFT',...
    'Callback', 'sdof(''plotting'', ''resp_fft'',get(gca, ''UserData''))');
load_fft = uimenu(option_menu,...
    'Label', 'Load FFT',...
    'Callback', 'sdof(''plotting'', ''load_fft'',get(gca, ''UserData''))');
impu_res = uimenu(option_menu,...
    'Label', 'Impulse Response',...
    'Callback', 'sdof(''plotting'', ''impulse'')');

% *****CREATE GUI'S FOR PROPERTIES*****
% System Properties
back_frm = uicontrol(data_fig, ...
    'Style', 'frame', ...
    'Units', 'normalized',...
    'Position', [ .03 .22 .3 .27 ]);
prop_txt = uicontrol(data_fig, ...
    'Style', 'text', ...
    'Units', 'normalized',...
    'String', 'SYSTEM PROPERTIES', ...
    'Position', [ .05 .42 .25 .06]);
% Editable text button for mass
mass_txt = uicontrol(data_fig, ...
    'Style', 'text', ...
    'Units', 'normalized',...
    'String', ' Mass', ...
    'Position', [ .05 .38 .15 .05]);
mass_inp = uicontrol(data_fig, ...
    'Style', 'edit', ...

```

```

'Units', 'normalized',...
'String', M, ...
'Position', [ .2 .38 .1 .05],...
'CallBack', [...
    'if str2num(get(gco,'String'))<=0,',...
    'errordlg('ERROR- Mass must be positive'),',...
    'end',',...
    'h=get(gco,'UserData');, ', ...
    'm=str2num(get(gco, 'String'));',',...
    'k=str2num(get(h(1),'String'));',',...
    'f=sqrt(k/m)/(2*pi);, ',...
    'set(h(2),'String',f);, ',...
    'dt = 1/(f*16);, ',...
    'set(h(3),'String',dt);, ' ] );
% Editable text button for Stiffness
stif_txt = uicontrol(data_fig, ...
    'Style', 'text', ...
    'Units', 'normalized',...
    'String', ' Stiffness', ...
    'Position', [ .05 .33 .15 .05]);
stif_inp = uicontrol(data_fig, ...
    'Style', 'edit', ...
    'Units', 'normalized',...
    'String', K, ...
    'Position', [ .2 .33 .1 .05 ],...
    'CallBack', [...
        'if str2num(get(gco,'String')) <= 0, ', ...
        'errordlg('ERROR- Stiffness must be positive'),',',...
        'else,',',...
        'h=get(gco,'UserData');, ', ...
        'm=str2num(get(h(1),'String'));',',...
        'k=str2num(get(gco, 'String'));',',...
        'f=sqrt(k/m)/(2*pi);, ',...
        'set(h(2),'String',f);, ',...
        'dt = 1/(f*16);, ',...
        'set(h(3),'String',dt);, ',...
        'end' ] );
% Editable text button for Damping Ratio
damp_txt = uicontrol(data_fig, ...
    'Style', 'text', ...
    'Units', 'normalized',...
    'String', ' % Damping ', ...
    'Position', [.05 .28 .15 .05 ]);
damp_inp = uicontrol(data_fig, ...
    'Style', 'edit', ...
    'Units', 'normalized',...
    'UserData', dyna_fig, ...
    'String', D, ...
    'Position', [ .2 .28 .1 .05 ],...
    'CallBack', [...
        'if str2num(get(gco,'String')) < 0, ', ...
        'errordlg('ERROR- Damping Ratio must be positive'),',',...
        'end' ] );
freq_txt = uicontrol(data_fig, ...
    'Style', 'text', ...
    'Units', 'normalized',...
    'String', ' Freq.(Hz) ', ...
    'Position', [.05 .23 .15 .05 ]);
freq_inp = uicontrol(data_fig, ...
    'Style', 'edit', ...
    'Units', 'normalized',...
    'String', F, ...

```

```

'Position', [ .2 .23 .1 .05 ],...
'CallBack', [...
    'h=get(gco, 'UserData');, ', ...
    'm=str2num(get(h(1), 'String')));, ', ...
    'f=str2num(get(gco, 'String')));, ', ...
    'k=m*(2*pi*f)^2;,', ...
    'set(h(2), 'String', k);, ', ...
    'dt = 1/(f*16);, ', ...
    'set(h(3), 'String', dt);'] );

%***** CREATE GUI'S FOR INITIAL CONDITIONS *****
% Initial Conditions
back_frm = uicontrol(data_fig, ...
    'Style', 'frame', ...
    'Units', 'normalized',...
    'Position', [ .03 .03 .3 .18 ]);
init_txt = uicontrol(data_fig, ...
    'Style', 'text', ...
    'Units', 'normalized',...
    'String', 'INITIAL CONDITIONS', ...
    'Position', [.05 .15 .25 .05]);
% Editable text button for Initial Displacement
idis_txt = uicontrol(data_fig, ...
    'Style', 'text', ...
    'Units', 'normalized',...
    'String', ' Displ. ', ...
    'Position', [.09 .09 .1 .05 ]);
idis_inp = uicontrol(data_fig, ...
    'Style', 'edit', ...
    'Units', 'normalized',...
    'String', Y0, ...
    'Position', [ .2 .09 .1 .05 ]);
% Editable text button for Initial Velocity
ivel_txt = uicontrol(data_fig, ...
    'Style', 'text', ...
    'Units', 'normalized',...
    'String', ' Velocity ', ...
    'Position', [.05 .04 .15 .05 ]);
ivel_inp = uicontrol(data_fig, ...
    'Style', 'edit', ...
    'Units', 'normalized',...
    'String', V0, ...
    'Position', [ .2 .04 .1 .05]);
setic_bt = uicontrol(data_fig, ...
    'Style', 'push', ...
    'Units', 'normalized',...
    'UserData', [idis_inp ivel_inp], ...
    'String', 'Reset', ...
    'Position', [ .04 .12 .07 .04 ],...
    'CallBack', [...
        'h=get(gco, 'UserData');, ', ...
        'set(h, 'String', 0.0);'] );

%*****CREATE GUI'S FOR LOADING*****
% Set default options for loading
back_frm = uicontrol(data_fig, ...
    'Style', 'frame', ...
    'Units', 'normalized',...
    'Position', [ .34 .03 .31 .46 ]);
lloc_pop = uicontrol(data_fig, ...
    'Style', 'popupmenu', ...
    'Units', 'normalized',...

```

```

    'UserData', dyna_fig, ...
    'String', 'FREE VIBRATION|FORCE ON MASS|SUPPORT MOTION', ...
    'Position', [ .37 .42 .26 .06], ...
    'CallBack', [...
'sdof('load', popupstr(gco), 1);'] );
load_pop = uicontrol(data_fig, ...
    'Style', 'popupmenu', ...
    'Units', 'normalized',...
    'String', 'Sinusoidal|Point Data|File', ...
    'Visible', 'off',...
    'Position', [ .37 .36 .26 .05], ...
    'CallBack', 'sdof('load', popupstr(gco), 0);']); %*****
% -----CONSTANT-----
% Editable text button for Loading Magnitude
lmg_txt = uicontrol(data_fig, ...
    'Style', 'text', ...
    'Units', 'normalized',...
    'String', ' Magitude ', ...
    'Visible', 'off',...
    'Position', [.35 .30 .15 .05 ]);
lmg_inp = uicontrol(data_fig, ...
    'Style', 'edit', ...
    'Units', 'normalized',...
    'Visible', 'off',...
    'Position', [ .48 .30 .1 .05 ]);
% -----HARMONIC-----
% Editable text button for Loading Frequency
lfre_txt = uicontrol(data_fig, ...
    'Style', 'text', ...
    'Units', 'normalized',...
    'String', ' Freq.(Hz) ', ...
    'Visible', 'off',...
    'Position', [.35 .25 .15 .05 ]);
lfre_inp = uicontrol(data_fig, ...
    'Style', 'edit', ...
    'Units', 'normalized',...
    'Visible', 'off',...
    'Position', [ .48 .25 .1 .05 ]);
sine_rad = uicontrol(data_fig, ...
    'Style', 'radio', ...
    'Units', 'normalized',...
    'Visible', 'off',...
    'Value', 1,...
    'String', 'Sine',...
    'Position', [ .42 .20 .16 .04 ], ...
    'CallBack', [...
    'set(gco, 'Value', 1);',...
    'set(get(gco, 'UserData'), 'Value', 0);']);
csin_rad = uicontrol(data_fig, ...
    'Style', 'radio', ...
    'Units', 'normalized',...
    'Visible', 'off',...
    'String', 'Cosine',...
    'Position', [ .42 .16 .16 .04 ],...
    'CallBack', [...
    'set(gco, 'Value', 1);',...
    'set(get(gco, 'UserData'), 'Value', 0);']);
set(sine_rad, 'UserData', csin_rad);
set(csin_rad, 'UserData', sine_rad);
% -----START/END TIME-----
% Editable text button for Initial Loading Time
tini_txt = uicontrol(data_fig, ...

```

```

        'Style', 'text', ...
        'Units', 'normalized',...
        'String', ' To ', ...
        'Visible', 'off',...
        'Position', [.35 .2 .15 .05 ]);
tini_inp = uicontrol(data_fig, ...
    'Style', 'edit', ...
    'Units', 'normalized',...
    'Visible', 'off',...
    'Position', [ .48 .2 .1 .05 ]);
% Editable text button for Final Loading Time
tfin_txt = uicontrol(data_fig, ...
    'Style', 'text', ...
    'Units', 'normalized',...
    'String', ' Tf ', ...
    'Visible', 'off',...
    'Position', [.35 .15 .15 .05 ]);
tfin_inp = uicontrol(data_fig, ...
    'Style', 'edit', ...
    'Units', 'normalized',...
    'Visible', 'off',...
    'Position', [ .48 .15 .1 .05 ]);
% -----STEP LOADING-----
% Editable text button for initial and final loading
imag_txt = uicontrol(data_fig, ...
    'Style', 'text', ...
    'Units', 'normalized',...
    'String', ' Po ', ...
    'Visible', 'off',...
    'Position', [.35 .30 .15 .05 ]);
imag_inp = uicontrol(data_fig, ...
    'Style', 'edit', ...
    'Units', 'normalized',...
    'Visible', 'off',...
    'Position', [ .48 .30 .1 .05 ]);
fmag_txt = uicontrol(data_fig, ...
    'Style', 'text', ...
    'Units', 'normalized',...
    'String', ' Pf ', ...
    'Visible', 'off',...
    'Position', [.35 .25 .15 .05 ]);
fmag_inp = uicontrol(data_fig, ...
    'Style', 'edit', ...
    'Units', 'normalized',...
    'Visible', 'off',...
    'Position', [ .48 .25 .1 .05 ]);

% -----POINT DATA -----
pd_hdl = zeros(7,2);

pd_hdl(1, 1) = uicontrol(data_fig, ...
    'Style', 'text', ...
    'Units', 'normalized',...
    'String', 'Time', ...
    'Visible', 'off',...
    'Position', [.42 .30 .1 .05 ]);
pd_hdl(1, 2) = uicontrol(data_fig, ...
    'Style', 'text', ...
    'Units', 'normalized',...
    'String', 'Mag.', ...
    'Visible', 'off',...
    'Position', [.52 .30 .1 .05 ]);

```

```

for i=1:3
    pd_hdl(4+i, 1) = uicontrol(data_fig, ...
        'Style', 'text', ...
        'Units', 'normalized',...
        'String', i, ...
        'Visible', 'off',...
        'Position', [.37 .30-i*0.05 .05 .05 ]);
end

pd_hdl(2, 1) = uicontrol(data_fig, ...
    'Style', 'edit', ...
    'Units', 'normalized',...
    'String', 0, ...
    'Visible', 'off',...
    'Position', [.42 .25 .1 .05 ],...
    'CallBack', [...
        'h=get(gco, ''UserData'');',...
        't1 = str2num(get(gco, ''String''));',...
        't2 = str2num(get(h(1), ''String''));',...
        't3 = str2num(get(h(2), ''String''));',...
        'if t1 < 0,',...
        'errordlg(''ERROR- Time must be greater than zero.''),',...
        'set(gco, ''String'',0),',...
        'end,',...
        'if t1 > t2,',...
        'set(h(1), ''String'',t1),',...
        'end,',...
        'if t1 > t3,',...
        'set(h(2), ''String'',t1),',...
        'end,'] );

pd_hdl(3, 1) = uicontrol(data_fig, ...
    'Style', 'edit', ...
    'Units', 'normalized',...
    'String', 0, ...
    'Visible', 'off',...
    'Position', [.42 .20 .1 .05 ],...
    'CallBack', [...
        'h=get(gco, ''UserData'');',...
        't2 = str2num(get(gco, ''String''));',...
        't1 = str2num(get(h(1), ''String''));',...
        't3 = str2num(get(h(2), ''String''));',...
        'if t2 < 0,',...
        'errordlg(''ERROR- Time must be greater than zero.''),',...
        'set(gco, ''String'',0),',...
        'end,',...
        'if t2 < t1,',...
        'set(h(1), ''String'',t2),',...
        'end,',...
        'if t2 > t3,',...
        'set(h(2), ''String'',t2),',...
        'end,'] );

pd_hdl(4, 1) = uicontrol(data_fig, ...
    'Style', 'edit', ...
    'Units', 'normalized',...
    'String', 0, ...
    'Visible', 'off',...
    'Position', [.42 .15 .1 .05 ],...
    'CallBack', [...
        'h=get(gco, ''UserData'');',...
        't3 = str2num(get(gco, ''String''));',...
        't1 = str2num(get(h(1), ''String''));',...

```

```

        't2 = str2num(get(h(2),'String'));',...
        'if t3 < 0,',...
        'errordlg(''ERROR- Time must be greater than zero.''),',...
        'set(gco, 'String',0),',...
        'end,',...
        'if t3 < t1,',...
        'set(h(1), 'String',t3),',...
        'end,',...
        'if t3 < t2,',...
        'set(h(2), 'String',t3),',...
        'end,'] );

for i=1:3
    pd_hdl(i+1, 2) = uicontrol(data_fig, ...
        'Style', 'edit', ...
        'Units', 'normalized',...
        'String', 0, ...
        'Visible', 'off',...
        'Position', [.52 .30-i*0.05 .1 .05 ]);
end

% -----FILE DATA LOADING-----
% Get loading from file
file_txt = uicontrol(data_fig, ...
    'Style', 'text', ...
    'Units', 'normalized',...
    'String', 'FILE NAME', ...
    'Visible', 'off',...
    'Position', [ .4 .3 .2 .05]);
file_inp = uicontrol(data_fig, ...
    'Style', 'edit', ...
    'Units', 'normalized',...
    'Visible', 'off',...
    'Position', [ .4 .26 .2 .05 ],...
    'Callback','sdo(''file_plot'',get(gco,'String'), gco);' );
fhelp_bt = uicontrol(data_fig, ...
    'Style', 'push', ...
    'Units', 'normalized',...
    'String', 'Format Help...', ...
    'Visible', 'off',...
    'Position', [ .42 .15 .16 .05 ],...
    'Callback','sdo(''button'', 'helpfile');' );

file_btn = uicontrol(data_fig, ...
    'Style', 'push', ...
    'Units', 'normalized',...
    'String', 'Open File...', ...
    'UserData', file_inp, ...
    'Visible', 'off',...
    'Position', [ .42 .2 .16 .05 ],...
    'Callback', [...
        'filename=uigetfile(''*.*'',''DATA FILE'', 100, 100);', ...
        'file_inp=get(gco, 'UserData');',...
        'set(file_inp, 'String', filename);',...
        'sdo(''file_plot'',filename, file_inp);' ]);
% ----- PERIODIC LOADING -----
perd_txt = uicontrol(data_fig, ...
    'Style', 'text', ...
    'Units', 'normalized',...
    'Visible', 'off',...
    'String', 'Period', ...
    'Position', [.35 .04 .15 .05 ]);

```



```

perd_inp = uicontrol(data_fig, ...
    'Style', 'edit', ...
    'Units', 'normalized',...
    'String', '1', ...
    'Visible', 'off',...
    'Position', [ .48 .04 .1 .05 ],...
    'Callback', [...
        'if str2num(get(gcf,'String'))< 0,',...
        'errordlg(''ERROR- Period must be greater than zero.''),',...
        'set(gcf, 'String',0),',...
        'end'] );
ltyp_pop = uicontrol(data_fig, ...
    'Style', 'popupmenu', ...
    'Units', 'normalized',...
    'UserData', [load_pop],...
    'String', 'Transient|Periodic|Harmonic', ...
    'Visible', 'off',...
    'Position', [ .4 .10 .2 .04 ],...
    'Callback', [...
        'h=get(gcf, 'UserData');', ...
        'sdof(''load'', popupstr(h));']);

%*****CREATE GUI'S FOR GRAPH*****
% Graph Properties
back_frm = uicontrol(data_fig, ...
    'Style', 'frame', ...
    'Units', 'normalized',...
    'Position', [ .66 .15 .3 .34 ]);
prop_txt = uicontrol(data_fig, ...
    'Style', 'text', ...
    'Units', 'normalized',...
    'String', 'DISPLAY PARAMETERS', ...
    'Position', [ .69 .41 .25 .06]);
% Time lenght of plot input
labetxt = uicontrol(data_fig, ...
    'Style', 'text', ...
    'Units', 'normalized',...
    'String', 'Max. Time: ', ...
    'Position', [ .67 .37 .15 .05]);
time_inp = uicontrol(data_fig, ...
    'Style', 'edit', ...
    'Units', 'normalized',...
    'String', 'TIME', ...
    'Position', [ .82 .37 .1 .05],...
    'Callback', [...
        'if str2num(get(gcf,'String'))<=0,',...
        'errordlg(''ERROR- Display Time must be positive.''),',...
        'end']);

% Time step for calculations
delt_txt = uicontrol(data_fig, ...
    'Style', 'text', ...
    'Units', 'normalized',...
    'String', ' Time Step: ', ...
    'Position', [ .67 .32 .15 .05]);
delt_inp = uicontrol(data_fig, ...
    'Style', 'edit', ...
    'Units', 'normalized',...
    'String', 'delT', ...
    'Position', [ .82 .32 .1 .05],...
    'Callback', [...
        'if str2num(get(gcf,'String'))<=0,',...

```

```

        'errordlg(''ERROR- Time Step must be positive''),',...
        'end']);

% Display Frequency for fourier data
minf_txt = uicontrol(data_fig, ...
    'Style', 'text', ...
    'Units', 'normalized',...
    'String', ' Min Freq. ', ...
    'Position', [ .67 .26 .15 .05]);
minf_inp = uicontrol(data_fig, ...
    'Style', 'edit', ...
    'Units', 'normalized',...
    'String', 0 , ...
    'Position', [ .82 .26 .1 .05],...
    'CallBack', [...
        'value = str2num(get(gco, 'String')); main =(gcf);',...
        'maxf = str2num(get( get(gco, 'UserData'), 'String'));',...
        'if value<=0,',...
        'errordlg(''ERROR- Display Frequency must be positive''),',...
        'elseif value >= maxf,',...
        'set(gcbo, 'String', 0);',...
        'errordlg(''ERROR- Min. Frequency must be less than Max.''),',...
        'else,',...
        'f_fig=findobj(''type'', 'figure'', 'Name'', 'RESPONSE FFT'');',...
        'if ~isempty(f_fig)',...
        'figure(f_fig); set(gca, 'XLim', [ value maxf] );',...
        'end;',...
        'f_fig=findobj(''type'', 'figure'', 'Name'', 'LOAD FFT'');',...
        'if ~isempty(f_fig);',...
        'figure(f_fig); set(gca, 'XLim', [ value maxf] );',...
        'end;',...
        'figure(main); end;'] );
maxf_txt = uicontrol(data_fig, ...
    'Style', 'text', ...
    'Units', 'normalized',...
    'String', ' Max Freq. ', ...
    'Position', [ .67 .21 .15 .05]);
maxf_inp = uicontrol(data_fig, ...
    'Style', 'edit', ...
    'Units', 'normalized',...
    'String', nyquist, ...
    'UserData', minf_inp,...
    'Position', [ .82 .21 .1 .05],...
    'CallBack', [...
        'value = str2num(get(gco, 'String')); main =(gcf);',...
        'minf = str2num(get( get(gco, 'UserData'), 'String'));',...
        'if value<=minf,',...
        'set(gco, 'String', minf+1);',...
        'errordlg(''ERROR- Max. Frequency must be greater than min''),',...
        'else,',...
        'f_fig=findobj(''type'', 'figure'', 'Name'', 'RESPONSE FFT'');',...
        'if ~isempty(f_fig)',...
        'figure(f_fig); set(gca, 'XLim', [minf value] );',...
        'end;',...
        'f_fig=findobj(''type'', 'figure'', 'Name'', 'LOAD FFT'');',...
        'if ~isempty(f_fig);',...
        'figure(f_fig); set(gca, 'XLim', [minf value] );',...
        'end;',...
        'figure(main); end;'] );
deft_btn = uicontrol(data_fig, ...
    'Style', 'push', ...
    'Units', 'normalized',...

```

```

        'UserData', [minf_inp maxf_inp delt_inp time_inp freq_inp],...
        'String', 'Reset Defaults', ...
        'Position', [ .71 .16 .19 .04 ],...
        'CallBack', 'sdof(''button'', 'default'', get(gco, 'UserData') )');
% *****CREATE CONTROL BUTTONS*****
% Graph Button [ .68 .21 .25 .06 ],
graph_bt = uicontrol(data_fig, ...
    'Style', 'push', ...
    'Units', 'normalized',...
    'String', 'Plot Response', ...
    'Position', [ .2 .51 .25 .06 ],...
    'CallBack', 'sdof(''plot'')');
% Clear Button [ .68 .14 .25 .06 ]
clear_bt = uicontrol(data_fig, ...
    'Style', 'push', ...
    'Units', 'normalized',...
    'String', 'Clear Screen', ...
    'Position', [ .55 .51 .25 .06 ],...
    'CallBack', 'sdof(''button'', 'clear'')');
% Close Button
help_btn = uicontrol(data_fig, ...
    'Style', 'push', ...
    'Units', 'normalized',...
    'String', 'Help/Info', ...
    'Position', [ .68 .09 .25 .05 ],...
    'CallBack', 'sdof(''button'', 'help'')');
% Close Button
close_bt = uicontrol(data_fig, ...
    'Style', 'push', ...
    'Units', 'normalized',...
    'String', 'Close', ...
    'Position', [ .68 .03 .25 .05 ],...
    'CallBack', 'sdof(''button'', 'close'')');
% Get Points with Mouse

% ***** SAVE OBJECT HANDLES *****
%Set UserData for objects
set(mass_inp, 'UserData', [stif_inp freq_inp delt_inp time_inp]);
set(stif_inp, 'UserData', [mass_inp freq_inp delt_inp time_inp]);
set(freq_inp, 'UserData', [mass_inp stif_inp delt_inp time_inp]);
set(delt_inp, 'UserData', [mass_inp stif_inp freq_inp time_inp]);
set(time_inp, 'UserData', [mass_inp stif_inp freq_inp delt_inp]);

set(pd_hdl(2, 1), 'UserData', [pd_hdl(3, 1) pd_hdl(4, 1)]);
set(pd_hdl(3, 1), 'UserData', [pd_hdl(2, 1) pd_hdl(4, 1)]);
set(pd_hdl(4, 1), 'UserData', [pd_hdl(2, 1) pd_hdl(3, 1)]);

set(resp_fft, 'UserData', [minf_inp maxf_inp]);
set(load_fft, 'UserData', [minf_inp maxf_inp]);
set(option_menu, 'UserData', [minf_inp maxf_inp]);
set(minf_inp, 'UserData', maxf_inp);

LOAD_HAND = [lmag_txt  lmag_inp
             lfre_txt  lfre_inp
             tini_txt  tini_inp
             tfin_txt  tfin_inp
             imag_txt  imag_inp
             fmag_txt  fmag_inp
             file_txt  file_inp
             file_btn  ltyp_pop
             perd_txt  perd_inp
             sine_rad  csin_rad

```

```

    lloc_pop  load_pop
    fhelpt_bt 0.0    ];
LOAD_HAND = [ LOAD_HAND ; pd_hdl ];

PROP_HAND = [mass_inp stif_inp damp_inp idis_inp ivel_inp load_pop, ...
             time_inp delt_inp];

%*****
%*           CHANGE LOAD OPTION: DISPLAY PROPER INPUT OPTIONS           *
%*****

elseif strcmp(option, 'load') %change in load type, reset load display
    load_type=arg1;
    resp_type=popupstr(LOAD_HAND(8,2));
    current=get(LOAD_HAND(8,2), 'Value');
    %For change in load condition, set to specified loading
    if nargin<3 %
        arg2=0;
    end;

    if arg2==1 & ~strcmp(load_type, 'FREE VIBRATION')
        load_type = popupstr(LOAD_HAND(11,2));
    end

    % Remove Harmonic option when changing from sinusoidal
    if ~strcmp(load_type, 'Sinusoidal') & strcmp(resp_type, 'Harmonic')
        current=1;
    end;
    if strcmp(load_type, 'Sinusoidal')
        set(LOAD_HAND(8,2), 'String','Transient|Periodic|Harmonic' );
    else
        set(LOAD_HAND(8,2), 'String','Transient|Periodic' );
    end;
    set(LOAD_HAND(8,2), 'Value', current);
    resp_type=popupstr(LOAD_HAND(8,2));

    % Select proper displays for the type of loading selected
    set(LOAD_HAND(1,:), 'Visible','off');
    set(LOAD_HAND(2,:), 'Visible','off');
    set(LOAD_HAND(3,:), 'Visible','off');
    set(LOAD_HAND(4,:), 'Visible','off');
    set(LOAD_HAND(5,:), 'Visible','off');
    set(LOAD_HAND(6,:), 'Visible','off');
    set(LOAD_HAND(7,:), 'Visible','off');
    set(LOAD_HAND(8,1), 'Visible','off');
    set(LOAD_HAND(8,2), 'Visible','on' );
    set(LOAD_HAND(9,:), 'Visible','off');
    set(LOAD_HAND(10,:), 'Visible','off');
    set(LOAD_HAND(11,2), 'Visible','on');
    set(LOAD_HAND(12,1), 'Visible','off');
    set(LOAD_HAND(13:19,:), 'Visible','off');

    if strcmp(load_type, 'FREE VIBRATION')
        set(LOAD_HAND(8,2), 'Visible','off');
        set(LOAD_HAND(11,2), 'Visible','off');
    elseif strcmp(load_type, 'Point Data')
        set(LOAD_HAND(13:19,:), 'Visible','on');
    elseif strcmp(load_type, 'Step Load')
        set(LOAD_HAND(1,:), 'Visible','on' );
        set(LOAD_HAND(3,:), 'Visible','on' );
        set(LOAD_HAND(4,:), 'Visible','on' );
    elseif strcmp(load_type, 'Sinusoidal')

```

```

        set(Load_HAND(1,:), 'Visible','on' );
        set(Load_HAND(2,:), 'Visible','on' );
        set(Load_HAND(3,:), 'Visible','on' );
        set(Load_HAND(4,:), 'Visible','on' );
    elseif strcmp(load_type, 'Triangular')
        set(Load_HAND(3,:), 'Visible','on' );
        set(Load_HAND(4,:), 'Visible','on' );
        set(Load_HAND(5,:), 'Visible','on' );
        set(Load_HAND(6,:), 'Visible','on' );
    elseif strcmp(load_type, 'File')
        set(Load_HAND(7,:), 'Visible','on' );
        set(Load_HAND(8,1), 'Visible','on' );
        set(Load_HAND(12,1), 'Visible','on' );
    end;

% Set period display if response is periodic
if strcmp(resp_type, 'Periodic') & ~strcmp(load_type, 'Free Vibration')
    set(Load_HAND(9,:), 'Visible','on' );
elseif strcmp(resp_type, 'Harmonic')
    set(Load_HAND(3,:), 'Visible','off');
    set(Load_HAND(4,:), 'Visible','off');
    set(Load_HAND(10,:), 'Visible','on');
end;

%*****
%*          PLOT GRAPH: READ INPUT AND PLOT GRAPH          *
%*****

elseif strcmp(option, 'plot')    %plot response with selected options
    setptr(gcf, 'watch');

    M =str2num(get (PROP_HAND(1), 'String'));; sdof('check', M, 0);
    K =str2num(get (PROP_HAND(2), 'String'));; sdof('check', K, 0);
    D =str2num(get (PROP_HAND(3), 'String'));; sdof('check', D, -.001);
    Y0 =str2num(get (PROP_HAND(4), 'String'));
    V0 =str2num(get (PROP_HAND(5), 'String'));
    TIME=str2num(get (PROP_HAND(7), 'String'));; sdof('check', TIME, 0);
    delT=str2num(get (PROP_HAND(8), 'String'));; sdof('check', delT, 0);
    applied_1 = popupstr(Load_HAND(11,1));
    load_type = popupstr(PROP_HAND(6));
    load_perd = popupstr(Load_HAND(8,2));

    if strcmp(applied_1, 'FREE VIBRATION')
        load_type = 'Free Vibration';
    end

%    Calculate system properties
D= 0.01*D;
C =D*2*sqrt(K*M);
W = sqrt(K/M);
Wd=W*sqrt(1-D^2);
T=(2*pi)/W;
%    Set matrices for using lsim function
A1=[ 0 1; -K/M -C/M];
A2=[ 0 ; 1/M];
A3=[1 0];
A4=0;

if (delT <= 0)
    delT=T/16;
end;
t1 = (0:delT:TIME);

```

```

s=size(t1,2);
true_s=s;
if strcmp(load_perd, 'Periodic')
    P =str2num(get(LOAD_HAND(9,2),'String'));, sdof('check', P, .001);
    p=round(P/delT); % make sure p is greater than one
    sdof('check', p, 1);
    if s<=p, s=p+1;, t1 = (0:delT:P+delT);, end;
end;
%*****FORM LOAD VECTOR*****
if strcmp(load_type, 'Free Vibration')
    LOAD = zeros(1,s);

elseif strcmp(load_type, 'Sinusoidal')
    P0 =str2num(get(LOAD_HAND(1,2),'String'));
    OM =str2num(get(LOAD_HAND(2,2),'String'));, sdof('check', OM, 0);
    if ~strcmp(load_perd, 'Harmonic')
        To =str2num(get(LOAD_HAND(3,2),'String'));, sdof('check', To, 0);
        Tf =str2num(get(LOAD_HAND(4,2),'String'));, sdof('check', Tf, To);
    else
        To = 0;, Tf = 0;
    end
    OM = OM*(2*pi);
    i=round(To/delT)+1;, f=round(Tf/delT);
    if (f>s), f=s;, end;
    LOAD = zeros(1,s);
    LOAD(i:f) = P0*sin(OM*t1(1:(f-i+1)));

elseif strcmp(load_type, 'Point Data')
    data = zeros(2,3);
    for i=1:3
        for j=1:2
            data(j,i) = str2num(get(LOAD_HAND(13+i,j),'String'));
        end
    end
    LOAD = zeros(1,size(t1,2));
    time = data(1,:);
    load = data(2,:);
    npts = 3;
    i=1; Po=0;, Pf=0;
    for j=1:(npts)
        f=round(time(j)/delT)+1;
        ft = f;
        if (f>s), f=s;, end;
        Po=Pf;
        LOAD(i) = Po;
        Pf= load(j);
        for c=(i+1):f, LOAD(c)= LOAD(c-1)+(Pf-Po)/(ft-i);, end;
        i=f;
    end

elseif strcmp(load_type, 'File')
    data = get(LOAD_HAND(7,2),'UserData');
    LOAD = zeros(1,size(t1,2));
    time = data(1,:);
    load = data(2,:);
    npts = size(time,2);
    i=1; Po=0;, Pf=0;
    for j=1:(npts)
        f=round(time(j)/delT)+1;
        ft = f;
        if (f>s), f=s;, end;
        Po=Pf;

```

```

LOAD(i)= Po;
Pf= load(j);
for c=(i+1):f, LOAD(c)= LOAD(c-1)+(Pf-Po)/(ft-i);, end;
i=f;
end
end

%For support motion, acc, multiply by mass to obtain rel dis.
if strcmp(applied_1, 'TEST')
LOAD=LOAD*(-M);
end;

%*****FOR PERIODIC LOADING, CYCLE LOADING*****
if (~strcmp(load_type, 'Free Vibration') & strcmp(load_perd, 'Periodic'))
Load_P = LOAD(1:p);
cycles=floor(s/p);, a=1;, b=1;
if cycles<2, b=p;, end;
if cycles < 100
for c=2:cycles
a=a+p;, b=a+p-1;
LOAD(a:b)=Load_P;
end
else
error('Please select a larger period');
error;
end
for c=(b+1):(s-1)
LOAD(c)=Load_P(c-b);
end
%SET INITIAL CONDITIONS FOR PERIODIC LOADING
if strcmp(applied_1, 'SUPPORT MOTION')
n=p+1;
t1_P=t1(1:n);, L_P=LOAD(1:n);, L_P(1)=0;
G1=zeros(1,n);, G2=zeros(1,n);
C1=W*W/Wd*(1-2*D^2);
C2=W*2*D;
C3=-W*W*D*W/Wd*(3-4*D*D);
C4=W*W*(1-4*D*D);
for c=1:n
G1(c)=exp(-D*W*t1(c))*(C1*sin(Wd*t1(c))+C2*cos(Wd*t1(c)));
G2(c)=exp(-D*W*t1(c))*(C3*sin(Wd*t1(c))+C4*cos(Wd*t1(c)));
end;
Disp=conv(L_P, G1)*delT;, Y=Disp(n);
Velo=conv(L_P, G2)*delT;, V=Velo(n);
s1=exp(-D*W*P)*sin(Wd*P);
c1=exp(-D*W*P)*cos(Wd*P);
mat1=[ 1-c1-D*(W/Wd)*s1 -s1/Wd
W^2*s1/Wd 1-c1+D*(W/Wd)*s1];
initc=mat1\[Y; V];
Y0=initc(1);, V0=initc(2);
set(PROP_HAND(4), 'String', Y0);
set(PROP_HAND(5), 'String', V0);
else
t1_P=t1(1:p+1);, L_P=LOAD(1:p+1);, L_P(1)=0;
[D1,V1] = lsim(A1, A2, A3, A4, L_P, t1_P, [0, 0]);

s1=exp(-D*W*P)*sin(Wd*P);
c1=exp(-D*W*P)*cos(Wd*P);
mat1=[ 1-c1-D*(W/Wd)*s1 -s1/Wd
W^2*s1/Wd 1-c1+D*(W/Wd)*s1];
initc=mat1\[V1(p+1,1); V1(p+1,2)];

```

```

        Y0=initc(1);, V0=initc(2);
        set(PROP_HAND(4),'String', Y0);
        set(PROP_HAND(5),'String', V0);
    end
end

%*****SINUSOIDAL- PHASE AND TRANSFER*****
if strcmp(load_type, 'Sinusoidal')
    r=OM/W;

    if strcmp(applied_1, 'SUPPORT MOTION')
        m=1/sqrt( (1-r^2)^2+(2*D*r)^2 )*sqrt(1+(2*D*r)^2);
        UO=1;
        ang=(2*D*r)^3/((1-r^2)+(2*D*r)^2);
        if ang<0, theta = atan ( ang ) + pi;
        else,      theta = atan ( ang );
        end
    else
        m=1/sqrt( (1-r^2)^2+(2*D*r)^2 );
        UO=P0/K;
        if r==1,      theta=pi/2;
        elseif r<1, theta= atan( (2*D*r)/( (1-r^2) ) );
        else,      theta= atan( (2*D*r)/( (1-r^2) ) )+pi;
        end
    end;
else
    m=0; , theta=0;
end
set(LOAD_HAND(2,2), 'UserData', [ m theta ]);

%*****SOLVE FOR RESOPONSE*****
if strcmp(load_type, 'Free Vibration')
    xplt1 = (Y0*cos(Wd*t1)+((V0+Y0*D*W)/Wd)*sin(Wd*t1)).*exp(-D*W*t1);

    %-----SINUSOIDAL HARMONIC-----
elseif strcmp(load_type, 'Sinusoidal') & strcmp(load_perd, 'Harmonic')
    sine=get(LOAD_HAND(10,1), 'Value');
    if sine==1
        xplt1= m*UO * sin(OM*t1-theta);, LOAD=P0*sin(OM*t1);
    else
        xplt1= m*UO * cos(OM*t1-theta);, LOAD=P0*cos(OM*t1);
    end;

    %-----SUPPORT MOTION-----
elseif strcmp(applied_1, 'SUPPORT MOTION')
    GIRF=zeros(1,s);
    C1=W*W/Wd*(1-2*D^2);
    C2=W*2*D;
    for c=1:s
        GIRF(c)=exp(-D*W*t1(c))*(C1*sin(Wd*t1(c))+C2*cos(Wd*t1(c)));
    end;

    tsize = size(t1, 2);
    gsize = size(GIRF, 2);
    if tsize >gsize
        t1=t1(1:gsize);
    elseif gsize > tsize
        GIRF = GIRF(1:tsize);
    end

    xplts=conv(LOAD,GIRF);, xplts=xplts(1:s)*delT;
    C2=(V0+D*W*Y0)/Wd;

```



```

for c=1:s
    xplt1(c)=xplts(c)+exp(-D*W*t1(c))*(C2*sin(Wd*t1(c))+Y0*cos(Wd*t1(c)));
end;

t1=t1(1:true_s);
xplt1=xplt1(1:true_s);
LOAD=LOAD(1:true_s);

%-----ALL OTHER LOADING-----
else
    LOAD(1)=0;
    tsize = size(t1, 2);
    lsize = size(LOAD, 2);
    if tsize >lsize
        t1=t1(1:lsize);
    elseif lsize > tsize
        LOAD = LOAD(1:tsize);
    end
    [xplt1,yplt1] = lsim(A1, A2, A3, A4, LOAD, t1, [Y0, V0]);
    t1=t1(1:true_s);
    xplt1=xplt1(1:true_s);
    LOAD=LOAD(1:true_s);
end;
set(PROF_HAND(6), 'UserData', [LOAD; t1]);
clr = clr_plts(PCOLOR);

%*****PLOT GRAPH OR RESPONSE*****
if strcmp(applied_l,'SUPPORT MOTION')
    plot (t1, xplt1, clr);
    ylabel('Absolute Response');
elseif strcmp(applied_l,'TEST')
    plot (t1, xplt1, clr);
    ylabel('Relative Displacement');
else
    plot (t1, xplt1, clr);
    ylabel('Displacement');
end
xlabel('Time');
PCOLOR=PCOLOR+1;
if (PCOLOR >= 7), PCOLOR = 1;, end;

trans_fig=findobj('type', 'figure', 'Name', 'TRANSFER');
if ~isempty(trans_fig)
    sdof('plotting', 'transfer');
end
load_fig=findobj('type', 'figure', 'Name', 'LOAD');
if ~isempty(load_fig) & ~strcmp(load_type,'Free Vibration')
    sdof('plotting', 'load');
end

lfft_fig=findobj('type', 'figure', 'Name', 'LOAD FFT');
if ~isempty(lfft_fig) & ~strcmp(load_type,'Free Vibration')
    sdof('plotting', 'load_fft');
end

rfft_fig=findobj('type', 'figure', 'Name', 'RESPONSE FFT');
if ~isempty(rfft_fig)
    sdof('plotting', 'resp_fft');
end

setptr(gcf, 'arrow');

```

```

%*****
%*          PLOT MENU OPTIONS          *
%*****

elseif strcmp(option, 'plotting')
    setptr(gcf, 'watch');
    main = gcf;
    if strcmp(arg1, 'resp_fft')
        %*****Response Fourier Transform*****
        h=get(gca, 'Children');

        rfft_fig=findobj('type', 'figure', 'Name', 'RESPONSE FFT');
        if isempty(rfft_fig)
            rfft_fig = figure( ...
                'Name', 'RESPONSE FFT', ...
                'Position', [725 40 300 200] ,...
                'UserData', arg2,...
                'NumberTitle', 'off', ...
                'Colormap', []);
        else
            figure(rfft_fig);
        end

        s=size(h,1);
        i=1;
        while i<s & ~strcmp(get(h(i), 'type'), 'line')
            i=i+1;
        end

        if ~strcmp(get(h(i), 'type'), 'line')
            %errorrdlg('No response to analyze');
        else
            data=get( h(i), 'YData');
            tdata=get( h(i), 'XData');
            delT = tdata(2);
            if PCOLOR == 1, pcolor = 1;
            else, pcolor = PCOLOR-1; end
            clr = clr_plts(pcolor);

            rsize=size(data,2);
            if strcmp( popupstr(Load_Hand(8,2)), 'Periodic')
                period = str2num( get(Load_Hand(9,2), 'String') );
                points = round(period/delT); sdof('check', points, 1);
                if rsize > points, rsize=points; end;

                data = data(1:rsize);
            end
            %interpolate to get points to the power of 2
            n=2;
            while n<rsize
                n=n*2;
            end
            response = linear_iteration(data, rsize, n);
            delT = delT*rsize/n;
            resp_fft=fft(response);
            absolute=abs(resp_fft(1:floor(n/2)))*delT;
            nyquist=1/(2*delT);
            fft_freq=(0:(n/2-1))/(n/2)*nyquist;
            hold on;
        end
    end
end

```

```

    plot(fft_freq, absolute, clr);
    h=get(gcf, 'UserData');
    minf = str2num(get(h(1), 'String'));
    maxf = str2num(get(h(2), 'string'));
    set(gca, 'XLim', [minf maxf]);
    title('Response FFT');
    xlabel('Frequency(Hz)');
    ylabel('Absolute Magnitude');
    grid on;
end

elseif strcmp(arg1, 'load')
%*****LOADING*****
main=gcf;
load_fig=findobj('type', 'figure', 'Name', 'LOAD');
if isempty(load_fig)
    load_fig = figure( ...
        'Name', 'LOAD',...
        'Position', [725 600 300 150] ,...
        'NumberTitle', 'off', ...
        'Colormap', []);
else
    figure(load_fig);
end
if PCOLOR == 1, pcolor = 1;
else, pcolor = PCOLOR-1;, end
clr = clr_plts(pcolor);
%Get load vector
data = get(PROP_HAND(6), 'UserData');
if ~isempty(data)
    LOAD = data(1,:);
    time = data(2,:);
hold on;
plot(time, LOAD, clr);
xlabel('Time');
ylabel('Excitation Magnitude')
grid on;
end
figure(main);

elseif strcmp(arg1, 'load_fft')
%*****LOADING FOURIER TRANSFORM*****
main=gcf;
load_fig=findobj('type', 'figure', 'Name', 'LOAD FFT');
if isempty(load_fig)
    load_fig = figure( ...
        'Name', 'LOAD FFT',...
        'Position', [725 600 300 120] ,...
        'UserData', arg2,...
        'NumberTitle', 'off', ...
        'Colormap', []);
else
    figure(load_fig);
end
if PCOLOR == 1, pcolor = 1;
else, pcolor = PCOLOR-1;, end
clr = clr_plts(pcolor);
%Get load vector
data = get(PROP_HAND(6), 'UserData');
if ~isempty(data)
    load = data(1,:);
    time = data(2,:);

```

```

        delT = time(2);
        lsize=size(load,2);
        if strcmp( popupstr(Load_HAND(8,2)), 'Periodic')
            period = str2num( get(Load_HAND(9,2), 'String') );
            points = round(period/delT); sdof('check', points, 1);
            if lsize > points, lsize=points;; end;

            load = load(1:lsize);
            time = time(1:lsize);
        end

        %interpolate to get points to the power of 2
        n=2;
        while n<lsize
            n=n*2;
        end
        LOAD = linear_iteration(load, lsize, n);
        delT = delT*lsize/n;
        load_fft=fft(LOAD);
        absolute=abs(load_fft(1:floor(n/2)))*delT;
        nyquist=1/(2*delT);
        fft_freq=(0:(n/2-1))/(n/2)*nyquist;
        hold on;
        plot(fft_freq, absolute, clr);
        h=get(gcf, 'UserData');
        minf = str2num(get(h(1), 'String'));

        maxf = str2num(get(h(2), 'string'));
        set(gca, 'XLim', [minf maxf]);
        title('LOAD FFT');
        xlabel('Frequency(Hz)');
        ylabel('Absolute Magnitude');
        grid on;
    end

elseif strcmp(arg1, 'impulse')
    %*****IMPULSE RESPONSE FUNCTION*****
elseif strcmp(arg1, 'transfer')
    %*****Transfer Function and Phase Angle*****
    %Determine if figure exist, else create one
    trans_fig=findobj('type', 'figure', 'Name', 'TRANSFER');
    if isempty(trans_fig)
        trans_fig = figure( ...
            'Name', 'TRANSFER', ...
            'Unit', 'pixels', ...
            'UserData', 0, ...
            'Position', [725 25 300 400] ,...
            'NumberTitle', 'off', ...
            'Colormap', []);
    else
        figure(trans_fig);
    end

    %Get Values from data.
    applied_l=popupstr(Load_HAND(11,1));
    M =str2num(get(Prop_HAND(1), 'String'));
    K =str2num(get(Prop_HAND(2), 'String'));
    D =str2num(get(Prop_HAND(3), 'String'));
    OM =str2num(get(Load_HAND(2,2), 'String'));
    ratio = get(Load_HAND(2,2), 'UserData');
    if isempty(ratio), ratio=[0 0];, end;
    D= 0.01*D;, OM=OM*2*pi;

```

```

W = sqrt(K/M);
r = (0:.01:3);
s=size(r,2);
trans = zeros(1,s);
phase = zeros(1,s);

if isempty(OM)
    r_v = 0;
else
    r_v= OM/W;
end

for i=1:s
    if strcmp(applied_1,'SUPPORT MOTION') %Transmisibility
        trans(i)=sqrt(1+(2*D*r(i))^2)/sqrt((1-r(i)^2)^2+(2*D*r(i))^2);
        ang(i)=(2*D*r(i))^3/((1-r(i)^2)+(2*D*r(i))^2);
        if ang(i)<0
            phase(i) = atan ( ang(i) ) + pi;
        else
            phase(i) = atan ( ang(i) );
        end

    else %Use transfer function
        trans(i) = 1/sqrt( (1-r(i)^2)^2+(2*D*r(i))^2 );
        if r(i)<1
            phase(i) = atan ( (2*D*r(i))/(1-r(i)^2) );
        elseif r(i)==1
            phase(i) = pi/2;
        else
            phase(i) = atan ( (2*D*r(i))/(1-r(i)^2) )+pi;
        end
    end
end

phase=phase*180/pi;

flag=get(gcf, 'UserData');
if PCOLOR == 1, pcolor = 1;
else, pcolor = PCOLOR-1;, end
clr = clr_plts(pcolor);

subplot(2, 1, 1);
hold on;
plot(r, trans, clr);
%grid on;
m=ratio(1);
x=[r_v r_v];, y=[0 m];
plot(x,y, clr);
if flag==0
    grid on;
    xlabel('Frequency Ratio');
    if strcmp(applied_1,'SUPPORT MOTION'), ylabel('Transmisibility');
    else, ylabel('Amplitude');, end;
    title('Transfer Function');
end;

subplot(2, 1, 2);
hold on;
plot(r, phase, clr)
%grid on;
set(gca, 'YLim', [0 180]);
theta=ratio(2)*180/pi;

```

```

    x=[r_v r_v];, y=[0 theta];
    plot(x,y, clr);
    if flag==0
        grid on;
        xlabel('Frequency Ratio');
        ylabel('Phase Angle(deg)');
    end;

    set(gcf, 'UserData', 1);

end
figure(main);
setptr(gcf, 'arrow');
%*****
%*          BUTTON CALLBACKS          *
%*****

elseif strcmp(option, 'button')

    if strcmp(arg1, 'close')
        fh=gcf;
        h=get(gcf, 'Userdata');
        trans=findobj('type', 'figure', 'Name', 'TRANSFER');
        load =findobj('type', 'figure', 'Name', 'LOAD');
        rfft =findobj('type', 'figure', 'Name', 'RESPONSE FFT');
        lfft =findobj('type', 'figure', 'Name', 'LOAD FFT');
        if ~isempty(trans), close(trans);, end;
        if ~isempty(load) , close(load);, end;
        if ~isempty(rfft) , close(rfft);, end;
        if ~isempty(lfft) , close(lfft);, end;
        close(fh);

    elseif strcmp(arg1, 'default')

        %min max dt t freq
        freq = str2num(get(arg2(5), 'String'));
        time = ceil(5/freq);
        dt = 1/(16*freq);
        minf = 0;
        maxf = ceil(1/(2*dt)); % nyquist
        set(arg2(1), 'String', minf);
        set(arg2(2), 'String', maxf);
        set(arg2(3), 'String', dt );
        set(arg2(4), 'String', time);

    elseif strcmp(arg1, 'help')

        ttlStr='VIBRATION OF SDOF';
        hlpStr= ...
        ['
        ' This interactive program plots the response '
        ' of a single degree of freedom system to the '
        ' specified forcing function. Enter the system'
        ' properties and loading conditions, then press'
        ' the PLOT button to obtain the response. '
        ' Previous plots remain on the screen, to allow'
        ' for comparison of the different results. To '
        ' clear the screen press the CLEAR button.  '];
        helpfun(ttlStr,hlpStr);

    elseif strcmp(arg1, 'helpfile')

```

```

ttlStr='Input File Format';
hlpStr= ...
['
    ' To input a random forcing function, such
    ' as an earthquake, prepare a file with the
    ' following format. Note the "Point Data"
    ' option that allows you to specify a function
    ' consisting of up to three points.
    ,
    ' File Format:
    ,
    ' number_of_points
    ' t1      P1
    ' t2      P2
    ' t3      P3
    ' ...
    ' tn      Pn
    '];
helpfun(ttlStr,hlpStr);

elseif strcmp(arg1, 'clear')
plot_btn=findobj('String', 'Plot Response');
set(plot_btn, 'Enable', 'off');
cla, PCOLOR=1;
main=gcf; %, h=get(main, 'UserData');
trans=findobj('type', 'figure', 'Name', 'TRANSFER');
load =findobj('type', 'figure', 'Name', 'LOAD');
rfft =findobj('type', 'figure', 'Name', 'RESPONSE FFT');
lfft =findobj('type', 'figure', 'Name', 'LOAD FFT');
if ~isempty(trans), figure(trans);
    subplot(2,1,1), cla, subplot(2,1,2), cla, end;
if ~isempty(load) , figure(load);, cla, end;
if ~isempty(rfft) , figure(rfft);, cla, end;
if ~isempty(lfft) , figure(lfft);, cla, end;
set(trans, 'UserData', 0);
figure(main)
set(plot_btn, 'Enable', 'on');
end
elseif strcmp(option, 'file_plot')
filename = arg1;
filedat = fopen(filename, 'r');
npts = fscanf(filedat, '%d', 1);
time = zeros(1,npts);
load = zeros(1,npts);
for j=1:npts
    time(j) = fscanf(filedat, '%g', 1);
    load(j) = fscanf(filedat, '%g', 1);
end
main=gcf;
load_fig=findobj('type', 'figure', 'Name', 'LOAD');
if isempty(load_fig)
    load_fig = figure( ...
        'Name', 'LOAD',...
        'Position', [725 450 300 275] ,...
        'NumberTitle', 'off', ...
        'Colormap', []);
else
    figure(load_fig);
end
if PCOLOR == 1, pcolor = 1;
else, pcolor = PCOLOR-1;, end
clr = clr_plts(pcolor);
%Get load vector

```

```

plot(time, load, clr);
title('Loading')
ylabel('magnitude')
grid;
figure(main);
data = [time; load];
set(arg2, 'UserData', data);

elseif strcmp(option, 'check')
    if isempty(arg1)
        errordlg('A required input is missing');
        error('A required input is missing');
        error;
    elseif arg1<arg2
        errordlg('Improper input specified, check data');
        error('Improper input specified, check data');
    end
end

function vec2=linear_iteration(vec1, n1, n2)
x=n1/n2;
vec = zeros(1,n2);
vec(1)=vec1(1);
for i=2:(n2-1)
    t=(i-1)*x+1;
    j=floor(t);
    e=t-j;
    vec(i)=(1-e)*vec1(j)+e*vec1(j+1);
end
vec(n2)=vec1(n1);
vec2=vec;

```