STUDY OF EARTHQUAKE RESPONSE SPECTRA

By

S.SATISH 1998B4A2437

Under the supervision of

Dr. SATYENDRA P. GUPTA
Group Leader, Civil Engineering Group

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE

Pilani, Rajasthan-333031.

May, 2003.

STUDY OF EARTHQUAKE RESPONSE SPECTRA

Submitted in Partial Fulfillment of the requirements of

BITS C421T: Thesis

By

S.SATISH 1998B4A2437

Under the supervision of

Dr. SATYENDRA P. GUPTA
Group Leader, Civil Engineering Group

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE

Pilani, Rajasthan-333031.

May, 2003.

ACKNOWLEDGEMENT

I would like to thank Dr. Satyendra P. Gupta for giving me an opportunity to work under him. His valuable encouragement and his kind advice were of great importance in the thesis work. I also thank all the members of the Civil Engineering association for rendering their help to me in completing the project.

CERTIFICATE

This is to certify that the Thesis entitled, "Study of Earthquake Response Spectra", and submitted by S.Satish 1998B4A2437 in partial fulfillment of the requirements of BITS C 421T Thesis embodies the work done by him under my supervision.

Date **Signature of the Supervisor**

Name & Designation
of Supervisor

Thesis Title: Study of Earthquake Response Spectra

Supervisor: Dr. Satyendra P. Gupta

Semester: Second Semester

Name of The Student: S.Satish ID NO: 1998B4A2437

ABSTRACT

This report deals with the method for finding the solution of a second degree equation, which is the equation of motion for a SDF system under earthquake excitation. Four methods are discussed and are compared with each other to find the most suitable method which can be used to solve the equation and thus plot the response spectra given the input earthquake accelerogram record of any earthquake. This response spectra thus obtained in the form of graph is used to find the maximum displacement of the buildings undergoing the earthquake motion. This method is available in IS code called the *Response Spectrum Method*. This report also describes the software which is developed to plot the ground response, response and response spectra given proper inputs.

Signature of the Student

(S.Satish)

Date:

Signature of the Supervisor

(Dr. Satyendra P.Gupta)

Civil Engineering Group

Date:

TABLE OF CONTENTS

CHAPTER – 1

INTRODUCTION

The technique of response spectra of strong motion earthquakes was introduced around 1940. Since then it has proved useful and informative in solving problems of design and analysis of structures subjected to strong earthquake motions. The spectrum is a plot of the peak values of a response quantity as a function of the natural vibration period T_n of the system, or a related parameter such as natural circular frequency w_n or cyclic frequency f_n. Each such plot is for a fixed damping ratio E, and several such plots for different values of E are included to cover the range of damping values encountered in actual structures. In a simpler way it can be said that spectra, calculated from the recorded ground acceleration, are plots of the maximum response to the earthquake of a simple oscillator over a range of value of their natural period and damping. The spectrum provides a description of the frequency characteristics of the ground motion and gives the maximum response of simple structures to the earthquake. By the superposition of different modes of response spectrum technique can be applied to the design and analysis of complex structures such as buildings, dams, bridges, nuclear power plant and other important buildings. The spectrum technique represents an intermediate approach between a design based on static loads and a complete integration of the equations of motion of the complex structures.

When force is represented by equation, it's possible to get close form solution. A set of response values of a quantity for a single degree freedom system can be obtained from the above equation, which will give a maximum value. Changing the time period of single degree freedom system will give another maximum value and so on. Plotting the maximum value verses time period will give response spectra of that quantity.

Strong motion earthquake records have been obtained infrequently in the past and the reduction to digital form, or equivalent analog form, and subsequent calculation of spectra have been performed on more or less on individual basis. However, starting from 1960's till now a number of strong motion instruments placed in the seismic regions of the world, and so several strong motion earthquake accelerogram data's are available

now. India itself has detailed data of two earthquakes (Koyana and Bhuj). The potential volume of the data and the development of the accelerograph indicate clearly that rapid and automated data processing for spectrum calculation procedures are needed. In an effort to fulfill a part of this need, this report details on new methods, which can be used for accurate and rapid computation of response spectra of strong motion earthquakes.

The digital computation of spectra requires the repeated numerical solution of the response of a simple oscillator is described by a second order, linear, inhomogeneous differential equation and if a digital description of the earthquake record is available, the response can be obtained by numerical integration. Several numerical integration techniques have been used for spectra calculation.

This report discusses about formulation of new methods for solving the differential equation governing the earthquake motion.

CHAPTER – 2

OBJECTIVE AND SCOPE

The main objective of this report is to develop and test methods for solving the differential equation for earthquake motion. It also compares the methods discussed. Testing is done using computer program, which displays the response spectra given the input earthquake data. The computer program would take the natural frequency and damping ratio of simple structure as well as accelerogram digitization time interval and acceleration value as input.

The scope of the thesis is restricted to elastic response spectra, where the only variables that determine the response are Time Period T_n and Damping Coefficient ζ. This report deals with the formulation and discussion of several methods for developing the response spectra for various ground motion excitation. Once the earthquake accelerogram record is fed as input to the software, it will give the digitized value as well as graphical representation of

1. Ground Response
2. Response given the time period and damping.
3. Response spectra given the time interval and damping.

CHAPTER – 3

LITERATURE REVIEW

The very first response spectrum was proposed by Benioff in 1934 and again in 1943 by M.A.Biot (Walter W.Hays, 1980) proposed it as a different method for determining the maximum amplitudes of response of ensemble of simple damped harmonic oscillators, when exited by a given ground motion time history.

G.W.Housner, father of earthquake engineering, accepted the concept of M.A.Biot, as a practical means of characterizing ground motion and their efforts on structures. The site-independent method was first introduced by G.W.Housner in 1959 (Walter W.Hays, 1980). He derived smooth normalized acceleration and velocity response spectra from the two horizontal components of ground acceleration recorded at four large earthquakes in the western United States. The earthquakes were: (1) 1934 Imperial Valley, California (M=6.5); (2) 1940 Imperial Valley, California (M=7.9); (3) 1952 Kern County, California (M=7.7) and (4) 1949 Puget Sound, Washington (M=7.1). The emperial distances ranged from 8 to 56 km. The recording sites were underlain by rock, stiff soil, and deep cohesionless soil. These spectra were scaled by a factor based on the spectrum intensity rather than the peak ground acceleration. G.W.Housner used electric analog technique. But now the availability of digital computers and progressive in the speed of digital computation has led to an increasing use of digital computers in the calculation of spectra.

In 1959, Newmark and Hall (Chopra.K.Anil, 2001) proposed a new technique for estimating site independent spectra. Their technique as based on the fact that the response spectrum over certain frequency ranges is related by an amplification factor to the peak values of ground acceleration, velocity and displacement. The amplification factor were statically determined from response spectra derived from the accellogram recorded at EL Centro from the 1940 Imperial valley, California earthquake and are a log-normal distribution. In this method, the estimated values of peak ground acceleration; velocity

and displacement for the site are plotted on tripartite logarithmic papers. Using the amplification factors that correspond to the desired percentage of critical damping, the peak ground motion values are amplified to give a smooth design response spectrum for the site.

Newmark and Hall in 1969 (Walter W.Hays, 1980) also proposed "standard" earthquake spectra for use whenever adequate details about the site ground motion parameters are available. These spectra values are based on peak ground motion values observed at El Centro from the Imperial Valley, California earthquake, but are about 50 percent more conservative than the El Centro values. The Imperial Valley accelerogram was recorded within 10 km of the fault; the levels of peak horizontal ground acceleration, velocity and displacement were respectively, 0.33g, 34.8 cm/s and 21.1 cm. The soil column at the El Centro site contained a 30-m-thick layer of stiff clay that amplified the input rock accelerations. The proposed standard earthquake had peak ground motion values of 0.5g, 60.9 cm/s and 45.7 cm, but it could be scaled to any peak ground acceleration level on the assumption that the three ground motion parameters are proportion to each other regardless of the level of shaking. Thus one can use peak ground acceleration of 1.0g. The studies by Newmark and Hall (1969) also provided an empirical basis for defining vertical ground motion spectra.

1n 1973, the United State Atomic Energy Commission (Walter W.Hays, 198) proposed guidelines for developing improved site-independent response spectra. It was based on two independent studies of the static properties of response spectra of earthquake ground motions by N.M.Newmark consulting services (1973) and J.A.Blume and associates, engineers (1973) was developed for defining site independent earthquake response spectra that are applicable to most sites. The only exceptions are the sites, which are relatively close to the epicenter of a postulated earthquake or sites, which have physical characteristics (e.g. Foundation deposits with well-defined frequency filtering characteristics) that could significantly enhance the spectral characteristics of ground motion in a portion of the spectral band of interest.

CHAPTER – 4

EARTHQUAKE RESPONSE OF LINEAR SYSTEMS

One of the most important applications of the theory of structural dynamics is in analyzing the response of structures to ground shaking caused by an earthquake. Of particular interest is the earthquake response of linear SDF (Single Degree of Freedom) systems to earthquake motions. By definition, linear systems are elastic systems, and we shall also refer them as elastic systems. Because earthquake can cause damage to many structures, we are also interested in the response of yielding or inelastic systems. A plot of the peak value of a response quantity as a function of the natural vibration period T_n of the system, or a related parameter such as circular frequency or cyclic frequency, is called the response spectrum for that quantity.

4.1 Earthquake Excitation

For engineering purposes the time variation of ground acceleration is the most useful way of defining the shaking of the ground during an earthquake. The ground acceleration $\ddot{u}(g)$ appears on the right side of the differential equation governing the response of structures to earthquake excitation. Thus, for given ground acceleration the problem to be solved is defined completely for an SDF system with known mass, stiffness, and damping properties.

The basic instrument to record three components of ground shaking during earthquakes is the strong-motion accelerograph, which does not record continuously but is triggered into motion by the first waves of the earthquake to arrive. This is because, even in earthquake prone regions, such as California and Japan, there may not be any strong ground motion to earthquakes record for months, or even years, at a time. Consequently, continual recordings of hundreds of such instruments would be a wasteful exercise. After triggering, the recording continues for some minutes or until the ground shaking falls

again to imperceptible levels. Clearly, the instruments must be regularly maintained and serviced so that they produce a record when shaking occurs.

The basic element of an accelerograph is a transducer element, which in its simplest form is an SDF mass-spring-damper system. Therefore, the transducer element is characterized by its natural frequency f_n and viscous damping ratio ζ; typically, f_n = 25 Hz and ζ = 60% for modern analog accelerographs. These transducer parameters enable the digital instrument to record, without excessive distortion, acceleration-time functions containing frequencies from very low up to say, 30Hz; the analog instrument is accurate over a narrower frequency range, say up to 15Hz.

Unfortunately, instrumental records of strong ground shaking were scarce for many years, and even today none or very few records may be obtained from a destructive earthquake in some parts of the world. For example, no strong-motion records were obtained from two earthquakes during 1993 that caused much destruction; Killari, Maharashtra, India, September 20, 1993; and Guam, a U.S. territory, August 8, 1993. Ideally, when a strong earthquake occurs it would be desirable to have many stations instrumented to record the ground motions. However, not knowing when earthquakes will occur and having limited budgets for installation and maintenance of instruments, it is only occasionally possible to obtain such recordings in the region of strongest shaking.

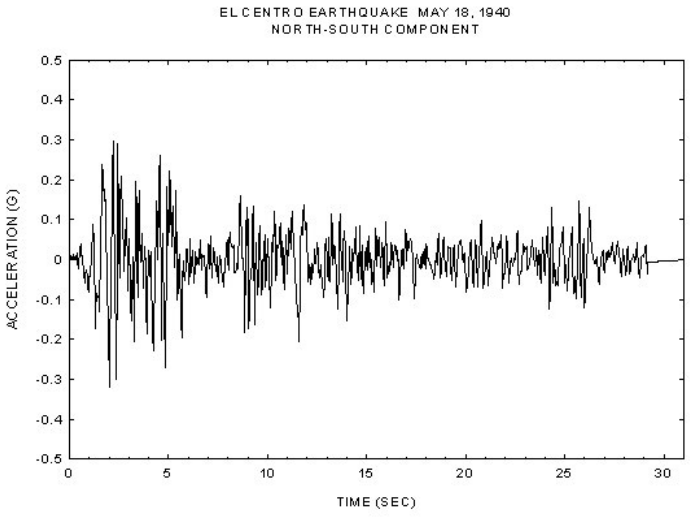

Figure 1

Figure 1 shows the north component of the ground motion recorded at a site in El Centro, California, during the Imperial Valley earthquake of May 18th, 1940. At this scale it becomes apparent that ground acceleration varies with time in a highly irregular manner. No matter how irregular the ground motion is presumed to be known and independent of the structural response. This is equivalent to saying that the foundation soil is rigid, implying no soil-structure interaction. If the structure were founded on very flexible soil, the motion of the structure and the resulting forces imposed on the underlying soil can modify the base-motion.

The ground acceleration is defined by numerical values at discrete time instants. These time instants should be closely spaced to describe accurately the highly irregular variation of acceleration with time. Typically, the time interval is chosen to be 1/100 to 1/50 of a second, requiring 1500 to 3000 ordinates to describe the ground motion of Figure 1.

The top curve in Figure 1 shows the variation of El Centro ground acceleration with time. The peak ground acceleration $\ddot{u}_{go}$ is 0.319g. The second is the ground velocity, obtained by integrating the acceleration-time function. The peak ground velocity $\dot{u}_{go}$ is 13.03 in/sec. Integration of ground velocity provides ground displacement, presented as the lowest trace. The peak ground displacement u_{go} is 8.4m. It is difficult to determine accurately the ground velocity and displacement because analog accelerographs do not record the initial part-until the accelerograph is triggered-of the acceleration-time function and thus the base (zero acceleration) line is known is unknown. Digital accelerographs overcome this problem by providing a short memory so that the onset of ground motion is measured.

In existence are several different versions of the El Centro ground motion. The variations among them arise from differences in (1) How the original analog trace of acceleration verses time was digitized into numerical data, and (2) The procedure chosen to introduce the missing baseline in the record.

4.2 Equation of Motion

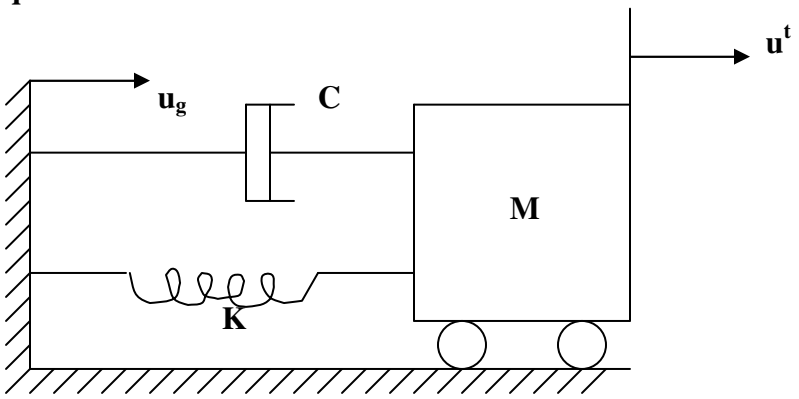

Figure 2

Consider the Spring-Mass system shown in Figure 2. The equation of equilibrium for the figure is

$W/g\ \ddot{\boldsymbol{u}} = W - (W+ku) - c\dot{\boldsymbol{u}}$

Where c=coefficient of viscous damping

And $m=W/g$, $w^2 = k/m = kg/W$

$2n = c/m = cg/W$

Thus $\ddot{u} + 2n\dot{u} + wu = 0$

This is the equation of motion for a free vibration. But in case of forced vibrations, the equation will slightly modify.

$\ddot{u} + 2n\dot{u} + wu = q$

where $q=Q/m = F(t')/m = f(t')$ = disturbing force per unit mass.

Assume that q is expressed as a function of dummy time variable *t'*. Then at any time instant *t'* we may calculate an incremental impulse *qdt'*. This impulse imparts to each unit of mass an instantaneous increase in velocity.

$du = qdt'$

Therefore, *du* = incremental displacement.

In earthquake-prone regions, the principle problem of structural dynamics that concern structural engineers is the behavior of structures subjected to earthquake-induced motion of the base of the structure. The displacement of the ground is denoted by u_g, the total (or absolute) displacement of the mass by u^t, and the relative displacement between the mass and ground by u. At each instant of time, these displacements are related by

$$u^t(t) = u(t) + u_g(t)$$

Both u^t and u_g refer to the same inertial frame of reference and their positive directions coincide.

The equation of motion for the idealized one-story system subjected to earthquake excitation can be derived by the concept of dynamic equilibrium. From the free-body diagram including the inertia force f_t, shown in **Fig**, the equation of dynamic equilibrium is

$$f_I + f_D + f_S = 0$$

Only the relative motion u between the mass and the base due to structural deformation produces elastic and damping forces (i.e., the rigid-body component of the displacement of the structure produces no internal forces). The inertia force f_I is related to the acceleration $\ddot{u}^t$ of the mass by

$$f_I = m\ddot{u}^t$$

By substitution, we get the equation of motion governing the motion of a linear SDF system shown in Figure 2 subjected to ground acceleration $\ddot{u}_g(t)$. Dividing this equation by m gives

$$\ddot{u}+2\zeta w_n\mathring{u}+w_n^2 u = \ddot{u}_g(t) \qquad \textbf{-(A)}$$

It is clear that for a given $\ddot{u}_g(t)$, the deformation response u(t) of the system depends only on the natural frequency w_n or the natural period T_n of the system and its damping ratio, ζ; writing formally, $u = u(t, T_n, \zeta)$. Thus any two system having the same values of T_n and ζ will have the same deformation response $u(t)$ even though one system may be more massive than the other or one may be stiffer than the other. Ground acceleration during

earthquakes varies irregularly to such an extent that analytical solution of the equation of motion must be ruled out. Therefore, numerical methods are necessary to determine the structural response and nay of the methods which is feasible and gives appropriate results can be used.

4.3 Response Quantities

Of greatest interest in structural engineering is the deformation of the system, or displacement $u(t)$ of the mass relative to the moving ground to which the internal forces are linearly related. These are the bending moments and shears in the beams and columns of the one story frame or the spring force in the system. Knowing the total displacement $u^t(t)$ of the mass would be useful in providing enough separation between adjacent buildings to prevent their pounding against each other during an earthquake. Pounding is the cause of damage to several buildings during almost every earthquake. Similarly, the total acceleration $\ddot{u}^t(t)$ of the mass would be needed if the structure is supporting sensitive equipment and the motion imparted to the equipment is to be determined. The numerical solution of Equation (A) can be implemented to provide results for relative quantities $u(t)$, $\mathring{u}(t)$ and $\ddot{u}(t)$ as well as total quantities $u^t(t)$, $\mathring{u}^t(t)$ and $\ddot{u}^t(t)$.

CHAPTER – 5

METHOD FOR SOLVING DIFFERENTIAL EQUATION FOR EARTHQUAKE EXCITATION

Consider the equation of motion for a damped system,

$$w/g\ d^2x/dt^2 + b\ dx/dt + kx = f(x, t) \qquad -(1)$$

Equation (1) can be modified and written as

$$d^2x/dt^2 = (g/w)\ (\ f(x, t) - b\ dx/dt - kx\ )$$

or in other terms, it can be written as

$$d^2x/dt^2 = F(t, x, dx/dt) \qquad -(2)$$

Where $F(t, x, dx/dt) = (g/w)\ (\ f(x, t) - b\ dx/dt - kx\ )$

This is a function of time, velocity and displacement. Let the initial values of the function and derivatives be

$$x(t_o) = x_o,\ x'(t_o) = x_o' \qquad -(3)$$

We convert this to a pair of two first order equations by simple expedient of defining the derivative as a second function.

Thus let,

$$dx/dt = y \qquad x(t_o)=x_o \qquad -(4)$$

Since $d^2x/dt^2 = d/dt\ (dx/dt) = dy/dt$, the equation becomes

$$dy/dt = F(t, x, y) \qquad y(t_o) = x_o' \qquad -(5)$$

Thus equation (4) and (5) together represent equation (3). For even higher orders, each of the lower derivatives are defined as a new function, giving a set of n first-order equations that correspond to the n^{th}-order differential equation. For a system of higher order

equations, each is similarly converted so that there is a set of n number first order equation results.

Thus for an n^{th} order differential equation

$y^{(n)} = f(x, y, y', y'', \ldots\ldots, y^{(n-1)})$ with initial values

$y(x_o) = A_1,\ y'(x_o) = A_2,\ y''(x_o) = A_3, \ldots\ldots\ldots,\ y^{(n-1)}(x_o) = A_n$

is converted to a system of n first-order differential equation by letting $y_1 = y$ and

$y_1' = y_2$

$y_2' = y_3$

……

……

$y'_{(n-1)} = y_n$

and $y_n' = f(x, y_1, y_2, \ldots\ldots, y_n)$

with initial conditions

$y_1(x_o) = A_1,\ y_2(x_o) = A_2,\ y_3(x_o) = A_3, \ldots\ldots y_n(x_o) = A_n$

For the differential equation governing the earthquake motion, the method is applied as follows.

$dy/dx = F(t, x, y)$

$F(t, x, y) = F(t, u, \dot{u}) = -\ddot{u}_g(t) - 2\zeta w_n \dot{u}(t) - w_n^2 u(t)$

Thus for a linear SDF system, the 2 first degree equations are

$du/dt = v$ with initial condition $u(0)=0$ -(6)

$dv/dt = F(t, u, \dot{u})$ with $\dot{u}(0) = v(0) = 0$ -(7)

Equation (6) and (7) are two single order differential equations and can be solved using existing numerical methods. The following **three** methods are used for solving them. The following methods are discussed as they give fairly close values for the solution. They are

1. Runge-Kutta Method
2. Runge-Kutta Fehlberg Method
3. Adam-Moulton Formula
4. Interpolation by Excitation

Runge-Kutta Method

The numerical methods such as Taylor series method, Euler and Modified Euler methods, though not very impressive, serve as good introduction to the next procedures. Although we can improve the accuracy of these methods by taking smaller step sizes, much greater accuracy can be obtained more efficiently by a group of methods after two German mathematicians, Runge and Kutta. They developed algorithms that solve a differential equation efficiently and yet are the equivalent of approximating the exact solution by matching the first n terms of the Taylor-series expansion. We will consider only the fourth order and fifth order Runge-Kutta methods, even though there are higher order methods. Actually, the modified Euler method is a second order Runge-Kutta method.

To impart some knowledge about how the Runge-Kutta method are developed, we will show the derivation of a simple second-order method. Here, the increment to y is a weighted average of two estimates of the increment which we call k_1 and k_2. Thus for the equation dy/dx = f(x, y),

$$y_{n+1} = y_n + ak_1 + k_2$$
$$k_1 = hf(x_n, y_n),$$
$$k_2 = hf(x_n + \alpha h, y_n + \beta k_1).$$

We can think of k_1 and k_2 as estimates of the change in y when x advances by h, because they are the product of the change in x and a value for the slope of the curve, dy/dx. The Runge-Kutta methods always use the simple Euler estimate as the first estimate of Δy; the other estimate is taken with x and y stepped up by the fractions α and β of h and of the estimate of Δy, k_1. Our problem is to devise a scheme of choosing the four parameters, a, b, α and β. The fourth order Runge-Kutta methods are most widely used and are derived in the similar fashion as for the second order methods. Greater complexity results from having to compare term through h^4, and this gives a set of 11 equations and 13 unknowns. The set of 11 equations can be solved with two unknowns chosen arbitrarily. The most commonly used set of values leads to the algorithm

$$y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4),$$
$$k_1 = hf(x_n, y_n),$$
$$k_2 = hf\left(x_n + \frac{h}{2}, y_n + \frac{k_1}{2}\right),$$
$$k_3 = hf\left(x_n + \frac{h}{2}, y_n + \frac{k_2}{2}\right),$$
$$k_3 = hf(x_n + h, y_n + k_3),$$

This method is can be used to solve the equation (A) a s follows,

Equation (7) rewritten as

$dv/dt = F(t, u, \dot{u})$ with $v(0)=0$

$k_{1,v} = hf(t, u(t), \dot{u}(t))$

$= h(-\ddot{u}_g(t) - 2\zeta w_n \dot{u}(t) - w_n^2 u(t))$

$k_{2,v} = hf(t, u(t) + h/2, \dot{u}(t)+k_{1,v}/2)$

$= h[-\ddot{u}_g(t) - 2\zeta w_n(\dot{u}(t)+k_{1,v}/2) - w_n^2(u(t)+h/2)]$

$k_{3,v} = hf(t, u(t) + h/2, \dot{u}(t)+k_{2,v}/2)$

$= h[-\ddot{u}_g(t) - 2\zeta w_n(\dot{u}(t)+k_{2,v}/2) - w_n^2(u(t)+h/2)]$

$k_{4,v} = hf(t, u(t) + h, \dot{u}(t)+k_{3,v})$

$= h[-\ddot{u}_g(t) - 2\zeta w_n(\dot{u}(t)+k_{3,v}) - w_n^2(u(t)+h)]$

$v_{(t+h)} = v_{(t)} + 1/6(k_{1,v} + 2k_{2,v} + 2k_{3,v} + k_{4,v})$

Equation (6) can be rewritten as

$du/dt = v$ with $u(0)=0$

$k_{1,u} = h\dot{u}(t)$

$k_{2,u} = h(\dot{u}(t) + k_{1,u}/2)$

$k_{3,u} = h(\dot{u}(t) + k_{2,u}/2)$

$k_{4,u} = h(\dot{u}(t) + k_{3,u})$

$u_{(t+h)} = u_{(t)} + 1/6(k_{1,u} + 2\, k_{2,u} + 2k_{3,u} + k_4)$

Runge-Kutta-Fehlberg Method

One way to determine whether the Runge-Kutta values are sufficiently accurate is to recompute the value at the end of each interval with the step size halved. If only a slight change in the value of y_{n+1} occurs, the results are accepted; if not, the step must be halved again until the results are satisfactory. This procedure is very expensive, however, For instance, to implement this way, we would need an additional seven function evaluation to determine the accuracy of our y_{n+1}. The best case then would require 4 + 7 = 11 function evaluations to go from (x_n, y_n) to (x_{n+1}, y_{n+1}).

A different approach evaluation uses two Runge-Kutta methods of different orders. For instance, we could use one fourth-order and one fifth-order method to move from (x_n, y_n) to (x_{n+1}, y_{n+1}). We could then compare our results at y_{n+1}. The Runge-Kutta-Fehlberg method, now one of the most popular of these methods, does just this. Only six functional evaluations (versus eleven) are required.

$$k_1 = hf(x_n, y_n)$$

$$k_2 = hf\left(x_n + \frac{h}{4}, y_n + \frac{k_1}{4}\right),$$

$$k_3 = hf\left(x_n + \frac{3h}{8}, y_n + \frac{3k_1}{32} + \frac{9k_2}{32}\right),$$

$$k_4 = hf\left(x_n + \frac{12h}{13}, y_n + \frac{1932k_1}{2197} - \frac{7200k_2}{2197} + \frac{7296k_3}{2197}\right),$$

$$k_5 = hf\left(x_n + h, y_n + \frac{439k_1}{216} - 8k_2 + \frac{3680k_3}{513} - \frac{845k_4}{4104}\right),$$

$$k_6 = hf\left(x_n + \frac{h}{2}, y_n + \frac{8k_1}{27} - 2k_2 + \frac{3544k_3}{2565} + \frac{1859k_4}{4104} - \frac{11k_5}{40}\right);$$

$$y_{n+1} = y_n + \left(\frac{16k_1}{135} + \frac{6656k_3}{12825} + \frac{28561k_4}{56430} - \frac{9k_5}{50} + \frac{2k_6}{55}\right)$$

This method is can be used to solve the equation (A) a s follows,

Equation (7) rewritten as

$dv/dt = F(t, u, \dot{u})$ with $v(0)=0$

$k_{1,v} = hf(t, u(t), \dot{u}(t))$

$= h(-\ddot{u}_g(t) - 2\zeta w_n \dot{u}(t) - w_n^2 u(t))$

$k_{2,v} = hF(t, u(t)+h/4, \dot{u}(t)+k_{1,v}/4)$

$= h(-\ddot{u}_g(t) - 2\zeta w_{n(}\dot{u}(t)+k_{1,v}/4) - w_n^2(u(t)+h/4))$

$k_{3,v} = hF(t, u(t)+3h/8, \dot{u}(t)+3k_{1,v}/32+9k_{2,}v/32)$

$= h(-\ddot{u}_g(t) - 2\zeta w_{n(}\dot{u}(t)+3k_{1,v}/32+9k_{2,}v/32) - w_n^2(u(t)+ 3h/8))$

$k_{4,v} = hF(t, u(t)+12h/13, \dot{u}(t)+1932k_{1,v}/2197-7200k_{2,}v/2197+7296k_{3,v}/2197)$

$=h(-\ddot{u}_g(t) - 2\zeta w_{n(}\dot{u}(t)+1932k_{1,v}/2197-7200k_{2,}v/2197+7296k_{3,v}/2197) - w_n^2(u(t)+ 12h/13))$

$k_{5,v} = hF(t, u(t)+h, \dot{u}(t)+439k_{1,v}/216-8k_{2,}v+3680k_{3,v}/513-845k_{4,v}/4104)$

$=h(-\ddot{u}_g(t) - 2\zeta w_{n(}\dot{u}(t)+ 439k_{1,v}/216-8k_{2,}v+3680k_{3,v}/513-845k_{4,v}/4104) - w_n^2(u(t)+ h))$

$k_{6,v} = hF(t, u(t)+h/2, \dot{u}(t)+8k_{1,v}/27-2k_{2,}v+3554k_{3,v}/2565-1859k_{4,v}/4104-11k_{5,v}/40)$

$=h(-\ddot{u}_g(t) - 2\zeta w_{n(}\dot{u}(t8k_{1,v}/27-2k_{2,}v+3554k_{3,v}/2565-1859k_{4,v}/4104-11k_{5,v}/40) - w_n^2(u(t)+ h/2))$

$$v_{(t+h)} = v_{(t)} + \left(\frac{16k_{1,v}}{135} + \frac{6656k_{3,v}}{12825} + \frac{28561k_{1,v}}{56430} - \frac{9k_{1,v}}{50} + \frac{2k_{1,v}}{55} \right)$$

For equation (6)

$du/dt = v$ with $u(0)=0$

$k_{1,u} = h\dot{u}(t)$

$k_{2,u} = h(\dot{u}(t)+k_{1,u}/4)$

$k_{3,u} = h(\dot{u}(t)+3k_{1,u}/32+9k_{2,u}/32)$

$k_{4,u} = h(\dot{u}(t)+1932k_{1,u}/2197-7200k_{2,u}/2197+7296k_{3,u}/2197)$

$k_{5,u} = h(\dot{u}(t)+439k_{1,u}/216-8k_{2,u}+3680k_{3,u}/513-845k_{4,u}/4104)$

$k_{6,u} = h\dot{u}(t)-8k_{1,u}/27+2k_{2,u}-3544k_{3,u}/2565+1859k_{4,u}/4104-11k_{5,u}/40)$

$$u_{(t+h)} = u_{(t)} + \left(\frac{16k_{1,u}}{135} + \frac{6656k_{3,u}}{12825} + \frac{28561k_{1,u}}{56430} - \frac{9k_{1,u}}{50} + \frac{2k_{1,u}}{55} \right)$$

Adams-Moulton Method

The Adams-Moulton method that does not have the same stability problem as the Milne method, but it is about as efficient. It also assumes a set of starting values already calculated by some other technique. Here we take a cubic through four points, from x_{n-3} to x_n, and integrate over one step, from $_n$ to x_{n+1}. We assume that $dy/dx=f(x, y)$.

Thus by getting the formula for the cubic interpolating polynomial by known methods and then integrating from $_n$ to x_{n+1}, we get the Adams-Moulton *predictor formula:*

Predictor:

$$y_{n+1} = y_n + h/24(55f_n - 59f_{n-1} + 37f_{n-2} - 9f_{n-3}) \qquad -(8)$$

After we have computed a tentative value for f at x_{n+1}, we use it together with the *f*-values at x_n, x_{n-1}, and x_{n-2} to construct another cubic polynomial that we integrate from x_n to x_{n+1}. This gives the Adams-Moulton *corrector formula:*

Corrector:

$$y_{n+1} = y_n + h/24(9f_{n+1} + 19f_n - 5f_{n-1} + f_{n-2}) \qquad -(9)$$

Substituting the values of velocity and displacement in equations (8) and (9), we can find the value of $v_{(t+h)}$ and $u_{(t+h)}$.

Method Based on Interpolation of Excitation

A highly efficient numerical procedure can be developed for linear systems by interpolating the excitation over each tme interval and developing the exact solution. If the time intervals are short, linear interpolation is satisfactory. The solution of equation for $h=t_{i+1}-t_i$ is given by

$$u = e^{-\xi w(t-t_i)}\left[C_1 \sin(w\sqrt{1-\xi^2}\,(t-t_1)) + C_2 \cos(w\sqrt{1-\xi^2}\,(t-t_1))\right] - \frac{a_i}{w^2} + \frac{2\xi\Delta a_i}{w^3 h} - \frac{1}{w^2}\frac{\Delta a_i}{h}(t-t_i)$$

in which C_1 and C_2 are constants of integration. Setting $u=u_i$ and $\mathring{u}=\mathring{u}_i$ at $t=t_i$ and solving for C_1 and C_2, it can be shown that

$$C_1 = \frac{1}{w\sqrt{1-\xi^2}}\left(\xi w u_i + \dot{u}_i - \frac{2\xi^2 - 1}{w^2}\frac{\Delta a_i}{h} + \frac{\xi a_i}{w}\right)$$

$$C_2 = u_i - \frac{2\xi\Delta a_i}{w^3 h} + \frac{a_i}{w^2}$$

Substituting these values of C_1 and C_2 into the equation will show that $\mathring{u}$ at $t=t_{i+1}$ are given by

$$\bar{u}_{i+1} = A(\xi, w, h)\bar{u}_i + B(\xi, w, h)\bar{a}_i$$

in which

$$\bar{u}_i = \begin{Bmatrix} u_i \\ \dot{u}_i \end{Bmatrix}$$

$$\bar{a}_i = \begin{Bmatrix} a_i \\ a_{i+1} \end{Bmatrix}$$

$$A = \begin{bmatrix} a_{11} a_{12} \\ a_{21} a_{22} \end{bmatrix}$$

$$B = \begin{bmatrix} b_{11} b_{12} \\ b_{21} b_{22} \end{bmatrix}$$

The elements of matrices A and B are given by

$$a_{11} = e^{-\xi wh}\left(\frac{\xi}{\sqrt{1-\xi^2}}\sin\left(w\sqrt{1-\xi^2}h\right) + \cos\left(w\sqrt{1-\xi^2}h\right)\right)$$

$$a_{12} = \frac{e^{-\xi wh}}{w\sqrt{1-\xi^2}}\sin\left(w\sqrt{1-\xi^2}h\right)$$

$$a_{21} = -\frac{w}{\sqrt{1-\xi^2}}e^{-\xi wh}\sin\left(w\sqrt{1-\xi^2}h\right)$$

$$a_{22} = e^{-\xi wh}\left(\cos\left(w\sqrt{1-\xi^2}h\right) - \frac{\xi}{\sqrt{1-\xi^2}}\sin\left(w\sqrt{1-\xi^2}h\right)\right)$$

$$b_{11} = e^{-\xi wh}\left[\left(\frac{2\xi^2-1}{w^2h} + \frac{\xi}{w}\right)\frac{\sin\left(w\sqrt{1-\xi^2}h\right)}{w\sqrt{1-\xi^2}} + \left(\frac{2\xi}{w^3h} + \frac{1}{w^2}\right)\cos\left(w\sqrt{1-\xi^2}h\right)\right] - \frac{2\xi}{w^3h}$$

$$b_{12} = e^{-\xi wh}\left[\left(\frac{2\xi^2-1}{w^2h}\right)\frac{\sin\left(w\sqrt{1-\xi^2}h\right)}{w\sqrt{1-\xi^2}} + \left(\frac{2\xi}{w^3h}\right)\cos\left(w\sqrt{1-\xi^2}h\right)\right] - \frac{1}{w^2} + \frac{2\xi}{w^3h}$$

$$b_{21} = e^{-\xi wh}\left[\left(\frac{2\xi^2-1}{w^2h} + \frac{\xi}{w}\right)\left(\cos\left(w\sqrt{1-\xi^2}h\right) - \frac{\xi}{\sqrt{1-\xi^2}}\sin\left(w\sqrt{1-\xi^2}h\right)\right) - \left(\frac{2\xi}{w^3h} + \frac{1}{w^2}\right)\left(w\sqrt{1-\xi^2}\sin\left(w\sqrt{1-\xi^2}h\right) + \xi w\cos\left(w\sqrt{1-\xi^2}h\right)\right)\right] + \frac{1}{w^2h}$$

$$b_{22} = -e^{-\xi wh}\left[\left(\frac{2\xi^2-1}{w^2h}\right)\left(\cos\left(w\sqrt{1-\xi^2}h\right) - \frac{\xi}{\sqrt{1-\xi^2}}\sin\left(w\sqrt{1-\xi^2}h\right)\right) - \left(\frac{2\xi}{w^3h}\right)\left(w\sqrt{1-\xi^2}\sin\left(w\sqrt{1-\xi^2}h\right) + \xi w\cos\left(w\sqrt{1-\xi^2}h\right)\right)\right] + \frac{1}{w^2h}$$

From the equation, it follows that he absolute acceleration, $\check{z}_i$, of the mass at time t_i is given by

$$\check{z}_i = \ddot{u}_i + a_i = -(2\zeta w \mathring{u} + w^2 u_i)$$

Hence the displacement and velocity of the oscillator are known at some time t_o, the stae of the oscillator at all subsequent times t_i can be computed exactly by step by step application of equations. The computational advantage of this approach lies ih the fact that A and B depends only on ζ, w, and h. ζ and w are constant during the calculation of each spectrum values, and if h is constant also, u_i, $\mathring{u}_i$ and $\check{z}_i$ can be evaluated by the execution of only ten multiplication operations for each step of integration. Matrices A and B, defined by the rather complicated expressions need to be evaluated only at the beginning of each spectrum calculation. If varying time intervals are used, it is necessary, in general, to complete A and B at each step of integration.

CHAPTER – 6

DEVELOPMENT OF SOFTWARE

Having developed the method to solve the differential equation governing the earthquake motion, there has to be a way by which the method can be tested to check whether it gives us satisfactory results. For this reason, software has also been developed as a part of the thesis to test those methods. But for the software to run, we have to have real time input data, which is the earthquake data. It is the earthquake accellogram record or the ground motion record.

Given the ground motion $\ddot{u}_g(t)$, the response spectrum can be developed by the implementation of the following steps:

1. Numerically define the ground acceleration $\ddot{u}_g(t)$; typically, the ground motion ordinates are defined every 0.02 seconds.
2. Select the natural vibration period T_n and damping ratio ζ of an SDF system.
3. Compute the deformation response $u(t)$ of this SDF system due to ground motion $\ddot{u}_g(t)$ by any of the numerical methods.
4. Determine u_o, the peak value of $u(t)$.
5. The spectral ordinates are $D = u_o$, $V = (2\pi/T_n)D$, and $A = (2\pi/T_n)^2 D$.
6. Repeat steps 2 to 5 for a range of T_n and ζ values covering all possible systems of engineering interest.
7. Present the results of steps 2 to 6 graphically to produce the response spectra.

Considerable computational effort is required to generate an earthquake response spectrum. A complete dynamic analysis to determine the time variation (or history) of the deformation of an SDF system provides the data for one point on the spectrum corresponding to the T_n and ζ of the system.

The input record for testing is the north-south component of the ground motion recorded at a site in El Centro, California during the Imperial Valley earthquake of May 18, 1940.

Numerical values for the ground acceleration in units of g, the acceleration due to gravity are given as the input. This includes 1559 data points at equal time spacing of 0.02 seconds, the first value at *t=0.02* sec; acceleration at *t=0* is zero.

The software is done in Visual Basic 6.0 Programming Language. It is user-friendly software and everything in the software is self-explanatory. In the following paragraphs, I shall explain the basic utilities of the software and the various components and options which are available. Shown below is the snapshot of the Software's screen.

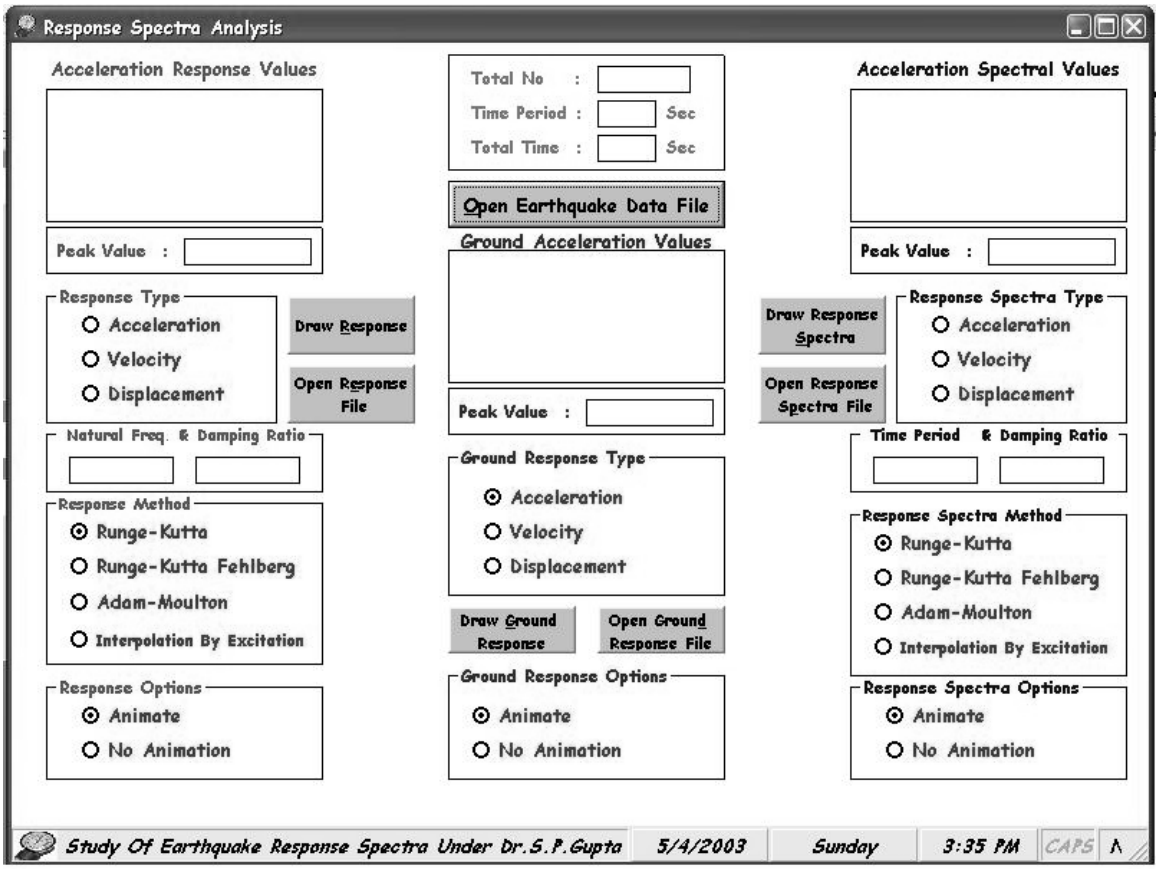

Figure 3 – Opening Screen

In the figure 3, you will note the three columns viz. Ground Response, Response and Response Spectrum. In each column you will see that there are three radio buttons defining the type of response you need to calculate and plot. They are Acceleration, Velocity and Displacement.

For plotting the ground response, you will have to select the type of ground response and then press the "Draw Ground Response" button. It will show a screen with the plot of the ground response.

For calculating the response or the response spectrum, you can choose the method that is to be used while calculating. There is also an option specifying whether you want to draw the graph dynamically or draw at once by selecting the Animate or No Animation option respectively. Let us see each of the options in detail.

Input data can be fed to the program as follows. Click on "Open Earthquake Data File" button. It will open a dialog as shown in figure 4.

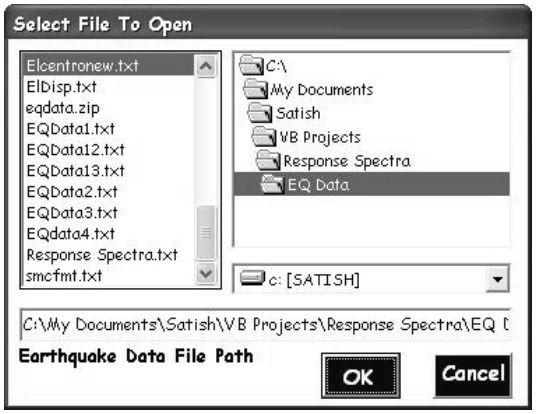

Figure 4 – File Dailog

Just go to the directory where the input Earthquake accellogram record is present and select the file and click on the "OK" Button. Note the middle column of the first screen. It is as shown in figure 5.

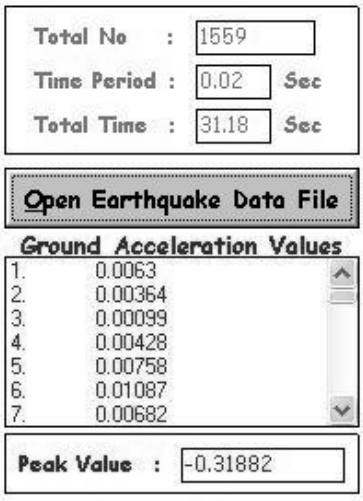

Figure 5 – Earthquake Data

You will see that once the earthquake data has been read, it will display the total number of data points, the time period or the time interval between each data point and the total time. Also the list box will be listed with all the input data points with their corresponding numbers. It also displays the peak value of the data.

6.1 Ground Response

You will also see that the Acceleration option is selected. Clicking on "Draw Ground Response" button will show a screen showing the ground acceleration response of the earthquake as shown in figure 6.

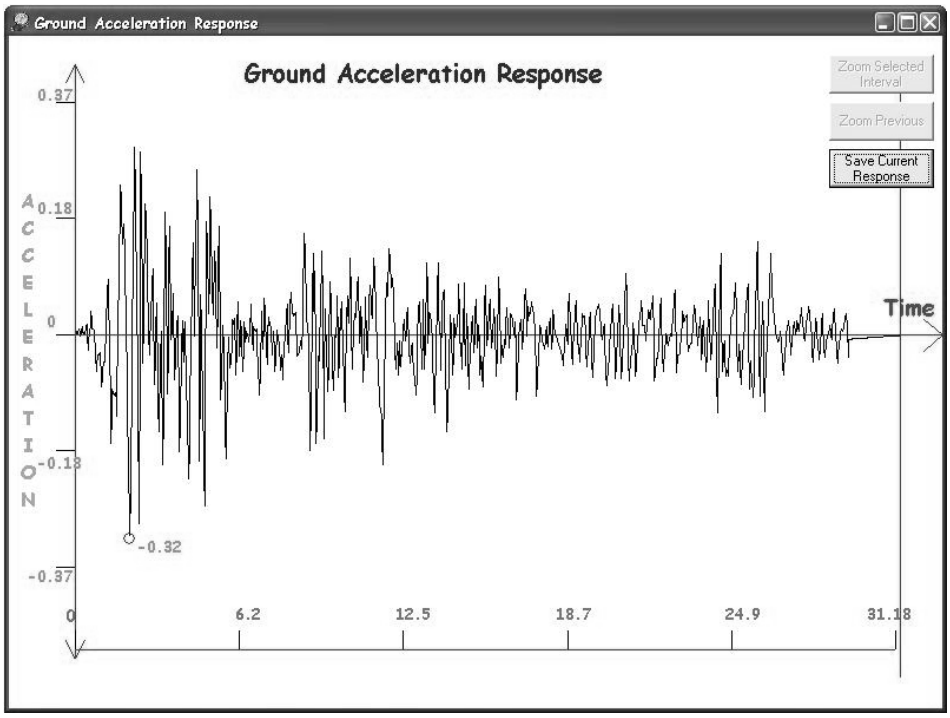

Figure 6 – Ground Acceleration of El Centro Earthquake

The ground response is plotted by trapiziodal method. It is because the ground response does not depend on the value of time period and damping. It is just pure integration. The screen shows the plot of the data with the peak value with time (in seconds) on x-axis and acceleration (in *g* - acceleration due to gravity) on y-axis. You can save this response as a response file and then retrieve it later using this software. You can also select a portion of the response by mouse and can zoom to that interval using the "Zoom Interval" button. You can also save this particular response. Pressing "Zoom Previous" button will bring it

to original response. Figure 7 shows the velocity and figure 8 shows displacement ground response for the earthquake data, which is got when the corresponding ground response options are selected.

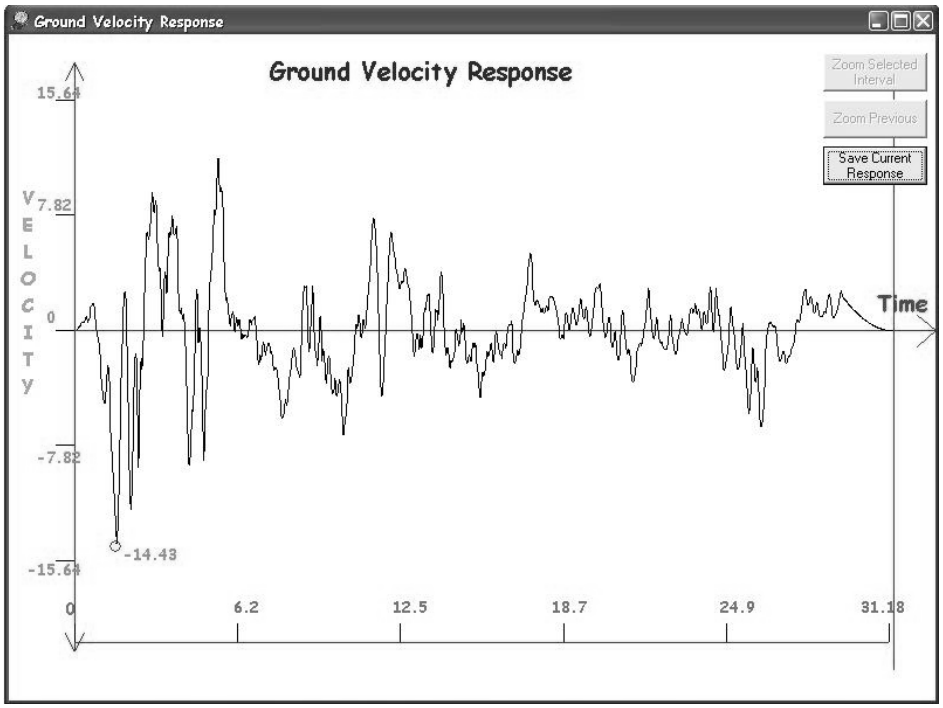

Figure 7 – Ground velocity of El Centro Earthquake

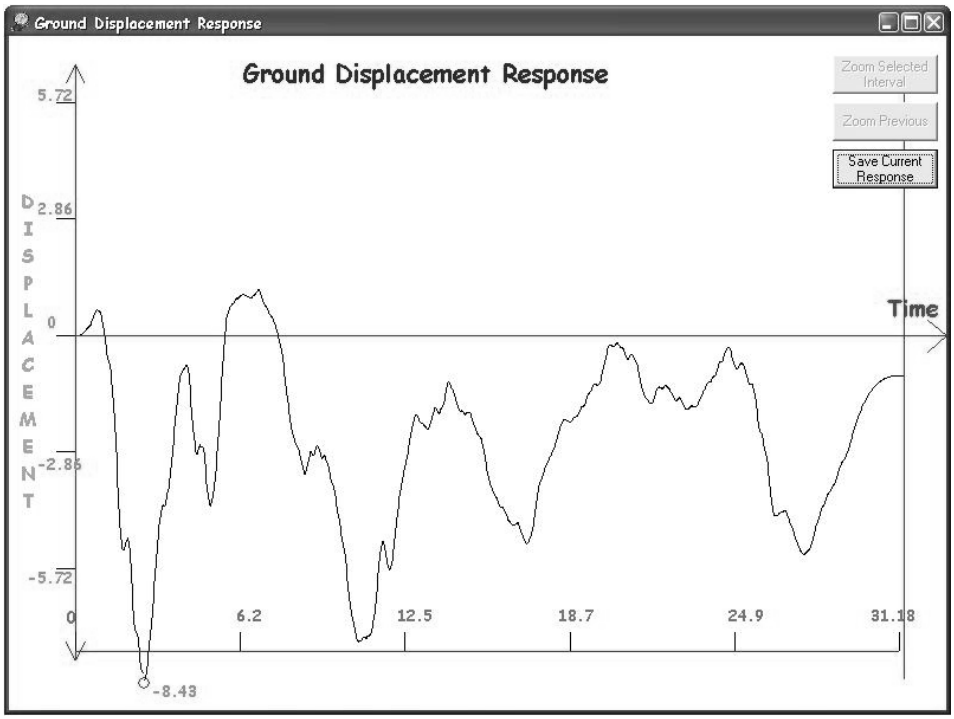

Figure 8 – Ground Displacement of El Centro Earthquake

6.2 Response

To plot a particular response for the data, there are two inputs to be fed in the text box. They are natural frequency and the damping ratio. Then select the method you want to use for finding the response. Then select the type of response you want by clicking on the respective option. Figure 9 shows the Displacement response for Time period = 2 sec and damping ratio ζ = 2% (given as input) using Runge-Kutta Method. You will see that for plotting a particular response, the only variables are the Time Period T_n and damping ratio ζ. Varying both will give us a series of response for the system.

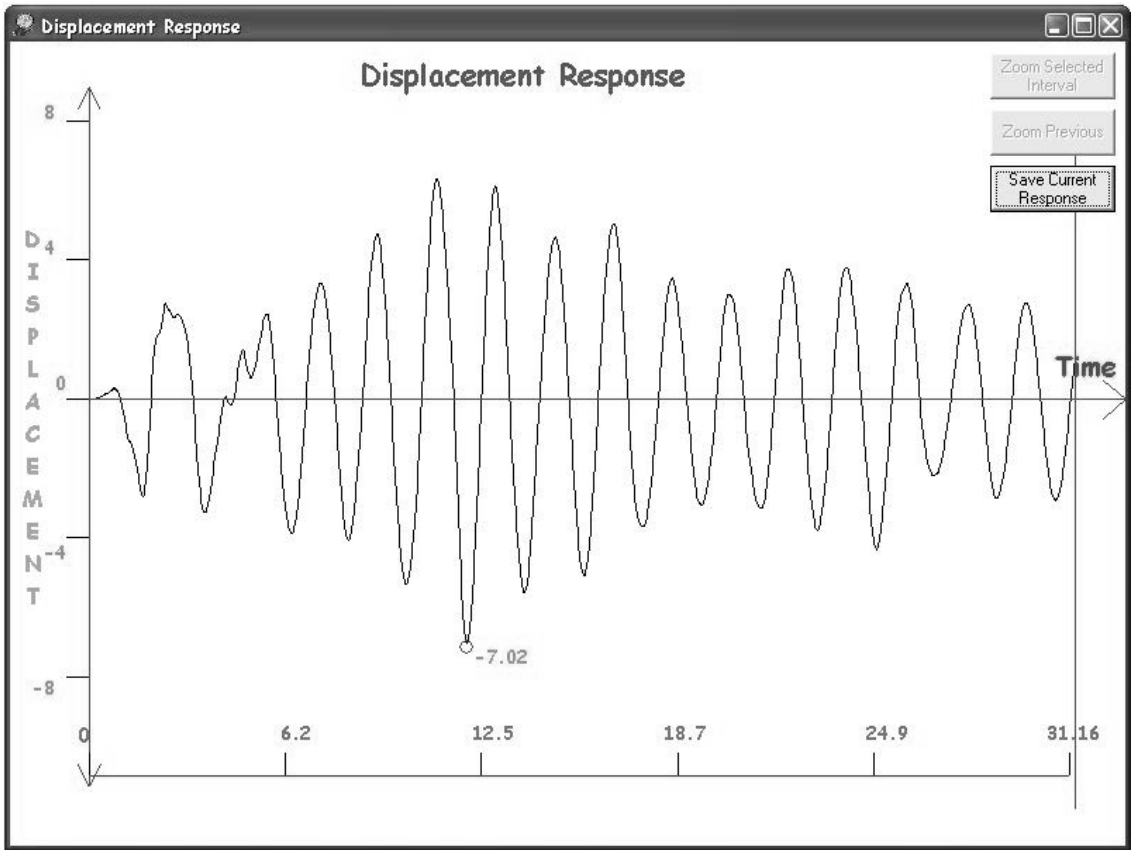

Figure 9 – Displacement Response for El Centro Earthquake with T_n = 2 sec & ζ = 2%

By varying the damping coefficient ζ, the displacement response for Time Period = 2 sec and Damping Ratio ζ = 5% using Runge-Kutta Method is shown in Figure 10.

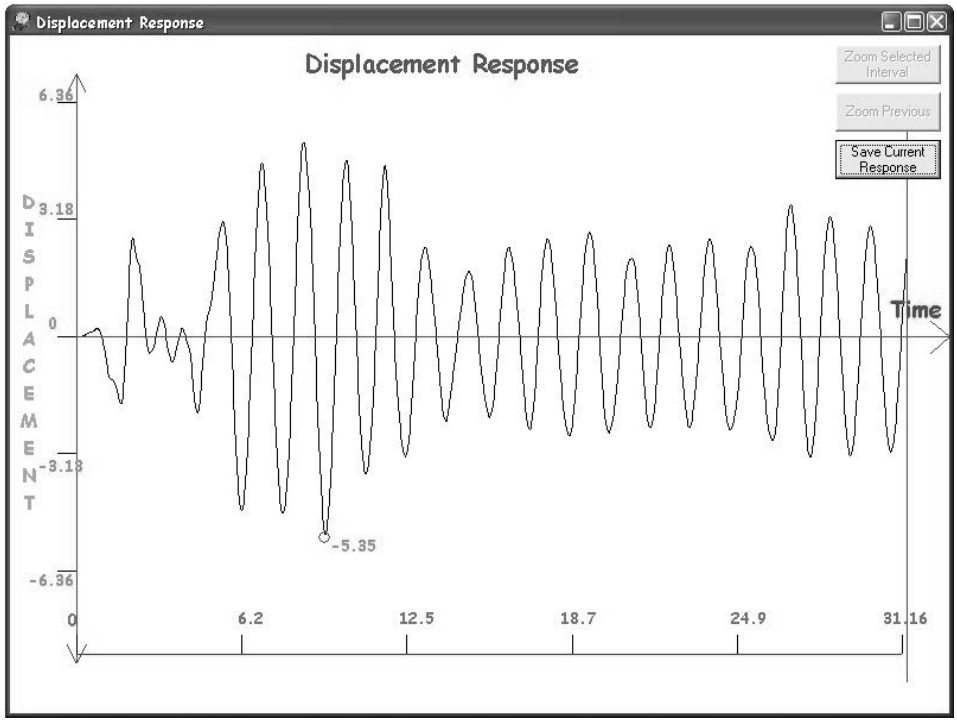

Figure 10 – Displacement Response for El Centro Earthquake with T_n = 2 sec & ζ = 5%

Figure 11 shows the acceleration response for the time period T_n = 0.5 sec and damping ζ = 2% using Runge-Kutta Method.

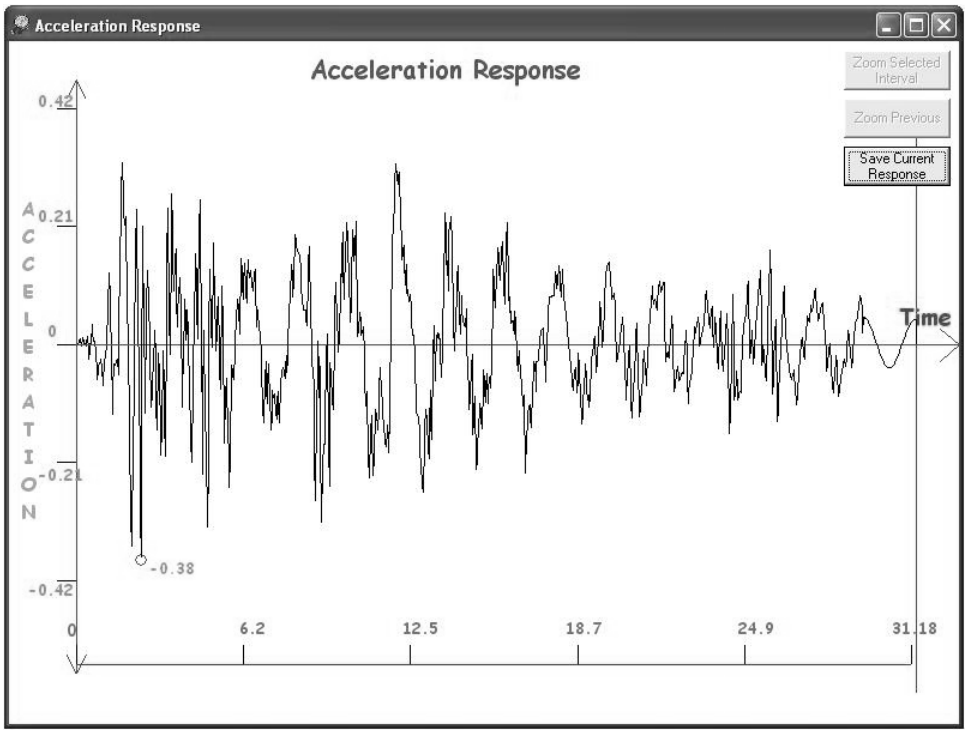

Figure 10 – Acceleration Response for El Centro Earthquake with T_n = 0.5 sec & ζ = 2%

6.3 Response Spectra

To plot response spectra for the data, there are two inputs to be fed in the text box. They are time period for which the spectrum has to be drawn and the damping ratio. Then select the method you want to use for finding the response spectra. Then select the type of response spectra you want by clicking on the respective option. Figure 12 shows the Displacement response spectrum for damping ratio $\zeta = 2\%$ (given as input) using Runge-Kutta Method. The time period is taken as 0.002 sec so that the response spectra will be plotted from 0 to 3.118 seconds. The peak value of the displacement response spectra is 64.37 inches as shown in the Figure 12.

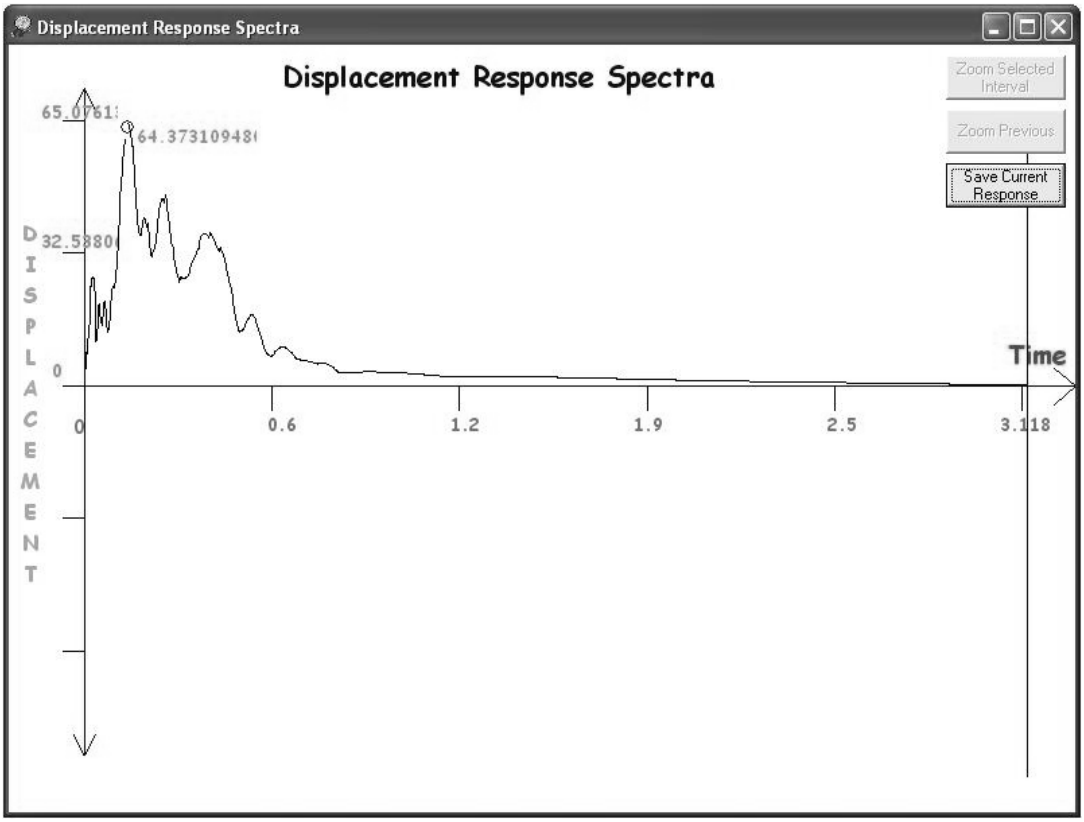

Figure 12 – Displacement Response Spectra for El Centro Earthquake with $\zeta = 2\%$

Similairly, we can draw the other two response spectra by selecting them from the response type options. Figure 13and 14 shows the velocity and acceleration response spectra for damping coefficient $\zeta = 2\%$ for El Centro Earthquake using Runge-Kutta method.

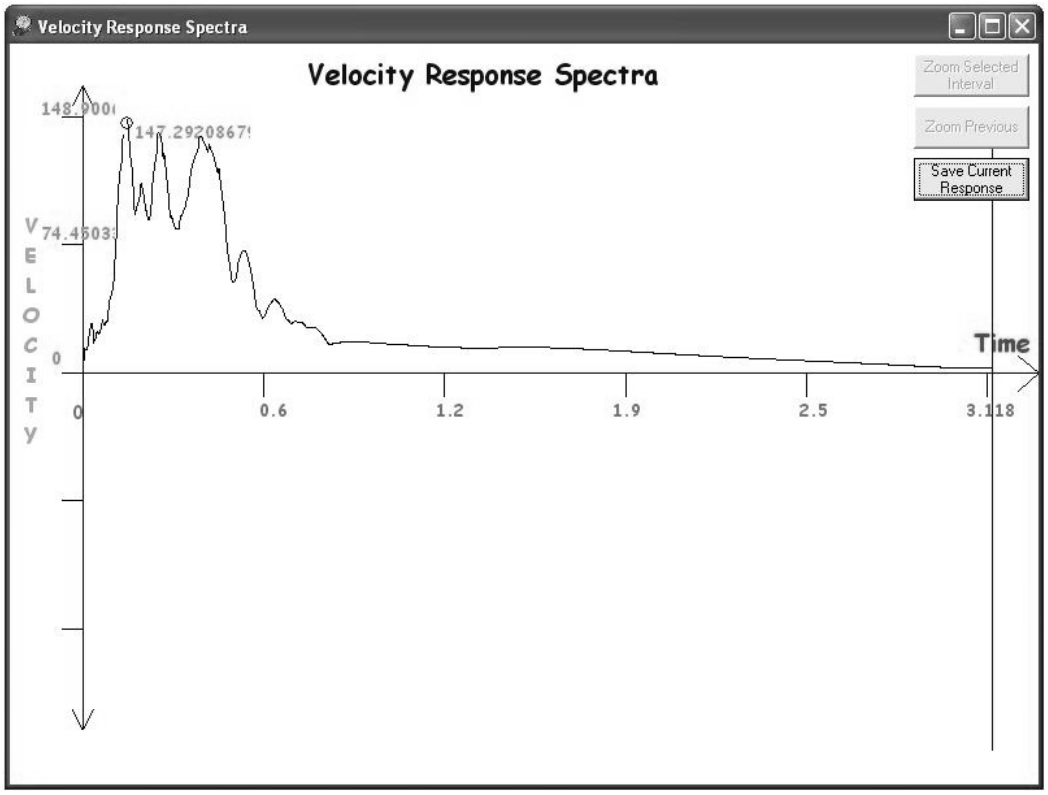

Figure 13 - Velocity Response Spectra for El Centro Earthquake with $\zeta = 2\%$

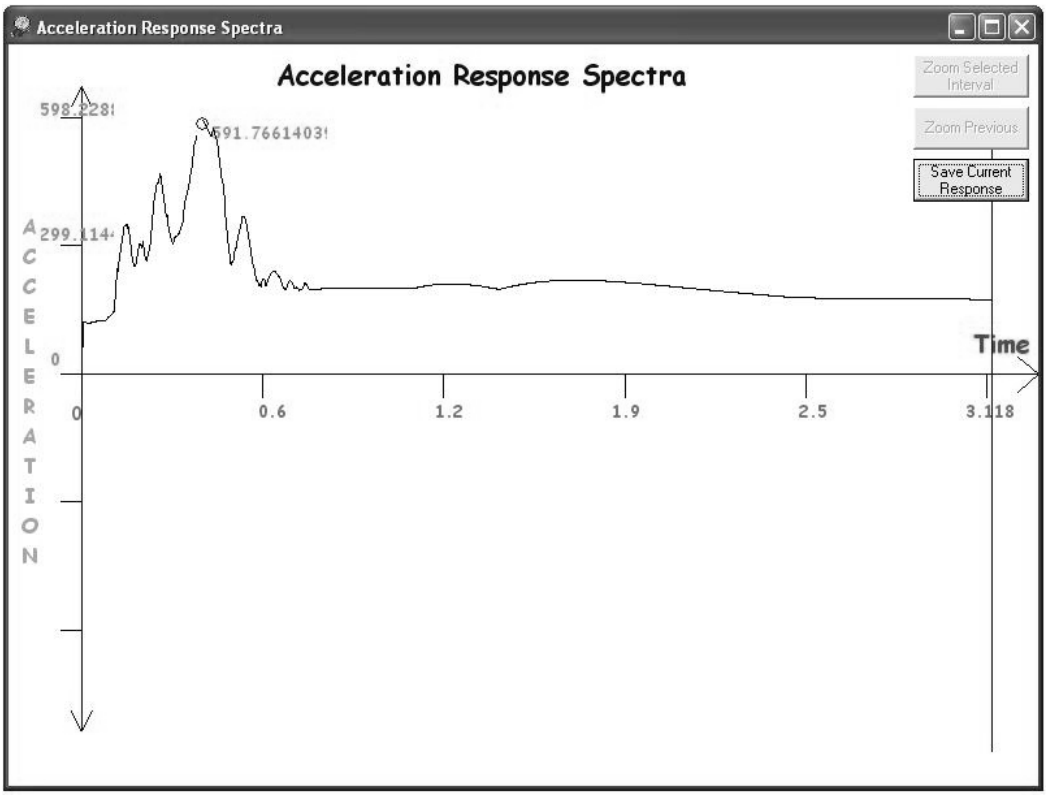

Figure 14 – Acceleration Response Spectra for El Centro Earthquake with $\zeta = 2\%$

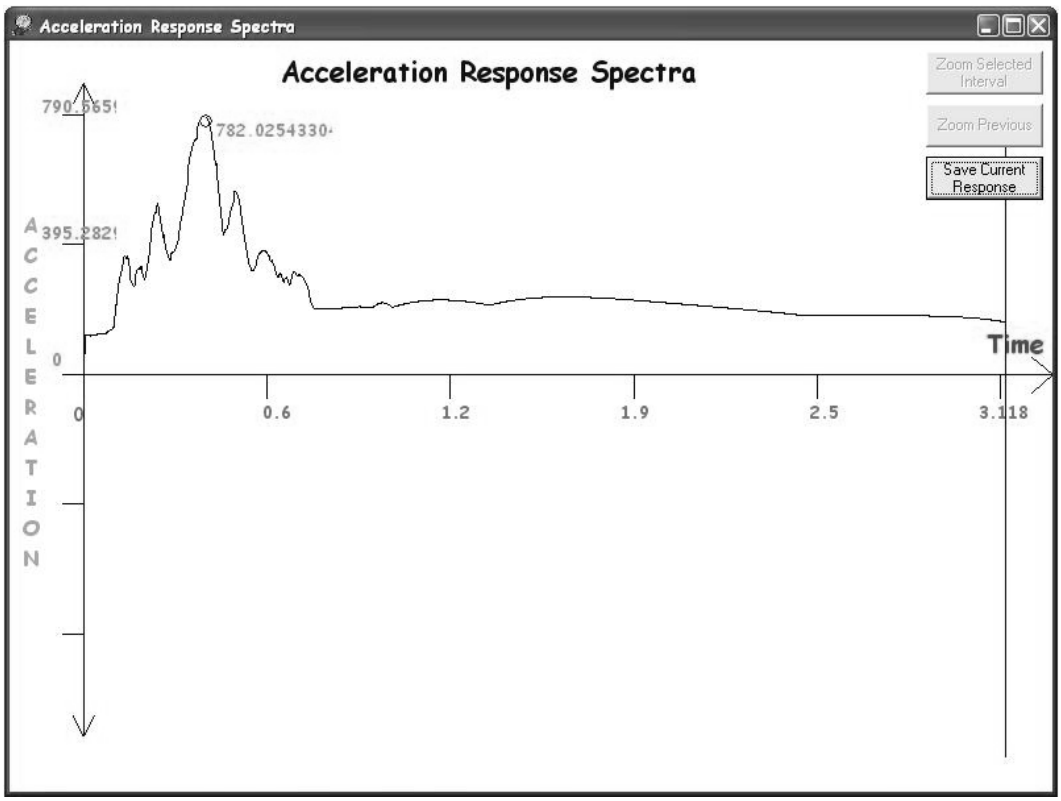

Figure 15 – Acceleration Response Spectra for El Centro Earthquake with $\zeta = 4\%$

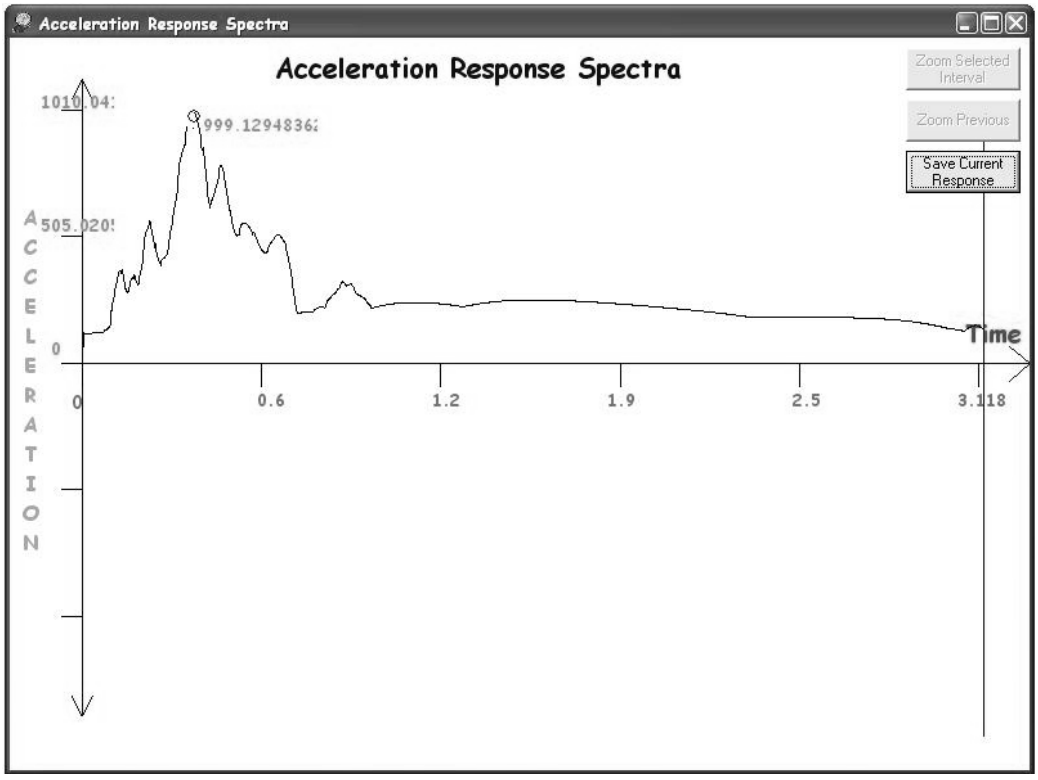

Figure 16 – Acceleration Response Spectra for El Centro Earthquake with $\zeta = 5\%$

CHAPTER – 7

DISCUSSION OF RESULTS

In this chapter, we will be discussing about the four methods that we have applied to draw the response and response spectra by comparing the values abtained by each one of them with the actual values.

Response Results

Let us discuss the first two methods first, as they are more or less the same algorithm. Figure 16 shows the plot of a particular response with Tn = 2 sec and ζ = 2% using both Runge Kutta as well as Runge-Kutta Fehlberg method. Thus plot of response and response spectra was done for the methods and was superimposed one over the other.

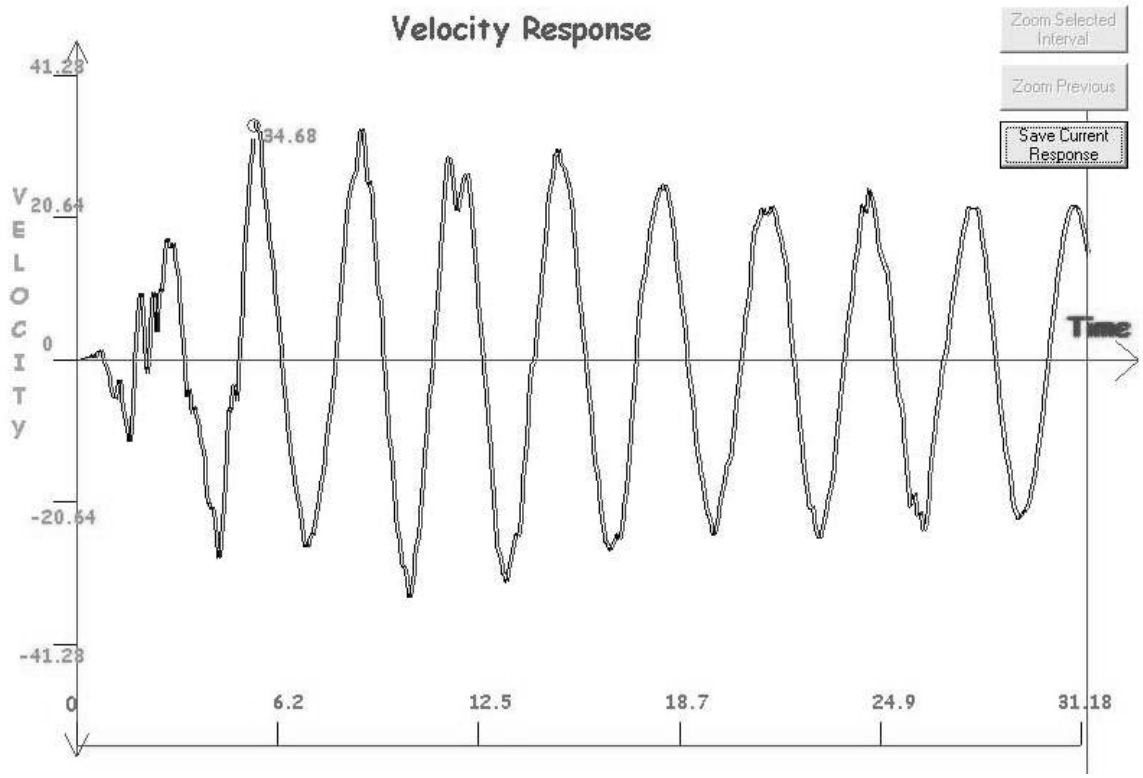

Figure 16 – Comparison between Runge-Kutta and Runge-Kutta-Fehlberg Method

If you look closely in the graph, you will note the two distinct lines representing the methods. Thus it is seen that both Runge-Kutta method and Runge-Kutta Fehlberg methods give close results. The maximum response using Runge-Kutta method is 34.64 while it is 34.68 using Runge-Kutta-Fehlberg method. Now let us try to check the values obtained by the other two methods.

Having compared the four methods, let us check whether the methods are close to the actual values. For that we need the actual response. Known Displacement response for the El Centro earthquake for Time Period = 0.5 sec and Damping Coefficient ζ = 2% is shown in Figure 17 with peak occurring at 2.61 inches.

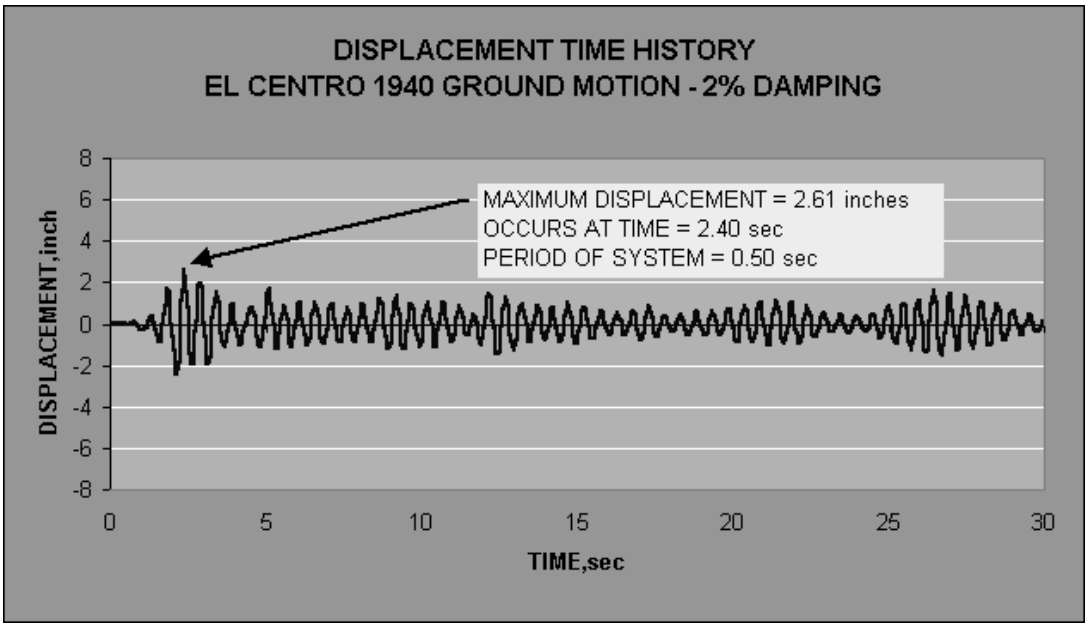

Figure 17 – Actual Displacement Response for El Centro Earthquake for T_n = 0.5 sec and ζ = 2%

Let us try to compare it with all the methods. Figure 18 shows the displacement response drawn using Runge-Kutta method. The peak value obtained is 2.22 inches.

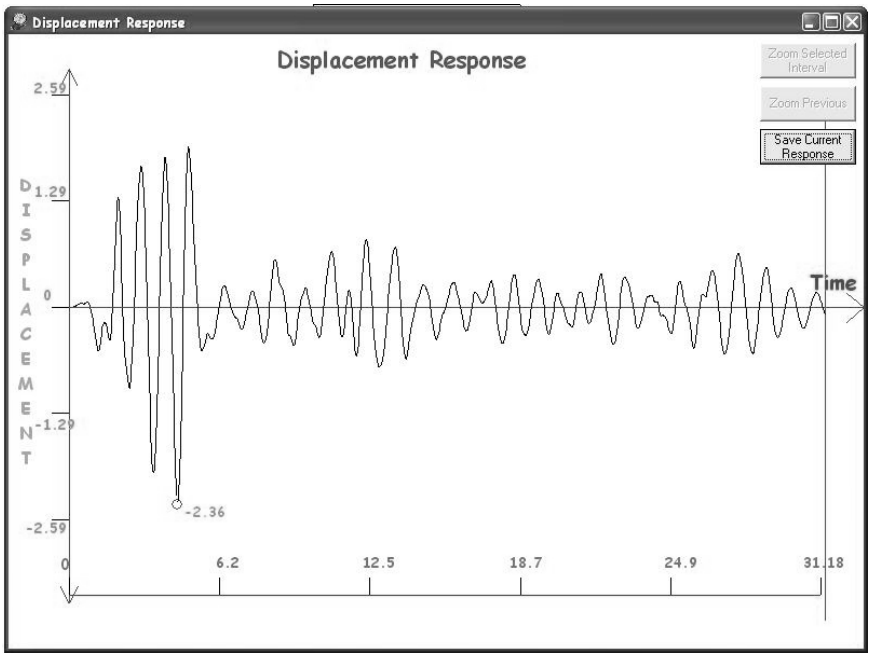

Figure 18 - Displacement Response for El Centro Earthquake for T_n = 0.5 sec and ζ = 2% using Runge-Kutta Method

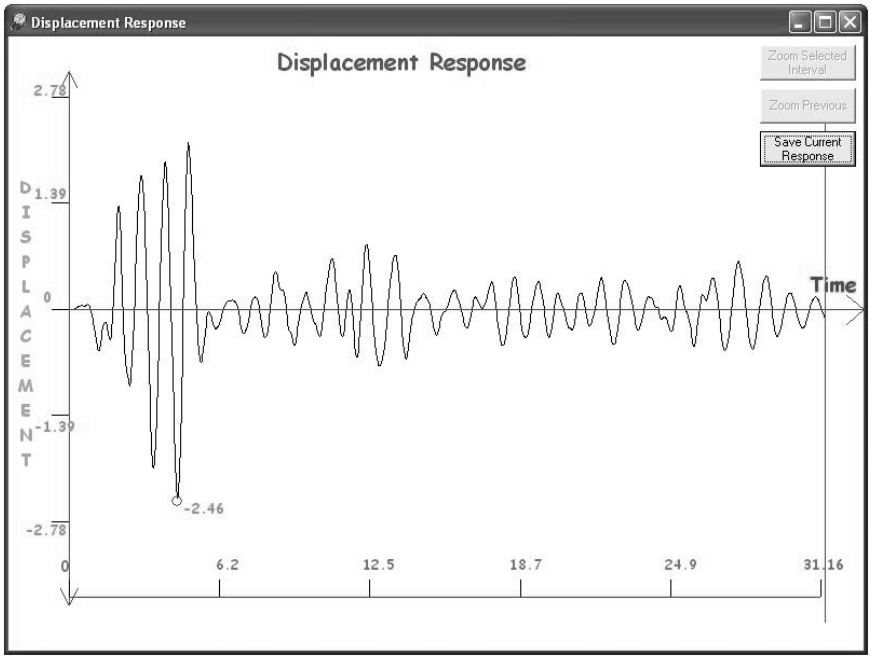

Figure 19 - Displacement Response for El Centro Earthquake for T_n = 0.5 sec and ζ = 2% using Runge-Kutta-Fehlberg Method

Figure 19 shows the Displacement response drawn using Runge-Kutta-Fehlberg method. Using Runge-Kutta-Fehlberg method yields 2.46 inches as the peak value.

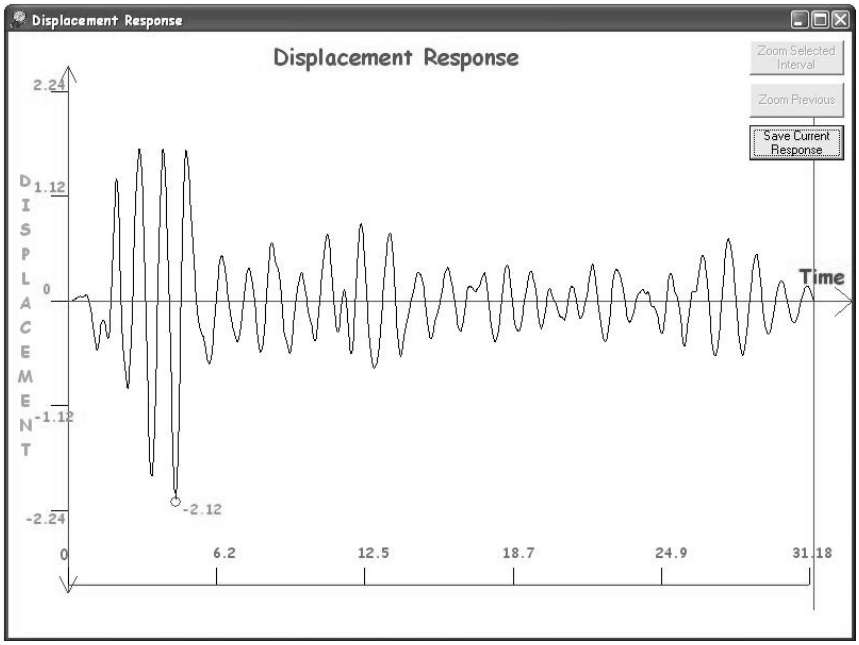

Figure 20 - Displacement Response for El Centro Earthquake for T_n = 0.5 sec and ζ = 2% using Adam-Moulton Method

Figure 20 shows the displacement response using Adam-Moulton method. It gave a peak value of 2.12 inches.

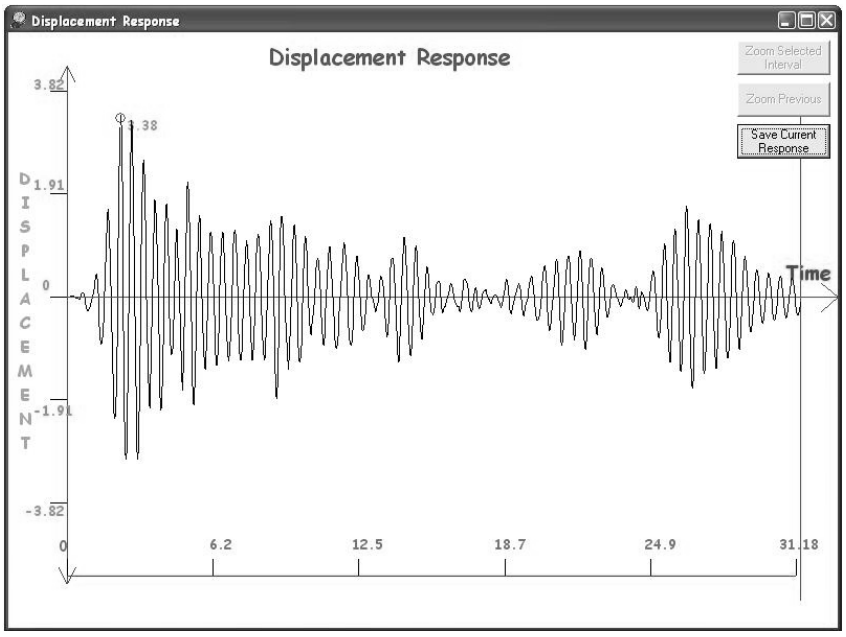

Figure 21 - Displacement Response for El Centro Earthquake for T_n = 0.5 sec and ζ = 2% using Interpolation of Excitation Method

Figure 21 shows the displacement response using Interpolation of Excitation method. It gave a peak value of 3.18 inches.

Superimposing the plots obtained by all the four methods, we obtain the following plot as shown in Figure 22.

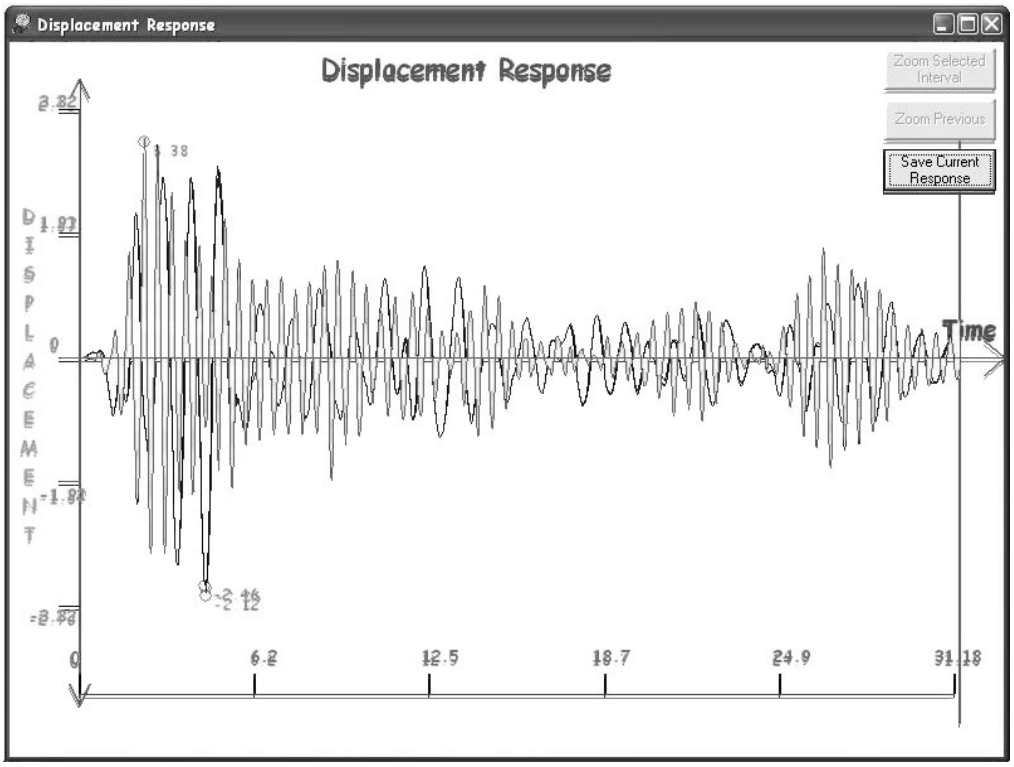

Figure 22 - Displacement Response for El Centro Earthquake for T_n = 0.5 sec and ζ = 2% for All the Four Methods

Comparing the four methods, we see that the plot by interpolation method gives a similair plot as that of the actual values. The other three methods gives more or less the same plot with little or no change in the values. The error obtained in each of the method is as follows

1. Runge-Kutta Method – 15%
2. Runge-Kutta-Fehlberg Method – 5.6%
3. Adam-Moulton Method – 18.1%
4. Interpolation of Excitation – 21%

Figure 23 shows yet another displacement response which is the actual response for the El Centro earthquake for Time Period = 1.59 sec and Damping = 2%. The peak response occurs at 5.89 inches.

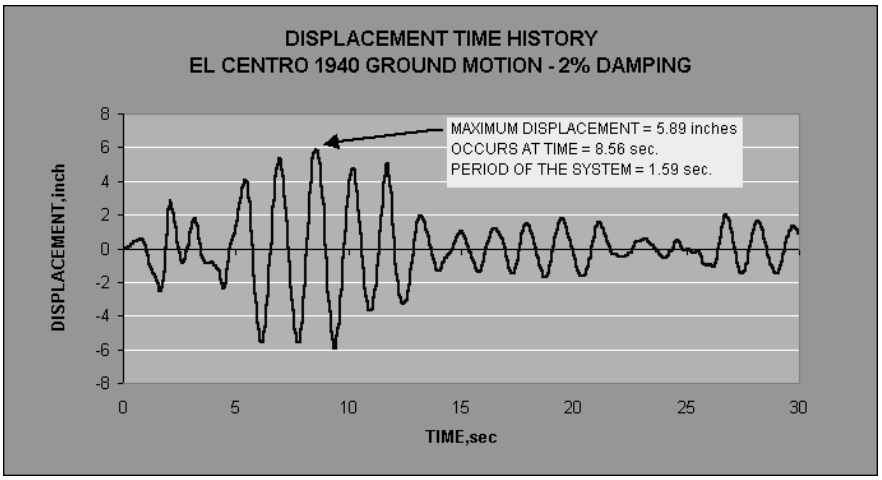

Figure 23– Actual Displacement Response for El Centro Earthquake for T_n = 1.59 sec and ζ = 2%

The response for the same values plotted using the software using Runge-Kutta Method is shown in Figure 24. The peak response occurs at 5.394 inches.

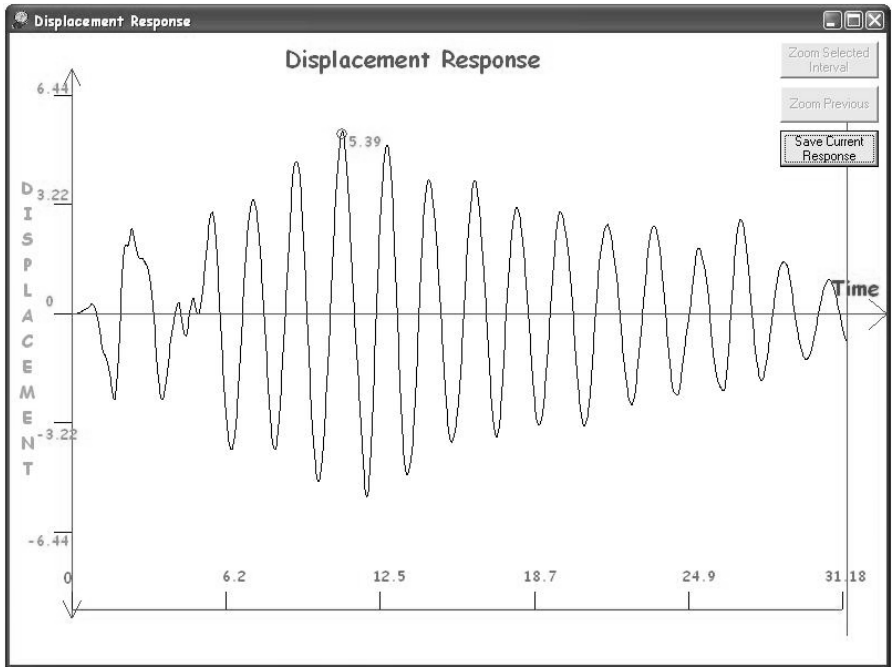

Figure 24 - Displacement Response for El Centro Earthquake for T_n = 0.5 sec and ζ = 2% using Runge-Kutta Method

The response for the same values plotted using the software using Runge-Kutta-Fehlberg Method is shown in Figure 25. The peak response occurs at 5.389 inches.

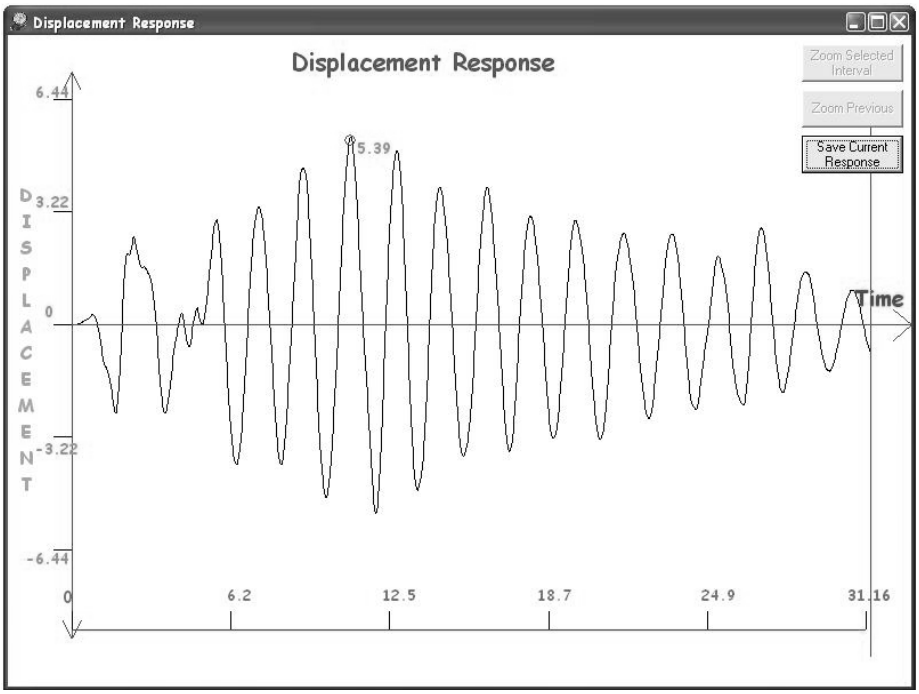

Figure 25 - Displacement Response for El Centro Earthquake for T_n = 0.5 sec and ζ = 2% using Runge-Kutta-Fehlberg Method

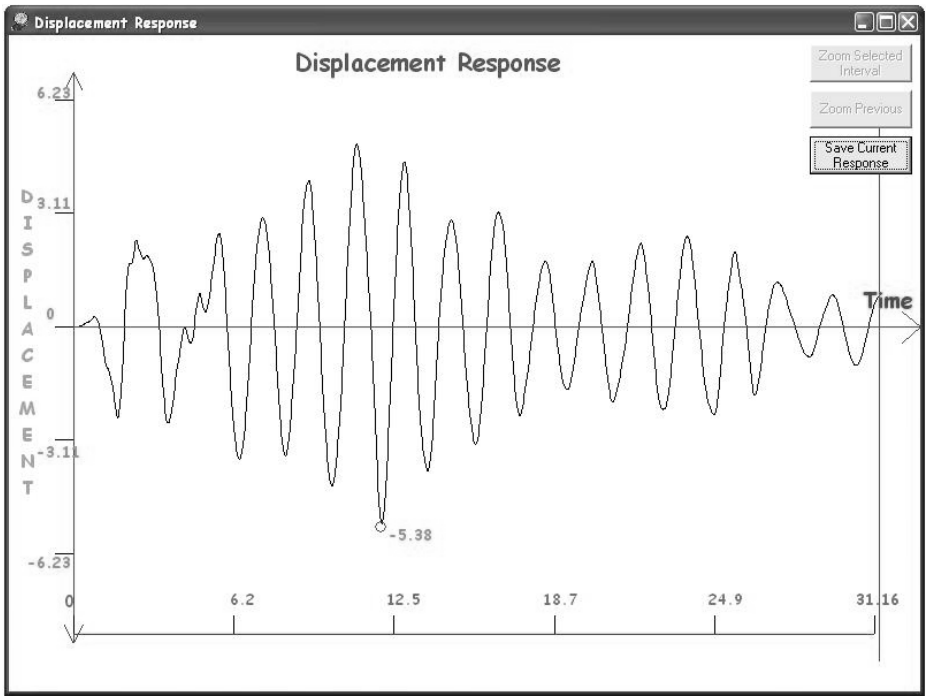

Figure 26 - Displacement Response for El Centro Earthquake for T_n = 0.5 sec and ζ = 2% using Adam-Moulton Method

The response for the same values plotted using the software using Adam-Moulton Method is shown in Figure 26. The peak response occurs at 5.378 inches.
The similair response using Interpolation method is shown in Figure 27. The peak response occurred at 5.23 inches.

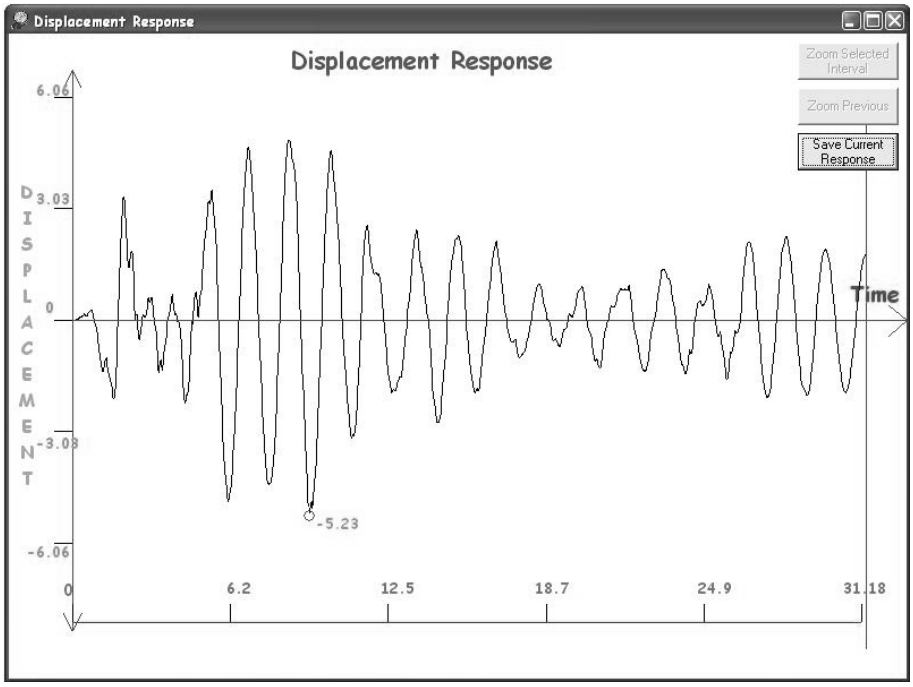

Figure 27 - Displacement Response for El Centro Earthquake for T_n = 0.5 sec and ζ = 2% using Interpolation by Excitation Method

And finally, Figure 28 shows the superimposed plot of all the four methods. You see that Runge-Kutta and Runge-Kutta-Fehlberg has overlapped one over the other and there is no visible difference between the two methods. Adam-Moulton method slightly varied from the Runge-Kutta and Runge-Kutta-Fehlberg method at various points but the variation is not to imminent. But the last method, which is the Interpolation of Excitation method, the response varied to a much greater extent. Comparing it with the actual values of the response, the error obtained in each of the method is as follows

1. Runge-Kutta Method – 8.5%
2. Runge-Kutta-Fehlberg Method – 8.5%
3. Adam-Moulton Method – 8.9%
4. Interpolation of Excitation – 11%

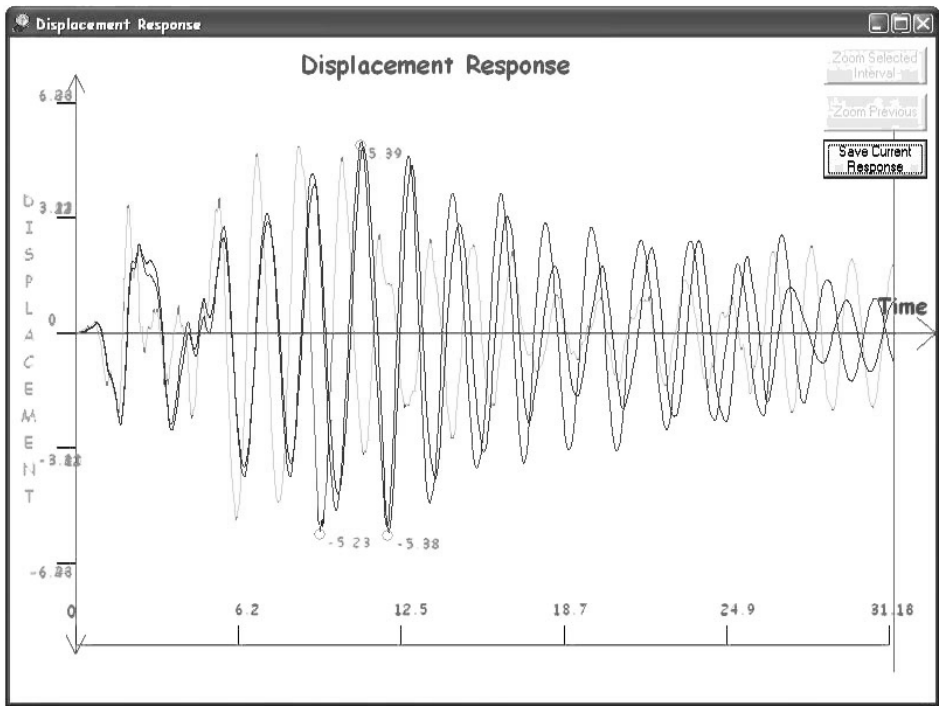

Figure 28 - Displacement Response for El Centro Earthquake for T_n = 0.5 sec and ζ = 2% for all the Four Method

Response Spectra Results

In drawing the response specta for the El Centro Earthquake, we have discussed and compared the four methods with the actual values of the spectra.

Also with known plots of response spectra, the methods can be compared with each other. Figure 29 shows the actual acceleration response spectra for ζ = 2%.

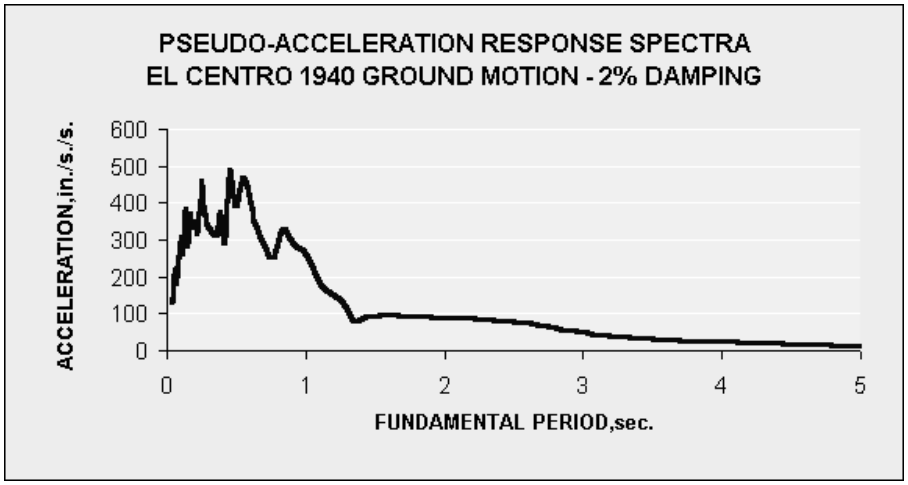

Figure 29 - Actual Acceleration Response Spectra for El Centro Earthquake with ζ = 2%

Figure 30 shows the acceleration response spectra for $\zeta = 2\%$ using Runge-Kutta Method.

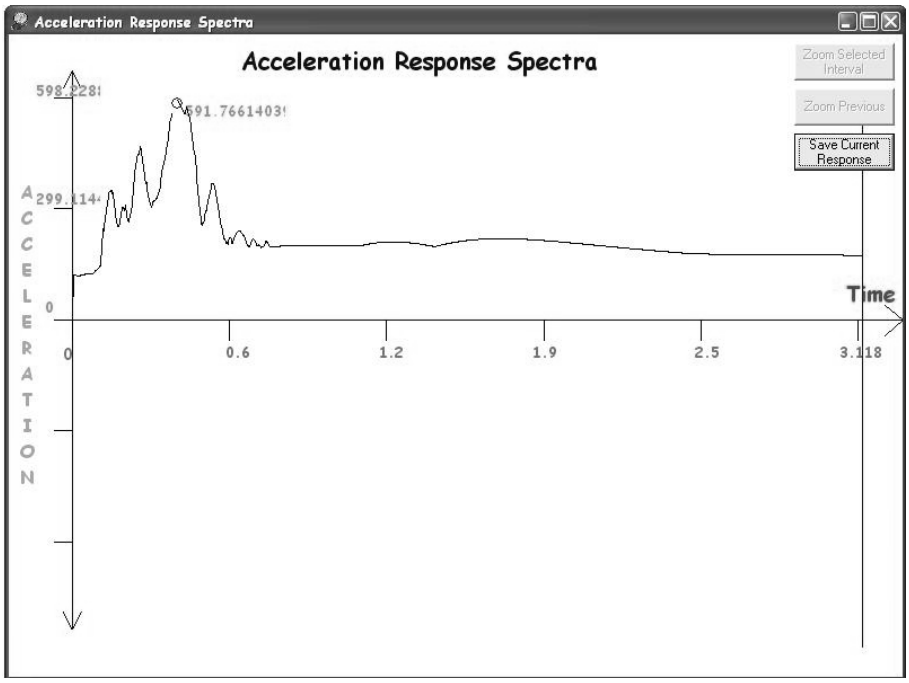

Figure 30 - Acceleration Response Spectra for El Centro Earthquake with $\zeta = 2\%$ using Runge-Kutta Method

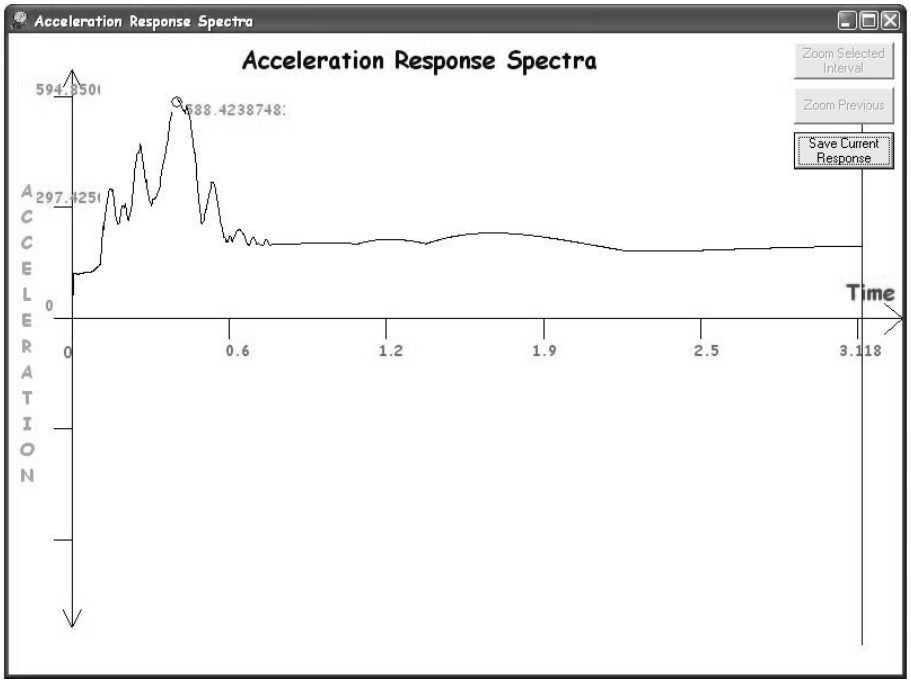

Figure 31 - Acceleration Response Spectra for El Centro Earthquake with $\zeta = 2\%$ using Runge-Kutta Method

Similairly, Figures 32 and 33 show the corresponding acceleration response spectra using Adam-Moulton and Interpolation method respectively.

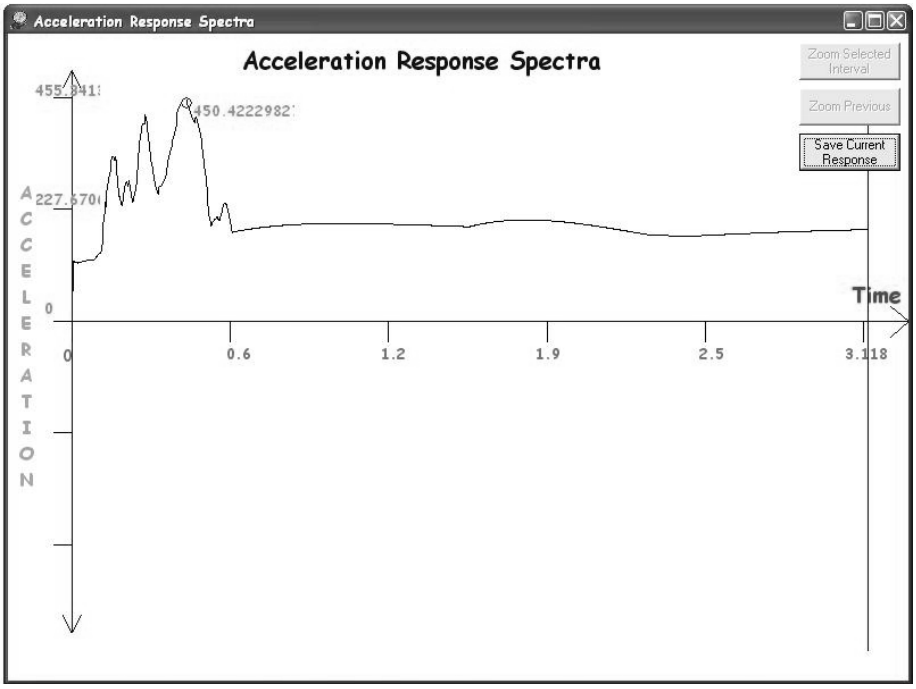

Figure 32 - Acceleration Response Spectra for El Centro Earthquake with $\zeta = 2\%$ using Adam-Moulton Method

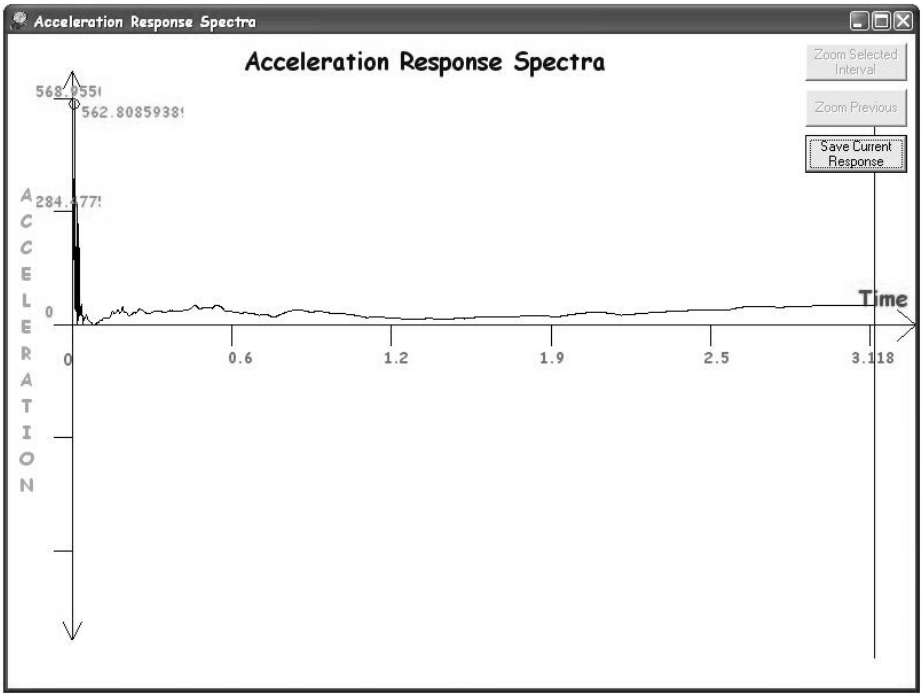

Figure 33 - Acceleration Response Spectra for El Centro Earthquake with $\zeta = 2\%$ using Interpolation by Excitation Method

Figure 34 shows the super-imposed plot of all the four methods.

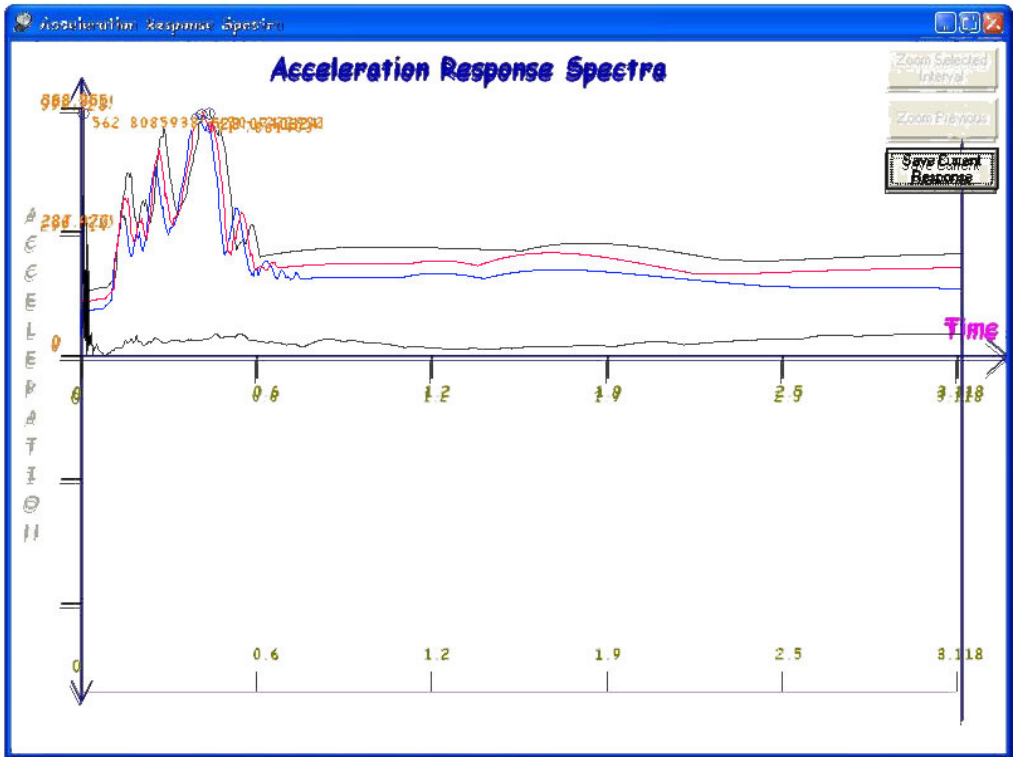

Figure 34 - Acceleration Response Spectra for El Centro Earthquake with $\zeta = 2\%$ for all the Four Methods

As it is seen from the Figure 34, all the methods seemed to give the peak values close to each other, but there is a major difference in the plot using Interpolation of Excitation technique. The other three methods have a uniform difference between each other.

The following table is the summarization of all the results

Method / Response	Runge-Kutta Method	Runge-Kutta-Fehlberg Method	Adam-Moulton Method	Interpolation by Excitation
Displacement Response $T_n = 0.5$ sec and $\zeta = 2\%$	**Peak** = 2.22 in. **Error** = 15%	**Peak** = 2.46 in. **Error** = 5.6%	**Peak** = 2.12 in. **Error** = 18.1%	**Peak** = 3.18 in. **Error** = 21%
Displacement Response $T_n = 2$ sec and $\zeta = 2\%$	**Peak** = 2.46 in. **Error** = 5.6%	**Peak** = 7.68 in. **Error** = 2.73%	**Peak** = 5.45 in. **Error** = 3%	**Peak** = 5.35 in. **Error** =2%
Displacement Response $T_n = 1.5$ sec and $\zeta = 2\%$	**Peak** = 5.394 in. **Error** = 8.5%	**Peak** = 5.389 in. **Error** = 8.5%	**Peak** = 5.378 in. **Error** = 8.9%	**Peak** = 5.23 in. **Error** = 11%
Acceleration Response Spectra $\zeta = 2\%$	**Peak** = 588 in/sec².	**Peak** = 591 in/sec².	**Peak** = 450 in/sec².	**Peak** = 562 in/sec².

From the table tabulating the results obtained for various responses, we can state that

a) Runge-Kutta-Fehlberg Method is the second most preferred of all the methods discussed as it gives more accurate results due to its closeness to the exact values.
b) Runge-Kutta method is also favorable as it also gives almost the same results to that of Runge-Kutta-Fehlberg method.
c) Adam-Mouton method has a reasonable margin of error, which makes it an approximate method to use while calculating the reponse and response spectra.
d) Interpolation of Excitation method not only gives values very close to actual ones but also it is the fastest with respect to computer time taken for plotting the response spectra than the other three methods. Thus we go for this method when we want to draw the spectra having accurate values.

CHAPTER – 8

CONCLUSION

This report has dealt with the formulation of a new method that has been developed to solve the differential equation governing earthquake excitation and applying this method on known methods and it has yielded satisfactory results when compared with the actual values. The user-friendly software has been developed to test the methods and to plot the response of the system given the input earthquake accelerogram record, and it provided the user with the plot as digitized values of the response which can be used for various purposes, for eg. in finding the maximum displacement of a building during an earthquake in a particular area. The response spectra thus obtained by the software can be used for this reason.

CHAPTER – 9

RECOMMENDATIONS

The following are some of the recommendations on the method and the software developed.

1. The approach in developing the method is correct as it is seen from the results obtained. But it can be still further developed to give more accurate results.
2. There are so many methods that are not discussed but here it is restricted to only four. These methods can be incorporated to find the validity of each of them using the software.
3. The software can be made to plot several response spectra in the same screen by varying the value of damping coefficient.

APPENDIX

The input data for the software is the north-south component of the ground motion recorded at a site in El Centro, California during the Imperial Valley earthquake of May 18, 1940. Numerical values for the ground acceleration in units of g, the acceleration due to gravity are given as the input. This includes 1559 data points (first parameter) at equal time spacing of 0.02 seconds (second parameter), the first value at *t=0.02* sec; acceleration at *t=0* is zero.

1559 0.02 0.006300 0.003640 0.000990 0.004280 0.007580 0.010870 0.006820 0.002770 -0.001280 0.003680 0.008640 0.013600 0.007270 0.000940 0.004200 0.002210 0.000210 0.004440 0.008670 0.012900 0.017130 -0.003430 -0.024000 -0.009920 0.004160 0.005280 0.016530 0.027790 0.039040 0.024490 0.009950 0.009610 0.009260 0.008920 -0.004860 -0.018640 -0.032420 -0.033650 -0.057230 -0.045340 -0.033460 -0.032010 -0.030560 -0.029110 -0.027660 -0.041160 -0.054660 -0.068160 -0.081660 -0.068460 -0.055270 -0.042080 -0.042590 -0.043110 -0.024280 -0.005450 0.013380 0.032210 0.051040 0.069870 0.088700 0.045240 0.001790 -0.041670 -0.085130 -0.128580 -0.172040 -0.129080 -0.086130 -0.089020 -0.091920 -0.094820 -0.093240 -0.091660 -0.094780 -0.097890 -0.129020 -0.076520 -0.024010 0.028490 0.080990 0.133500 0.186000 0.238500 0.219930 0.201350 0.182770 0.164200 0.145620 0.161430 0.177250 0.132150 0.087050 0.041960 -0.003140 -0.048240 -0.093340 -0.138430 -0.183530 -0.228630 -0.273720 -0.318820 -0.250240 -0.181660 -0.113090 -0.044510 0.024070 0.092650 0.161230 0.229810 0.298390 0.231970 0.165540 0.099120 0.032700 -0.033720 -0.100140 -0.166560 -0.232990 -0.299410 -0.004210 0.290990 0.223800 0.156620 0.089430 0.022240 -0.044950 0.018340 0.081630 0.144910 0.208200 0.189730 0.171250 0.137590 0.103930 0.070270 0.036610 0.002950 -0.030710 -0.005610 0.019480 0.044580 0.064680 0.084780 0.104870 0.058950 0.013030 -0.032890 -0.078820 -0.035560 0.007710 0.050970 0.010130 -0.030710 -0.071560 -0.112400 -0.153240 -0.113140 -0.073040 -0.032940 0.007150 -0.063500 -0.134150 -0.204800 -0.124820 -0.044850 0.035130 0.115100 0.195080 0.123010 0.050940 -0.021130 -0.093200 -0.026630 0.039950 0.106530 0.173110 0.112830 0.052550 -0.007720 0.010640 0.029000 0.047370 0.065730 0.020210 -0.025300 -0.070810 -0.041070 -0.011330 0.002880 0.017090 0.031310 -0.022780 -0.076860 -0.130950 -0.185040 -0.143470 -0.101900 -0.060340 -0.018770 0.022800 -0.009960 -0.042720 -0.021470 -0.000210 0.021040 -0.014590 -0.050220 -0.085850 -0.121480 -0.157110 -0.192740 -0.228370 -0.181450 -0.134530 -0.087610 -0.040690 0.006230 0.053160 0.100080 0.147000 0.097540 0.048080 -0.001380 0.051410 0.104200 0.156990 0.209790 0.262580 0.169960 0.077340 -0.015270 -0.107890 -0.200510 -0.067860 0.064790 0.016710 -0.031370 -0.079450 -0.127530 -0.175610 -0.223690 -0.271770 -0.158510 -0.045250 0.068020 0.181280 0.144640 0.108000 0.071370 0.034730 0.096660 0.158600 0.220530 0.182960 0.145380 0.107800 0.070230 0.032650 0.066490 0.100330 0.134170 0.103370 0.072570 0.041770 0.010970 -0.019830 0.044380 0.108600 0.172810 0.104160 0.035510 -0.033150 -0.101800 -0.072620 -0.043440 -0.014260 0.014920 -0.020250 -0.055430 -0.090600 -0.125780 -0.160950 -0.196130 -0.147840 -0.099550 -0.051270 -0.002980 -0.019520 -0.036050 -0.052590 -0.041820 -0.031060 -0.029030 -0.026990 0.025150 0.017700 0.022130 0.026560

0.004190 -0.018190 -0.040570 -0.062940 -0.024170 0.014600 0.053370 0.024280 -0.004800 -0.033890 -0.005570 0.022740 0.006790 -0.009150 -0.025090 -0.041030 -0.056980 -0.018260 0.020460 0.004540 -0.011380 -0.002150 0.007080 0.004960 0.002850 0.000740 -0.005340 -0.011410 0.003610 0.018630 0.033650 0.048670 0.030400 0.012130 -0.006140 -0.024410 0.013750 0.010990 0.008230 0.005470 0.008120 0.010770 -0.006920 -0.024610 -0.042300 -0.059990 -0.077680 -0.095380 -0.062090 -0.028800 0.004480 0.037770 0.017730 -0.002310 -0.022350 0.017910 0.058160 0.037380 0.016600 -0.004180 -0.024960 -0.045740 -0.020710 0.004320 0.029350 0.015260 0.018060 0.020860 0.007930 -0.005010 -0.017950 -0.030890 -0.018410 -0.005930 0.006550 -0.025190 -0.056930 -0.040450 -0.023980 -0.007500 0.008970 0.003840 -0.001290 -0.006420 -0.011560 -0.026190 -0.040820 -0.055450 -0.043660 -0.031880 -0.069640 -0.056340 -0.043030 -0.029720 -0.016420 -0.003110 0.010200 0.023500 0.036810 0.050110 0.024360 -0.001390 -0.027140 -0.003090 0.020960 0.045010 0.069060 0.057730 0.046400 0.035070 0.033570 0.032070 0.030570 0.032500 0.034440 0.036370 0.013480 -0.009420 -0.032310 -0.029970 -0.030950 -0.031920 -0.025880 -0.019840 -0.013790 -0.007750 -0.014490 -0.021230 0.015230 0.051700 0.088160 0.124630 0.161090 0.129870 0.098640 0.067410 0.036180 0.004950 0.004200 0.003450 0.002690 -0.059220 -0.121120 -0.183030 -0.120430 -0.057820 0.004790 0.067400 0.130010 0.083730 0.037450 0.069790 0.102130 -0.035170 -0.172470 -0.137630 -0.102780 -0.067940 -0.033100 -0.036470 -0.039840 -0.005170 0.029500 0.064170 0.098830 0.133500 0.059240 -0.015030 -0.089290 -0.163550 -0.060960 0.041640 0.015510 -0.010610 -0.036740 -0.062870 -0.088990 -0.054300 -0.019610 0.015080 0.049770 0.084460 0.050230 0.016000 -0.018230 -0.052460 -0.086690 -0.067690 -0.048700 -0.029700 -0.010710 0.008290 -0.003140 0.029660 0.062460 -0.002340 -0.067140 -0.040510 -0.013880 0.012740 0.008050 0.030240 0.052430 0.023510 -0.005410 -0.034320 -0.063240 -0.092150 -0.121070 -0.084500 -0.047940 -0.011370 0.025200 0.061770 0.040280 0.018800 0.044560 0.070320 0.096080 0.121840 0.063500 0.005170 -0.053170 -0.031240 -0.009300 0.012630 0.034570 0.032830 0.031090 0.029350 0.045110 0.060870 0.076630 0.092390 0.057420 0.022450 -0.012520 0.006800 0.026110 0.045430 0.015710 -0.014020 -0.043740 -0.073470 -0.039900 -0.006330 0.027240 0.060800 0.036690 0.012580 -0.011530 -0.035640 -0.006770 0.022100 0.050980 0.079850 0.069150 0.058450 0.047750 0.037060 0.026360 0.058220 0.090090 0.121960 0.100690 0.079430 0.058160 0.036890 0.015630 -0.005640 -0.026900 -0.048170 -0.069440 -0.090700 -0.111970 -0.115210 -0.118460 -0.121700 -0.124940 -0.165000 -0.205050 -0.157130 -0.109210 -0.061290 -0.013370 0.034550 0.082470 0.075760 0.069060 0.062360 0.087350 0.112350 0.137340 0.121750 0.106160 0.090570 0.074980 0.080110 0.085240 0.090370 0.062080 0.033780 0.005490 -0.022810 -0.054440 -0.040300 -0.026150 -0.012010 -0.020280 -0.028550 -0.062430 -0.035240 -0.008050 -0.049480 -0.036430 -0.023370 -0.033680 -0.018790 -0.003890 0.011000 0.025890 0.014460 0.003030 -0.008400 0.004630 0.017660 0.030690 0.043720 0.021650 -0.000420 -0.022490 -0.044560 -0.036380 -0.028190 -0.020010 -0.011820 -0.024450 -0.037070 -0.049690 -0.058820 -0.067950 -0.077070 -0.086200 -0.095330 -0.062760 -0.030180 0.002390 0.034960 0.043990 0.053010 0.031760 0.010510 -0.010730 -0.031980 -0.053230 0.001860 0.056960 0.019850 -0.017260 -0.054380 -0.012040 0.030310 0.072650 0.114990 0.072370 0.029750 -0.012880 0.012120 0.037110 0.035170

0.033230 0.018530 0.003830 0.003420 -0.021810 -0.047040 -0.072270 -0.097500 -0.122730 -0.083170 -0.043620 -0.004070 0.035490 0.075040 0.114600 0.077690 0.040780 0.003870 0.002840 0.001820 -0.055130 0.047320 0.052230 0.057150 0.062060 0.066980 0.071890 0.027050 -0.017790 -0.062630 -0.107470 -0.152320 -0.125910 -0.099500 -0.073090 -0.046680 -0.020270 0.006140 0.032550 0.008590 -0.015370 -0.039320 -0.063280 -0.033220 -0.003150 0.026910 0.011960 -0.003000 0.003350 0.009700 0.016050 0.022390 0.042150 0.061910 0.081670 0.034770 -0.012120 -0.013090 -0.014070 -0.052740 -0.025440 0.001860 0.029160 0.056460 0.083760 0.017540 -0.048690 -0.020740 0.007220 0.035170 -0.005280 -0.045720 -0.086170 -0.069600 -0.053030 -0.036460 -0.019890 -0.003320 0.013250 0.029820 0.011010 -0.007810 -0.026620 -0.005630 0.015360 0.036350 0.057340 0.031590 0.005840 -0.019920 -0.002010 0.015890 -0.010240 -0.036360 -0.062490 -0.047800 -0.033110 -0.049410 -0.065700 -0.082000 -0.049800 -0.017600 0.014600 0.046800 0.079000 0.047500 0.016000 -0.015500 -0.001020 0.013470 0.027950 0.042440 0.056920 0.037810 0.018700 -0.000410 -0.019520 -0.004270 0.010980 0.026230 0.041480 0.018210 -0.005060 -0.008740 -0.037260 -0.065790 -0.026000 0.013800 0.053590 0.093380 0.058830 0.024290 -0.010260 -0.044800 -0.010830 -0.018690 -0.026550 -0.034410 -0.025030 -0.015640 -0.006260 -0.010090 -0.013920 0.014900 0.043720 0.034630 0.020980 0.007330 -0.006320 -0.019970 0.007670 0.035320 0.034090 0.032870 0.031640 0.024030 0.016420 0.009820 0.003220 -0.003390 0.022020 -0.019410 -0.060850 -0.102280 -0.078470 -0.054660 -0.030840 -0.007030 0.016780 0.019460 0.022140 0.024830 0.018090 -0.002020 -0.022130 -0.002780 0.016560 0.035900 0.055250 0.074590 0.062030 0.049480 0.036920 -0.001450 0.045990 0.040790 0.035580 0.030370 0.036260 0.042150 0.048030 0.053920 0.049470 0.045020 0.040560 0.036110 0.031660 0.006140 -0.019370 -0.044890 -0.070400 -0.095920 -0.077450 -0.058990 -0.040520 -0.022060 -0.003590 0.014870 0.010050 0.005230 0.000410 -0.004410 -0.009230 -0.011890 -0.015230 -0.018560 -0.021900 -0.009830 0.002240 0.014310 0.003350 -0.007600 -0.018560 -0.007370 0.003830 0.015020 0.026220 0.010160 -0.005900 -0.021960 -0.001210 0.019530 0.040270 0.028260 0.016250 0.004240 0.001960 -0.000310 -0.002580 -0.004860 -0.007130 -0.009410 -0.011680 -0.013960 -0.017500 -0.021040 -0.024580 -0.028130 -0.031670 -0.035210 -0.042050 -0.048890 -0.035590 -0.022290 -0.008990 0.004310 0.017620 0.007140 -0.003340 -0.013830 0.013140 0.040110 0.067080 0.048200 0.029320 0.010430 -0.008450 -0.027330 -0.046210 -0.031550 -0.016880 -0.002220 0.012440 0.026830 0.041210 0.055590 0.032530 0.009460 -0.013600 -0.014320 -0.015040 -0.015760 -0.042090 -0.026850 -0.011610 0.003630 0.018870 0.034110 0.031150 0.028190 0.029170 0.030150 0.031130 0.003880 -0.023370 -0.050620 -0.038200 -0.025790 -0.013370 -0.000950 0.011460 0.023880 0.036290 0.010470 -0.015350 -0.041170 -0.066990 -0.052070 -0.037150 -0.022220 -0.007300 0.007620 0.022540 0.037470 0.040010 0.042560 0.045070 0.047590 0.050100 0.045450 0.040800 0.028760 0.016710 0.004670 -0.007380 -0.001160 0.005060 0.011280 0.017500 -0.002110 -0.021730 -0.041350 -0.060960 -0.080580 -0.069950 -0.059310 -0.048680 -0.038050 -0.025570 -0.013100 -0.000630 0.011850 0.024320 0.036800 0.049270 0.029740 0.010210 -0.009320 -0.028840 -0.048370 -0.067900 -0.048620 -0.029340 -0.010060 0.009220 0.028510 0.047790 0.024560 0.001330 -0.021900 -0.045130 -0.068360 -0.049780 -0.031200 -0.012620 0.005960 0.024530 0.043110 0.061690 0.080270

0.098850 0.064520 0.030190 -0.004140 -0.038480 -0.072810 -0.059990 -0.047170 -0.034350 -0.032310 -0.030280 -0.028240 -0.003960 0.020320 0.003130 -0.014060 -0.031240 -0.048430 -0.065620 -0.051320 -0.037020 -0.022720 -0.008430 0.005870 0.020170 0.026980 0.033790 0.040610 0.047420 0.054230 0.035350 0.016470 0.016220 0.015980 0.015740 0.007470 -0.000800 -0.009070 0.000720 0.010510 0.020300 0.030090 0.039890 0.034780 0.029670 0.024570 0.030750 0.036940 0.043130 0.049310 0.055500 0.061680 -0.005260 -0.072200 -0.063360 -0.054510 -0.045660 -0.036810 -0.036780 -0.036750 -0.036720 -0.017650 0.001430 0.020510 0.039580 0.058660 0.035560 0.012450 -0.010660 -0.033760 -0.056870 -0.045020 -0.033170 -0.021310 -0.009460 0.002390 -0.002080 -0.006540 -0.011010 -0.015480 -0.012000 -0.008510 -0.005030 -0.001540 0.001950 0.000510 -0.000920 0.011350 0.023630 0.035900 0.048180 0.060450 0.072730 0.028470 -0.015790 -0.060040 -0.050690 -0.041340 -0.031990 -0.031350 -0.030710 -0.030070 -0.018630 -0.007190 0.004250 0.015700 0.027140 0.038580 0.029750 0.020920 0.023340 0.025760 0.028190 0.030610 0.033040 0.013710 -0.005610 -0.024940 -0.022080 -0.019230 -0.016380 -0.013530 -0.012610 -0.011700 -0.001690 0.008330 0.018340 0.028350 0.038360 0.048380 0.037490 0.026600 0.015710 0.004820 -0.006070 -0.016960 -0.007800 0.001360 0.010520 0.019680 0.028840 -0.005040 -0.038930 -0.023420 -0.007910 0.007590 0.023100 0.007070 -0.008950 -0.024980 -0.041000 -0.057030 -0.029200 -0.001370 0.026450 0.054280 0.035870 0.017460 -0.000960 -0.019370 -0.037780 -0.022810 -0.007840 0.007130 0.022100 0.037070 0.052040 0.067010 0.081980 0.030850 -0.020270 -0.071400 -0.122530 -0.086440 -0.050350 -0.014260 0.021830 0.057920 0.094000 0.130090 0.036110 -0.057870 -0.048020 -0.038170 -0.028320 -0.018460 -0.008610 -0.036520 -0.064440 -0.061690 -0.058940 -0.056180 -0.060730 -0.065280 -0.046280 -0.027280 -0.008290 0.010710 0.029700 0.031380 0.033060 0.034740 0.036420 0.045740 0.055060 0.064390 0.073710 0.083030 0.036050 -0.010920 -0.057900 -0.046960 -0.036020 -0.025080 -0.014140 -0.035610 -0.057080 -0.078550 -0.063040 -0.047530 -0.032030 -0.016520 -0.001020 0.009220 0.019460 0.029700 0.039930 0.050170 0.060410 0.070650 0.080890 -0.001920 -0.084730 -0.070320 -0.055900 -0.041480 -0.052960 -0.064430 -0.075900 -0.087380 -0.098850 -0.067980 -0.037100 -0.006230 0.024650 0.055530 0.086400 0.117280 0.148150 0.087150 0.026150 -0.034850 -0.095840 -0.071000 -0.046160 -0.021320 0.003530 0.028370 0.053210 -0.004690 -0.062580 -0.120480 -0.099600 -0.078720 -0.057840 -0.036960 -0.016080 0.004800 0.025680 0.046560 0.067440 0.088320 0.109200 0.130080 0.109950 0.089820 0.069690 0.049550 0.040060 0.030560 0.021070 0.011580 0.007800 0.004020 0.000240 -0.003540 -0.007320 -0.011100 -0.007800 -0.004500 -0.001200 0.002100 0.005400 -0.008310 -0.022030 -0.035750 -0.049470 -0.063190 -0.050460 -0.037730 -0.025000 -0.012270 0.000460 0.004820 0.009190 0.013550 0.017910 0.022280 0.008830 -0.004620 -0.018070 -0.031520 -0.022760 -0.014010 -0.005260 0.003500 0.012250 0.021010 0.014370 0.007730 0.001100 0.008230 0.015370 0.022510 0.017130 0.011750 0.006370 0.013760 0.021140 0.028520 0.035910 0.043290 0.034580 0.025870 0.017150 0.008440 -0.000270 -0.008980 -0.001260 0.006450 0.014170 0.020390 0.026610 0.032830 0.039050 0.045270 0.036390 0.027500 0.018620 0.009740 0.000860 -0.013330 -0.027520 -0.041710 -0.028120 -0.014530 -0.000940 0.012640 0.026230 0.016900 0.007560 -0.001770 -0.011110 -0.020440 -0.029770 -0.039110 -0.024420 -0.009730 0.004960 0.019650

0.034340 0.020540 0.006740 -0.007060 -0.020860 -0.034660 -0.026630 -0.018600 -0.010570 -0.002540 -0.000630 0.001280 0.003190 0.005100 0.009990 0.014880 0.007910 0.000930 -0.006050 0.003420 0.012880 0.022350 0.031810 0.041280 0.027070 0.012870 -0.001340 -0.015540 -0.029750 -0.043950 -0.036120 -0.028280 -0.020440 -0.012600 -0.004760 0.003070 0.010910 0.009840 0.008760 0.007680 0.006610 0.012340 0.018070 0.023800 0.029530 0.035260 0.027840 0.020420 0.013000 -0.034150 -0.006280 -0.006210 -0.006150 -0.006090 -0.006020 -0.005960 -0.005900 -0.005830 -0.005770 -0.005710 -0.005640 -0.005580 -0.005520 -0.005450 -0.005390 -0.005320 -0.005260 -0.005200 -0.005130 -0.005070 -0.005010 -0.004940 -0.004880 -0.004820 -0.004750 -0.004690 -0.004630 -0.004560 -0.004500 -0.004440 -0.004370 -0.004310 -0.004250 -0.004180 -0.004120 -0.004060 -0.003990 -0.003930 -0.003870 -0.003800 -0.003740 -0.003680 -0.003610 -0.003550 -0.003490 -0.003420 -0.003360 -0.003300 -0.003230 -0.003170 -0.003110 -0.003040 -0.002980 -0.002920 -0.002850 -0.002790 -0.002730 -0.002660 -0.002600 -0.002540 -0.002470 -0.002410 -0.002350 -0.002280 -0.002220 -0.002160 -0.002090 -0.002030 -0.001970 -0.001900 -0.001840 -0.001780 -0.001710 -0.001650 -0.001580 -0.001520 -0.001460 -0.001390 -0.001330 -0.001270 -0.001200 -0.001140 -0.001080 -0.001010 -0.000950 -0.000890 -0.000820 -0.000760 -0.000700 -0.000630 -0.000570 -0.000510 -0.000440 -0.000380 -0.000320 -0.000250 -0.000190 -0.000130 -0.000060 0.000000 0.000000

The following is the source code of the software done in Visual Basic 6.0 environment. The comments are placed in start of the functions.

```
Dim h As Double              //Variables Declaration
Dim m As Double
Dim wd As Double
Dim a As Double
Dim e As Double
Dim P(10000) As Double
Dim X(10000) As Double
Dim Y(10000) As Double
Dim Aa(10000) As Double
Dim B(10000) As Double

Dim term1 As Double
Dim term2 As Double
Dim u(10000) As Double
Dim all() As String
Dim t As Double
Dim n As Integer
Dim filenum
Dim str As String
Dim i As Integer
Dim j As Integer
Dim k As Integer

Dim scx As Double
Dim scy As Double
Dim scxr As Double
Dim scyr As Double
Dim scxrs As Double
Dim scyrs As Double

Dim tt As Double
Dim ttr As Double
Dim ttrs As Double

Dim x1 As Integer
Dim y1 As Integer
Dim x1r As Integer
Dim y1r As Integer
Dim x1rs As Integer
Dim y1rs As Integer

Dim max As Double
Dim min As Double
Dim maxr As Double
Dim minr As Double
Dim maxrs As Double
Dim minrs As Double
Dim ani As Integer
Dim anir As Integer
Dim anirs As Integer
```

```
Dim ct As Integer
Dim ctr As Integer
Dim ctrs As Integer
Dim no As Integer
Dim nor As Integer
Dim nors As Integer
Dim cur As Integer
Dim curr As Integer
Dim currs As Integer

Dim xval As Integer
Dim yval As Integer
Dim xvalr As Integer
Dim yvalr As Integer
Dim xvalrs As Integer
Dim yvalrs As Integer

Dim xx1 As Integer
Dim xx2 As Integer
Dim xx1r As Integer
Dim xx2r As Integer
Dim xx1rs As Integer
Dim xx2rs As Integer

Dim m1 As Double
Dim m2 As Double
Dim m3 As Double
Dim m4 As Double

Dim area As Double
Dim filenum1
Dim tn As Double
Dim eta As Double

Dim k11 As Double
Dim k12 As Double
Dim k13 As Double
Dim k14 As Double
Dim k15 As Double
Dim k16 As Double
Dim k21 As Double
Dim k22 As Double
Dim k23 As Double
Dim k24 As Double
Dim k25 As Double
Dim k26 As Double

Dim du(10000) As Double
Dim uu(10000) As Double
Dim ddu(10000) As Double
Dim acc(10000) As Double
Dim r(10000) As Double
Dim rs(10000) As Double

Dim wn As Double
```

```
Dim a11 As Double
Dim a12 As Double
Dim a21 As Double
Dim a22 As Double
Dim b11 As Double
Dim b12 As Double
Dim b21 As Double
Dim b22 As Double

Private Sub Command1_Click()          //Open Earthquake Data File
Text2.Text = "1"
Form2.Show (1)
End Sub

Public Function Calculate()           //Reads the ground response data from the file
On Error GoTo err1:
filenum = FreeFile
Open Form2.Text1.Text For Input As filenum
   str = Input(LOF(filenum), filenum)
   all = Split(str, " ")
   n = CInt(all(0))
   h = CDbl(all(1))
   List1.Clear
   List2.Clear
   List2.Clear
   Option6.Value = False
   Option7.Value = False
   Option8.Value = False
   Option16.Value = False
   Option17.Value = False
   Option15.Value = False
   Text8.Text = ""
   Text12.Text = ""
   For i = 2 To n + 1
      If (Option3.Value = False) Then
         P(i - 2) = CDbl(all(i)) * 386.0886
      Else
         P(i - 2) = CDbl(all(i))
      End If
   Next
   If (Option3.Value = True) Then
      For i = 0 To n - 1
         u(i) = P(i)
      Next
   End If
   If (Option4.Value = True) Then
      List1.Clear
      Cumm
   End If
   If (Option5.Value = True) Then
      Cumm
      For i = 0 To n - 1
         P(i) = u(i)
      Next
      Cumm
   End If
```

```
   For i = 0 To n - 1
      List1.AddItem ((i + 1) & "." & vbTab & u(i) & "")
   Next
Close filenum
Text3.Text = n
Text4.Text = h
max = Abs(u(0))
For i = 1 To n - 1
   If (Abs(u(i)) > Abs(max)) Then
      max = u(i)
   End If
Next
Text5.Text = max
Text7.Text = n * h
Exit Function
err1:
   MsgBox ("Error Opening File " & Form2.Text1.Text)
   List1.Clear
   List2.Clear
   List3.Clear
   Text8.Text = ""
   Text12.Text = ""
   Option6.Value = False
   Option7.Value = False
   Option8.Value = False
   Option16.Value = False
   Option17.Value = False
   Option15.Value = False
End Function
```

Public Function Calculater() **//Reads the response data from the file**

```
On Error GoTo err1:
filenum = FreeFile
Open Form2.Text1.Text For Input As filenum
   str = Input(LOF(filenum), filenum)
   all = Split(str, vbNewLine)
   n = CInt(all(0))
   h = CDbl(all(1))
   List1.Clear
   List2.Clear
   List3.Clear
   Option6.Value = False
   Option7.Value = False
   Option8.Value = False
   Option16.Value = False
   Option17.Value = False
   Option15.Value = False
   Text8.Text = ""
   Text12.Text = ""
   For i = 2 To n + 1
      If (Option6.Value = False) Then
         P(i - 2) = CDbl(all(i)) * 386.0886
      Else
         P(i - 2) = CDbl(all(i))
      End If
   Next
```

```
   For i = 0 To n - 1
      r(i) = P(i)
   Next
   For i = 0 To n - 1
      List2.AddItem ((i + 1) & "." & vbTab & r(i) & "")
   Next
Close filenum
Text3.Text = n
Text4.Text = h
max = Abs(r(0))
For i = 1 To n - 1
   If (Abs(r(i)) > Abs(max)) Then
      max = r(i)
   End If
Next
Text8.Text = max
Text7.Text = n * h
Exit Function
err1:
   MsgBox ("Error Opening File " & Form2.Text1.Text)
   List3.Clear
   List2.Clear
   List1.Clear
   Text8.Text = ""
   Text12.Text = ""
   Option6.Value = False
   Option7.Value = False
   Option8.Value = False
   Option16.Value = False
   Option17.Value = False
   Option15.Value = False
End Function

Public Function Calculaters()                    //Reads the response spectra data from the file
On Error GoTo err22:
filenum = FreeFile
Open Form2.Text1.Text For Input As filenum
   str = Input(LOF(filenum), filenum)
   all = Split(str, vbNewLine)
   n = CInt(all(0))
   tn = CDbl(all(1))
   List1.Clear
   List2.Clear
   List3.Clear
   Option6.Value = False
   Option7.Value = False
   Option8.Value = False
   Option16.Value = False
   Option17.Value = False
   Option15.Value = False
   Text5.Text = ""
   Text8.Text = ""
   Text12.Text = ""
   For i = 2 To n + 1
      If (Option17.Value = False) Then
         P(i - 2) = CDbl(all(i)) * 386.0886
```

```
      Else
         P(i - 2) = CDbl(all(i))
      End If
   Next
   For i = 0 To n - 1
      rs(i) = P(i)
   Next
   For i = 0 To n - 1
      List3.AddItem ((i + 1) & "." & vbTab & rs(i) & "")
   Next
Close filenum
Text3.Text = n
Text4.Text = tn
maxrs = Abs(rs(0))
For i = 1 To n - 1
   If (Abs(r(i)) > Abs(maxrs)) Then
      maxrs = rs(i)
   End If
Next
Text12.Text = maxrs
Text7.Text = n * tn
Exit Function
err22:
   MsgBox ("Error Opening File " & Form2.Text1.Text)
   List3.Clear
   List2.Clear
   List1.Clear
   Text8.Text = ""
   Text12.Text = ""
   Option16.Value = False
   Option17.Value = False
   Option15.Value = False
End Function

Private Sub Command2_Click()                    //Draws the Ground Response
On Error GoTo err2:
   Form3.Shape1.Visible = False
   Form3.Label14.Visible = False
If (List1.ListCount > 0) Then
   If (Option3.Value = True) Then
      Form3.Caption = "Ground Acceleration Response"
      Form3.Label1.Caption = "Ground Acceleration Response"
      Form3.Text4.Text = "ACCELERATION"
   End If
   If (Option4.Value = True) Then
      Form3.Caption = "Ground Velocity Response"
      Form3.Label1.Caption = "Ground Velocity Response"
      Form3.Text4.Text = "VELOCITY"
   End If
   If (Option5.Value = True) Then
      Form3.Caption = "Ground Displacement Response"
      Form3.Label1.Caption = "Ground Displacement Response"
      Form3.Text4.Text = "DISPLACEMENT"
   End If
   xx1 = 0
   xx2 = n - 1
```

```
mark3
ct = 0
Form3.Label4.Caption = Text7.Text
Form3.Show
Form3.Text1.Text = "1"
scx = 10500 / (n * h)
max = u(0)
min = u(0)
For i = 1 To n - 1
   If (u(i) > max) Then
      max = u(i)
   End If
   If (u(i) < min) Then
      min = u(i)
   End If
Next
scy = (yval * 1.3) / (max - min)
Form3.Label10.Caption = CInt(100 * 1477.5 / scy) / 100
Form3.Label11.Caption = CInt(100 * (-1) * 1477.5 / scy) / 100
Form3.Label12.Caption = CInt(100 * 1477.5 * 2 / scy) / 100
Form3.Label13.Caption = CInt(100 * (-1) * 1477.5 * 2 / scy) / 100
tt = h * 2
no = Int((n / 30000) * 33)
If (no = 0) Then
   no = 1
End If
If (Option1.Value = False) Then
   Form3.Line (xval, yval)-(xval + Int(tt * scx), yval - Int(u(0) * scy))
   x1 = xval + Int(tt * scx)
   y1 = yval - Int(u(0) * scy)
   tt = tt + h
   For i = 1 To n - 1
      Form3.Line (x1, y1)-(xval + Int(tt * scx), yval - Int(u(i) * scy))
      If (Abs(max) < Abs(min) And u(i) = min) Then
         Form3.Shape1.Left = xval + Int(tt * scx) - 75
         Form3.Shape1.Top = yval - Int(u(i) * scy) - 75
         Form3.Label14.Left = xval + Int(tt * scx) + 100
         Form3.Label14.Top = yval - Int(u(i) * scy)
         Form3.Label14.Caption = CInt(100 * min) / 100
         Form3.Shape1.Visible = True
         Form3.Label14.Visible = True
      End If
      If (Abs(max) > Abs(min) And u(i) = max) Then
         Form3.Shape1.Left = xval + Int(tt * scx) - 75
         Form3.Shape1.Top = yval - Int(u(i) * scy) - 75
         Form3.Label14.Left = xval + Int(tt * scx) + 100
         Form3.Label14.Top = yval - Int(u(i) * scy)
         Form3.Label14.Caption = CInt(1000 * max) / 100
         Form3.Shape1.Visible = True
         Form3.Label14.Visible = True
      End If
      x1 = xval + Int(tt * scx)
      y1 = yval - Int(u(i) * scy)
      tt = tt + h
   Next
   Form3.Text1.Text = ""
```

```
   Else
      Timer1.Interval = 1
      Exit Sub
   End If
End If
Exit Sub
err2:
   MsgBox ("Error In Drawing Ground Response")
   Form3.Visible = False
   Exit Sub
End Sub

Private Sub Command3_Click()              //Open Ground Response File
Text2.Text = "2"
Form2.Show (1)
End Sub

Private Sub Command4_Click()              //Open Response File
Text2.Text = "3"
Form2.Show (1)
End Sub

Private Sub Command5_Click()              //Draws the Response
On Error GoTo err12:
   Form5.Shape1.Visible = False
   Form5.Label14.Visible = False
If (List2.ListCount > 0) Then
   If (Text6.Text <> "" And Text7.Text <> "") Then
      wn = CDbl(Text6.Text)
      eta = CDbl(Text7.Text)
   End If
   If (Option6.Value = True) Then
      Form5.Caption = "Acceleration Response"
      Form5.Label1.Caption = "Acceleration Response"
      Form5.Text4.Text = "ACCELERATION"
   End If
   If (Option7.Value = True) Then
      Form5.Caption = "Displacement Response"
      Form5.Label1.Caption = "Displacement Response"
      Form5.Text4.Text = "DISPLACEMENT"
   End If
   If (Option8.Value = True) Then
      Form5.Caption = "Velocity Response"
      Form5.Label1.Caption = "Velocity Response"
      Form5.Text4.Text = "VELOCITY"
   End If
   xx1r = 0
   xx2r = n - 1
   mark5
   ctr = 0
   Form5.Label4.Caption = Text7.Text
   Form5.Show
   Form5.Text1.Text = "1"
   scxr = 10500 / (n * h)
   maxr = r(0)
   minr = r(0)
```

```
    For i = 1 To n - 1
        If (r(i) > maxr) Then
            maxr = r(i)
        End If
        If (r(i) < minr) Then
            minr = r(i)
        End If
    Next
    scyr = (yvalr * 1.3) / (maxr - minr)
    Form5.Label10.Caption = CInt(100 * 1477.5 / scyr) / 100
    Form5.Label11.Caption = CInt(100 * (-1) * 1477.5 / scyr) / 100
    Form5.Label12.Caption = CInt(100 * 1477.5 * 2 / scyr) / 100
    Form5.Label13.Caption = CInt(100 * (-1) * 1477.5 * 2 / scyr) / 100
    ttr = h
    nor = Int((n / 30000) * 33)
    If (nor = 0) Then
        nor = 1
    End If
    If (Option10.Value = False) Then
        Form5.Line (xvalr, yvalr)-(xvalr + Int(ttr * scxr), yvalr - Int(r(0) * scyr))
        x1r = xvalr + Int(ttr * scxr)
        y1r = yvalr - Int(r(0) * scyr)
        ttr = ttr + h
        For i = 1 To n - 1
            Form5.Line (x1r, y1r)-(xvalr + Int(ttr * scxr), yvalr - Int(r(i) * scyr))
            If (Abs(maxr) < Abs(min) And r(i) = minr) Then
                Form5.Shape1.Left = xvalr + Int(ttr * scxr) - 75
                Form5.Shape1.Top = yvalr - Int(r(i) * scyr) - 75
                Form5.Label14.Left = xvalr + Int(ttr * scxr) + 100
                Form5.Label14.Top = yvalr - Int(r(i) * scyr)
                Form5.Label14.Caption = CInt(100 * minr) / 100
                Form5.Shape1.Visible = True
                Form5.Label14.Visible = True
            End If
            If (Abs(maxr) > Abs(minr) And r(i) = maxr) Then
                Form5.Shape1.Left = xvalr + Int(ttr * scxr) - 75
                Form3.Shape1.Top = yvalr - Int(r(i) * scyr) - 75
                Form5.Label14.Left = xvalr + Int(ttr * scxr) + 100
                Form5.Label14.Top = yvalr - Int(r(i) * scyr)
                Form5.Label14.Caption = CInt(1000 * maxr) / 100
                Form5.Shape1.Visible = True
                Form5.Label14.Visible = True
            End If
            x1r = xvalr + Int(ttr * scxr)
            y1r = yvalr - Int(r(i) * scyr)
            ttr = ttr + h
        Next
        Form5.Text1.Text = ""
    Else
        Timer2.Interval = 1
        Exit Sub
    End If
End If
Exit Sub
err12:
    'MsgBox ("Error In Drawing Response")
```

```
   'Form5.Visible = False
   Resume Next
   Exit Sub
End Sub

Private Sub Command6_Click()                //Open Response Spectra File
Text2.Text = "4"
Form2.Show (1)
End Sub

Private Sub Command7_Click()                //Draws Response Spectra
On Error GoTo err20:
   Form6.Shape1.Visible = False
   Form6.Label14.Visible = False
If (List3.ListCount > 0) Then
   If (Text10.Text <> "" And Text11.Text <> "") Then
      wn = CDbl(Text10.Text)
      eta = CDbl(Text11.Text)
   End If
   If (Option17.Value = True) Then
      Form6.Caption = "Acceleration Response Spectra"
      Form6.Label1.Caption = "Acceleration Response Spectra"
      Form6.Text4.Text = "ACCELERATION"
   End If
   If (Option16.Value = True) Then
      Form6.Caption = "Displacement Response Spectra"
      Form6.Label1.Caption = "Displacement Response Spectra"
      Form6.Text4.Text = "DISPLACEMENT"
   End If
   If (Option15.Value = True) Then
      Form6.Caption = "Velocity Response Spectra"
      Form6.Label1.Caption = "Velocity Response Spectra"
      Form6.Text4.Text = "VELOCITY"
   End If
   xx1rs = 0
   xx2rs = n - 1
   mark6
   ctrs = 0
   Form6.Label4.Caption = n * tn
   Form6.Show
   Form6.Text1.Text = "1"
   scxrs = 10500 / (n * CDbl(Text10.Text))
   maxrs = rs(0)
   minrs = rs(0)
   For i = 1 To n - 1
      If (rs(i) > maxrs) Then
         maxrs = rs(i)
      End If
      If (rs(i) < minrs) Then
         minrs = rs(i)
      End If
   Next
   scyrs = (yvalrs) / (maxrs * 1.3) ' - minrs)
   Form6.Label10.Caption = 1477.5 / scyrs 'CInt(100 * 1477.5 / scyrs) / 100
   Form6.Label11.Caption = "" 'CInt(100 * (-1) * 1477.5 / scyrs) / 100
   Form6.Label12.Caption = 1477.5 / scyrs * 2 'CInt(100 * 1477.5 * 2 / scyrs) / 100
```

```
   Form6.Label13.Caption = "" 'CInt(100 * (-1) * 1477.5 * 2 / scyrs) / 100
   ttrs = 2 * CDbl(Text10.Text)
   nors = Int((n / 30000) * 15)
   If (nors = 0) Then
      nors = 1
   End If
   If (Option11.Value = False) Then
      Form6.Line (xvalrs, yvalrs)-(xvalrs + Int(ttrs * scxrs), yvalrs - Int(rs(0) * scyrs))
      x1rs = xvalrs + Int(ttrs * scxrs)
      y1rs = yvalrs - Int(rs(0) * scyrs)
      ttrs = CDbl(Text10.Text)
      For i = 1 To n - 1
         Form6.Line (x1rs, y1rs)-(xvalrs + Int(ttrs * scxrs), yvalrs - Int(rs(i) * scyrs))
         'If (Abs(maxrs) < Abs(minrs) And rs(i) = minrs) Then
            'Form6.Shape1.Left = xvalrs + Int(ttrs * scxrs) - 75
            'Form6.Shape1.Top = yvalrs - Int(rs(i) * scyrs) - 75
            'Form6.Label14.Left = xvalrs + Int(ttrs * scxrs) + 100
            'Form6.Label14.Top = yvalrs - Int(rs(i) * scyrs)
            'Form6.Label14.Caption = minrs 'CInt(100 * minrs) / 100
            'Form6.Shape1.Visible = True
            'Form6.Label14.Visible = True
         'End If
         If (rs(i) = maxrs) Then
            Form6.Shape1.Left = xvalrs + Int(ttrs * scxrs) - 75
            Form6.Shape1.Top = yvalrs - Int(rs(i) * scyrs) - 75
            Form6.Label14.Left = xvalrs + Int(ttrs * scxrs) + 100
            Form6.Label14.Top = yvalrs - Int(rs(i) * scyrs)
            Form6.Label14.Caption = maxrs 'CInt(1000 * maxrs) / 100
            Form6.Shape1.Visible = True
            Form6.Label14.Visible = True
         End If
         x1rs = xvalrs + Int(ttrs * scxrs)
         y1rs = yvalrs - Int(rs(i) * scyrs)
         ttrs = ttrs + CDbl(Text10.Text)
      Next
      Form6.Text1.Text = ""
   Else
      Timer3.Interval = 1
      Exit Sub
   End If
End If
Exit Sub
err20:
   MsgBox ("Error In Drawing Response Spectra")
   Form6.Visible = False
   Exit Sub
End Sub

Private Sub Form_Load()          //Initialization of variables
ct = 0
ctr = 0
ctrs = 0
xval = 840
xvalr = 840
xvalrs = 840
yval = 3800
```

```
yvalr = 3800
yvalrs = 3800
StatusBar2.Panels(3).Text = Format(Date, "dddd")
eta = 0.6
wn = 1
End Sub

Private Sub Form_Unload(Cancel As Integer)                    //End the Program
End
End Sub

Private Sub Option12_Click()                    //Runge-Kutta-Fehlberg Method for Response
If (Option6.Value = True) Then
   Option6_Click
End If
If (Option7.Value = True) Then
   Option7_Click
End If
If (Option8.Value = True) Then
   Option8_Click
End If
End Sub

Private Sub Option13_Click()                    //Runge-Kutta Method for Response
If (Option6.Value = True) Then
   Option6_Click
End If
If (Option7.Value = True) Then
   Option7_Click
End If
If (Option8.Value = True) Then
   Option8_Click
End If
End Sub

Private Sub Option15_Click()                    //Velocity Response Spectra
On Error GoTo err18:
If (List1.ListCount > 0) Then
   tn = CDbl(Text10.Text)
   eta = CDbl(Text11.Text)
   For i = 2 To n + 1
      P(i - 2) = CDbl(all(i)) * 386.0886
   Next
   wn = 0
   List3.Clear
   'For i = 0 To n - 1
   k = 0
   Do While (k < n)
      If (Option19.Value = True) Then
         RKF
      End If
      If (Option18.Value = True) Then
         RK
      End If
      If (Option21.Value = True) Then
         AMF
```

```
      End If
      If (Option23.Value = True) Then
         IBE
      End If
      maxrs = Abs(ddu(0))
      For j = 1 To n - 1
         If (Abs(ddu(j)) > Abs(maxrs)) Then
            maxrs = ddu(j)
         End If
      Next
      'List3.AddItem ((i + 1) & "." & vbTab & wn)
      rs(k) = Abs(maxrs)
      wn = wn + tn
      k = k + 1
   Loop
   'Next
   Label4.Caption = "Velocity Spectral Values"
   For i = 0 To n - 1
      'rs(i) = ddu(i)
      List3.AddItem ((i + 1) & "." & vbTab & rs(i))
   Next
   maxrs = Abs(rs(0))
   For i = 1 To n - 1
      If (Abs(rs(i)) > Abs(maxrs)) Then
         maxrs = rs(i)
      End If
   Next
   Text12.Text = maxrs
Else
   Option15.Value = False
End If
Exit Sub
err18:
   Option15.Value = False
   MsgBox ("Enter Valid Values For Time Period And Damping Ratio")
End Sub

Private Sub Option16_Click()                    //Displacement Response Spectra
On Error GoTo err19:
If (List1.ListCount > 0) Then
   tn = CDbl(Text10.Text)
   eta = CDbl(Text11.Text)
   For i = 2 To n + 1
      P(i - 2) = CDbl(all(i)) * 386.0886
   Next
   wn = 0
   List3.Clear
   'For i = 0 To n - 1
   k = 0
   Do While (k < n)
      If (Option19.Value = True) Then
         RKF
      End If
      If (Option18.Value = True) Then
         RK
      End If
```

```
      If (Option21.Value = True) Then
         AMF
      End If
      If (Option23.Value = True) Then
         IBE
      End If
      maxrs = Abs(du(0))
      For j = 1 To n - 1
         If (Abs(du(j)) > Abs(maxrs)) Then
            maxrs = du(j)
         End If
      Next
      'List3.AddItem ((i + 1) & "." & vbTab & wn)
      rs(k) = Abs(maxrs)
      wn = wn + tn
      k = k + 1
   Loop
   'Next
   Label4.Caption = "Displacement Spectral Values"
   For i = 0 To n - 1
      'rs(i) = ddu(i)
      List3.AddItem ((i + 1) & "." & vbTab & rs(i))
   Next
   maxrs = Abs(rs(0))
   For i = 1 To n - 1
      If (Abs(rs(i)) > Abs(maxrs)) Then
         maxrs = rs(i)
      End If
   Next
   Text12.Text = maxrs
Else
   Option16.Value = False
End If
Exit Sub
err19:
   Option16.Value = False
   MsgBox ("Enter Valid Values For Time Period And Damping Ratio")
End Sub

Private Sub Option17_Click()                    //Acceleration Response Spectra
On Error GoTo err17:
If (List1.ListCount > 0) Then
   tn = CDbl(Text10.Text)
   eta = CDbl(Text11.Text)
   For i = 2 To n + 1
      P(i - 2) = CDbl(all(i)) * 386.0886
   Next
   wn = 0
   List3.Clear
   'For i = 0 To n - 1
   k = 0
   Do While (k < n)
      If (Option19.Value = True) Then
         RKF
      End If
      If (Option18.Value = True) Then
```

```
        RK
      End If
      If (Option21.Value = True) Then
        AMF
      End If
      If (Option23.Value = True) Then
        IBE
      End If
      maxrs = Abs(acc(0))
      For j = 1 To n - 1
        If (Abs(acc(j)) > Abs(maxrs)) Then
          maxrs = acc(j)
        End If
      Next
      'List3.AddItem ((i + 1) & "." & vbTab & wn)
      rs(k) = Abs(maxrs)
      wn = wn + tn
      k = k + 1
    Loop
    'Next
    Label4.Caption = "Velocity Spectral Values"
    For i = 0 To n - 1
      'rs(i) = ddu(i)
      List3.AddItem ((i + 1) & "." & vbTab & rs(i))
    Next
    maxrs = Abs(rs(0))
    For i = 1 To n - 1
      If (Abs(rs(i)) > Abs(maxrs)) Then
        maxrs = rs(i)
      End If
    Next
    Text12.Text = maxrs
  Else
    Option17.Value = False
  End If
  Exit Sub
  err17:
    Option17.Value = False
    MsgBox ("Enter Valid Values For Time Period And Damping Ratio")
  End Sub

  Private Sub Option18_Click()          //Runge-Kutta Method for Response Spectra
  If (Option17.Value = True) Then
    Option17_Click
  End If
  If (Option15.Value = True) Then
    Option15_Click
  End If
  If (Option16.Value = True) Then
    Option16_Click
  End If
  End Sub

  Private Sub Option19_Click()          //Runge-Kutta-Fehlberg Method for Response Spectra
  If (Option17.Value = True) Then
    Option17_Click
```

```
End If
If (Option15.Value = True) Then
   Option15_Click
End If
If (Option16.Value = True) Then
   Option16_Click
End If
End Sub

Private Sub Option20_Click()          //Adam-Moulton Method for Response
If (Option6.Value = True) Then
   Option6_Click
End If
If (Option7.Value = True) Then
   Option7_Click
End If
If (Option8.Value = True) Then
   Option8_Click
End If
End Sub

Private Sub Option21_Click()          //Adam-Moulton Method for Response Spectra
If (Option17.Value = True) Then
   Option17_Click
End If
If (Option15.Value = True) Then
   Option15_Click
End If
If (Option16.Value = True) Then
   Option16_Click
End If
End Sub

Private Sub Option22_Click()          //Interpolation by Excitation for Response
If (Option6.Value = True) Then
   Option6_Click
End If
If (Option7.Value = True) Then
   Option7_Click
End If
If (Option8.Value = True) Then
   Option8_Click
End If
End Sub

Private Sub Option23_Click()          //Interpolation by Excitation for Response Spectra
If (Option17.Value = True) Then
   Option17_Click
End If
If (Option15.Value = True) Then
   Option15_Click
End If
If (Option16.Value = True) Then
   Option16_Click
End If
End Sub
```

```
Private Sub Option3_Click()                    //Acceleration Ground Response
If (List1.ListCount > 0) Then
   Label1.Caption = "Ground Acceleration Values"
   List1.Clear
   For i = 2 To n + 1
      P(i - 2) = CDbl(all(i))
   Next
   If (Option3.Value = True) Then
      For i = 0 To n - 1
         u(i) = P(i)
      Next
   End If
   For i = 0 To n - 1
      List1.AddItem ((i + 1) & "." & vbTab & u(i) & "")
   Next
   Text3.Text = n
   Text4.Text = h
   max = Abs(u(0))
   For i = 1 To n - 1
      If (Abs(u(i)) > Abs(max)) Then
         max = u(i)
      End If
   Next
   Text5.Text = max
   Text7.Text = n * h
End If
End Sub

Private Sub Option4_Click()                    //Velocity Ground Response
If (List1.ListCount > 0) Then
   Label1.Caption = "Ground Velocity Values"
   List1.Clear
   For i = 2 To n + 1
      P(i - 2) = CDbl(all(i)) * 386.0886
   Next
   If (Option4.Value = True) Then
      List1.Clear
      Cumm
   End If
   For i = 0 To n - 1
      List1.AddItem ((i + 1) & "." & vbTab & u(i) & "")
   Next
   Text3.Text = n
   Text4.Text = h
   max = Abs(u(0))
   For i = 1 To n - 1
      If (Abs(u(i)) > Abs(max)) Then
         max = u(i)
      End If
   Next
   Text5.Text = max
   Text7.Text = n * h
End If
End Sub
```

```
Private Sub Option5_Click()                    //Displacement Ground Response
If (List1.ListCount > 0) Then
   Label1.Caption = "Ground Displacement Values"
   List1.Clear
   For i = 2 To n + 1
      P(i - 2) = CDbl(all(i)) * 386.0886
   Next
   If (Option5.Value = True) Then
      Cumm
      For i = 0 To n - 1
         P(i) = u(i)
      Next
      Cumm
   End If
   For i = 0 To n - 1
      List1.AddItem ((i + 1) & "." & vbTab & u(i) & "")
   Next
   Text3.Text = n
   Text4.Text = h
   max = Abs(u(0))
   For i = 1 To n - 1
      If (Abs(u(i)) > Abs(max)) Then
         max = u(i)
      End If
   Next
   Text5.Text = max
   Text7.Text = n * h
End If
End Sub

Private Sub Option6_Click()                    //Acceleration Response
On Error GoTo err13:
If (List1.ListCount > 0) Then
   wn = CDbl(Text6.Text)
   eta = CDbl(Text9.Text)
   List2.Clear
   Label3.Caption = "Acceleration Response Values"
   For i = 0 To n - 1
      P(i) = all(i + 2)
   Next
   If (Option12.Value = True) Then
      RKF
   End If
   If (Option13.Value = True) Then
      RK
   End If
   If (Option20.Value = True) Then
      AMF
   End If
   If (Option22.Value = True) Then
      IBE
   End If
   For i = 0 To n - 1
      r(i) = acc(i)
      List2.AddItem ((i + 1) & "." & vbTab & acc(i))
   Next
```

```
   maxr = Abs(acc(0))
   For i = 1 To n - 1
      If (Abs(acc(i)) > Abs(maxr)) Then
         maxr = acc(i)
      End If
   Next
   Text8.Text = maxr
Else
   Option6.Value = False
End If
Exit Sub
err13:
   Option6.Value = False
   MsgBox ("Enter Valid Values For Time Period And Damping Ratio")
End Sub

Private Sub Option7_Click()                    //Velocity Response
On Error GoTo err14:
If (List1.ListCount > 0) Then
   wn = CDbl(Text6.Text)
   eta = CDbl(Text9.Text)
   For i = 2 To n + 1
      P(i - 2) = CDbl(all(i)) * 386.0886
   Next
   If (Option12.Value = True) Then
      RKF
   End If
   If (Option13.Value = True) Then
      RK
   End If
   If (Option20.Value = True) Then
      AMF
   End If
   If (Option22.Value = True) Then
      IBE
   End If
   List2.Clear
   Label3.Caption = "Displacement Response Values"
   For i = 0 To n - 1
      r(i) = du(i)
      List2.AddItem ((i + 1) & "." & vbTab & du(i))
   Next
   maxr = Abs(du(0))
   For i = 1 To n - 1
      If (Abs(du(i)) > Abs(maxr)) Then
         maxr = du(i)
      End If
   Next
   Text8.Text = maxr
End If
Exit Sub
err14:
   Option7.Value = False
   MsgBox ("Enter Valid Values For Time Period And Damping Ratio")
End Sub
```

```
Private Sub Option8_Click()                    //Displacement Response
On Error GoTo err15:
If (List1.ListCount > 0) Then
   wn = CDbl(Text6.Text)
   eta = CDbl(Text9.Text)
   For i = 2 To n + 1
      P(i - 2) = CDbl(all(i)) * 386.0886
   Next
   If (Option12.Value = True) Then
      RKF
   End If
   If (Option13.Value = True) Then
      RK
   End If
   If (Option20.Value = True) Then
      AMF
   End If
   If (Option22.Value = True) Then
      IBE
   End If
   List2.Clear
   Label3.Caption = "Velocity Response Values"
   For i = 0 To n - 1
      r(i) = ddu(i)
      List2.AddItem ((i + 1) & "." & vbTab & ddu(i))
   Next
   maxr = Abs(ddu(0))
   For i = 1 To n - 1
      If (Abs(ddu(i)) > Abs(maxr)) Then
         maxr = ddu(i)
      End If
   Next
   Text8.Text = maxr
Else
   Option8.Value = False
End If
Exit Sub
err15:
   Option8.Value = False
   MsgBox ("Enter Valid Values For Time Period And Damping Ratio")
End Sub

Private Sub Text10_KeyPress(KeyAscii As Integer)          //Time Period for Response Spectra
On Error GoTo err23:
If (KeyAscii = 13 And wn <> CDbl(Text10.Text)) Then
   If (Option17.Value = True) Then
      Option17_Click
   End If
   If (Option15.Value = True) Then
      Option15_Click
   End If
   If (Option16.Value = True) Then
      Option16_Click
   End If
End If
Exit Sub
```

```
err23:
   Exit Sub
End Sub

Private Sub Text11_KeyPress(KeyAscii As Integer)          //Damping for Response Spectra
On Error GoTo err24:
If (KeyAscii = 13 And eta <> CDbl(Text11.Text)) Then
   If (Option17.Value = True) Then
      Option17_Click
   End If
   If (Option15.Value = True) Then
      Option15_Click
   End If
   If (Option16.Value = True) Then
      Option16_Click
   End If
End If
Exit Sub
err24:
   Exit Sub
End Sub

Private Sub Text6_KeyPress(KeyAscii As Integer)           //Natural Frequency for Response
On Error GoTo err16:
If (KeyAscii = 13 And wn <> CDbl(Text6.Text)) Then
   If (Option6.Value = True) Then
      Option6_Click
   End If
   If (Option7.Value = True) Then
      Option7_Click
   End If
   If (Option8.Value = True) Then
      Option8_Click
   End If
End If
Exit Sub
err16:
   Exit Sub
End Sub

Private Sub Text9_KeyPress(KeyAscii As Integer)           //Damping for Response
On Error GoTo err17:
If (KeyAscii = 13 And eta <> CDbl(Text9.Text)) Then
   If (Option6.Value = True) Then
      Option6_Click
   End If
   If (Option7.Value = True) Then
      Option7_Click
   End If
   If (Option8.Value = True) Then
      Option8_Click
   End If
End If
err17:
   Exit Sub
End Sub
```

```
Private Sub Timer1_Timer()                    //Animate Ground Response
On Error GoTo err6:
Do While (cur < no)
   If (ct > n - 1) Then
      ani = 0
      Timer1.Interval = 0
      Form3.Text1.Text = ""
      Exit Sub
   End If
   If (ct = 0) Then
      Form3.Line (xval, yval)-(xval + Int(tt * scx), yval - Int(u(0) * scy))
      x1 = xval + Int(tt * scx)
      y1 = yval - Int(u(0) * scy)
      tt = tt + h
   Else
      Form3.Line (x1, y1)-(xval + Int(tt * scx), yval - Int(u(ct) * scy))
      If (Abs(max) < Abs(min) And u(ct) = min) Then
         Form3.Shape1.Left = xval + Int(tt * scx) - 75
         Form3.Shape1.Top = yval - Int(u(ct) * scy) - 75
         Form3.Label14.Left = xval + Int(tt * scx) + 100
         Form3.Label14.Top = yval - Int(u(ct) * scy)
         Form3.Label14.Caption = CInt(100 * min) / 100
         Form3.Shape1.Visible = True
         Form3.Label14.Visible = True
      End If
      If (Abs(max) > Abs(min) And u(ct) = max) Then
         Form3.Shape1.Left = xval + Int(tt * scx) - 75
         Form3.Shape1.Top = yval - Int(u(ct) * scy) - 75
         Form3.Label14.Left = xval + Int(tt * scx) + 100
         Form3.Label14.Top = yval - Int(u(ct) * scy)
         Form3.Label14.Caption = CInt(100 * max) / 100
         Form3.Shape1.Visible = True
         Form3.Label14.Visible = True
      End If
      x1 = xval + Int(tt * scx)
      y1 = yval - Int(u(ct) * scy)
      tt = tt + h
   End If
   ct = ct + 1
   cur = cur + 1
Loop
Exit Sub
err6:
   Resume Next
End Sub

Public Function save()                    //Save Ground Response
filenum1 = FreeFile
Open Text1 For Output As filenum1
   Print #filenum1, n & ""
   Print #filenum1, h & ""
   For i = 0 To n - 1
      Print #filenum1, u(i) & ""
   Next
Close filenum1
End Function
```

```
Public Function saver()                    //Save Response
filenum1 = FreeFile
Open Text1 For Output As filenum1
   Print #filenum1, n & ""
   Print #filenum1, h & ""
   For i = 0 To n - 1
      Print #filenum1, r(i) & ""
   Next
Close filenum1
End Function

Public Function savers()                   //Save Response Spectrum
filenum1 = FreeFile
Open Text1 For Output As filenum1
   Print #filenum1, n & ""
   Print #filenum1, tn & ""
   For i = 0 To n - 1
      Print #filenum1, rs(i) & ""
   Next
Close filenum1
End Function

Public Function CalculateDisp()            //Calculate Displacement
On Error GoTo err1:
filenum = FreeFile
Open Form2.Text1.Text For Input As filenum
   str = Input(LOF(filenum), filenum)
   all = Split(str, vbNewLine)
   n = CInt(all(0))
   h = CDbl(all(1))
   List1.Clear
   Option6.Value = False
   Option7.Value = False
   Option8.Value = False
   For i = 2 To n + 1
      u(i - 2) = CDbl(all(i))
      List1.AddItem ((i - 1) & "." & vbTab & u(i - 2) & "")
   Next
Close filenum
Text3.Text = n
Text4.Text = h
max = Abs(u(0))
For i = 1 To n - 1
   If (Abs(u(i)) > Abs(max)) Then
      max = u(i)
   End If
Next
Text5.Text = max
Text7.Text = n * h
Exit Function
err1:
   MsgBox ("Error Opening File " & Form2.Text1.Text)
   List1.Clear
   List2.Clear
   Text8.Text = ""
   Option6.Value = False
```

```
   Option7.Value = False
   Option8.Value = False
End Function

Public Function Repaint()                    //Repaint Ground Response
On Error GoTo err4:
   scx = 10500 / ((xx2 - xx1 + 1) * h)
   max = u(xx1)
   min = u(xx1)
   For i = xx1 To xx2
      If (u(i) > max) Then
         max = u(i)
      End If
      If (u(i) < min) Then
         min = u(i)
      End If
   Next
   scy = (yval * 1.3) / (max - min)
   tt = h
   Form3.Line (xval, yval)-(xval + Int(tt * scx), yval - Int(u(xx1) * scy))
   x1 = xval + Int(tt * scx)
   y1 = yval - Int(u(xx1) * scy)
   tt = tt + h
   For i = xx1 To xx2
      Form3.Line (x1, y1)-(xval + Int(tt * scx), yval - Int(u(i) * scy))
      x1 = xval + Int(tt * scx)
      y1 = yval - Int(u(i) * scy)
      tt = tt + h
   Next
Exit Function
err4:
   Resume Next
End Function

Public Function Repaintr()                   //Repaint Response
On Error GoTo err10:
   scxr = 10500 / ((xx2r - xx1r + 1) * h)
   maxr = r(xx1r)
   minr = r(xx1r)
   For i = xx1r To xx2r
      If (r(i) > maxr) Then
         maxr = r(i)
      End If
      If (r(i) < minr) Then
         minr = r(i)
      End If
   Next
   scyr = (yvalr * 1.3) / (maxr - minr)
   ttr = h
   Form5.Line (xvalr, yvalr)-(xvalr + Int(ttr * scxr), yvalr - Int(r(xx1r) * scyr))
   x1r = xvalr + Int(ttr * scxr)
   y1r = yvalr - Int(r(xx1r) * scyr)
   ttr = ttr + h
   For i = xx1r To xx2r
      Form5.Line (x1r, y1r)-(xvalr + Int(ttr * scxr), yvalr - Int(r(i) * scyr))
      x1r = xvalr + Int(ttr * scxr)
```

```
      y1r = yvalr - Int(r(i) * scyr)
      ttr = ttr + h
   Next
Exit Function
err10:
   Resume Next
End Function

Public Function Repaintrs()                    //Repaint Response Spectrum
On Error GoTo err25:
   scxrs = 10500 / ((xx2rs - xx1rs + 1) * tn)
   maxrs = rs(xx1rs)
   minrs = rs(xx1rs)
   For i = xx1rs To xx2rs
      If (rs(i) > maxrs) Then
         maxrs = rs(i)
      End If
      If (rs(i) < minrs) Then
         minrs = rs(i)
      End If
   Next
   scyrs = (yvalrs) / (maxrs * 1.3) '- minrs)
   ttrs = 0
   Form6.Line (xvalrs, yvalrs)-(xvalrs + Int(ttrs * scxrs), yvalrs - Int(r(xx1rs) * scyrs))
   x1rs = xvalrs + Int(ttrs * scxrs)
   y1rs = yvalrs - Int(rs(xx1rs) * scyrs)
   ttrs = ttrs + tn
   For i = xx1rs To xx2rs
      Form6.Line (x1rs, y1rs)-(xvalrs + Int(ttrs * scxrs), yvalrs - Int(rs(i) * scyrs))
      If (rs(i) = maxrs) Then
         Form6.Shape1.Left = x1rs + Int(ttrs * scxrs) - 75
         Form6.Shape1.Top = y1rs - Int(rs(i) * scyrs) - 75
         Form6.Label14.Left = x1rs + Int(ttrs * scxrs) + 100
         Form6.Label14.Top = y1rs - Int(rs(i) * scyrs)
         Form6.Label14.Caption = maxrs 'CInt(100 * maxrs) / 100
         Form6.Shape1.Visible = True
         Form6.Label14.Visible = True
      End If
      x1rs = xvalrs + Int(ttrs * scxrs)
      y1rs = yvalrs - Int(rs(i) * scyrs)
      ttrs = ttrs + tn
   Next
Exit Function
err25:
   Resume Next
End Function

Public Function Zoom()                 //Zoom Ground Response
xx1 = CInt(Form3.Text2.Text)
xx2 = CInt(Form3.Text3.Text)
xx1 = Int(xx1 / (scx * h))
xx2 = Int(xx2 / (scx * h))
Form3.Label3.Caption = (xx1 * h)
Form3.Label4.Caption = (xx2 * h)
If (xx2 > n - 1) Then
   xx2 = n - 1
```

```
End If
If (xx1 < 0) Then
   xx1 = 0
End If
mark3
End Function

Public Function Zoomr()              //Zoom Response
xx1r = CInt(Form5.Text2.Text)
xx2r = CInt(Form5.Text3.Text)
xx1r = Int(xx1r / (scxr * h))
xx2r = Int(xx2r / (scxr * h))
Form5.Label3.Caption = (xx1r * h)
Form5.Label4.Caption = (xx2r * h)
If (xx2r > n - 1) Then
   xx2r = n - 1
End If
If (xx1r < 0) Then
   xx1r = 0
End If
mark5
End Function

Public Function Zoomrs()                    //Zoom Response Spectra
xx1rs = CInt(Form6.Text2.Text)
xx2rs = CInt(Form6.Text3.Text)
xx1rs = Int(xx1rs / (scxrs * tn))
xx2rs = Int(xx2rs / (scxrs * tn))
Form6.Label3.Caption = (xx1rs * tn)
Form6.Label4.Caption = (xx2rs * tn)
If (xx2rs > n - 1) Then
   xx2rs = n - 1
End If
If (xx1rs < 0) Then
   xx1rs = 0
End If
mark6
End Function

Public Function Original()                  //Zoom to Previous Ground Response
xx1 = 0
xx2 = n - 1
xx1r = 0
xx2r = n - 1
xx1rs = 0
xx2rs = n - 1
Form3.Label3.Caption = (xx1 * h)
Form3.Label4.Caption = (xx2 * h)
Form5.Label3.Caption = (xx1r * h)
Form5.Label4.Caption = (xx2r * h)
Form6.Label3.Caption = (xx1rs * tn)
Form6.Label4.Caption = (xx2rs * tn)
End Function
```

```
Public Function SaveCurDisp()                    //Save Ground Response File
filenum1 = FreeFile
Open Text1 For Output As filenum1
   Print #filenum1, (xx2 - xx1 + 1) & ""
   Print #filenum1, h & ""
   For i = xx1 To xx2
      Print #filenum1, u(i) & ""
   Next
Close filenum1
End Function

Public Function Trapizoidal()                    //Trapiziodal Rule
t = h
a = h / (2 * m * wd)
   For i = 0 To n - 1
      X(i) = Exp(e * wd * t) * P(i) * Cos(wd * t)
      Y(i) = Exp(e * wd * t) * P(i) * Sin(wd * t)
      Aa(i) = X(0)
      If (i > 0) Then
         For j = 1 To i - 1
            Aa(i) = Aa(i) + 2 * X(j)
         Next
      End If
      If (i > 1) Then
         Aa(i) = Aa(i) + X(i)
      End If
      B(i) = Y(0)
      If (i > 0) Then
         For j = 1 To i - 1
            B(i) = B(i) + 2 * Y(j)
         Next
      End If
      If (i > 1) Then
         B(i) = B(i) + Y(i)
      End If
      Aa(i) = Aa(i) * a
      B(i) = B(i) * a
      term1 = Aa(i) * Exp((-1) * wd * t) * Sin(wd * t) / (t)
      term2 = B(i) * Exp((-1) * wd * t) * Sin(wd * t) / (t)
      u(i) = term1 - term2
      t = t + h
   Next
End Function

Public Function CummOpp()                        //Ground Displacement
t = h
u(0) = t * P(0)
t = t + h
For i = 1 To n - 1
   u(i) = u(i - 1) + ((t * P(i)) / i)
   t = t + h
Next
End Function
```

```
Public Function Cumm()                    //Ground Velocity
t = h
u(0) = t * P(0)
t = t + h
For i = 1 To n - 1
   area = (h * P(i)) + (0.5 * (P(i) - P(i - 1)) * h)
   u(i) = u(i - 1) + (area)
   t = t + h
Next
End Function

Public Function RK()                      //Runge-Kutta Method
du(0) = 0
ddu(0) = 0
For i = 0 To n - 2
   k11 = h * (((1) * P(i)) - (2 * eta * wn * du(i)) - (wn * wn * ddu(i)))
   k12 = h * (((1) * P(i)) - (2 * eta * wn * (du(i) + (k11 / 2))) - (wn * wn * (ddu(i) + (h / 2))))
   k13 = h * (((1) * P(i)) - (2 * eta * wn * (du(i) + (k12 / 2))) - (wn * wn * (ddu(i) + (h / 2))))
   k14 = h * (((1) * P(i)) - (2 * eta * wn * (du(i) + k13)) - (wn * wn * (ddu(i) + h)))
   ddu(i + 1) = (ddu(i) + ((1 / 6) * (k11 + (2 * k12) + (2 * k13) + k14)))
   k21 = h * ddu(i)
   k22 = h * (ddu(i) + (k21 / 2))
   k23 = h * (ddu(i) + (k22 / 2))
   k24 = h * (ddu(i) + k23)
   du(i + 1) = (du(i) + ((1 / 6) * (k21 + (2 * k22) + (2 * k23) + k24)))
   acc(i) = P(i) - (2 * eta * wn * du(i)) - (wn * wn * ddu(i))
Next
End Function

Public Function RKF()                     //Runge-Kutta-Fehlberg Method
du(0) = 0
ddu(0) = 0
For i = 0 To n - 2
   k11 = h * (((1) * P(i)) - (2 * eta * wn * du(i)) - (wn * wn * ddu(i)))

   k12 = h * (((1) * P(i)) - (2 * eta * wn * (du(i) + (k11 / 4))) - (wn * wn * (ddu(i) + (h / 4))))

   k13 = h * (((1) * P(i)) - (2 * eta * wn * (du(i) + (3 * k11 / 32) + (9 * k12 / 32))) - (wn * wn * (ddu(i) +
(3 * h / 8))))

   k14 = h * (((1) * P(i)) - (2 * eta * wn * (du(i) + (1932 * k11 / 2197) - (7200 * k12 / 2197) + (7296 * k13
/ 2197))) - (wn * wn * (ddu(i) + (12 * h / 13))))

   k15 = h * (((1) * P(i)) - (2 * eta * wn * (du(i) + (439 * k11 / 216) - (8 * k12) + (3680 * k13 / 513) - (845
* k14 / 4104))) - (wn * wn * (ddu(i) + h)))

   k16 = h * (((1) * P(i)) - (2 * eta * wn * (du(i) - (8 * k11 / 27) + (2 * k12) - (3544 * k13 / 2565) + (1859 *
k14 / 4104) - (11 * k15 / 40))) - (wn * wn * (ddu(i) + (h / 2))))

   ddu(i + 1) = ddu(i) + ((16 * k11 / 135) + (6656 * k13 / 12825) + (28561 * k14 / 56430) - (9 * k15 / 50)
+ (2 * k16 / 55))

   k21 = h * ddu(i)
   k22 = h * (ddu(i) + (k21 / 4))
   k23 = h * (ddu(i) + ((3 * k21 / 32) + (9 * k22 / 32)))
   k24 = h * (ddu(i) + ((1932 * k21 / 2197) - (7200 * k22 / 2197) + (7296 * k23 / 2197)))
```

```
   k25 = h * (ddu(i) + ((439 * k21 / 216) - (8 * k22) + (3680 * k23 / 513) - (845 * k24 / 4104)))
   k26 = h * (ddu(i) - ((8 * k21 / 27) + (2 * k22) - (3544 * k23 / 2565) + (1859 * k24 / 4104) - (11 * k25 / 40)))

   du(i + 1) = du(i) + ((16 * k21 / 135) + (6656 * k23 / 12825) + (28561 * k24 / 56430) - (9 * k25 / 50) + (2 * k26 / 55))

   acc(i) = P(i) - (2 * eta * wn * du(i)) - (wn * wn * ddu(i))
Next
End Function

Public Function AMF()                    //Adam-Moulton Method
du(0) = 0
ddu(0) = 0
For i = 0 To 3
   k11 = h * (((1) * P(i)) - (2 * eta * wn * du(i)) - (wn * wn * ddu(i)))
   k12 = h * (((1) * P(i)) - (2 * eta * wn * (du(i) + (k11 / 2))) - (wn * wn * (ddu(i) + (h / 2))))
   k13 = h * (((1) * P(i)) - (2 * eta * wn * (du(i) + (k12 / 2))) - (wn * wn * (ddu(i) + (h / 2))))
   k14 = h * (((1) * P(i)) - (2 * eta * wn * (du(i) + k13)) - (wn * wn * (ddu(i) + h)))
   ddu(i + 1) = (ddu(i) + ((1 / 6) * (k11 + (2 * k12) + (2 * k13) + k14)))

   k21 = h * ddu(i)
   k22 = h * (ddu(i) + (k21 / 2))
   k23 = h * (ddu(i) + (k22 / 2))
   k24 = h * (ddu(i) + k23)
   du(i + 1) = (du(i) + ((1 / 6) * (k21 + (2 * k22) + (2 * k23) + k24)))

   acc(i) = P(i) - (2 * eta * wn * du(i)) - (wn * wn * ddu(i))
Next
For i = 4 To n - 2
   ddu(i + 1) = ddu(i) + ((h / 24) * ((55 * du(i)) - (59 * du(i - 1)) + (37 * du(i - 2)) - (9 * du(i - 3))))
   du(i + 1) = du(i) + ((h / 24) * ((55 * acc(i)) - (59 * acc(i - 1)) + (37 * acc(i - 2)) - (9 * acc(i - 3))))
   acc(i + 1) = P(i) - (2 * eta * wn * du(i)) - (wn * wn * ddu(i))
Next
End Function

Public Function IBE()                    //Method of Interpolation by Excitation
For i = 0 To n - 1
   du(i) = 0
   ddu(i) = 0
Next
a11 = 0
a12 = 0
a21 = 0
a22 = 0
b11 = 0
b12 = 0
b21 = 0
b22 = 0
a11 = Exp((-1) * eta * wn * h) * (((eta / (1 - (eta * eta)) ^ 0.5) * Sin(wn * h * (1 - (eta * eta)) ^ 0.5)) + Cos(wn * h * (1 - (eta * eta)) ^ 0.5))

a12 = Exp((-1) * eta * wn * h) * Sin(wn * h * (1 - (eta * eta)) ^ 0.5) / (wn * (1 - (eta * eta)) ^ 0.5)

a21 = (-1) * wn * Exp((-1) * eta * wn * h) * wn * Sin(wn * h * (1 - (eta * eta)) ^ 0.5) / ((1 - (eta * eta)) ^ 0.5)
```

```
a22 = (Exp((-1) * eta * wn * h) * Cos(wn * h * (1 - (eta * eta)) ^ 0.5)) - ((eta / (1 - (eta * eta)) ^ 0.5) *
Sin(wn * h * (1 - (eta * eta)) ^ 0.5))

b11 = ((((((2 * eta * eta) - 1) / (wn * wn * h)) + (eta / wn)) * Sin(wn * h * (1 - (eta * eta)) ^ 0.5) / (wn * (1 -
(eta * eta)) ^ 0.5)) + (((((2 * eta * eta) - 1) / (wn * wn * wn * h)) + (1 / (wn * wn))) * Cos(wn * h * (1 - (eta
* eta)) ^ 0.5)) * Exp((-1) * eta * wn * h)) - ((2 * eta) / (wn * wn * wn * h))

b12 = ((((((2 * eta * eta) - 1) / (wn * wn * h))) * Sin(wn * h * (1 - (eta * eta)) ^ 0.5) / (wn * (1 - (eta * eta))
^ 0.5)) + (((((2 * eta * eta) - 1) / (wn * wn * wn * h))) * Cos(wn * h * (1 - (eta * eta)) ^ 0.5)) * (-1) * Exp((-
1) * eta * wn * h)) - (1 / (wn * wn)) + ((2 * eta) / (wn * wn * wn * h))

b21 = (((((((2 * eta * eta) - 1) / (wn * wn * h)) + (eta / wn)) * ((-1) * (eta / ((1 - (eta * eta)) ^ 0.5)) * Sin(wn
* h * (1 - (eta * eta)) ^ 0.5) / (wn * (1 - (eta * eta)) ^ 0.5)) + (Cos(wn * h * (1 - (eta * eta)) ^ 0.5)))) - ((((((2
* eta) - 1) / (wn * wn * wn * h)) + (1 / (wn * wn))) * (wn * ((1 - (eta * eta)) ^ 0.5) * Sin(wn * h * (1 - (eta *
eta)) ^ 0.5)) + (wn * eta * Cos(wn * h * (1 - (eta * eta)) ^ 0.5)))) * Exp((-1) * eta * wn * h)) + ((1) / (wn *
wn * h))

b22 = (((((((2 * eta * eta) - 1) / (wn * wn * h))) * ((-1) * (eta / ((1 - (eta * eta)) ^ 0.5)) * Sin(wn * h * (1 -
(eta * eta)) ^ 0.5) / (wn * (1 - (eta * eta)) ^ 0.5)) + (Cos(wn * h * (1 - (eta * eta)) ^ 0.5)))) - ((((((2 * eta) -
1) / (wn * wn * wn * h))) * (wn * ((1 - (eta * eta)) ^ 0.5) * Sin(wn * h * (1 - (eta * eta)) ^ 0.5)) + (wn * eta
* Cos(wn * h * (1 - (eta * eta)) ^ 0.5)))) * (-1) * Exp((-1) * eta * wn * h)) - ((1) / (wn * wn * h))

For i = 0 To n - 2
   ddu(i + 1) = (a11 * ddu(i)) + (a12 * du(i)) + (b11 * P(i)) + (b12 * P(i + 1))
   du(i) = (a21 * ddu(i)) + (a22 * du(i)) + (b21 * P(i)) + (b22 * P(i + 1))
   acc(i + 1) = (-1) * ((2 * eta * wn * du(i + 1)) + (wn * wn * ddu(i + 1)))
Next
End Function

Private Sub Timer2_Timer()                    //Animate Response
On Error GoTo err8:
curr = 0
Do While (curr < nor)
   If (ctr > n - 1) Then
      anir = 0
      Timer2.Interval = 0
      Form5.Text1.Text = ""
      Exit Sub
   End If
   If (ctr = 0) Then
      Form5.Line (xvalr, yvalr)-(xvalr + Int(ttr * scxr), yvalr - Int(r(0) * scyr))
      x1r = xvalr + Int(ttr * scxr)
      y1r = yvalr - Int(r(0) * scyr)
      ttr = ttr + h
   Else
      Form5.Line (x1r, y1r)-(xvalr + Int(ttr * scxr), yvalr - Int(r(ctr) * scyr))
      If (Abs(maxr) < Abs(minr) And r(ctr) = minr) Then
         Form5.Shape1.Left = xvalr + Int(ttr * scxr) - 75
         Form5.Shape1.Top = yvalr - Int(r(ctr) * scyr) - 75
         Form5.Label14.Left = xvalr + Int(ttr * scxr) + 100
         Form5.Label14.Top = yvalr - Int(r(ctr) * scyr)
         Form5.Label14.Caption = CInt(100 * minr) / 100
         Form5.Shape1.Visible = True
         Form5.Label14.Visible = True
      End If
```

```
    If (Abs(maxr) > Abs(minr) And r(ctr) = maxr) Then
       Form5.Shape1.Left = xvalr + Int(ttr * scxr) - 75
       Form5.Shape1.Top = yvalr - Int(r(ctr) * scyr) - 75
       Form5.Label14.Left = xvalr + Int(ttr * scxr) + 100
       Form5.Label14.Top = yvalr - Int(r(ctr) * scyr)
       Form5.Label14.Caption = CInt(100 * maxr) / 100
       Form5.Shape1.Visible = True
       Form5.Label14.Visible = True
    End If
    x1r = xvalr + Int(ttr * scxr)
    y1r = yvalr - Int(r(ctr) * scyr)
    ttr = ttr + h
  End If
  ctr = ctr + 1
  curr = curr + 1
Loop
Exit Sub
err8:
  Resume Next
End Sub

Private Sub Timer3_Timer()                    //Animate Response Spectra
On Error GoTo err24:
currs = 0
Do While (currs < nors)
  If (ctrs > n - 1) Then
    anirs = 0
    Timer3.Interval = 0
    Form6.Text1.Text = ""
    Exit Sub
  End If
  If (ctrs = 0) Then
    Form6.Line (xvalrs, yvalrs)-(xvalrs + Int(ttrs * scxrs), yvalrs - Int(rs(0) * scyrs))
    x1rs = xvalrs + Int(ttrs * scxrs)
    y1rs = yvalrs - Int(rs(0) * scyrs)
    ttrs = ttrs + tn
  Else
    Form6.Line (x1rs, y1rs)-(xvalrs + Int(ttrs * scxrs), yvalrs - Int(rs(ctrs) * scyrs))
    'If (Abs(maxrs) < Abs(minrs) And rs(ctrs) = minrs) Then
      'Form6.Shape1.Left = xvalrs + Int(ttrs * scxrs) - 75
      'Form6.Shape1.Top = yvalrs - Int(r(ctrs) * scyrs) - 75
      'Form6.Label14.Left = xvalrs + Int(ttrs * scxrs) + 100
      'Form6.Label14.Top = yvalrs - Int(rs(ctrs) * scyrs)
      'Form6.Label14.Caption = CInt(100 * minrs) / 100
      'Form6.Shape1.Visible = True
      'Form6.Label14.Visible = True
    'End If
    If (rs(ctrs) = maxrs) Then
      Form6.Shape1.Left = xvalrs + Int(ttrs * scxrs) - 75
      Form6.Shape1.Top = yvalrs - Int(rs(ctrs) * scyrs) - 75
      Form6.Label14.Left = xvalrs + Int(ttrs * scxrs) + 100
      Form6.Label14.Top = yvalrs - Int(rs(ctrs) * scyrs)
      Form6.Label14.Caption = maxrs 'CInt(100 * maxrs) / 100
      Form6.Shape1.Visible = True
      Form6.Label14.Visible = True
    End If
```

```
      x1rs = xvalrs + Int(ttrs * scxrs)
      y1rs = yvalrs - Int(rs(ctrs) * scyrs)
      ttrs = ttrs + tn
   End If
   ctrs = ctrs + 1
   currs = currs + 1
Loop
Exit Sub
err24:
   Resume Next
End Sub

Public Function mark3()                    //Mark Points in Ground Response
On Error GoTo err11:
Form3.Label5.Caption = CInt(10 * ((xx1 * h) + ((xx2 - xx1) * h / 5))) / 10
Form3.Label6.Caption = CInt(10 * ((xx1 * h) + (2 * (xx2 - xx1) * h / 5))) / 10
Form3.Label7.Caption = CInt(10 * ((xx1 * h) + (3 * (xx2 - xx1) * h / 5))) / 10
Form3.Label8.Caption = CInt(10 * ((xx1 * h) + (4 * (xx2 - xx1) * h / 5))) / 10
Exit Function
err11:
   Resume Next
End Function

Public Function mark5()                    //Mark Points in Response
On Error GoTo err12:
Form5.Label5.Caption = CInt(10 * ((xx1r * h) + ((xx2r - xx1r) * h / 5))) / 10
Form5.Label6.Caption = CInt(10 * ((xx1r * h) + (2 * (xx2r - xx1r) * h / 5))) / 10
Form5.Label7.Caption = CInt(10 * ((xx1r * h) + (3 * (xx2r - xx1r) * h / 5))) / 10
Form5.Label8.Caption = CInt(10 * ((xx1r * h) + (4 * (xx2r - xx1r) * h / 5))) / 10
Exit Function
err12:
   Resume Next
End Function

Public Function mark6()                    //Mark Points in Response Spectrum
On Error GoTo err13:
Form6.Label5.Caption = CInt(10 * ((xx1rs * h) + ((xx2rs - xx1rs) * CDbl(Text10.Text) / 5))) / 10
Form6.Label6.Caption = CInt(10 * ((xx1rs * h) + (2 * (xx2rs - xx1rs) * CDbl(Text10.Text) / 5))) / 10
Form6.Label7.Caption = CInt(10 * ((xx1rs * h) + (3 * (xx2rs - xx1rs) * CDbl(Text10.Text) / 5))) / 10
Form6.Label8.Caption = CInt(10 * ((xx1rs * h) + (4 * (xx2rs - xx1rs) * CDbl(Text10.Text) / 5))) / 10
Exit Function
err13:
   Resume Next
End Function
```

REFERENCES

1. Chopra, K., Anil, (1995) "Dynamics of Structures", Text Book, ISBN 0-13-855214.

2. Humar, L., Jagmohan (1990) "Dynamics of Structures", Engle Cliffs, N.J. Prentice Hall.

3. Timoshenko, P., Stephen (1999) "Applied Numerical Analysis", Pearson Education Inc., Sixth Edition.

4. William Weaver, Jr. and others, "Vibrational Problems in Engineering", John Wiley & Sons, Fifth Edition, 1990.

5. University of Berkeley Earthquake Engineering Research Centre. http://peer.berkeley.edu/research/

6. United States Geological Survey. http://mapping.usgs.gov/esic/to_order.html

APPENDIX

The input data for the software is the north-south component of the ground motion recorded at a site in El Centro, California during the Imperial Valley earthquake of May 18, 1940. Numerical values for the ground acceleration in units of g, the acceleration due to gravity are given as the input. This includes 1559 data points (first parameter) at equal time spacing of 0.02 seconds (second parameter), the first value at *t=0.02* sec; acceleration at *t=0* is zero.

1559 0.02 0.006300 0.003640 0.000990 0.004280 0.007580 0.010870 0.006820 0.002770 -0.001280 0.003680 0.008640 0.013600 0.007270 0.000940 0.004200 0.002210 0.000210 0.004440 0.008670 0.012900 0.017130 -0.003430 -0.024000 -0.009920 0.004160 0.005280 0.016530 0.027790 0.039040 0.024490 0.009950 0.009610 0.009260 0.008920 -0.004860 -0.018640 -0.032420 -0.033650 -0.057230 -0.045340 -0.033460 -0.032010 -0.030560 -0.029110 -0.027660 -0.041160 -0.054660 -0.068160 -0.081660 -0.068460 -0.055270 -0.042080 -0.042590 -0.043110 -0.024280 -0.005450 0.013380 0.032210 0.051040 0.069870 0.088700 0.045240 0.001790 -0.041670 -0.085130 -0.128580 -0.172040 -0.129080 -0.086130 -0.089020 -0.091920 -0.094820 -0.093240 -0.091660 -0.094780 -0.097890 -0.129020 -0.076520 -0.024010 0.028490 0.080990 0.133500 0.186000 0.238500 0.219930 0.201350 0.182770 0.164200 0.145620 0.161430 0.177250 0.132150 0.087050 0.041960 -0.003140 -0.048240 -0.093340 -0.138430 -0.183530 -0.228630 -0.273720 -0.318820 -0.250240 -0.181660 -0.113090 -0.044510 0.024070 0.092650 0.161230 0.229810 0.298390 0.231970 0.165540 0.099120 0.032700 -0.033720 -0.100140 -0.166560 -0.232990 -0.299410 -0.004210 0.290990 0.223800 0.156620 0.089430 0.022240 -0.044950 0.018340 0.081630 0.144910 0.208200 0.189730 0.171250 0.137590 0.103930 0.070270 0.036610 0.002950 -0.030710 -0.005610 0.019480 0.044580 0.064680 0.084780 0.104870 0.058950 0.013030 -0.032890 -0.078820 -0.035560 0.007710 0.050970 0.010130 -0.030710 -0.071560 -0.112400 -0.153240 -0.113140 -0.073040 -0.032940 0.007150 -0.063500 -0.134150 -0.204800 -0.124820 -0.044850 0.035130 0.115100 0.195080 0.123010 0.050940 -0.021130 -0.093200 -0.026630 0.039950 0.106530 0.173110 0.112830 0.052550 -0.007720 0.010640 0.029000 0.047370 0.065730 0.020210 -0.025300 -0.070810 -0.041070 -0.011330 0.002880 0.017090 0.031310 -0.022780 -0.076860 -0.130950 -0.185040 -0.143470 -0.101900 -0.060340 -0.018770 0.022800 -0.009960 -0.042720 -0.021470 -0.000210 0.021040 -0.014590 -0.050220 -0.085850 -0.121480 -0.157110 -0.192740 -0.228370 -0.181450 -0.134530 -0.087610 -0.040690 0.006230 0.053160 0.100080 0.147000 0.097540 0.048080 -0.001380 0.051410 0.104200 0.156990 0.209790 0.262580 0.169960 0.077340 -0.015270 -0.107890 -0.200510 -0.067860 0.064790 0.016710 -0.031370 -0.079450 -0.127530 -0.175610 -0.223690 -0.271770 -0.158510 -0.045250 0.068020 0.181280 0.144640 0.108000 0.071370 0.034730 0.096660 0.158600 0.220530 0.182960 0.145380 0.107800 0.070230 0.032650 0.066490 0.100330 0.134170 0.103370 0.072570 0.041770 0.010970 -0.019830 0.044380 0.108600 0.172810 0.104160 0.035510 -0.033150 -0.101800 -0.072620 -0.043440 -0.014260 0.014920 -0.020250 -0.055430 -0.090600 -0.125780 -0.160950 -0.196130 -0.147840 -0.099550 -0.051270 -0.002980 -0.019520 -0.036050 -0.052590 -0.041820 -0.031060 -0.029030 -0.026990 0.025150 0.017700 0.022130 0.026560

0.004190 -0.018190 -0.040570 -0.062940 -0.024170 0.014600 0.053370 0.024280 -0.004800 -0.033890 -0.005570 0.022740 0.006790 -0.009150 -0.025090 -0.041030 -0.056980 -0.018260 0.020460 0.004540 -0.011380 -0.002150 0.007080 0.004960 0.002850 0.000740 -0.005340 -0.011410 0.003610 0.018630 0.033650 0.048670 0.030400 0.012130 -0.006140 -0.024410 0.013750 0.010990 0.008230 0.005470 0.008120 0.010770 -0.006920 -0.024610 -0.042300 -0.059990 -0.077680 -0.095380 -0.062090 -0.028800 0.004480 0.037770 0.017730 -0.002310 -0.022350 0.017910 0.058160 0.037380 0.016600 -0.004180 -0.024960 -0.045740 -0.020710 0.004320 0.029350 0.015260 0.018060 0.020860 0.007930 -0.005010 -0.017950 -0.030890 -0.018410 -0.005930 0.006550 -0.025190 -0.056930 -0.040450 -0.023980 -0.007500 0.008970 0.003840 -0.001290 -0.006420 -0.011560 -0.026190 -0.040820 -0.055450 -0.043660 -0.031880 -0.069640 -0.056340 -0.043030 -0.029720 -0.016420 -0.003110 0.010200 0.023500 0.036810 0.050110 0.024360 -0.001390 -0.027140 -0.003090 0.020960 0.045010 0.069060 0.057730 0.046400 0.035070 0.033570 0.032070 0.030570 0.032500 0.034440 0.036370 0.013480 -0.009420 -0.032310 -0.029970 -0.030950 -0.031920 -0.025880 -0.019840 -0.013790 -0.007750 -0.014490 -0.021230 0.015230 0.051700 0.088160 0.124630 0.161090 0.129870 0.098640 0.067410 0.036180 0.004950 0.004200 0.003450 0.002690 -0.059220 -0.121120 -0.183030 -0.120430 -0.057820 0.004790 0.067400 0.130010 0.083730 0.037450 0.069790 0.102130 -0.035170 -0.172470 -0.137630 -0.102780 -0.067940 -0.033100 -0.036470 -0.039840 -0.005170 0.029500 0.064170 0.098830 0.133500 0.059240 -0.015030 -0.089290 -0.163550 -0.060960 0.041640 0.015510 -0.010610 -0.036740 -0.062870 -0.088990 -0.054300 -0.019610 0.015080 0.049770 0.084460 0.050230 0.016000 -0.018230 -0.052460 -0.086690 -0.067690 -0.048700 -0.029700 -0.010710 0.008290 -0.003140 0.029660 0.062460 -0.002340 -0.067140 -0.040510 -0.013880 0.012740 0.008050 0.030240 0.052430 0.023510 -0.005410 -0.034320 -0.063240 -0.092150 -0.121070 -0.084500 -0.047940 -0.011370 0.025200 0.061770 0.040280 0.018800 0.044560 0.070320 0.096080 0.121840 0.063500 0.005170 -0.053170 -0.031240 -0.009300 0.012630 0.034570 0.032830 0.031090 0.029350 0.045110 0.060870 0.076630 0.092390 0.057420 0.022450 -0.012520 0.006800 0.026110 0.045430 0.015710 -0.014020 -0.043740 -0.073470 -0.039900 -0.006330 0.027240 0.060800 0.036690 0.012580 -0.011530 -0.035640 -0.006770 0.022100 0.050980 0.079850 0.069150 0.058450 0.047750 0.037060 0.026360 0.058220 0.090090 0.121960 0.100690 0.079430 0.058160 0.036890 0.015630 -0.005640 -0.026900 -0.048170 -0.069440 -0.090700 -0.111970 -0.115210 -0.118460 -0.121700 -0.124940 -0.165000 -0.205050 -0.157130 -0.109210 -0.061290 -0.013370 0.034550 0.082470 0.075760 0.069060 0.062360 0.087350 0.112350 0.137340 0.121750 0.106160 0.090570 0.074980 0.080110 0.085240 0.090370 0.062080 0.033780 0.005490 -0.022810 -0.054440 -0.040300 -0.026150 -0.012010 -0.020280 -0.028550 -0.062430 -0.035240 -0.008050 -0.049480 -0.036430 -0.023370 -0.033680 -0.018790 -0.003890 0.011000 0.025890 0.014460 0.003030 -0.008400 0.004630 0.017660 0.030690 0.043720 0.021650 -0.000420 -0.022490 -0.044560 -0.036380 -0.028190 -0.020010 -0.011820 -0.024450 -0.037070 -0.049690 -0.058820 -0.067950 -0.077070 -0.086200 -0.095330 -0.062760 -0.030180 0.002390 0.034960 0.043990 0.053010 0.031760 0.010510 -0.010730 -0.031980 -0.053230 0.001860 0.056960 0.019850 -0.017260 -0.054380 -0.012040 0.030310 0.072650 0.114990 0.072370 0.029750 -0.012880 0.012120 0.037110 0.035170

0.033230 0.018530 0.003830 0.003420 -0.021810 -0.047040 -0.072270 -0.097500 -0.122730 -0.083170 -0.043620 -0.004070 0.035490 0.075040 0.114600 0.077690 0.040780 0.003870 0.002840 0.001820 -0.055130 0.047320 0.052230 0.057150 0.062060 0.066980 0.071890 0.027050 -0.017790 -0.062630 -0.107470 -0.152320 -0.125910 -0.099500 -0.073090 -0.046680 -0.020270 0.006140 0.032550 0.008590 -0.015370 -0.039320 -0.063280 -0.033220 -0.003150 0.026910 0.011960 -0.003000 0.003350 0.009700 0.016050 0.022390 0.042150 0.061910 0.081670 0.034770 -0.012120 -0.013090 -0.014070 -0.052740 -0.025440 0.001860 0.029160 0.056460 0.083760 0.017540 -0.048690 -0.020740 0.007220 0.035170 -0.005280 -0.045720 -0.086170 -0.069600 -0.053030 -0.036460 -0.019890 -0.003320 0.013250 0.029820 0.011010 -0.007810 -0.026620 -0.005630 0.015360 0.036350 0.057340 0.031590 0.005840 -0.019920 -0.002010 0.015890 -0.010240 -0.036360 -0.062490 -0.047800 -0.033110 -0.049410 -0.065700 -0.082000 -0.049800 -0.017600 0.014600 0.046800 0.079000 0.047500 0.016000 -0.015500 -0.001020 0.013470 0.027950 0.042440 0.056920 0.037810 0.018700 -0.000410 -0.019520 -0.004270 0.010980 0.026230 0.041480 0.018210 -0.005060 -0.008740 -0.037260 -0.065790 -0.026000 0.013800 0.053590 0.093380 0.058830 0.024290 -0.010260 -0.044800 -0.010830 -0.018690 -0.026550 -0.034410 -0.025030 -0.015640 -0.006260 -0.010090 -0.013920 0.014900 0.043720 0.034630 0.020980 0.007330 -0.006320 -0.019970 0.007670 0.035320 0.034090 0.032870 0.031640 0.024030 0.016420 0.009820 0.003220 -0.003390 0.022020 -0.019410 -0.060850 -0.102280 -0.078470 -0.054660 -0.030840 -0.007030 0.016780 0.019460 0.022140 0.024830 0.018090 -0.002020 -0.022130 -0.002780 0.016560 0.035900 0.055250 0.074590 0.062030 0.049480 0.036920 -0.001450 0.045990 0.040790 0.035580 0.030370 0.036260 0.042150 0.048030 0.053920 0.049470 0.045020 0.040560 0.036110 0.031660 0.006140 -0.019370 -0.044890 -0.070400 -0.095920 -0.077450 -0.058990 -0.040520 -0.022060 -0.003590 0.014870 0.010050 0.005230 0.000410 -0.004410 -0.009230 -0.011890 -0.015230 -0.018560 -0.021900 -0.009830 0.002240 0.014310 0.003350 -0.007600 -0.018560 -0.007370 0.003830 0.015020 0.026220 0.010160 -0.005900 -0.021960 -0.001210 0.019530 0.040270 0.028260 0.016250 0.004240 0.001960 -0.000310 -0.002580 -0.004860 -0.007130 -0.009410 -0.011680 -0.013960 -0.017500 -0.021040 -0.024580 -0.028130 -0.031670 -0.035210 -0.042050 -0.048890 -0.035590 -0.022290 -0.008990 0.004310 0.017620 0.007140 -0.003340 -0.013830 0.013140 0.040110 0.067080 0.048200 0.029320 0.010430 -0.008450 -0.027330 -0.046210 -0.031550 -0.016880 -0.002220 0.012440 0.026830 0.041210 0.055590 0.032530 0.009460 -0.013600 -0.014320 -0.015040 -0.015760 -0.042090 -0.026850 -0.011610 0.003630 0.018870 0.034110 0.031150 0.028190 0.029170 0.030150 0.031130 0.003880 -0.023370 -0.050620 -0.038200 -0.025790 -0.013370 -0.000950 0.011460 0.023880 0.036290 0.010470 -0.015350 -0.041170 -0.066990 -0.052070 -0.037150 -0.022220 -0.007300 0.007620 0.022540 0.037470 0.040010 0.042560 0.045070 0.047590 0.050100 0.045450 0.040800 0.028760 0.016710 0.004670 -0.007380 -0.001160 0.005060 0.011280 0.017500 -0.002110 -0.021730 -0.041350 -0.060960 -0.080580 -0.069950 -0.059310 -0.048680 -0.038050 -0.025570 -0.013100 -0.000630 0.011850 0.024320 0.036800 0.049270 0.029740 0.010210 -0.009320 -0.028840 -0.048370 -0.067900 -0.048620 -0.029340 -0.010060 0.009220 0.028510 0.047790 0.024560 0.001330 -0.021900 -0.045130 -0.068360 -0.049780 -0.031200 -0.012620 0.005960 0.024530 0.043110 0.061690 0.080270

0.098850 0.064520 0.030190 -0.004140 -0.038480 -0.072810 -0.059990 -0.047170 -0.034350 -0.032310 -0.030280 -0.028240 -0.003960 0.020320 0.003130 -0.014060 -0.031240 -0.048430 -0.065620 -0.051320 -0.037020 -0.022720 -0.008430 0.005870 0.020170 0.026980 0.033790 0.040610 0.047420 0.054230 0.035350 0.016470 0.016220 0.015980 0.015740 0.007470 -0.000800 -0.009070 0.000720 0.010510 0.020300 0.030090 0.039890 0.034780 0.029670 0.024570 0.030750 0.036940 0.043130 0.049310 0.055500 0.061680 -0.005260 -0.072200 -0.063360 -0.054510 -0.045660 -0.036810 -0.036780 -0.036750 -0.036720 -0.017650 0.001430 0.020510 0.039580 0.058660 0.035560 0.012450 -0.010660 -0.033760 -0.056870 -0.045020 -0.033170 -0.021310 -0.009460 0.002390 -0.002080 -0.006540 -0.011010 -0.015480 -0.012000 -0.008510 -0.005030 -0.001540 0.001950 0.000510 -0.000920 0.011350 0.023630 0.035900 0.048180 0.060450 0.072730 0.028470 -0.015790 -0.060040 -0.050690 -0.041340 -0.031990 -0.031350 -0.030710 -0.030070 -0.018630 -0.007190 0.004250 0.015700 0.027140 0.038580 0.029750 0.020920 0.023340 0.025760 0.028190 0.030610 0.033040 0.013710 -0.005610 -0.024940 -0.022080 -0.019230 -0.016380 -0.013530 -0.012610 -0.011700 -0.001690 0.008330 0.018340 0.028350 0.038360 0.048380 0.037490 0.026600 0.015710 0.004820 -0.006070 -0.016960 -0.007800 0.001360 0.010520 0.019680 0.028840 -0.005040 -0.038930 -0.023420 -0.007910 0.007590 0.023100 0.007070 -0.008950 -0.024980 -0.041000 -0.057030 -0.029200 -0.001370 0.026450 0.054280 0.035870 0.017460 -0.000960 -0.019370 -0.037780 -0.022810 -0.007840 0.007130 0.022100 0.037070 0.052040 0.067010 0.081980 0.030850 -0.020270 -0.071400 -0.122530 -0.086440 -0.050350 -0.014260 0.021830 0.057920 0.094000 0.130090 0.036110 -0.057870 -0.048020 -0.038170 -0.028320 -0.018460 -0.008610 -0.036520 -0.064440 -0.061690 -0.058940 -0.056180 -0.060730 -0.065280 -0.046280 -0.027280 -0.008290 0.010710 0.029700 0.031380 0.033060 0.034740 0.036420 0.045740 0.055060 0.064390 0.073710 0.083030 0.036050 -0.010920 -0.057900 -0.046960 -0.036020 -0.025080 -0.014140 -0.035610 -0.057080 -0.078550 -0.063040 -0.047530 -0.032030 -0.016520 -0.001020 0.009220 0.019460 0.029700 0.039930 0.050170 0.060410 0.070650 0.080890 -0.001920 -0.084730 -0.070320 -0.055900 -0.041480 -0.052960 -0.064430 -0.075900 -0.087380 -0.098850 -0.067980 -0.037100 -0.006230 0.024650 0.055530 0.086400 0.117280 0.148150 0.087150 0.026150 -0.034850 -0.095840 -0.071000 -0.046160 -0.021320 0.003530 0.028370 0.053210 -0.004690 -0.062580 -0.120480 -0.099600 -0.078720 -0.057840 -0.036960 -0.016080 0.004800 0.025680 0.046560 0.067440 0.088320 0.109200 0.130080 0.109950 0.089820 0.069690 0.049550 0.040060 0.030560 0.021070 0.011580 0.007800 0.004020 0.000240 -0.003540 -0.007320 -0.011100 -0.007800 -0.004500 -0.001200 0.002100 0.005400 -0.008310 -0.022030 -0.035750 -0.049470 -0.063190 -0.050460 -0.037730 -0.025000 -0.012270 0.000460 0.004820 0.009190 0.013550 0.017910 0.022280 0.008830 -0.004620 -0.018070 -0.031520 -0.022760 -0.014010 -0.005260 0.003500 0.012250 0.021010 0.014370 0.007730 0.001100 0.008230 0.015370 0.022510 0.017130 0.011750 0.006370 0.013760 0.021140 0.028520 0.035910 0.043290 0.034580 0.025870 0.017150 0.008440 -0.000270 -0.008980 -0.001260 0.006450 0.014170 0.020390 0.026610 0.032830 0.039050 0.045270 0.036390 0.027500 0.018620 0.009740 0.000860 -0.013330 -0.027520 -0.041710 -0.028120 -0.014530 -0.000940 0.012640 0.026230 0.016900 0.007560 -0.001770 -0.011110 -0.020440 -0.029770 -0.039110 -0.024420 -0.009730 0.004960 0.019650

0.034340 0.020540 0.006740 -0.007060 -0.020860 -0.034660 -0.026630 -0.018600 -0.010570 -0.002540 -0.000630 0.001280 0.003190 0.005100 0.009990 0.014880 0.007910 0.000930 -0.006050 0.003420 0.012880 0.022350 0.031810 0.041280 0.027070 0.012870 -0.001340 -0.015540 -0.029750 -0.043950 -0.036120 -0.028280 -0.020440 -0.012600 -0.004760 0.003070 0.010910 0.009840 0.008760 0.007680 0.006610 0.012340 0.018070 0.023800 0.029530 0.035260 0.027840 0.020420 0.013000 -0.034150 -0.006280 -0.006210 -0.006150 -0.006090 -0.006020 -0.005960 -0.005900 -0.005830 -0.005770 -0.005710 -0.005640 -0.005580 -0.005520 -0.005450 -0.005390 -0.005320 -0.005260 -0.005200 -0.005130 -0.005070 -0.005010 -0.004940 -0.004880 -0.004820 -0.004750 -0.004690 -0.004630 -0.004560 -0.004500 -0.004440 -0.004370 -0.004310 -0.004250 -0.004180 -0.004120 -0.004060 -0.003990 -0.003930 -0.003870 -0.003800 -0.003740 -0.003680 -0.003610 -0.003550 -0.003490 -0.003420 -0.003360 -0.003300 -0.003230 -0.003170 -0.003110 -0.003040 -0.002980 -0.002920 -0.002850 -0.002790 -0.002730 -0.002660 -0.002600 -0.002540 -0.002470 -0.002410 -0.002350 -0.002280 -0.002220 -0.002160 -0.002090 -0.002030 -0.001970 -0.001900 -0.001840 -0.001780 -0.001710 -0.001650 -0.001580 -0.001520 -0.001460 -0.001390 -0.001330 -0.001270 -0.001200 -0.001140 -0.001080 -0.001010 -0.000950 -0.000890 -0.000820 -0.000760 -0.000700 -0.000630 -0.000570 -0.000510 -0.000440 -0.000380 -0.000320 -0.000250 -0.000190 -0.000130 -0.000060 0.000000 0.000000

The following is the source code of the software done in Visual Basic 6.0 environment. The comments are placed in start of the functions.

```
Dim h As Double                 //Variables Declaration
Dim m As Double
Dim wd As Double
Dim a As Double
Dim e As Double
Dim P(10000) As Double
Dim X(10000) As Double
Dim Y(10000) As Double
Dim Aa(10000) As Double
Dim B(10000) As Double

Dim term1 As Double
Dim term2 As Double
Dim u(10000) As Double
Dim all() As String
Dim t As Double
Dim n As Integer
Dim filenum
Dim str As String
Dim i As Integer
Dim j As Integer
Dim k As Integer

Dim scx As Double
Dim scy As Double
Dim scxr As Double
Dim scyr As Double
Dim scxrs As Double
Dim scyrs As Double

Dim tt As Double
Dim ttr As Double
Dim ttrs As Double

Dim x1 As Integer
Dim y1 As Integer
Dim x1r As Integer
Dim y1r As Integer
Dim x1rs As Integer
Dim y1rs As Integer

Dim max As Double
Dim min As Double
Dim maxr As Double
Dim minr As Double
Dim maxrs As Double
Dim minrs As Double
Dim ani As Integer
Dim anir As Integer
Dim anirs As Integer
```

```
Dim ct As Integer
Dim ctr As Integer
Dim ctrs As Integer
Dim no As Integer
Dim nor As Integer
Dim nors As Integer
Dim cur As Integer
Dim curr As Integer
Dim currs As Integer

Dim xval As Integer
Dim yval As Integer
Dim xvalr As Integer
Dim yvalr As Integer
Dim xvalrs As Integer
Dim yvalrs As Integer

Dim xx1 As Integer
Dim xx2 As Integer
Dim xx1r As Integer
Dim xx2r As Integer
Dim xx1rs As Integer
Dim xx2rs As Integer

Dim m1 As Double
Dim m2 As Double
Dim m3 As Double
Dim m4 As Double

Dim area As Double
Dim filenum1
Dim tn As Double
Dim eta As Double

Dim k11 As Double
Dim k12 As Double
Dim k13 As Double
Dim k14 As Double
Dim k15 As Double
Dim k16 As Double
Dim k21 As Double
Dim k22 As Double
Dim k23 As Double
Dim k24 As Double
Dim k25 As Double
Dim k26 As Double

Dim du(10000) As Double
Dim uu(10000) As Double
Dim ddu(10000) As Double
Dim acc(10000) As Double
Dim r(10000) As Double
Dim rs(10000) As Double

Dim wn As Double
```

```
Dim a11 As Double
Dim a12 As Double
Dim a21 As Double
Dim a22 As Double
Dim b11 As Double
Dim b12 As Double
Dim b21 As Double
Dim b22 As Double

Private Sub Command1_Click()          //Open Earthquake Data File
Text2.Text = "1"
Form2.Show (1)
End Sub

Public Function Calculate()           //Reads the ground response data from the file
On Error GoTo err1:
filenum = FreeFile
Open Form2.Text1.Text For Input As filenum
   str = Input(LOF(filenum), filenum)
   all = Split(str, " ")
   n = CInt(all(0))
   h = CDbl(all(1))
   List1.Clear
   List2.Clear
   List2.Clear
   Option6.Value = False
   Option7.Value = False
   Option8.Value = False
   Option16.Value = False
   Option17.Value = False
   Option15.Value = False
   Text8.Text = ""
   Text12.Text = ""
   For i = 2 To n + 1
      If (Option3.Value = False) Then
         P(i - 2) = CDbl(all(i)) * 386.0886
      Else
         P(i - 2) = CDbl(all(i))
      End If
   Next
   If (Option3.Value = True) Then
      For i = 0 To n - 1
         u(i) = P(i)
      Next
   End If
   If (Option4.Value = True) Then
      List1.Clear
      Cumm
   End If
   If (Option5.Value = True) Then
      Cumm
      For i = 0 To n - 1
         P(i) = u(i)
      Next
      Cumm
   End If
```

```
   For i = 0 To n - 1
      List1.AddItem ((i + 1) & "." & vbTab & u(i) & "")
   Next
Close filenum
Text3.Text = n
Text4.Text = h
max = Abs(u(0))
For i = 1 To n - 1
   If (Abs(u(i)) > Abs(max)) Then
      max = u(i)
   End If
Next
Text5.Text = max
Text7.Text = n * h
Exit Function
err1:
   MsgBox ("Error Opening File " & Form2.Text1.Text)
   List1.Clear
   List2.Clear
   List3.Clear
   Text8.Text = ""
   Text12.Text = ""
   Option6.Value = False
   Option7.Value = False
   Option8.Value = False
   Option16.Value = False
   Option17.Value = False
   Option15.Value = False
End Function

Public Function Calculater()                    //Reads the response data from the file
On Error GoTo err1:
filenum = FreeFile
Open Form2.Text1.Text For Input As filenum
   str = Input(LOF(filenum), filenum)
   all = Split(str, vbNewLine)
   n = CInt(all(0))
   h = CDbl(all(1))
   List1.Clear
   List2.Clear
   List3.Clear
   Option6.Value = False
   Option7.Value = False
   Option8.Value = False
   Option16.Value = False
   Option17.Value = False
   Option15.Value = False
   Text8.Text = ""
   Text12.Text = ""
   For i = 2 To n + 1
      If (Option6.Value = False) Then
         P(i - 2) = CDbl(all(i)) * 386.0886
      Else
         P(i - 2) = CDbl(all(i))
      End If
   Next
```

```
    For i = 0 To n - 1
        r(i) = P(i)
    Next
    For i = 0 To n - 1
        List2.AddItem ((i + 1) & "." & vbTab & r(i) & "")
    Next
Close filenum
Text3.Text = n
Text4.Text = h
max = Abs(r(0))
For i = 1 To n - 1
    If (Abs(r(i)) > Abs(max)) Then
        max = r(i)
    End If
Next
Text8.Text = max
Text7.Text = n * h
Exit Function
err1:
    MsgBox ("Error Opening File " & Form2.Text1.Text)
    List3.Clear
    List2.Clear
    List1.Clear
    Text8.Text = ""
    Text12.Text = ""
    Option6.Value = False
    Option7.Value = False
    Option8.Value = False
    Option16.Value = False
    Option17.Value = False
    Option15.Value = False
End Function

Public Function Calculaters()                    //Reads the response spectra data from the file
On Error GoTo err22:
filenum = FreeFile
Open Form2.Text1.Text For Input As filenum
    str = Input(LOF(filenum), filenum)
    all = Split(str, vbNewLine)
    n = CInt(all(0))
    tn = CDbl(all(1))
    List1.Clear
    List2.Clear
    List3.Clear
    Option6.Value = False
    Option7.Value = False
    Option8.Value = False
    Option16.Value = False
    Option17.Value = False
    Option15.Value = False
    Text5.Text = ""
    Text8.Text = ""
    Text12.Text = ""
    For i = 2 To n + 1
        If (Option17.Value = False) Then
            P(i - 2) = CDbl(all(i)) * 386.0886
```

```
        Else
            P(i - 2) = CDbl(all(i))
        End If
    Next
    For i = 0 To n - 1
        rs(i) = P(i)
    Next
    For i = 0 To n - 1
        List3.AddItem ((i + 1) & "." & vbTab & rs(i) & "")
    Next
Close filenum
Text3.Text = n
Text4.Text = tn
maxrs = Abs(rs(0))
For i = 1 To n - 1
    If (Abs(r(i)) > Abs(maxrs)) Then
        maxrs = rs(i)
    End If
Next
Text12.Text = maxrs
Text7.Text = n * tn
Exit Function
err22:
    MsgBox ("Error Opening File " & Form2.Text1.Text)
    List3.Clear
    List2.Clear
    List1.Clear
    Text8.Text = ""
    Text12.Text = ""
    Option16.Value = False
    Option17.Value = False
    Option15.Value = False
End Function

Private Sub Command2_Click()                    //Draws the Ground Response
On Error GoTo err2:
    Form3.Shape1.Visible = False
    Form3.Label14.Visible = False
If (List1.ListCount > 0) Then
    If (Option3.Value = True) Then
        Form3.Caption = "Ground Acceleration Response"
        Form3.Label1.Caption = "Ground Acceleration Response"
        Form3.Text4.Text = "ACCELERATION"
    End If
    If (Option4.Value = True) Then
        Form3.Caption = "Ground Velocity Response"
        Form3.Label1.Caption = "Ground Velocity Response"
        Form3.Text4.Text = "VELOCITY"
    End If
    If (Option5.Value = True) Then
        Form3.Caption = "Ground Displacement Response"
        Form3.Label1.Caption = "Ground Displacement Response"
        Form3.Text4.Text = "DISPLACEMENT"
    End If
    xx1 = 0
    xx2 = n - 1
```

```
mark3
ct = 0
Form3.Label4.Caption = Text7.Text
Form3.Show
Form3.Text1.Text = "1"
scx = 10500 / (n * h)
max = u(0)
min = u(0)
For i = 1 To n - 1
   If (u(i) > max) Then
      max = u(i)
   End If
   If (u(i) < min) Then
      min = u(i)
   End If
Next
scy = (yval * 1.3) / (max - min)
Form3.Label10.Caption = CInt(100 * 1477.5 / scy) / 100
Form3.Label11.Caption = CInt(100 * (-1) * 1477.5 / scy) / 100
Form3.Label12.Caption = CInt(100 * 1477.5 * 2 / scy) / 100
Form3.Label13.Caption = CInt(100 * (-1) * 1477.5 * 2 / scy) / 100
tt = h * 2
no = Int((n / 30000) * 33)
If (no = 0) Then
   no = 1
End If
If (Option1.Value = False) Then
   Form3.Line (xval, yval)-(xval + Int(tt * scx), yval - Int(u(0) * scy))
   x1 = xval + Int(tt * scx)
   y1 = yval - Int(u(0) * scy)
   tt = tt + h
   For i = 1 To n - 1
      Form3.Line (x1, y1)-(xval + Int(tt * scx), yval - Int(u(i) * scy))
      If (Abs(max) < Abs(min) And u(i) = min) Then
         Form3.Shape1.Left = xval + Int(tt * scx) - 75
         Form3.Shape1.Top = yval - Int(u(i) * scy) - 75
         Form3.Label14.Left = xval + Int(tt * scx) + 100
         Form3.Label14.Top = yval - Int(u(i) * scy)
         Form3.Label14.Caption = CInt(100 * min) / 100
         Form3.Shape1.Visible = True
         Form3.Label14.Visible = True
      End If
      If (Abs(max) > Abs(min) And u(i) = max) Then
         Form3.Shape1.Left = xval + Int(tt * scx) - 75
         Form3.Shape1.Top = yval - Int(u(i) * scy) - 75
         Form3.Label14.Left = xval + Int(tt * scx) + 100
         Form3.Label14.Top = yval - Int(u(i) * scy)
         Form3.Label14.Caption = CInt(1000 * max) / 100
         Form3.Shape1.Visible = True
         Form3.Label14.Visible = True
      End If
      x1 = xval + Int(tt * scx)
      y1 = yval - Int(u(i) * scy)
      tt = tt + h
   Next
   Form3.Text1.Text = ""
```

```
   Else
      Timer1.Interval = 1
      Exit Sub
   End If
End If
Exit Sub
err2:
   MsgBox ("Error In Drawing Ground Response")
   Form3.Visible = False
   Exit Sub
End Sub

Private Sub Command3_Click()              //Open Ground Response File
Text2.Text = "2"
Form2.Show (1)
End Sub

Private Sub Command4_Click()              //Open Response File
Text2.Text = "3"
Form2.Show (1)
End Sub

Private Sub Command5_Click()              //Draws the Response
On Error GoTo err12:
   Form5.Shape1.Visible = False
   Form5.Label14.Visible = False
If (List2.ListCount > 0) Then
   If (Text6.Text <> "" And Text7.Text <> "") Then
      wn = CDbl(Text6.Text)
      eta = CDbl(Text7.Text)
   End If
   If (Option6.Value = True) Then
      Form5.Caption = "Acceleration Response"
      Form5.Label1.Caption = "Acceleration Response"
      Form5.Text4.Text = "ACCELERATION"
   End If
   If (Option7.Value = True) Then
      Form5.Caption = "Displacement Response"
      Form5.Label1.Caption = "Displacement Response"
      Form5.Text4.Text = "DISPLACEMENT"
   End If
   If (Option8.Value = True) Then
      Form5.Caption = "Velocity Response"
      Form5.Label1.Caption = "Velocity Response"
      Form5.Text4.Text = "VELOCITY"
   End If
   xx1r = 0
   xx2r = n - 1
   mark5
   ctr = 0
   Form5.Label4.Caption = Text7.Text
   Form5.Show
   Form5.Text1.Text = "1"
   scxr = 10500 / (n * h)
   maxr = r(0)
   minr = r(0)
```

```
    For i = 1 To n - 1
       If (r(i) > maxr) Then
          maxr = r(i)
       End If
       If (r(i) < minr) Then
          minr = r(i)
       End If
    Next
    scyr = (yvalr * 1.3) / (maxr - minr)
    Form5.Label10.Caption = CInt(100 * 1477.5 / scyr) / 100
    Form5.Label11.Caption = CInt(100 * (-1) * 1477.5 / scyr) / 100
    Form5.Label12.Caption = CInt(100 * 1477.5 * 2 / scyr) / 100
    Form5.Label13.Caption = CInt(100 * (-1) * 1477.5 * 2 / scyr) / 100
    ttr = h
    nor = Int((n / 30000) * 33)
    If (nor = 0) Then
       nor = 1
    End If
    If (Option10.Value = False) Then
       Form5.Line (xvalr, yvalr)-(xvalr + Int(ttr * scxr), yvalr - Int(r(0) * scyr))
       x1r = xvalr + Int(ttr * scxr)
       y1r = yvalr - Int(r(0) * scyr)
       ttr = ttr + h
       For i = 1 To n - 1
          Form5.Line (x1r, y1r)-(xvalr + Int(ttr * scxr), yvalr - Int(r(i) * scyr))
          If (Abs(maxr) < Abs(min) And r(i) = minr) Then
             Form5.Shape1.Left = xvalr + Int(ttr * scxr) - 75
             Form5.Shape1.Top = yvalr - Int(r(i) * scyr) - 75
             Form5.Label14.Left = xvalr + Int(ttr * scxr) + 100
             Form5.Label14.Top = yvalr - Int(r(i) * scyr)
             Form5.Label14.Caption = CInt(100 * minr) / 100
             Form5.Shape1.Visible = True
             Form5.Label14.Visible = True
          End If
          If (Abs(maxr) > Abs(minr) And r(i) = maxr) Then
             Form5.Shape1.Left = xvalr + Int(ttr * scxr) - 75
             Form3.Shape1.Top = yvalr - Int(r(i) * scyr) - 75
             Form5.Label14.Left = xvalr + Int(ttr * scxr) + 100
             Form5.Label14.Top = yvalr - Int(r(i) * scyr)
             Form5.Label14.Caption = CInt(1000 * maxr) / 100
             Form5.Shape1.Visible = True
             Form5.Label14.Visible = True
          End If
          x1r = xvalr + Int(ttr * scxr)
          y1r = yvalr - Int(r(i) * scyr)
          ttr = ttr + h
       Next
       Form5.Text1.Text = ""
    Else
       Timer2.Interval = 1
       Exit Sub
    End If
End If
Exit Sub
err12:
    'MsgBox ("Error In Drawing Response")
```

```
    'Form5.Visible = False
    Resume Next
    Exit Sub
End Sub

Private Sub Command6_Click()                //Open Response Spectra File
Text2.Text = "4"
Form2.Show (1)
End Sub

Private Sub Command7_Click()                //Draws Response Spectra
On Error GoTo err20:
    Form6.Shape1.Visible = False
    Form6.Label14.Visible = False
If (List3.ListCount > 0) Then
    If (Text10.Text <> "" And Text11.Text <> "") Then
        wn = CDbl(Text10.Text)
        eta = CDbl(Text11.Text)
    End If
    If (Option17.Value = True) Then
        Form6.Caption = "Acceleration Response Spectra"
        Form6.Label1.Caption = "Acceleration Response Spectra"
        Form6.Text4.Text = "ACCELERATION"
    End If
    If (Option16.Value = True) Then
        Form6.Caption = "Displacement Response Spectra"
        Form6.Label1.Caption = "Displacement Response Spectra"
        Form6.Text4.Text = "DISPLACEMENT"
    End If
    If (Option15.Value = True) Then
        Form6.Caption = "Velocity Response Spectra"
        Form6.Label1.Caption = "Velocity Response Spectra"
        Form6.Text4.Text = "VELOCITY"
    End If
    xx1rs = 0
    xx2rs = n - 1
    mark6
    ctrs = 0
    Form6.Label4.Caption = n * tn
    Form6.Show
    Form6.Text1.Text = "1"
    scxrs = 10500 / (n * CDbl(Text10.Text))
    maxrs = rs(0)
    minrs = rs(0)
    For i = 1 To n - 1
        If (rs(i) > maxrs) Then
            maxrs = rs(i)
        End If
        If (rs(i) < minrs) Then
            minrs = rs(i)
        End If
    Next
    scyrs = (yvalrs) / (maxrs * 1.3) ' - minrs)
    Form6.Label10.Caption = 1477.5 / scyrs 'CInt(100 * 1477.5 / scyrs) / 100
    Form6.Label11.Caption = "" 'CInt(100 * (-1) * 1477.5 / scyrs) / 100
    Form6.Label12.Caption = 1477.5 / scyrs * 2 'CInt(100 * 1477.5 * 2 / scyrs) / 100
```

```
   Form6.Label13.Caption = "" 'CInt(100 * (-1) * 1477.5 * 2 / scyrs) / 100
   ttrs = 2 * CDbl(Text10.Text)
   nors = Int((n / 30000) * 15)
   If (nors = 0) Then
      nors = 1
   End If
   If (Option11.Value = False) Then
      Form6.Line (xvalrs, yvalrs)-(xvalrs + Int(ttrs * scxrs), yvalrs - Int(rs(0) * scyrs))
      x1rs = xvalrs + Int(ttrs * scxrs)
      y1rs = yvalrs - Int(rs(0) * scyrs)
      ttrs = CDbl(Text10.Text)
      For i = 1 To n - 1
         Form6.Line (x1rs, y1rs)-(xvalrs + Int(ttrs * scxrs), yvalrs - Int(rs(i) * scyrs))
         'If (Abs(maxrs) < Abs(minrs) And rs(i) = minrs) Then
            'Form6.Shape1.Left = xvalrs + Int(ttrs * scxrs) - 75
            'Form6.Shape1.Top = yvalrs - Int(rs(i) * scyrs) - 75
            'Form6.Label14.Left = xvalrs + Int(ttrs * scxrs) + 100
            'Form6.Label14.Top = yvalrs - Int(rs(i) * scyrs)
            'Form6.Label14.Caption = minrs 'CInt(100 * minrs) / 100
            'Form6.Shape1.Visible = True
            'Form6.Label14.Visible = True
         'End If
         If (rs(i) = maxrs) Then
            Form6.Shape1.Left = xvalrs + Int(ttrs * scxrs) - 75
            Form6.Shape1.Top = yvalrs - Int(rs(i) * scyrs) - 75
            Form6.Label14.Left = xvalrs + Int(ttrs * scxrs) + 100
            Form6.Label14.Top = yvalrs - Int(rs(i) * scyrs)
            Form6.Label14.Caption = maxrs 'CInt(1000 * maxrs) / 100
            Form6.Shape1.Visible = True
            Form6.Label14.Visible = True
         End If
         x1rs = xvalrs + Int(ttrs * scxrs)
         y1rs = yvalrs - Int(rs(i) * scyrs)
         ttrs = ttrs + CDbl(Text10.Text)
      Next
      Form6.Text1.Text = ""
   Else
      Timer3.Interval = 1
      Exit Sub
   End If
End If
Exit Sub
err20:
   MsgBox ("Error In Drawing Response Spectra")
   Form6.Visible = False
   Exit Sub
End Sub

Private Sub Form_Load()          //Initialization of variables
ct = 0
ctr = 0
ctrs = 0
xval = 840
xvalr = 840
xvalrs = 840
yval = 3800
```

```
yvalr = 3800
yvalrs = 3800
StatusBar2.Panels(3).Text = Format(Date, "dddd")
eta = 0.6
wn = 1
End Sub

Private Sub Form_Unload(Cancel As Integer)                    //End the Program
End
End Sub

Private Sub Option12_Click()                    //Runge-Kutta-Fehlberg Method for Response
If (Option6.Value = True) Then
   Option6_Click
End If
If (Option7.Value = True) Then
   Option7_Click
End If
If (Option8.Value = True) Then
   Option8_Click
End If
End Sub

Private Sub Option13_Click()                    //Runge-Kutta Method for Response
If (Option6.Value = True) Then
   Option6_Click
End If
If (Option7.Value = True) Then
   Option7_Click
End If
If (Option8.Value = True) Then
   Option8_Click
End If
End Sub

Private Sub Option15_Click()                    //Velocity Response Spectra
On Error GoTo err18:
If (List1.ListCount > 0) Then
   tn = CDbl(Text10.Text)
   eta = CDbl(Text11.Text)
   For i = 2 To n + 1
      P(i - 2) = CDbl(all(i)) * 386.0886
   Next
   wn = 0
   List3.Clear
   'For i = 0 To n - 1
   k = 0
   Do While (k < n)
      If (Option19.Value = True) Then
         RKF
      End If
      If (Option18.Value = True) Then
         RK
      End If
      If (Option21.Value = True) Then
         AMF
```

```
        End If
        If (Option23.Value = True) Then
            IBE
        End If
        maxrs = Abs(ddu(0))
        For j = 1 To n - 1
            If (Abs(ddu(j)) > Abs(maxrs)) Then
                maxrs = ddu(j)
            End If
        Next
        'List3.AddItem ((i + 1) & "." & vbTab & wn)
        rs(k) = Abs(maxrs)
        wn = wn + tn
        k = k + 1
    Loop
    'Next
    Label4.Caption = "Velocity Spectral Values"
    For i = 0 To n - 1
        'rs(i) = ddu(i)
        List3.AddItem ((i + 1) & "." & vbTab & rs(i))
    Next
    maxrs = Abs(rs(0))
    For i = 1 To n - 1
        If (Abs(rs(i)) > Abs(maxrs)) Then
            maxrs = rs(i)
        End If
    Next
    Text12.Text = maxrs
Else
    Option15.Value = False
End If
Exit Sub
err18:
    Option15.Value = False
    MsgBox ("Enter Valid Values For Time Period And Damping Ratio")
End Sub

Private Sub Option16_Click()                    //Displacement Response Spectra
On Error GoTo err19:
If (List1.ListCount > 0) Then
    tn = CDbl(Text10.Text)
    eta = CDbl(Text11.Text)
    For i = 2 To n + 1
        P(i - 2) = CDbl(all(i)) * 386.0886
    Next
    wn = 0
    List3.Clear
    'For i = 0 To n - 1
    k = 0
    Do While (k < n)
        If (Option19.Value = True) Then
            RKF
        End If
        If (Option18.Value = True) Then
            RK
        End If
```

```
      If (Option21.Value = True) Then
         AMF
      End If
      If (Option23.Value = True) Then
         IBE
      End If
      maxrs = Abs(du(0))
      For j = 1 To n - 1
         If (Abs(du(j)) > Abs(maxrs)) Then
            maxrs = du(j)
         End If
      Next
      'List3.AddItem ((i + 1) & "." & vbTab & wn)
      rs(k) = Abs(maxrs)
      wn = wn + tn
      k = k + 1
   Loop
   'Next
   Label4.Caption = "Displacement Spectral Values"
   For i = 0 To n - 1
      'rs(i) = ddu(i)
      List3.AddItem ((i + 1) & "." & vbTab & rs(i))
   Next
   maxrs = Abs(rs(0))
   For i = 1 To n - 1
      If (Abs(rs(i)) > Abs(maxrs)) Then
         maxrs = rs(i)
      End If
   Next
   Text12.Text = maxrs
Else
   Option16.Value = False
End If
Exit Sub
err19:
   Option16.Value = False
   MsgBox ("Enter Valid Values For Time Period And Damping Ratio")
End Sub

Private Sub Option17_Click()                    //Acceleration Response Spectra
On Error GoTo err17:
If (List1.ListCount > 0) Then
   tn = CDbl(Text10.Text)
   eta = CDbl(Text11.Text)
   For i = 2 To n + 1
      P(i - 2) = CDbl(all(i)) * 386.0886
   Next
   wn = 0
   List3.Clear
   'For i = 0 To n - 1
   k = 0
   Do While (k < n)
      If (Option19.Value = True) Then
         RKF
      End If
      If (Option18.Value = True) Then
```

```
        RK
      End If
      If (Option21.Value = True) Then
        AMF
      End If
      If (Option23.Value = True) Then
        IBE
      End If
      maxrs = Abs(acc(0))
      For j = 1 To n - 1
        If (Abs(acc(j)) > Abs(maxrs)) Then
          maxrs = acc(j)
        End If
      Next
      'List3.AddItem ((i + 1) & "." & vbTab & wn)
      rs(k) = Abs(maxrs)
      wn = wn + tn
      k = k + 1
    Loop
    'Next
    Label4.Caption = "Velocity Spectral Values"
    For i = 0 To n - 1
      'rs(i) = ddu(i)
      List3.AddItem ((i + 1) & "." & vbTab & rs(i))
    Next
    maxrs = Abs(rs(0))
    For i = 1 To n - 1
      If (Abs(rs(i)) > Abs(maxrs)) Then
        maxrs = rs(i)
      End If
    Next
    Text12.Text = maxrs
  Else
    Option17.Value = False
  End If
  Exit Sub
  err17:
    Option17.Value = False
    MsgBox ("Enter Valid Values For Time Period And Damping Ratio")
  End Sub

  Private Sub Option18_Click()          //Runge-Kutta Method for Response Spectra
  If (Option17.Value = True) Then
    Option17_Click
  End If
  If (Option15.Value = True) Then
    Option15_Click
  End If
  If (Option16.Value = True) Then
    Option16_Click
  End If
  End Sub

  Private Sub Option19_Click()          //Runge-Kutta-Fehlberg Method for Response Spectra
  If (Option17.Value = True) Then
    Option17_Click
```

```
End If
If (Option15.Value = True) Then
   Option15_Click
End If
If (Option16.Value = True) Then
   Option16_Click
End If
End Sub

Private Sub Option20_Click()          //Adam-Moulton Method for Response
If (Option6.Value = True) Then
   Option6_Click
End If
If (Option7.Value = True) Then
   Option7_Click
End If
If (Option8.Value = True) Then
   Option8_Click
End If
End Sub

Private Sub Option21_Click()          //Adam-Moulton Method for Response Spectra
If (Option17.Value = True) Then
   Option17_Click
End If
If (Option15.Value = True) Then
   Option15_Click
End If
If (Option16.Value = True) Then
   Option16_Click
End If
End Sub

Private Sub Option22_Click()          //Interpolation by Excitation for Response
If (Option6.Value = True) Then
   Option6_Click
End If
If (Option7.Value = True) Then
   Option7_Click
End If
If (Option8.Value = True) Then
   Option8_Click
End If
End Sub

Private Sub Option23_Click()          //Interpolation by Excitation for Response Spectra
If (Option17.Value = True) Then
   Option17_Click
End If
If (Option15.Value = True) Then
   Option15_Click
End If
If (Option16.Value = True) Then
   Option16_Click
End If
End Sub
```

```
Private Sub Option3_Click()                    //Acceleration Ground Response
If (List1.ListCount > 0) Then
   Label1.Caption = "Ground Acceleration Values"
   List1.Clear
   For i = 2 To n + 1
      P(i - 2) = CDbl(all(i))
   Next
   If (Option3.Value = True) Then
      For i = 0 To n - 1
         u(i) = P(i)
      Next
   End If
   For i = 0 To n - 1
      List1.AddItem ((i + 1) & "." & vbTab & u(i) & "")
   Next
   Text3.Text = n
   Text4.Text = h
   max = Abs(u(0))
   For i = 1 To n - 1
      If (Abs(u(i)) > Abs(max)) Then
         max = u(i)
      End If
   Next
   Text5.Text = max
   Text7.Text = n * h
End If
End Sub

Private Sub Option4_Click()                    //Velocity Ground Response
If (List1.ListCount > 0) Then
   Label1.Caption = "Ground Velocity Values"
   List1.Clear
   For i = 2 To n + 1
      P(i - 2) = CDbl(all(i)) * 386.0886
   Next
   If (Option4.Value = True) Then
      List1.Clear
      Cumm
   End If
   For i = 0 To n - 1
      List1.AddItem ((i + 1) & "." & vbTab & u(i) & "")
   Next
   Text3.Text = n
   Text4.Text = h
   max = Abs(u(0))
   For i = 1 To n - 1
      If (Abs(u(i)) > Abs(max)) Then
         max = u(i)
      End If
   Next
   Text5.Text = max
   Text7.Text = n * h
End If
End Sub
```

```
Private Sub Option5_Click()                    //Displacement Ground Response
If (List1.ListCount > 0) Then
   Label1.Caption = "Ground Displacement Values"
   List1.Clear
   For i = 2 To n + 1
      P(i - 2) = CDbl(all(i)) * 386.0886
   Next
   If (Option5.Value = True) Then
      Cumm
      For i = 0 To n - 1
         P(i) = u(i)
      Next
      Cumm
   End If
   For i = 0 To n - 1
      List1.AddItem ((i + 1) & "." & vbTab & u(i) & "")
   Next
   Text3.Text = n
   Text4.Text = h
   max = Abs(u(0))
   For i = 1 To n - 1
      If (Abs(u(i)) > Abs(max)) Then
         max = u(i)
      End If
   Next
   Text5.Text = max
   Text7.Text = n * h
End If
End Sub

Private Sub Option6_Click()                    //Acceleration Response
On Error GoTo err13:
If (List1.ListCount > 0) Then
   wn = CDbl(Text6.Text)
   eta = CDbl(Text9.Text)
   List2.Clear
   Label3.Caption = "Acceleration Response Values"
   For i = 0 To n - 1
      P(i) = all(i + 2)
   Next
   If (Option12.Value = True) Then
      RKF
   End If
   If (Option13.Value = True) Then
      RK
   End If
   If (Option20.Value = True) Then
      AMF
   End If
   If (Option22.Value = True) Then
      IBE
   End If
   For i = 0 To n - 1
      r(i) = acc(i)
      List2.AddItem ((i + 1) & "." & vbTab & acc(i))
   Next
```

```
   maxr = Abs(acc(0))
   For i = 1 To n - 1
      If (Abs(acc(i)) > Abs(maxr)) Then
         maxr = acc(i)
      End If
   Next
   Text8.Text = maxr
Else
   Option6.Value = False
End If
Exit Sub
err13:
   Option6.Value = False
   MsgBox ("Enter Valid Values For Time Period And Damping Ratio")
End Sub

Private Sub Option7_Click()                    //Velocity Response
On Error GoTo err14:
If (List1.ListCount > 0) Then
   wn = CDbl(Text6.Text)
   eta = CDbl(Text9.Text)
   For i = 2 To n + 1
      P(i - 2) = CDbl(all(i)) * 386.0886
   Next
   If (Option12.Value = True) Then
      RKF
   End If
   If (Option13.Value = True) Then
      RK
   End If
   If (Option20.Value = True) Then
      AMF
   End If
   If (Option22.Value = True) Then
      IBE
   End If
   List2.Clear
   Label3.Caption = "Displacement Response Values"
   For i = 0 To n - 1
      r(i) = du(i)
      List2.AddItem ((i + 1) & "." & vbTab & du(i))
   Next
   maxr = Abs(du(0))
   For i = 1 To n - 1
      If (Abs(du(i)) > Abs(maxr)) Then
         maxr = du(i)
      End If
   Next
   Text8.Text = maxr
End If
Exit Sub
err14:
   Option7.Value = False
   MsgBox ("Enter Valid Values For Time Period And Damping Ratio")
End Sub
```

```
Private Sub Option8_Click()                    //Displacement Response
On Error GoTo err15:
If (List1.ListCount > 0) Then
   wn = CDbl(Text6.Text)
   eta = CDbl(Text9.Text)
   For i = 2 To n + 1
      P(i - 2) = CDbl(all(i)) * 386.0886
   Next
   If (Option12.Value = True) Then
      RKF
   End If
   If (Option13.Value = True) Then
      RK
   End If
   If (Option20.Value = True) Then
      AMF
   End If
   If (Option22.Value = True) Then
      IBE
   End If
   List2.Clear
   Label3.Caption = "Velocity Response Values"
   For i = 0 To n - 1
      r(i) = ddu(i)
      List2.AddItem ((i + 1) & "." & vbTab & ddu(i))
   Next
   maxr = Abs(ddu(0))
   For i = 1 To n - 1
      If (Abs(ddu(i)) > Abs(maxr)) Then
         maxr = ddu(i)
      End If
   Next
   Text8.Text = maxr
Else
   Option8.Value = False
End If
Exit Sub
err15:
   Option8.Value = False
   MsgBox ("Enter Valid Values For Time Period And Damping Ratio")
End Sub

Private Sub Text10_KeyPress(KeyAscii As Integer)          //Time Period for Response Spectra
On Error GoTo err23:
If (KeyAscii = 13 And wn <> CDbl(Text10.Text)) Then
   If (Option17.Value = True) Then
      Option17_Click
   End If
   If (Option15.Value = True) Then
      Option15_Click
   End If
   If (Option16.Value = True) Then
      Option16_Click
   End If
End If
Exit Sub
```

```
err23:
   Exit Sub
End Sub

Private Sub Text11_KeyPress(KeyAscii As Integer)          //Damping for Response Spectra
On Error GoTo err24:
If (KeyAscii = 13 And eta <> CDbl(Text11.Text)) Then
   If (Option17.Value = True) Then
      Option17_Click
   End If
   If (Option15.Value = True) Then
      Option15_Click
   End If
   If (Option16.Value = True) Then
      Option16_Click
   End If
End If
Exit Sub
err24:
   Exit Sub
End Sub

Private Sub Text6_KeyPress(KeyAscii As Integer)           //Natural Frequency for Response
On Error GoTo err16:
If (KeyAscii = 13 And wn <> CDbl(Text6.Text)) Then
   If (Option6.Value = True) Then
      Option6_Click
   End If
   If (Option7.Value = True) Then
      Option7_Click
   End If
   If (Option8.Value = True) Then
      Option8_Click
   End If
End If
Exit Sub
err16:
   Exit Sub
End Sub

Private Sub Text9_KeyPress(KeyAscii As Integer)           //Damping for Response
On Error GoTo err17:
If (KeyAscii = 13 And eta <> CDbl(Text9.Text)) Then
   If (Option6.Value = True) Then
      Option6_Click
   End If
   If (Option7.Value = True) Then
      Option7_Click
   End If
   If (Option8.Value = True) Then
      Option8_Click
   End If
End If
err17:
   Exit Sub
End Sub
```

```
Private Sub Timer1_Timer()                    //Animate Ground Response
On Error GoTo err6:
Do While (cur < no)
   If (ct > n - 1) Then
      ani = 0
      Timer1.Interval = 0
      Form3.Text1.Text = ""
      Exit Sub
   End If
   If (ct = 0) Then
      Form3.Line (xval, yval)-(xval + Int(tt * scx), yval - Int(u(0) * scy))
      x1 = xval + Int(tt * scx)
      y1 = yval - Int(u(0) * scy)
      tt = tt + h
   Else
      Form3.Line (x1, y1)-(xval + Int(tt * scx), yval - Int(u(ct) * scy))
      If (Abs(max) < Abs(min) And u(ct) = min) Then
         Form3.Shape1.Left = xval + Int(tt * scx) - 75
         Form3.Shape1.Top = yval - Int(u(ct) * scy) - 75
         Form3.Label14.Left = xval + Int(tt * scx) + 100
         Form3.Label14.Top = yval - Int(u(ct) * scy)
         Form3.Label14.Caption = CInt(100 * min) / 100
         Form3.Shape1.Visible = True
         Form3.Label14.Visible = True
      End If
      If (Abs(max) > Abs(min) And u(ct) = max) Then
         Form3.Shape1.Left = xval + Int(tt * scx) - 75
         Form3.Shape1.Top = yval - Int(u(ct) * scy) - 75
         Form3.Label14.Left = xval + Int(tt * scx) + 100
         Form3.Label14.Top = yval - Int(u(ct) * scy)
         Form3.Label14.Caption = CInt(100 * max) / 100
         Form3.Shape1.Visible = True
         Form3.Label14.Visible = True
      End If
      x1 = xval + Int(tt * scx)
      y1 = yval - Int(u(ct) * scy)
      tt = tt + h
   End If
   ct = ct + 1
   cur = cur + 1
Loop
Exit Sub
err6:
   Resume Next
End Sub

Public Function save()                    //Save Ground Response
filenum1 = FreeFile
Open Text1 For Output As filenum1
   Print #filenum1, n & ""
   Print #filenum1, h & ""
   For i = 0 To n - 1
      Print #filenum1, u(i) & ""
   Next
Close filenum1
End Function
```

```
Public Function saver()                 //Save Response
filenum1 = FreeFile
Open Text1 For Output As filenum1
   Print #filenum1, n & ""
   Print #filenum1, h & ""
   For i = 0 To n - 1
      Print #filenum1, r(i) & ""
   Next
Close filenum1
End Function

Public Function savers()                //Save Response Spectrum
filenum1 = FreeFile
Open Text1 For Output As filenum1
   Print #filenum1, n & ""
   Print #filenum1, tn & ""
   For i = 0 To n - 1
      Print #filenum1, rs(i) & ""
   Next
Close filenum1
End Function

Public Function CalculateDisp()                 //Calculate Displacement
On Error GoTo err1:
filenum = FreeFile
Open Form2.Text1.Text For Input As filenum
   str = Input(LOF(filenum), filenum)
   all = Split(str, vbNewLine)
   n = CInt(all(0))
   h = CDbl(all(1))
   List1.Clear
   Option6.Value = False
   Option7.Value = False
   Option8.Value = False
   For i = 2 To n + 1
      u(i - 2) = CDbl(all(i))
      List1.AddItem ((i - 1) & "." & vbTab & u(i - 2) & "")
   Next
Close filenum
Text3.Text = n
Text4.Text = h
max = Abs(u(0))
For i = 1 To n - 1
   If (Abs(u(i)) > Abs(max)) Then
      max = u(i)
   End If
Next
Text5.Text = max
Text7.Text = n * h
Exit Function
err1:
   MsgBox ("Error Opening File " & Form2.Text1.Text)
   List1.Clear
   List2.Clear
   Text8.Text = ""
   Option6.Value = False
```

```
   Option7.Value = False
   Option8.Value = False
End Function

Public Function Repaint()                    //Repaint Ground Response
On Error GoTo err4:
   scx = 10500 / ((xx2 - xx1 + 1) * h)
   max = u(xx1)
   min = u(xx1)
   For i = xx1 To xx2
      If (u(i) > max) Then
         max = u(i)
      End If
      If (u(i) < min) Then
         min = u(i)
      End If
   Next
   scy = (yval * 1.3) / (max - min)
   tt = h
   Form3.Line (xval, yval)-(xval + Int(tt * scx), yval - Int(u(xx1) * scy))
   x1 = xval + Int(tt * scx)
   y1 = yval - Int(u(xx1) * scy)
   tt = tt + h
   For i = xx1 To xx2
      Form3.Line (x1, y1)-(xval + Int(tt * scx), yval - Int(u(i) * scy))
      x1 = xval + Int(tt * scx)
      y1 = yval - Int(u(i) * scy)
      tt = tt + h
   Next
Exit Function
err4:
   Resume Next
End Function

Public Function Repaintr()                   //Repaint Response
On Error GoTo err10:
   scxr = 10500 / ((xx2r - xx1r + 1) * h)
   maxr = r(xx1r)
   minr = r(xx1r)
   For i = xx1r To xx2r
      If (r(i) > maxr) Then
         maxr = r(i)
      End If
      If (r(i) < minr) Then
         minr = r(i)
      End If
   Next
   scyr = (yvalr * 1.3) / (maxr - minr)
   ttr = h
   Form5.Line (xvalr, yvalr)-(xvalr + Int(ttr * scxr), yvalr - Int(r(xx1r) * scyr))
   x1r = xvalr + Int(ttr * scxr)
   y1r = yvalr - Int(r(xx1r) * scyr)
   ttr = ttr + h
   For i = xx1r To xx2r
      Form5.Line (x1r, y1r)-(xvalr + Int(ttr * scxr), yvalr - Int(r(i) * scyr))
      x1r = xvalr + Int(ttr * scxr)
```

```
      y1r = yvalr - Int(r(i) * scyr)
      ttr = ttr + h
   Next
Exit Function
err10:
   Resume Next
End Function

Public Function Repaintrs()                    //Repaint Response Spectrum
On Error GoTo err25:
   scxrs = 10500 / ((xx2rs - xx1rs + 1) * tn)
   maxrs = rs(xx1rs)
   minrs = rs(xx1rs)
   For i = xx1rs To xx2rs
      If (rs(i) > maxrs) Then
         maxrs = rs(i)
      End If
      If (rs(i) < minrs) Then
         minrs = rs(i)
      End If
   Next
   scyrs = (yvalrs) / (maxrs * 1.3) '- minrs)
   ttrs = 0
   Form6.Line (xvalrs, yvalrs)-(xvalrs + Int(ttrs * scxrs), yvalrs - Int(r(xx1rs) * scyrs))
   x1rs = xvalrs + Int(ttrs * scxrs)
   y1rs = yvalrs - Int(rs(xx1rs) * scyrs)
   ttrs = ttrs + tn
   For i = xx1rs To xx2rs
      Form6.Line (x1rs, y1rs)-(xvalrs + Int(ttrs * scxrs), yvalrs - Int(rs(i) * scyrs))
      If (rs(i) = maxrs) Then
         Form6.Shape1.Left = x1rs + Int(ttrs * scxrs) - 75
         Form6.Shape1.Top = y1rs - Int(rs(i) * scyrs) - 75
         Form6.Label14.Left = x1rs + Int(ttrs * scxrs) + 100
         Form6.Label14.Top = y1rs - Int(rs(i) * scyrs)
         Form6.Label14.Caption = maxrs 'CInt(100 * maxrs) / 100
         Form6.Shape1.Visible = True
         Form6.Label14.Visible = True
      End If
      x1rs = xvalrs + Int(ttrs * scxrs)
      y1rs = yvalrs - Int(rs(i) * scyrs)
      ttrs = ttrs + tn
   Next
Exit Function
err25:
   Resume Next
End Function

Public Function Zoom()                 //Zoom Ground Response
xx1 = CInt(Form3.Text2.Text)
xx2 = CInt(Form3.Text3.Text)
xx1 = Int(xx1 / (scx * h))
xx2 = Int(xx2 / (scx * h))
Form3.Label3.Caption = (xx1 * h)
Form3.Label4.Caption = (xx2 * h)
If (xx2 > n - 1) Then
   xx2 = n - 1
```

```
End If
If (xx1 < 0) Then
   xx1 = 0
End If
mark3
End Function

Public Function Zoomr()              //Zoom Response
xx1r = CInt(Form5.Text2.Text)
xx2r = CInt(Form5.Text3.Text)
xx1r = Int(xx1r / (scxr * h))
xx2r = Int(xx2r / (scxr * h))
Form5.Label3.Caption = (xx1r * h)
Form5.Label4.Caption = (xx2r * h)
If (xx2r > n - 1) Then
   xx2r = n - 1
End If
If (xx1r < 0) Then
   xx1r = 0
End If
mark5
End Function

Public Function Zoomrs()                    //Zoom Response Spectra
xx1rs = CInt(Form6.Text2.Text)
xx2rs = CInt(Form6.Text3.Text)
xx1rs = Int(xx1rs / (scxrs * tn))
xx2rs = Int(xx2rs / (scxrs * tn))
Form6.Label3.Caption = (xx1rs * tn)
Form6.Label4.Caption = (xx2rs * tn)
If (xx2rs > n - 1) Then
   xx2rs = n - 1
End If
If (xx1rs < 0) Then
   xx1rs = 0
End If
mark6
End Function

Public Function Original()                  //Zoom to Previous Ground Response
xx1 = 0
xx2 = n - 1
xx1r = 0
xx2r = n - 1
xx1rs = 0
xx2rs = n - 1
Form3.Label3.Caption = (xx1 * h)
Form3.Label4.Caption = (xx2 * h)
Form5.Label3.Caption = (xx1r * h)
Form5.Label4.Caption = (xx2r * h)
Form6.Label3.Caption = (xx1rs * tn)
Form6.Label4.Caption = (xx2rs * tn)
End Function
```

```
Public Function SaveCurDisp()                    //Save Ground Response File
filenum1 = FreeFile
Open Text1 For Output As filenum1
   Print #filenum1, (xx2 - xx1 + 1) & ""
   Print #filenum1, h & ""
   For i = xx1 To xx2
      Print #filenum1, u(i) & ""
   Next
Close filenum1
End Function

Public Function Trapizoidal()                    //Trapiziodal Rule
t = h
a = h / (2 * m * wd)
   For i = 0 To n - 1
      X(i) = Exp(e * wd * t) * P(i) * Cos(wd * t)
      Y(i) = Exp(e * wd * t) * P(i) * Sin(wd * t)
      Aa(i) = X(0)
      If (i > 0) Then
         For j = 1 To i - 1
            Aa(i) = Aa(i) + 2 * X(j)
         Next
      End If
      If (i > 1) Then
         Aa(i) = Aa(i) + X(i)
      End If
      B(i) = Y(0)
      If (i > 0) Then
         For j = 1 To i - 1
            B(i) = B(i) + 2 * Y(j)
         Next
      End If
      If (i > 1) Then
         B(i) = B(i) + Y(i)
      End If
      Aa(i) = Aa(i) * a
      B(i) = B(i) * a
      term1 = Aa(i) * Exp((-1) * wd * t) * Sin(wd * t) / (t)
      term2 = B(i) * Exp((-1) * wd * t) * Sin(wd * t) / (t)
      u(i) = term1 - term2
      t = t + h
   Next
End Function

Public Function CummOpp()                        //Ground Displacement
t = h
u(0) = t * P(0)
t = t + h
For i = 1 To n - 1
   u(i) = u(i - 1) + ((t * P(i)) / i)
   t = t + h
Next
End Function
```

```
Public Function Cumm()                    //Ground Velocity
t = h
u(0) = t * P(0)
t = t + h
For i = 1 To n - 1
   area = (h * P(i)) + (0.5 * (P(i) - P(i - 1)) * h)
   u(i) = u(i - 1) + (area)
   t = t + h
Next
End Function

Public Function RK()                      //Runge-Kutta Method
du(0) = 0
ddu(0) = 0
For i = 0 To n - 2
   k11 = h * (((1) * P(i)) - (2 * eta * wn * du(i)) - (wn * wn * ddu(i)))
   k12 = h * (((1) * P(i)) - (2 * eta * wn * (du(i) + (k11 / 2))) - (wn * wn * (ddu(i) + (h / 2))))
   k13 = h * (((1) * P(i)) - (2 * eta * wn * (du(i) + (k12 / 2))) - (wn * wn * (ddu(i) + (h / 2))))
   k14 = h * (((1) * P(i)) - (2 * eta * wn * (du(i) + k13)) - (wn * wn * (ddu(i) + h)))
   ddu(i + 1) = (ddu(i) + ((1 / 6) * (k11 + (2 * k12) + (2 * k13) + k14)))
   k21 = h * ddu(i)
   k22 = h * (ddu(i) + (k21 / 2))
   k23 = h * (ddu(i) + (k22 / 2))
   k24 = h * (ddu(i) + k23)
   du(i + 1) = (du(i) + ((1 / 6) * (k21 + (2 * k22) + (2 * k23) + k24)))
   acc(i) = P(i) - (2 * eta * wn * du(i)) - (wn * wn * ddu(i))
Next
End Function

Public Function RKF()                     //Runge-Kutta-Fehlberg Method
du(0) = 0
ddu(0) = 0
For i = 0 To n - 2
   k11 = h * (((1) * P(i)) - (2 * eta * wn * du(i)) - (wn * wn * ddu(i)))

   k12 = h * (((1) * P(i)) - (2 * eta * wn * (du(i) + (k11 / 4))) - (wn * wn * (ddu(i) + (h / 4))))

   k13 = h * (((1) * P(i)) - (2 * eta * wn * (du(i) + (3 * k11 / 32) + (9 * k12 / 32))) - (wn * wn * (ddu(i) +
(3 * h / 8))))

   k14 = h * (((1) * P(i)) - (2 * eta * wn * (du(i) + (1932 * k11 / 2197) - (7200 * k12 / 2197) + (7296 * k13
/ 2197))) - (wn * wn * (ddu(i) + (12 * h / 13))))

   k15 = h * (((1) * P(i)) - (2 * eta * wn * (du(i) + (439 * k11 / 216) - (8 * k12) + (3680 * k13 / 513) - (845
* k14 / 4104))) - (wn * wn * (ddu(i) + h)))

   k16 = h * (((1) * P(i)) - (2 * eta * wn * (du(i) - (8 * k11 / 27) + (2 * k12) - (3544 * k13 / 2565) + (1859 *
k14 / 4104) - (11 * k15 / 40))) - (wn * wn * (ddu(i) + (h / 2))))

   ddu(i + 1) = ddu(i) + ((16 * k11 / 135) + (6656 * k13 / 12825) + (28561 * k14 / 56430) - (9 * k15 / 50)
+ (2 * k16 / 55))

   k21 = h * ddu(i)
   k22 = h * (ddu(i) + (k21 / 4))
   k23 = h * (ddu(i) + ((3 * k21 / 32) + (9 * k22 / 32)))
   k24 = h * (ddu(i) + ((1932 * k21 / 2197) - (7200 * k22 / 2197) + (7296 * k23 / 2197)))
```

```
   k25 = h * (ddu(i) + ((439 * k21 / 216) - (8 * k22) + (3680 * k23 / 513) - (845 * k24 / 4104)))
   k26 = h * (ddu(i) - ((8 * k21 / 27) + (2 * k22) - (3544 * k23 / 2565) + (1859 * k24 / 4104) - (11 * k25 / 40)))

   du(i + 1) = du(i) + ((16 * k21 / 135) + (6656 * k23 / 12825) + (28561 * k24 / 56430) - (9 * k25 / 50) + (2 * k26 / 55))

   acc(i) = P(i) - (2 * eta * wn * du(i)) - (wn * wn * ddu(i))
Next
End Function

Public Function AMF()                    //Adam-Moulton Method
du(0) = 0
ddu(0) = 0
For i = 0 To 3
   k11 = h * (((1) * P(i)) - (2 * eta * wn * du(i)) - (wn * wn * ddu(i)))
   k12 = h * (((1) * P(i)) - (2 * eta * wn * (du(i) + (k11 / 2))) - (wn * wn * (ddu(i) + (h / 2))))
   k13 = h * (((1) * P(i)) - (2 * eta * wn * (du(i) + (k12 / 2))) - (wn * wn * (ddu(i) + (h / 2))))
   k14 = h * (((1) * P(i)) - (2 * eta * wn * (du(i) + k13)) - (wn * wn * (ddu(i) + h)))
   ddu(i + 1) = (ddu(i) + ((1 / 6) * (k11 + (2 * k12) + (2 * k13) + k14)))

   k21 = h * ddu(i)
   k22 = h * (ddu(i) + (k21 / 2))
   k23 = h * (ddu(i) + (k22 / 2))
   k24 = h * (ddu(i) + k23)
   du(i + 1) = (du(i) + ((1 / 6) * (k21 + (2 * k22) + (2 * k23) + k24)))

   acc(i) = P(i) - (2 * eta * wn * du(i)) - (wn * wn * ddu(i))
Next
For i = 4 To n - 2
   ddu(i + 1) = ddu(i) + ((h / 24) * ((55 * du(i)) - (59 * du(i - 1)) + (37 * du(i - 2)) - (9 * du(i - 3))))
   du(i + 1) = du(i) + ((h / 24) * ((55 * acc(i)) - (59 * acc(i - 1)) + (37 * acc(i - 2)) - (9 * acc(i - 3))))
   acc(i + 1) = P(i) - (2 * eta * wn * du(i)) - (wn * wn * ddu(i))
Next
End Function

Public Function IBE()                    //Method of Interpolation by Excitation
For i = 0 To n - 1
   du(i) = 0
   ddu(i) = 0
Next
a11 = 0
a12 = 0
a21 = 0
a22 = 0
b11 = 0
b12 = 0
b21 = 0
b22 = 0
a11 = Exp((-1) * eta * wn * h) * (((eta / (1 - (eta * eta)) ^ 0.5) * Sin(wn * h * (1 - (eta * eta)) ^ 0.5)) + Cos(wn * h * (1 - (eta * eta)) ^ 0.5))

a12 = Exp((-1) * eta * wn * h) * Sin(wn * h * (1 - (eta * eta)) ^ 0.5) / (wn * (1 - (eta * eta)) ^ 0.5)

a21 = (-1) * wn * Exp((-1) * eta * wn * h) * wn * Sin(wn * h * (1 - (eta * eta)) ^ 0.5) / ((1 - (eta * eta)) ^ 0.5)
```

```
a22 = (Exp((-1) * eta * wn * h) * Cos(wn * h * (1 - (eta * eta)) ^ 0.5)) - ((eta / (1 - (eta * eta)) ^ 0.5) *
Sin(wn * h * (1 - (eta * eta)) ^ 0.5))

b11 = ((((((2 * eta * eta) - 1) / (wn * wn * h)) + (eta / wn)) * Sin(wn * h * (1 - (eta * eta)) ^ 0.5) / (wn * (1 -
(eta * eta)) ^ 0.5)) + (((((2 * eta * eta) - 1) / (wn * wn * wn * h)) + (1 / (wn * wn))) * Cos(wn * h * (1 - (eta
* eta)) ^ 0.5)) * Exp((-1) * eta * wn * h)) - ((2 * eta) / (wn * wn * wn * h))

b12 = ((((((2 * eta * eta) - 1) / (wn * wn * h))) * Sin(wn * h * (1 - (eta * eta)) ^ 0.5) / (wn * (1 - (eta * eta))
^ 0.5)) + (((((2 * eta * eta) - 1) / (wn * wn * wn * h))) * Cos(wn * h * (1 - (eta * eta)) ^ 0.5)) * (-1) * Exp((-
1) * eta * wn * h)) - (1 / (wn * wn)) + ((2 * eta) / (wn * wn * wn * h))

b21 = (((((((2 * eta * eta) - 1) / (wn * wn * h)) + (eta / wn)) * ((-1) * (eta / ((1 - (eta * eta)) ^ 0.5)) * Sin(wn
* h * (1 - (eta * eta)) ^ 0.5) / (wn * (1 - (eta * eta)) ^ 0.5)) + (Cos(wn * h * (1 - (eta * eta)) ^ 0.5)))) - ((((((2
* eta) - 1) / (wn * wn * wn * h)) + (1 / (wn * wn))) * (wn * ((1 - (eta * eta)) ^ 0.5) * Sin(wn * h * (1 - (eta *
eta)) ^ 0.5)) + (wn * eta * Cos(wn * h * (1 - (eta * eta)) ^ 0.5)))) * Exp((-1) * eta * wn * h)) + ((1) / (wn *
wn * h))

b22 = (((((((2 * eta * eta) - 1) / (wn * wn * h))) * ((-1) * (eta / ((1 - (eta * eta)) ^ 0.5)) * Sin(wn * h * (1 -
(eta * eta)) ^ 0.5) / (wn * (1 - (eta * eta)) ^ 0.5)) + (Cos(wn * h * (1 - (eta * eta)) ^ 0.5)))) - ((((((2 * eta) -
1) / (wn * wn * wn * h))) * (wn * ((1 - (eta * eta)) ^ 0.5) * Sin(wn * h * (1 - (eta * eta)) ^ 0.5)) + (wn * eta
* Cos(wn * h * (1 - (eta * eta)) ^ 0.5)))) * (-1) * Exp((-1) * eta * wn * h)) - ((1) / (wn * wn * h))

For i = 0 To n - 2
   ddu(i + 1) = (a11 * ddu(i)) + (a12 * du(i)) + (b11 * P(i)) + (b12 * P(i + 1))
   du(i) = (a21 * ddu(i)) + (a22 * du(i)) + (b21 * P(i)) + (b22 * P(i + 1))
   acc(i + 1) = (-1) * ((2 * eta * wn * du(i + 1)) + (wn * wn * ddu(i + 1)))
Next
End Function

Private Sub Timer2_Timer()                    //Animate Response
On Error GoTo err8:
curr = 0
Do While (curr < nor)
   If (ctr > n - 1) Then
      anir = 0
      Timer2.Interval = 0
      Form5.Text1.Text = ""
      Exit Sub
   End If
   If (ctr = 0) Then
      Form5.Line (xvalr, yvalr)-(xvalr + Int(ttr * scxr), yvalr - Int(r(0) * scyr))
      x1r = xvalr + Int(ttr * scxr)
      y1r = yvalr - Int(r(0) * scyr)
      ttr = ttr + h
   Else
      Form5.Line (x1r, y1r)-(xvalr + Int(ttr * scxr), yvalr - Int(r(ctr) * scyr))
      If (Abs(maxr) < Abs(minr) And r(ctr) = minr) Then
         Form5.Shape1.Left = xvalr + Int(ttr * scxr) - 75
         Form5.Shape1.Top = yvalr - Int(r(ctr) * scyr) - 75
         Form5.Label14.Left = xvalr + Int(ttr * scxr) + 100
         Form5.Label14.Top = yvalr - Int(r(ctr) * scyr)
         Form5.Label14.Caption = CInt(100 * minr) / 100
         Form5.Shape1.Visible = True
         Form5.Label14.Visible = True
      End If
```

```
    If (Abs(maxr) > Abs(minr) And r(ctr) = maxr) Then
       Form5.Shape1.Left = xvalr + Int(ttr * scxr) - 75
       Form5.Shape1.Top = yvalr - Int(r(ctr) * scyr) - 75
       Form5.Label14.Left = xvalr + Int(ttr * scxr) + 100
       Form5.Label14.Top = yvalr - Int(r(ctr) * scyr)
       Form5.Label14.Caption = CInt(100 * maxr) / 100
       Form5.Shape1.Visible = True
       Form5.Label14.Visible = True
    End If
    x1r = xvalr + Int(ttr * scxr)
    y1r = yvalr - Int(r(ctr) * scyr)
    ttr = ttr + h
  End If
  ctr = ctr + 1
  curr = curr + 1
Loop
Exit Sub
err8:
  Resume Next
End Sub

Private Sub Timer3_Timer()                    //Animate Response Spectra
On Error GoTo err24:
currs = 0
Do While (currs < nors)
  If (ctrs > n - 1) Then
    anirs = 0
    Timer3.Interval = 0
    Form6.Text1.Text = ""
    Exit Sub
  End If
  If (ctrs = 0) Then
    Form6.Line (xvalrs, yvalrs)-(xvalrs + Int(ttrs * scxrs), yvalrs - Int(rs(0) * scyrs))
    x1rs = xvalrs + Int(ttrs * scxrs)
    y1rs = yvalrs - Int(rs(0) * scyrs)
    ttrs = ttrs + tn
  Else
    Form6.Line (x1rs, y1rs)-(xvalrs + Int(ttrs * scxrs), yvalrs - Int(rs(ctrs) * scyrs))
    'If (Abs(maxrs) < Abs(minrs) And rs(ctrs) = minrs) Then
       'Form6.Shape1.Left = xvalrs + Int(ttrs * scxrs) - 75
       'Form6.Shape1.Top = yvalrs - Int(r(ctrs) * scyrs) - 75
       'Form6.Label14.Left = xvalrs + Int(ttrs * scxrs) + 100
       'Form6.Label14.Top = yvalrs - Int(rs(ctrs) * scyrs)
       'Form6.Label14.Caption = CInt(100 * minrs) / 100
       'Form6.Shape1.Visible = True
       'Form6.Label14.Visible = True
    'End If
    If (rs(ctrs) = maxrs) Then
       Form6.Shape1.Left = xvalrs + Int(ttrs * scxrs) - 75
       Form6.Shape1.Top = yvalrs - Int(rs(ctrs) * scyrs) - 75
       Form6.Label14.Left = xvalrs + Int(ttrs * scxrs) + 100
       Form6.Label14.Top = yvalrs - Int(rs(ctrs) * scyrs)
       Form6.Label14.Caption = maxrs 'CInt(100 * maxrs) / 100
       Form6.Shape1.Visible = True
       Form6.Label14.Visible = True
    End If
```

```
      x1rs = xvalrs + Int(ttrs * scxrs)
      y1rs = yvalrs - Int(rs(ctrs) * scyrs)
      ttrs = ttrs + tn
   End If
   ctrs = ctrs + 1
   currs = currs + 1
Loop
Exit Sub
err24:
   Resume Next
End Sub

Public Function mark3()                    //Mark Points in Ground Response
On Error GoTo err11:
Form3.Label5.Caption = CInt(10 * ((xx1 * h) + ((xx2 - xx1) * h / 5))) / 10
Form3.Label6.Caption = CInt(10 * ((xx1 * h) + (2 * (xx2 - xx1) * h / 5))) / 10
Form3.Label7.Caption = CInt(10 * ((xx1 * h) + (3 * (xx2 - xx1) * h / 5))) / 10
Form3.Label8.Caption = CInt(10 * ((xx1 * h) + (4 * (xx2 - xx1) * h / 5))) / 10
Exit Function
err11:
   Resume Next
End Function

Public Function mark5()                    //Mark Points in Response
On Error GoTo err12:
Form5.Label5.Caption = CInt(10 * ((xx1r * h) + ((xx2r - xx1r) * h / 5))) / 10
Form5.Label6.Caption = CInt(10 * ((xx1r * h) + (2 * (xx2r - xx1r) * h / 5))) / 10
Form5.Label7.Caption = CInt(10 * ((xx1r * h) + (3 * (xx2r - xx1r) * h / 5))) / 10
Form5.Label8.Caption = CInt(10 * ((xx1r * h) + (4 * (xx2r - xx1r) * h / 5))) / 10
Exit Function
err12:
   Resume Next
End Function

Public Function mark6()                    //Mark Points in Response Spectrum
On Error GoTo err13:
Form6.Label5.Caption = CInt(10 * ((xx1rs * h) + ((xx2rs - xx1rs) * CDbl(Text10.Text) / 5))) / 10
Form6.Label6.Caption = CInt(10 * ((xx1rs * h) + (2 * (xx2rs - xx1rs) * CDbl(Text10.Text) / 5))) / 10
Form6.Label7.Caption = CInt(10 * ((xx1rs * h) + (3 * (xx2rs - xx1rs) * CDbl(Text10.Text) / 5))) / 10
Form6.Label8.Caption = CInt(10 * ((xx1rs * h) + (4 * (xx2rs - xx1rs) * CDbl(Text10.Text) / 5))) / 10
Exit Function
err13:
   Resume Next
End Function
```

The following is the source code of the software done in Visual Basic 6.0 environment. The comments are placed in start of the functions.

```
Dim h As Double              //Variables Declaration
Dim m As Double
Dim wd As Double
Dim a As Double
Dim e As Double
Dim P(10000) As Double
Dim X(10000) As Double
Dim Y(10000) As Double
Dim Aa(10000) As Double
Dim B(10000) As Double

Dim term1 As Double
Dim term2 As Double
Dim u(10000) As Double
Dim all() As String
Dim t As Double
Dim n As Integer
Dim filenum
Dim str As String
Dim i As Integer
Dim j As Integer
Dim k As Integer

Dim scx As Double
Dim scy As Double
Dim scxr As Double
Dim scyr As Double
Dim scxrs As Double
Dim scyrs As Double

Dim tt As Double
Dim ttr As Double
Dim ttrs As Double

Dim x1 As Integer
Dim y1 As Integer
Dim x1r As Integer
Dim y1r As Integer
Dim x1rs As Integer
Dim y1rs As Integer

Dim max As Double
Dim min As Double
Dim maxr As Double
Dim minr As Double
Dim maxrs As Double
Dim minrs As Double
Dim ani As Integer
Dim anir As Integer
Dim anirs As Integer
```

```
Dim ct As Integer
Dim ctr As Integer
Dim ctrs As Integer
Dim no As Integer
Dim nor As Integer
Dim nors As Integer
Dim cur As Integer
Dim curr As Integer
Dim currs As Integer

Dim xval As Integer
Dim yval As Integer
Dim xvalr As Integer
Dim yvalr As Integer
Dim xvalrs As Integer
Dim yvalrs As Integer

Dim xx1 As Integer
Dim xx2 As Integer
Dim xx1r As Integer
Dim xx2r As Integer
Dim xx1rs As Integer
Dim xx2rs As Integer

Dim m1 As Double
Dim m2 As Double
Dim m3 As Double
Dim m4 As Double

Dim area As Double
Dim filenum1
Dim tn As Double
Dim eta As Double

Dim k11 As Double
Dim k12 As Double
Dim k13 As Double
Dim k14 As Double
Dim k15 As Double
Dim k16 As Double
Dim k21 As Double
Dim k22 As Double
Dim k23 As Double
Dim k24 As Double
Dim k25 As Double
Dim k26 As Double

Dim du(10000) As Double
Dim uu(10000) As Double
Dim ddu(10000) As Double
Dim acc(10000) As Double
Dim r(10000) As Double
Dim rs(10000) As Double

Dim wn As Double
```

```
Dim a11 As Double
Dim a12 As Double
Dim a21 As Double
Dim a22 As Double
Dim b11 As Double
Dim b12 As Double
Dim b21 As Double
Dim b22 As Double

Private Sub Command1_Click()          //Open Earthquake Data File
Text2.Text = "1"
Form2.Show (1)
End Sub

Public Function Calculate()           //Reads the ground response data from the file
On Error GoTo err1:
filenum = FreeFile
Open Form2.Text1.Text For Input As filenum
   str = Input(LOF(filenum), filenum)
   all = Split(str, " ")
   n = CInt(all(0))
   h = CDbl(all(1))
   List1.Clear
   List2.Clear
   List2.Clear
   Option6.Value = False
   Option7.Value = False
   Option8.Value = False
   Option16.Value = False
   Option17.Value = False
   Option15.Value = False
   Text8.Text = ""
   Text12.Text = ""
   For i = 2 To n + 1
      If (Option3.Value = False) Then
         P(i - 2) = CDbl(all(i)) * 386.0886
      Else
         P(i - 2) = CDbl(all(i))
      End If
   Next
   If (Option3.Value = True) Then
      For i = 0 To n - 1
         u(i) = P(i)
      Next
   End If
   If (Option4.Value = True) Then
      List1.Clear
      Cumm
   End If
   If (Option5.Value = True) Then
      Cumm
      For i = 0 To n - 1
         P(i) = u(i)
      Next
      Cumm
   End If
```

```
   For i = 0 To n - 1
      List1.AddItem ((i + 1) & "." & vbTab & u(i) & "")
   Next
Close filenum
Text3.Text = n
Text4.Text = h
max = Abs(u(0))
For i = 1 To n - 1
   If (Abs(u(i)) > Abs(max)) Then
      max = u(i)
   End If
Next
Text5.Text = max
Text7.Text = n * h
Exit Function
err1:
   MsgBox ("Error Opening File " & Form2.Text1.Text)
   List1.Clear
   List2.Clear
   List3.Clear
   Text8.Text = ""
   Text12.Text = ""
   Option6.Value = False
   Option7.Value = False
   Option8.Value = False
   Option16.Value = False
   Option17.Value = False
   Option15.Value = False
End Function

Public Function Calculater()                    //Reads the response data from the file
On Error GoTo err1:
filenum = FreeFile
Open Form2.Text1.Text For Input As filenum
   str = Input(LOF(filenum), filenum)
   all = Split(str, vbNewLine)
   n = CInt(all(0))
   h = CDbl(all(1))
   List1.Clear
   List2.Clear
   List3.Clear
   Option6.Value = False
   Option7.Value = False
   Option8.Value = False
   Option16.Value = False
   Option17.Value = False
   Option15.Value = False
   Text8.Text = ""
   Text12.Text = ""
   For i = 2 To n + 1
      If (Option6.Value = False) Then
         P(i - 2) = CDbl(all(i)) * 386.0886
      Else
         P(i - 2) = CDbl(all(i))
      End If
   Next
```

```
   For i = 0 To n - 1
      r(i) = P(i)
   Next
   For i = 0 To n - 1
      List2.AddItem ((i + 1) & "." & vbTab & r(i) & "")
   Next
Close filenum
Text3.Text = n
Text4.Text = h
max = Abs(r(0))
For i = 1 To n - 1
   If (Abs(r(i)) > Abs(max)) Then
      max = r(i)
   End If
Next
Text8.Text = max
Text7.Text = n * h
Exit Function
err1:
   MsgBox ("Error Opening File " & Form2.Text1.Text)
   List3.Clear
   List2.Clear
   List1.Clear
   Text8.Text = ""
   Text12.Text = ""
   Option6.Value = False
   Option7.Value = False
   Option8.Value = False
   Option16.Value = False
   Option17.Value = False
   Option15.Value = False
End Function

Public Function Calculaters()                    //Reads the response spectra data from the file
On Error GoTo err22:
filenum = FreeFile
Open Form2.Text1.Text For Input As filenum
   str = Input(LOF(filenum), filenum)
   all = Split(str, vbNewLine)
   n = CInt(all(0))
   tn = CDbl(all(1))
   List1.Clear
   List2.Clear
   List3.Clear
   Option6.Value = False
   Option7.Value = False
   Option8.Value = False
   Option16.Value = False
   Option17.Value = False
   Option15.Value = False
   Text5.Text = ""
   Text8.Text = ""
   Text12.Text = ""
   For i = 2 To n + 1
      If (Option17.Value = False) Then
         P(i - 2) = CDbl(all(i)) * 386.0886
```

```
      Else
         P(i - 2) = CDbl(all(i))
      End If
   Next
   For i = 0 To n - 1
      rs(i) = P(i)
   Next
   For i = 0 To n - 1
      List3.AddItem ((i + 1) & "." & vbTab & rs(i) & "")
   Next
Close filenum
Text3.Text = n
Text4.Text = tn
maxrs = Abs(rs(0))
For i = 1 To n - 1
   If (Abs(r(i)) > Abs(maxrs)) Then
      maxrs = rs(i)
   End If
Next
Text12.Text = maxrs
Text7.Text = n * tn
Exit Function
err22:
   MsgBox ("Error Opening File " & Form2.Text1.Text)
   List3.Clear
   List2.Clear
   List1.Clear
   Text8.Text = ""
   Text12.Text = ""
   Option16.Value = False
   Option17.Value = False
   Option15.Value = False
End Function

Private Sub Command2_Click()                    //Draws the Ground Response
On Error GoTo err2:
   Form3.Shape1.Visible = False
   Form3.Label14.Visible = False
If (List1.ListCount > 0) Then
   If (Option3.Value = True) Then
      Form3.Caption = "Ground Acceleration Response"
      Form3.Label1.Caption = "Ground Acceleration Response"
      Form3.Text4.Text = "ACCELERATION"
   End If
   If (Option4.Value = True) Then
      Form3.Caption = "Ground Velocity Response"
      Form3.Label1.Caption = "Ground Velocity Response"
      Form3.Text4.Text = "VELOCITY"
   End If
   If (Option5.Value = True) Then
      Form3.Caption = "Ground Displacement Response"
      Form3.Label1.Caption = "Ground Displacement Response"
      Form3.Text4.Text = "DISPLACEMENT"
   End If
   xx1 = 0
   xx2 = n - 1
```

```
mark3
ct = 0
Form3.Label4.Caption = Text7.Text
Form3.Show
Form3.Text1.Text = "1"
scx = 10500 / (n * h)
max = u(0)
min = u(0)
For i = 1 To n - 1
   If (u(i) > max) Then
      max = u(i)
   End If
   If (u(i) < min) Then
      min = u(i)
   End If
Next
scy = (yval * 1.3) / (max - min)
Form3.Label10.Caption = CInt(100 * 1477.5 / scy) / 100
Form3.Label11.Caption = CInt(100 * (-1) * 1477.5 / scy) / 100
Form3.Label12.Caption = CInt(100 * 1477.5 * 2 / scy) / 100
Form3.Label13.Caption = CInt(100 * (-1) * 1477.5 * 2 / scy) / 100
tt = h * 2
no = Int((n / 30000) * 33)
If (no = 0) Then
   no = 1
End If
If (Option1.Value = False) Then
   Form3.Line (xval, yval)-(xval + Int(tt * scx), yval - Int(u(0) * scy))
   x1 = xval + Int(tt * scx)
   y1 = yval - Int(u(0) * scy)
   tt = tt + h
   For i = 1 To n - 1
      Form3.Line (x1, y1)-(xval + Int(tt * scx), yval - Int(u(i) * scy))
      If (Abs(max) < Abs(min) And u(i) = min) Then
         Form3.Shape1.Left = xval + Int(tt * scx) - 75
         Form3.Shape1.Top = yval - Int(u(i) * scy) - 75
         Form3.Label14.Left = xval + Int(tt * scx) + 100
         Form3.Label14.Top = yval - Int(u(i) * scy)
         Form3.Label14.Caption = CInt(100 * min) / 100
         Form3.Shape1.Visible = True
         Form3.Label14.Visible = True
      End If
      If (Abs(max) > Abs(min) And u(i) = max) Then
         Form3.Shape1.Left = xval + Int(tt * scx) - 75
         Form3.Shape1.Top = yval - Int(u(i) * scy) - 75
         Form3.Label14.Left = xval + Int(tt * scx) + 100
         Form3.Label14.Top = yval - Int(u(i) * scy)
         Form3.Label14.Caption = CInt(1000 * max) / 100
         Form3.Shape1.Visible = True
         Form3.Label14.Visible = True
      End If
      x1 = xval + Int(tt * scx)
      y1 = yval - Int(u(i) * scy)
      tt = tt + h
   Next
   Form3.Text1.Text = ""
```

```
   Else
      Timer1.Interval = 1
      Exit Sub
   End If
End If
Exit Sub
err2:
   MsgBox ("Error In Drawing Ground Response")
   Form3.Visible = False
   Exit Sub
End Sub

Private Sub Command3_Click()              //Open Ground Response File
Text2.Text = "2"
Form2.Show (1)
End Sub

Private Sub Command4_Click()              //Open Response File
Text2.Text = "3"
Form2.Show (1)
End Sub

Private Sub Command5_Click()              //Draws the Response
On Error GoTo err12:
   Form5.Shape1.Visible = False
   Form5.Label14.Visible = False
If (List2.ListCount > 0) Then
   If (Text6.Text <> "" And Text7.Text <> "") Then
      wn = CDbl(Text6.Text)
      eta = CDbl(Text7.Text)
   End If
   If (Option6.Value = True) Then
      Form5.Caption = "Acceleration Response"
      Form5.Label1.Caption = "Acceleration Response"
      Form5.Text4.Text = "ACCELERATION"
   End If
   If (Option7.Value = True) Then
      Form5.Caption = "Displacement Response"
      Form5.Label1.Caption = "Displacement Response"
      Form5.Text4.Text = "DISPLACEMENT"
   End If
   If (Option8.Value = True) Then
      Form5.Caption = "Velocity Response"
      Form5.Label1.Caption = "Velocity Response"
      Form5.Text4.Text = "VELOCITY"
   End If
   xx1r = 0
   xx2r = n - 1
   mark5
   ctr = 0
   Form5.Label4.Caption = Text7.Text
   Form5.Show
   Form5.Text1.Text = "1"
   scxr = 10500 / (n * h)
   maxr = r(0)
   minr = r(0)
```

```
    For i = 1 To n - 1
       If (r(i) > maxr) Then
          maxr = r(i)
       End If
       If (r(i) < minr) Then
          minr = r(i)
       End If
    Next
    scyr = (yvalr * 1.3) / (maxr - minr)
    Form5.Label10.Caption = CInt(100 * 1477.5 / scyr) / 100
    Form5.Label11.Caption = CInt(100 * (-1) * 1477.5 / scyr) / 100
    Form5.Label12.Caption = CInt(100 * 1477.5 * 2 / scyr) / 100
    Form5.Label13.Caption = CInt(100 * (-1) * 1477.5 * 2 / scyr) / 100
    ttr = h
    nor = Int((n / 30000) * 33)
    If (nor = 0) Then
       nor = 1
    End If
    If (Option10.Value = False) Then
       Form5.Line (xvalr, yvalr)-(xvalr + Int(ttr * scxr), yvalr - Int(r(0) * scyr))
       x1r = xvalr + Int(ttr * scxr)
       y1r = yvalr - Int(r(0) * scyr)
       ttr = ttr + h
       For i = 1 To n - 1
          Form5.Line (x1r, y1r)-(xvalr + Int(ttr * scxr), yvalr - Int(r(i) * scyr))
          If (Abs(maxr) < Abs(min) And r(i) = minr) Then
             Form5.Shape1.Left = xvalr + Int(ttr * scxr) - 75
             Form5.Shape1.Top = yvalr - Int(r(i) * scyr) - 75
             Form5.Label14.Left = xvalr + Int(ttr * scxr) + 100
             Form5.Label14.Top = yvalr - Int(r(i) * scyr)
             Form5.Label14.Caption = CInt(100 * minr) / 100
             Form5.Shape1.Visible = True
             Form5.Label14.Visible = True
          End If
          If (Abs(maxr) > Abs(minr) And r(i) = maxr) Then
             Form5.Shape1.Left = xvalr + Int(ttr * scxr) - 75
             Form3.Shape1.Top = yvalr - Int(r(i) * scyr) - 75
             Form5.Label14.Left = xvalr + Int(ttr * scxr) + 100
             Form5.Label14.Top = yvalr - Int(r(i) * scyr)
             Form5.Label14.Caption = CInt(1000 * maxr) / 100
             Form5.Shape1.Visible = True
             Form5.Label14.Visible = True
          End If
          x1r = xvalr + Int(ttr * scxr)
          y1r = yvalr - Int(r(i) * scyr)
          ttr = ttr + h
       Next
       Form5.Text1.Text = ""
    Else
       Timer2.Interval = 1
       Exit Sub
    End If
End If
Exit Sub
err12:
    'MsgBox ("Error In Drawing Response")
```

```
   'Form5.Visible = False
   Resume Next
   Exit Sub
End Sub

Private Sub Command6_Click()                //Open Response Spectra File
Text2.Text = "4"
Form2.Show (1)
End Sub

Private Sub Command7_Click()                //Draws Response Spectra
On Error GoTo err20:
   Form6.Shape1.Visible = False
   Form6.Label14.Visible = False
If (List3.ListCount > 0) Then
   If (Text10.Text <> "" And Text11.Text <> "") Then
      wn = CDbl(Text10.Text)
      eta = CDbl(Text11.Text)
   End If
   If (Option17.Value = True) Then
      Form6.Caption = "Acceleration Response Spectra"
      Form6.Label1.Caption = "Acceleration Response Spectra"
      Form6.Text4.Text = "ACCELERATION"
   End If
   If (Option16.Value = True) Then
      Form6.Caption = "Displacement Response Spectra"
      Form6.Label1.Caption = "Displacement Response Spectra"
      Form6.Text4.Text = "DISPLACEMENT"
   End If
   If (Option15.Value = True) Then
      Form6.Caption = "Velocity Response Spectra"
      Form6.Label1.Caption = "Velocity Response Spectra"
      Form6.Text4.Text = "VELOCITY"
   End If
   xx1rs = 0
   xx2rs = n - 1
   mark6
   ctrs = 0
   Form6.Label4.Caption = n * tn
   Form6.Show
   Form6.Text1.Text = "1"
   scxrs = 10500 / (n * CDbl(Text10.Text))
   maxrs = rs(0)
   minrs = rs(0)
   For i = 1 To n - 1
      If (rs(i) > maxrs) Then
         maxrs = rs(i)
      End If
      If (rs(i) < minrs) Then
         minrs = rs(i)
      End If
   Next
   scyrs = (yvalrs) / (maxrs * 1.3) ' - minrs)
   Form6.Label10.Caption = 1477.5 / scyrs 'CInt(100 * 1477.5 / scyrs) / 100
   Form6.Label11.Caption = "" 'CInt(100 * (-1) * 1477.5 / scyrs) / 100
   Form6.Label12.Caption = 1477.5 / scyrs * 2 'CInt(100 * 1477.5 * 2 / scyrs) / 100
```

```
   Form6.Label13.Caption = "" 'CInt(100 * (-1) * 1477.5 * 2 / scyrs) / 100
   ttrs = 2 * CDbl(Text10.Text)
   nors = Int((n / 30000) * 15)
   If (nors = 0) Then
      nors = 1
   End If
   If (Option11.Value = False) Then
      Form6.Line (xvalrs, yvalrs)-(xvalrs + Int(ttrs * scxrs), yvalrs - Int(rs(0) * scyrs))
      x1rs = xvalrs + Int(ttrs * scxrs)
      y1rs = yvalrs - Int(rs(0) * scyrs)
      ttrs = CDbl(Text10.Text)
      For i = 1 To n - 1
         Form6.Line (x1rs, y1rs)-(xvalrs + Int(ttrs * scxrs), yvalrs - Int(rs(i) * scyrs))
         'If (Abs(maxrs) < Abs(minrs) And rs(i) = minrs) Then
            'Form6.Shape1.Left = xvalrs + Int(ttrs * scxrs) - 75
            'Form6.Shape1.Top = yvalrs - Int(rs(i) * scyrs) - 75
            'Form6.Label14.Left = xvalrs + Int(ttrs * scxrs) + 100
            'Form6.Label14.Top = yvalrs - Int(rs(i) * scyrs)
            'Form6.Label14.Caption = minrs 'CInt(100 * minrs) / 100
            'Form6.Shape1.Visible = True
            'Form6.Label14.Visible = True
         'End If
         If (rs(i) = maxrs) Then
            Form6.Shape1.Left = xvalrs + Int(ttrs * scxrs) - 75
            Form6.Shape1.Top = yvalrs - Int(rs(i) * scyrs) - 75
            Form6.Label14.Left = xvalrs + Int(ttrs * scxrs) + 100
            Form6.Label14.Top = yvalrs - Int(rs(i) * scyrs)
            Form6.Label14.Caption = maxrs 'CInt(1000 * maxrs) / 100
            Form6.Shape1.Visible = True
            Form6.Label14.Visible = True
         End If
         x1rs = xvalrs + Int(ttrs * scxrs)
         y1rs = yvalrs - Int(rs(i) * scyrs)
         ttrs = ttrs + CDbl(Text10.Text)
      Next
      Form6.Text1.Text = ""
   Else
      Timer3.Interval = 1
      Exit Sub
   End If
End If
Exit Sub
err20:
   MsgBox ("Error In Drawing Response Spectra")
   Form6.Visible = False
   Exit Sub
End Sub

Private Sub Form_Load()          //Initialization of variables
ct = 0
ctr = 0
ctrs = 0
xval = 840
xvalr = 840
xvalrs = 840
yval = 3800
```

```
yvalr = 3800
yvalrs = 3800
StatusBar2.Panels(3).Text = Format(Date, "dddd")
eta = 0.6
wn = 1
End Sub

Private Sub Form_Unload(Cancel As Integer)                    //End the Program
End
End Sub

Private Sub Option12_Click()                    //Runge-Kutta-Fehlberg Method for Response
If (Option6.Value = True) Then
   Option6_Click
End If
If (Option7.Value = True) Then
   Option7_Click
End If
If (Option8.Value = True) Then
   Option8_Click
End If
End Sub

Private Sub Option13_Click()                    //Runge-Kutta Method for Response
If (Option6.Value = True) Then
   Option6_Click
End If
If (Option7.Value = True) Then
   Option7_Click
End If
If (Option8.Value = True) Then
   Option8_Click
End If
End Sub

Private Sub Option15_Click()                    //Velocity Response Spectra
On Error GoTo err18:
If (List1.ListCount > 0) Then
   tn = CDbl(Text10.Text)
   eta = CDbl(Text11.Text)
   For i = 2 To n + 1
      P(i - 2) = CDbl(all(i)) * 386.0886
   Next
   wn = 0
   List3.Clear
   'For i = 0 To n - 1
   k = 0
   Do While (k < n)
      If (Option19.Value = True) Then
         RKF
      End If
      If (Option18.Value = True) Then
         RK
      End If
      If (Option21.Value = True) Then
         AMF
```

```
      End If
      If (Option23.Value = True) Then
         IBE
      End If
      maxrs = Abs(ddu(0))
      For j = 1 To n - 1
         If (Abs(ddu(j)) > Abs(maxrs)) Then
            maxrs = ddu(j)
         End If
      Next
      'List3.AddItem ((i + 1) & "." & vbTab & wn)
      rs(k) = Abs(maxrs)
      wn = wn + tn
      k = k + 1
   Loop
   'Next
   Label4.Caption = "Velocity Spectral Values"
   For i = 0 To n - 1
      'rs(i) = ddu(i)
      List3.AddItem ((i + 1) & "." & vbTab & rs(i))
   Next
   maxrs = Abs(rs(0))
   For i = 1 To n - 1
      If (Abs(rs(i)) > Abs(maxrs)) Then
         maxrs = rs(i)
      End If
   Next
   Text12.Text = maxrs
Else
   Option15.Value = False
End If
Exit Sub
err18:
   Option15.Value = False
   MsgBox ("Enter Valid Values For Time Period And Damping Ratio")
End Sub

Private Sub Option16_Click()                    //Displacement Response Spectra
On Error GoTo err19:
If (List1.ListCount > 0) Then
   tn = CDbl(Text10.Text)
   eta = CDbl(Text11.Text)
   For i = 2 To n + 1
      P(i - 2) = CDbl(all(i)) * 386.0886
   Next
   wn = 0
   List3.Clear
   'For i = 0 To n - 1
   k = 0
   Do While (k < n)
      If (Option19.Value = True) Then
         RKF
      End If
      If (Option18.Value = True) Then
         RK
      End If
```

```
      If (Option21.Value = True) Then
         AMF
      End If
      If (Option23.Value = True) Then
         IBE
      End If
      maxrs = Abs(du(0))
      For j = 1 To n - 1
         If (Abs(du(j)) > Abs(maxrs)) Then
            maxrs = du(j)
         End If
      Next
      'List3.AddItem ((i + 1) & "." & vbTab & wn)
      rs(k) = Abs(maxrs)
      wn = wn + tn
      k = k + 1
   Loop
   'Next
   Label4.Caption = "Displacement Spectral Values"
   For i = 0 To n - 1
      'rs(i) = ddu(i)
      List3.AddItem ((i + 1) & "." & vbTab & rs(i))
   Next
   maxrs = Abs(rs(0))
   For i = 1 To n - 1
      If (Abs(rs(i)) > Abs(maxrs)) Then
         maxrs = rs(i)
      End If
   Next
   Text12.Text = maxrs
Else
   Option16.Value = False
End If
Exit Sub
err19:
   Option16.Value = False
   MsgBox ("Enter Valid Values For Time Period And Damping Ratio")
End Sub

Private Sub Option17_Click()                    //Acceleration Response Spectra
On Error GoTo err17:
If (List1.ListCount > 0) Then
   tn = CDbl(Text10.Text)
   eta = CDbl(Text11.Text)
   For i = 2 To n + 1
      P(i - 2) = CDbl(all(i)) * 386.0886
   Next
   wn = 0
   List3.Clear
   'For i = 0 To n - 1
   k = 0
   Do While (k < n)
      If (Option19.Value = True) Then
         RKF
      End If
      If (Option18.Value = True) Then
```

```
        RK
      End If
      If (Option21.Value = True) Then
        AMF
      End If
      If (Option23.Value = True) Then
        IBE
      End If
      maxrs = Abs(acc(0))
      For j = 1 To n - 1
        If (Abs(acc(j)) > Abs(maxrs)) Then
          maxrs = acc(j)
        End If
      Next
      'List3.AddItem ((i + 1) & "." & vbTab & wn)
      rs(k) = Abs(maxrs)
      wn = wn + tn
      k = k + 1
    Loop
    'Next
    Label4.Caption = "Velocity Spectral Values"
    For i = 0 To n - 1
      'rs(i) = ddu(i)
      List3.AddItem ((i + 1) & "." & vbTab & rs(i))
    Next
    maxrs = Abs(rs(0))
    For i = 1 To n - 1
      If (Abs(rs(i)) > Abs(maxrs)) Then
        maxrs = rs(i)
      End If
    Next
    Text12.Text = maxrs
  Else
    Option17.Value = False
  End If
  Exit Sub
  err17:
    Option17.Value = False
    MsgBox ("Enter Valid Values For Time Period And Damping Ratio")
  End Sub

  Private Sub Option18_Click()          //Runge-Kutta Method for Response Spectra
  If (Option17.Value = True) Then
    Option17_Click
  End If
  If (Option15.Value = True) Then
    Option15_Click
  End If
  If (Option16.Value = True) Then
    Option16_Click
  End If
  End Sub

  Private Sub Option19_Click()          //Runge-Kutta-Fehlberg Method for Response Spectra
  If (Option17.Value = True) Then
    Option17_Click
```

```
End If
If (Option15.Value = True) Then
   Option15_Click
End If
If (Option16.Value = True) Then
   Option16_Click
End If
End Sub

Private Sub Option20_Click()          //Adam-Moulton Method for Response
If (Option6.Value = True) Then
   Option6_Click
End If
If (Option7.Value = True) Then
   Option7_Click
End If
If (Option8.Value = True) Then
   Option8_Click
End If
End Sub

Private Sub Option21_Click()          //Adam-Moulton Method for Response Spectra
If (Option17.Value = True) Then
   Option17_Click
End If
If (Option15.Value = True) Then
   Option15_Click
End If
If (Option16.Value = True) Then
   Option16_Click
End If
End Sub

Private Sub Option22_Click()          //Interpolation by Excitation for Response
If (Option6.Value = True) Then
   Option6_Click
End If
If (Option7.Value = True) Then
   Option7_Click
End If
If (Option8.Value = True) Then
   Option8_Click
End If
End Sub

Private Sub Option23_Click()          //Interpolation by Excitation for Response Spectra
If (Option17.Value = True) Then
   Option17_Click
End If
If (Option15.Value = True) Then
   Option15_Click
End If
If (Option16.Value = True) Then
   Option16_Click
End If
End Sub
```

```
Private Sub Option3_Click()                    //Acceleration Ground Response
If (List1.ListCount > 0) Then
   Label1.Caption = "Ground Acceleration Values"
   List1.Clear
   For i = 2 To n + 1
      P(i - 2) = CDbl(all(i))
   Next
   If (Option3.Value = True) Then
      For i = 0 To n - 1
         u(i) = P(i)
      Next
   End If
   For i = 0 To n - 1
      List1.AddItem ((i + 1) & "." & vbTab & u(i) & "")
   Next
   Text3.Text = n
   Text4.Text = h
   max = Abs(u(0))
   For i = 1 To n - 1
      If (Abs(u(i)) > Abs(max)) Then
         max = u(i)
      End If
   Next
   Text5.Text = max
   Text7.Text = n * h
End If
End Sub

Private Sub Option4_Click()                    //Velocity Ground Response
If (List1.ListCount > 0) Then
   Label1.Caption = "Ground Velocity Values"
   List1.Clear
   For i = 2 To n + 1
      P(i - 2) = CDbl(all(i)) * 386.0886
   Next
   If (Option4.Value = True) Then
      List1.Clear
      Cumm
   End If
   For i = 0 To n - 1
      List1.AddItem ((i + 1) & "." & vbTab & u(i) & "")
   Next
   Text3.Text = n
   Text4.Text = h
   max = Abs(u(0))
   For i = 1 To n - 1
      If (Abs(u(i)) > Abs(max)) Then
         max = u(i)
      End If
   Next
   Text5.Text = max
   Text7.Text = n * h
End If
End Sub
```

```
Private Sub Option5_Click()                    //Displacement Ground Response
If (List1.ListCount > 0) Then
   Label1.Caption = "Ground Displacement Values"
   List1.Clear
   For i = 2 To n + 1
      P(i - 2) = CDbl(all(i)) * 386.0886
   Next
   If (Option5.Value = True) Then
      Cumm
      For i = 0 To n - 1
         P(i) = u(i)
      Next
      Cumm
   End If
   For i = 0 To n - 1
      List1.AddItem ((i + 1) & "." & vbTab & u(i) & "")
   Next
   Text3.Text = n
   Text4.Text = h
   max = Abs(u(0))
   For i = 1 To n - 1
      If (Abs(u(i)) > Abs(max)) Then
         max = u(i)
      End If
   Next
   Text5.Text = max
   Text7.Text = n * h
End If
End Sub

Private Sub Option6_Click()                    //Acceleration Response
On Error GoTo err13:
If (List1.ListCount > 0) Then
   wn = CDbl(Text6.Text)
   eta = CDbl(Text9.Text)
   List2.Clear
   Label3.Caption = "Acceleration Response Values"
   For i = 0 To n - 1
      P(i) = all(i + 2)
   Next
   If (Option12.Value = True) Then
      RKF
   End If
   If (Option13.Value = True) Then
      RK
   End If
   If (Option20.Value = True) Then
      AMF
   End If
   If (Option22.Value = True) Then
      IBE
   End If
   For i = 0 To n - 1
      r(i) = acc(i)
      List2.AddItem ((i + 1) & "." & vbTab & acc(i))
   Next
```

```
   maxr = Abs(acc(0))
   For i = 1 To n - 1
      If (Abs(acc(i)) > Abs(maxr)) Then
         maxr = acc(i)
      End If
   Next
   Text8.Text = maxr
Else
   Option6.Value = False
End If
Exit Sub
err13:
   Option6.Value = False
   MsgBox ("Enter Valid Values For Time Period And Damping Ratio")
End Sub

Private Sub Option7_Click()                    //Velocity Response
On Error GoTo err14:
If (List1.ListCount > 0) Then
   wn = CDbl(Text6.Text)
   eta = CDbl(Text9.Text)
   For i = 2 To n + 1
      P(i - 2) = CDbl(all(i)) * 386.0886
   Next
   If (Option12.Value = True) Then
      RKF
   End If
   If (Option13.Value = True) Then
      RK
   End If
   If (Option20.Value = True) Then
      AMF
   End If
   If (Option22.Value = True) Then
      IBE
   End If
   List2.Clear
   Label3.Caption = "Displacement Response Values"
   For i = 0 To n - 1
      r(i) = du(i)
      List2.AddItem ((i + 1) & "." & vbTab & du(i))
   Next
   maxr = Abs(du(0))
   For i = 1 To n - 1
      If (Abs(du(i)) > Abs(maxr)) Then
         maxr = du(i)
      End If
   Next
   Text8.Text = maxr
End If
Exit Sub
err14:
   Option7.Value = False
   MsgBox ("Enter Valid Values For Time Period And Damping Ratio")
End Sub
```

```
Private Sub Option8_Click()                    //Displacement Response
On Error GoTo err15:
If (List1.ListCount > 0) Then
   wn = CDbl(Text6.Text)
   eta = CDbl(Text9.Text)
   For i = 2 To n + 1
      P(i - 2) = CDbl(all(i)) * 386.0886
   Next
   If (Option12.Value = True) Then
      RKF
   End If
   If (Option13.Value = True) Then
      RK
   End If
   If (Option20.Value = True) Then
      AMF
   End If
   If (Option22.Value = True) Then
      IBE
   End If
   List2.Clear
   Label3.Caption = "Velocity Response Values"
   For i = 0 To n - 1
      r(i) = ddu(i)
      List2.AddItem ((i + 1) & "." & vbTab & ddu(i))
   Next
   maxr = Abs(ddu(0))
   For i = 1 To n - 1
      If (Abs(ddu(i)) > Abs(maxr)) Then
         maxr = ddu(i)
      End If
   Next
   Text8.Text = maxr
Else
   Option8.Value = False
End If
Exit Sub
err15:
   Option8.Value = False
   MsgBox ("Enter Valid Values For Time Period And Damping Ratio")
End Sub

Private Sub Text10_KeyPress(KeyAscii As Integer)          //Time Period for Response Spectra
On Error GoTo err23:
If (KeyAscii = 13 And wn <> CDbl(Text10.Text)) Then
   If (Option17.Value = True) Then
      Option17_Click
   End If
   If (Option15.Value = True) Then
      Option15_Click
   End If
   If (Option16.Value = True) Then
      Option16_Click
   End If
End If
Exit Sub
```

```
err23:
   Exit Sub
End Sub

Private Sub Text11_KeyPress(KeyAscii As Integer)          //Damping for Response Spectra
On Error GoTo err24:
If (KeyAscii = 13 And eta <> CDbl(Text11.Text)) Then
   If (Option17.Value = True) Then
      Option17_Click
   End If
   If (Option15.Value = True) Then
      Option15_Click
   End If
   If (Option16.Value = True) Then
      Option16_Click
   End If
End If
Exit Sub
err24:
   Exit Sub
End Sub

Private Sub Text6_KeyPress(KeyAscii As Integer)           //Natural Frequency for Response
On Error GoTo err16:
If (KeyAscii = 13 And wn <> CDbl(Text6.Text)) Then
   If (Option6.Value = True) Then
      Option6_Click
   End If
   If (Option7.Value = True) Then
      Option7_Click
   End If
   If (Option8.Value = True) Then
      Option8_Click
   End If
End If
Exit Sub
err16:
   Exit Sub
End Sub

Private Sub Text9_KeyPress(KeyAscii As Integer)           //Damping for Response
On Error GoTo err17:
If (KeyAscii = 13 And eta <> CDbl(Text9.Text)) Then
   If (Option6.Value = True) Then
      Option6_Click
   End If
   If (Option7.Value = True) Then
      Option7_Click
   End If
   If (Option8.Value = True) Then
      Option8_Click
   End If
End If
err17:
   Exit Sub
End Sub
```

```
Private Sub Timer1_Timer()                    //Animate Ground Response
On Error GoTo err6:
Do While (cur < no)
   If (ct > n - 1) Then
      ani = 0
      Timer1.Interval = 0
      Form3.Text1.Text = ""
      Exit Sub
   End If
   If (ct = 0) Then
      Form3.Line (xval, yval)-(xval + Int(tt * scx), yval - Int(u(0) * scy))
      x1 = xval + Int(tt * scx)
      y1 = yval - Int(u(0) * scy)
      tt = tt + h
   Else
      Form3.Line (x1, y1)-(xval + Int(tt * scx), yval - Int(u(ct) * scy))
      If (Abs(max) < Abs(min) And u(ct) = min) Then
         Form3.Shape1.Left = xval + Int(tt * scx) - 75
         Form3.Shape1.Top = yval - Int(u(ct) * scy) - 75
         Form3.Label14.Left = xval + Int(tt * scx) + 100
         Form3.Label14.Top = yval - Int(u(ct) * scy)
         Form3.Label14.Caption = CInt(100 * min) / 100
         Form3.Shape1.Visible = True
         Form3.Label14.Visible = True
      End If
      If (Abs(max) > Abs(min) And u(ct) = max) Then
         Form3.Shape1.Left = xval + Int(tt * scx) - 75
         Form3.Shape1.Top = yval - Int(u(ct) * scy) - 75
         Form3.Label14.Left = xval + Int(tt * scx) + 100
         Form3.Label14.Top = yval - Int(u(ct) * scy)
         Form3.Label14.Caption = CInt(100 * max) / 100
         Form3.Shape1.Visible = True
         Form3.Label14.Visible = True
      End If
      x1 = xval + Int(tt * scx)
      y1 = yval - Int(u(ct) * scy)
      tt = tt + h
   End If
   ct = ct + 1
   cur = cur + 1
Loop
Exit Sub
err6:
   Resume Next
End Sub

Public Function save()                    //Save Ground Response
filenum1 = FreeFile
Open Text1 For Output As filenum1
   Print #filenum1, n & ""
   Print #filenum1, h & ""
   For i = 0 To n - 1
      Print #filenum1, u(i) & ""
   Next
Close filenum1
End Function
```

```
Public Function saver()                    //Save Response
filenum1 = FreeFile
Open Text1 For Output As filenum1
   Print #filenum1, n & ""
   Print #filenum1, h & ""
   For i = 0 To n - 1
      Print #filenum1, r(i) & ""
   Next
Close filenum1
End Function

Public Function savers()                   //Save Response Spectrum
filenum1 = FreeFile
Open Text1 For Output As filenum1
   Print #filenum1, n & ""
   Print #filenum1, tn & ""
   For i = 0 To n - 1
      Print #filenum1, rs(i) & ""
   Next
Close filenum1
End Function

Public Function CalculateDisp()            //Calculate Displacement
On Error GoTo err1:
filenum = FreeFile
Open Form2.Text1.Text For Input As filenum
   str = Input(LOF(filenum), filenum)
   all = Split(str, vbNewLine)
   n = CInt(all(0))
   h = CDbl(all(1))
   List1.Clear
   Option6.Value = False
   Option7.Value = False
   Option8.Value = False
   For i = 2 To n + 1
      u(i - 2) = CDbl(all(i))
      List1.AddItem ((i - 1) & "." & vbTab & u(i - 2) & "")
   Next
Close filenum
Text3.Text = n
Text4.Text = h
max = Abs(u(0))
For i = 1 To n - 1
   If (Abs(u(i)) > Abs(max)) Then
      max = u(i)
   End If
Next
Text5.Text = max
Text7.Text = n * h
Exit Function
err1:
   MsgBox ("Error Opening File " & Form2.Text1.Text)
   List1.Clear
   List2.Clear
   Text8.Text = ""
   Option6.Value = False
```

```
   Option7.Value = False
   Option8.Value = False
End Function

Public Function Repaint()                    //Repaint Ground Response
On Error GoTo err4:
   scx = 10500 / ((xx2 - xx1 + 1) * h)
   max = u(xx1)
   min = u(xx1)
   For i = xx1 To xx2
      If (u(i) > max) Then
         max = u(i)
      End If
      If (u(i) < min) Then
         min = u(i)
      End If
   Next
   scy = (yval * 1.3) / (max - min)
   tt = h
   Form3.Line (xval, yval)-(xval + Int(tt * scx), yval - Int(u(xx1) * scy))
   x1 = xval + Int(tt * scx)
   y1 = yval - Int(u(xx1) * scy)
   tt = tt + h
   For i = xx1 To xx2
      Form3.Line (x1, y1)-(xval + Int(tt * scx), yval - Int(u(i) * scy))
      x1 = xval + Int(tt * scx)
      y1 = yval - Int(u(i) * scy)
      tt = tt + h
   Next
Exit Function
err4:
   Resume Next
End Function

Public Function Repaintr()                   //Repaint Response
On Error GoTo err10:
   scxr = 10500 / ((xx2r - xx1r + 1) * h)
   maxr = r(xx1r)
   minr = r(xx1r)
   For i = xx1r To xx2r
      If (r(i) > maxr) Then
         maxr = r(i)
      End If
      If (r(i) < minr) Then
         minr = r(i)
      End If
   Next
   scyr = (yvalr * 1.3) / (maxr - minr)
   ttr = h
   Form5.Line (xvalr, yvalr)-(xvalr + Int(ttr * scxr), yvalr - Int(r(xx1r) * scyr))
   x1r = xvalr + Int(ttr * scxr)
   y1r = yvalr - Int(r(xx1r) * scyr)
   ttr = ttr + h
   For i = xx1r To xx2r
      Form5.Line (x1r, y1r)-(xvalr + Int(ttr * scxr), yvalr - Int(r(i) * scyr))
      x1r = xvalr + Int(ttr * scxr)
```

```
      y1r = yvalr - Int(r(i) * scyr)
      ttr = ttr + h
   Next
Exit Function
err10:
   Resume Next
End Function

Public Function Repaintrs()                    //Repaint Response Spectrum
On Error GoTo err25:
   scxrs = 10500 / ((xx2rs - xx1rs + 1) * tn)
   maxrs = rs(xx1rs)
   minrs = rs(xx1rs)
   For i = xx1rs To xx2rs
      If (rs(i) > maxrs) Then
         maxrs = rs(i)
      End If
      If (rs(i) < minrs) Then
         minrs = rs(i)
      End If
   Next
   scyrs = (yvalrs) / (maxrs * 1.3) '- minrs)
   ttrs = 0
   Form6.Line (xvalrs, yvalrs)-(xvalrs + Int(ttrs * scxrs), yvalrs - Int(r(xx1rs) * scyrs))
   x1rs = xvalrs + Int(ttrs * scxrs)
   y1rs = yvalrs - Int(rs(xx1rs) * scyrs)
   ttrs = ttrs + tn
   For i = xx1rs To xx2rs
      Form6.Line (x1rs, y1rs)-(xvalrs + Int(ttrs * scxrs), yvalrs - Int(rs(i) * scyrs))
      If (rs(i) = maxrs) Then
         Form6.Shape1.Left = x1rs + Int(ttrs * scxrs) - 75
         Form6.Shape1.Top = y1rs - Int(rs(i) * scyrs) - 75
         Form6.Label14.Left = x1rs + Int(ttrs * scxrs) + 100
         Form6.Label14.Top = y1rs - Int(rs(i) * scyrs)
         Form6.Label14.Caption = maxrs 'CInt(100 * maxrs) / 100
         Form6.Shape1.Visible = True
         Form6.Label14.Visible = True
      End If
      x1rs = xvalrs + Int(ttrs * scxrs)
      y1rs = yvalrs - Int(rs(i) * scyrs)
      ttrs = ttrs + tn
   Next
Exit Function
err25:
   Resume Next
End Function

Public Function Zoom()                 //Zoom Ground Response
xx1 = CInt(Form3.Text2.Text)
xx2 = CInt(Form3.Text3.Text)
xx1 = Int(xx1 / (scx * h))
xx2 = Int(xx2 / (scx * h))
Form3.Label3.Caption = (xx1 * h)
Form3.Label4.Caption = (xx2 * h)
If (xx2 > n - 1) Then
   xx2 = n - 1
```

```
End If
If (xx1 < 0) Then
   xx1 = 0
End If
mark3
End Function

Public Function Zoomr()              //Zoom Response
xx1r = CInt(Form5.Text2.Text)
xx2r = CInt(Form5.Text3.Text)
xx1r = Int(xx1r / (scxr * h))
xx2r = Int(xx2r / (scxr * h))
Form5.Label3.Caption = (xx1r * h)
Form5.Label4.Caption = (xx2r * h)
If (xx2r > n - 1) Then
   xx2r = n - 1
End If
If (xx1r < 0) Then
   xx1r = 0
End If
mark5
End Function

Public Function Zoomrs()                    //Zoom Response Spectra
xx1rs = CInt(Form6.Text2.Text)
xx2rs = CInt(Form6.Text3.Text)
xx1rs = Int(xx1rs / (scxrs * tn))
xx2rs = Int(xx2rs / (scxrs * tn))
Form6.Label3.Caption = (xx1rs * tn)
Form6.Label4.Caption = (xx2rs * tn)
If (xx2rs > n - 1) Then
   xx2rs = n - 1
End If
If (xx1rs < 0) Then
   xx1rs = 0
End If
mark6
End Function

Public Function Original()                  //Zoom to Previous Ground Response
xx1 = 0
xx2 = n - 1
xx1r = 0
xx2r = n - 1
xx1rs = 0
xx2rs = n - 1
Form3.Label3.Caption = (xx1 * h)
Form3.Label4.Caption = (xx2 * h)
Form5.Label3.Caption = (xx1r * h)
Form5.Label4.Caption = (xx2r * h)
Form6.Label3.Caption = (xx1rs * tn)
Form6.Label4.Caption = (xx2rs * tn)
End Function
```

```
Public Function SaveCurDisp()                    //Save Ground Response File
filenum1 = FreeFile
Open Text1 For Output As filenum1
   Print #filenum1, (xx2 - xx1 + 1) & ""
   Print #filenum1, h & ""
   For i = xx1 To xx2
      Print #filenum1, u(i) & ""
   Next
Close filenum1
End Function

Public Function Trapizoidal()                    //Trapiziodal Rule
t = h
a = h / (2 * m * wd)
   For i = 0 To n - 1
      X(i) = Exp(e * wd * t) * P(i) * Cos(wd * t)
      Y(i) = Exp(e * wd * t) * P(i) * Sin(wd * t)
      Aa(i) = X(0)
      If (i > 0) Then
         For j = 1 To i - 1
            Aa(i) = Aa(i) + 2 * X(j)
         Next
      End If
      If (i > 1) Then
         Aa(i) = Aa(i) + X(i)
      End If
      B(i) = Y(0)
      If (i > 0) Then
         For j = 1 To i - 1
            B(i) = B(i) + 2 * Y(j)
         Next
      End If
      If (i > 1) Then
         B(i) = B(i) + Y(i)
      End If
      Aa(i) = Aa(i) * a
      B(i) = B(i) * a
      term1 = Aa(i) * Exp((-1) * wd * t) * Sin(wd * t) / (t)
      term2 = B(i) * Exp((-1) * wd * t) * Sin(wd * t) / (t)
      u(i) = term1 - term2
      t = t + h
   Next
End Function

Public Function CummOpp()                        //Ground Displacement
t = h
u(0) = t * P(0)
t = t + h
For i = 1 To n - 1
   u(i) = u(i - 1) + ((t * P(i)) / i)
   t = t + h
Next
End Function
```

```
Public Function Cumm()                    //Ground Velocity
t = h
u(0) = t * P(0)
t = t + h
For i = 1 To n - 1
   area = (h * P(i)) + (0.5 * (P(i) - P(i - 1)) * h)
   u(i) = u(i - 1) + (area)
   t = t + h
Next
End Function

Public Function RK()                      //Runge-Kutta Method
du(0) = 0
ddu(0) = 0
For i = 0 To n - 2
   k11 = h * (((1) * P(i)) - (2 * eta * wn * du(i)) - (wn * wn * ddu(i)))
   k12 = h * (((1) * P(i)) - (2 * eta * wn * (du(i) + (k11 / 2))) - (wn * wn * (ddu(i) + (h / 2))))
   k13 = h * (((1) * P(i)) - (2 * eta * wn * (du(i) + (k12 / 2))) - (wn * wn * (ddu(i) + (h / 2))))
   k14 = h * (((1) * P(i)) - (2 * eta * wn * (du(i) + k13)) - (wn * wn * (ddu(i) + h)))
   ddu(i + 1) = (ddu(i) + ((1 / 6) * (k11 + (2 * k12) + (2 * k13) + k14)))
   k21 = h * ddu(i)
   k22 = h * (ddu(i) + (k21 / 2))
   k23 = h * (ddu(i) + (k22 / 2))
   k24 = h * (ddu(i) + k23)
   du(i + 1) = (du(i) + ((1 / 6) * (k21 + (2 * k22) + (2 * k23) + k24)))
   acc(i) = P(i) - (2 * eta * wn * du(i)) - (wn * wn * ddu(i))
Next
End Function

Public Function RKF()                     //Runge-Kutta-Fehlberg Method
du(0) = 0
ddu(0) = 0
For i = 0 To n - 2
   k11 = h * (((1) * P(i)) - (2 * eta * wn * du(i)) - (wn * wn * ddu(i)))

   k12 = h * (((1) * P(i)) - (2 * eta * wn * (du(i) + (k11 / 4))) - (wn * wn * (ddu(i) + (h / 4))))

   k13 = h * (((1) * P(i)) - (2 * eta * wn * (du(i) + (3 * k11 / 32) + (9 * k12 / 32))) - (wn * wn * (ddu(i) +
(3 * h / 8))))

   k14 = h * (((1) * P(i)) - (2 * eta * wn * (du(i) + (1932 * k11 / 2197) - (7200 * k12 / 2197) + (7296 * k13
/ 2197))) - (wn * wn * (ddu(i) + (12 * h / 13))))

   k15 = h * (((1) * P(i)) - (2 * eta * wn * (du(i) + (439 * k11 / 216) - (8 * k12) + (3680 * k13 / 513) - (845
* k14 / 4104))) - (wn * wn * (ddu(i) + h)))

   k16 = h * (((1) * P(i)) - (2 * eta * wn * (du(i) - (8 * k11 / 27) + (2 * k12) - (3544 * k13 / 2565) + (1859 *
k14 / 4104) - (11 * k15 / 40))) - (wn * wn * (ddu(i) + (h / 2))))

   ddu(i + 1) = ddu(i) + ((16 * k11 / 135) + (6656 * k13 / 12825) + (28561 * k14 / 56430) - (9 * k15 / 50)
+ (2 * k16 / 55))

   k21 = h * ddu(i)
   k22 = h * (ddu(i) + (k21 / 4))
   k23 = h * (ddu(i) + ((3 * k21 / 32) + (9 * k22 / 32)))
   k24 = h * (ddu(i) + ((1932 * k21 / 2197) - (7200 * k22 / 2197) + (7296 * k23 / 2197)))
```

```
   k25 = h * (ddu(i) + ((439 * k21 / 216) - (8 * k22) + (3680 * k23 / 513) - (845 * k24 / 4104)))
   k26 = h * (ddu(i) - ((8 * k21 / 27) + (2 * k22) - (3544 * k23 / 2565) + (1859 * k24 / 4104) - (11 * k25 / 40)))

   du(i + 1) = du(i) + ((16 * k21 / 135) + (6656 * k23 / 12825) + (28561 * k24 / 56430) - (9 * k25 / 50) + (2 * k26 / 55))

   acc(i) = P(i) - (2 * eta * wn * du(i)) - (wn * wn * ddu(i))
Next
End Function

Public Function AMF()                    //Adam-Moulton Method
du(0) = 0
ddu(0) = 0
For i = 0 To 3
   k11 = h * (((1) * P(i)) - (2 * eta * wn * du(i)) - (wn * wn * ddu(i)))
   k12 = h * (((1) * P(i)) - (2 * eta * wn * (du(i) + (k11 / 2))) - (wn * wn * (ddu(i) + (h / 2))))
   k13 = h * (((1) * P(i)) - (2 * eta * wn * (du(i) + (k12 / 2))) - (wn * wn * (ddu(i) + (h / 2))))
   k14 = h * (((1) * P(i)) - (2 * eta * wn * (du(i) + k13)) - (wn * wn * (ddu(i) + h)))
   ddu(i + 1) = (ddu(i) + ((1 / 6) * (k11 + (2 * k12) + (2 * k13) + k14)))

   k21 = h * ddu(i)
   k22 = h * (ddu(i) + (k21 / 2))
   k23 = h * (ddu(i) + (k22 / 2))
   k24 = h * (ddu(i) + k23)
   du(i + 1) = (du(i) + ((1 / 6) * (k21 + (2 * k22) + (2 * k23) + k24)))

   acc(i) = P(i) - (2 * eta * wn * du(i)) - (wn * wn * ddu(i))
Next
For i = 4 To n - 2
   ddu(i + 1) = ddu(i) + ((h / 24) * ((55 * du(i)) - (59 * du(i - 1)) + (37 * du(i - 2)) - (9 * du(i - 3))))
   du(i + 1) = du(i) + ((h / 24) * ((55 * acc(i)) - (59 * acc(i - 1)) + (37 * acc(i - 2)) - (9 * acc(i - 3))))
   acc(i + 1) = P(i) - (2 * eta * wn * du(i)) - (wn * wn * ddu(i))
Next
End Function

Public Function IBE()                    //Method of Interpolation by Excitation
For i = 0 To n - 1
   du(i) = 0
   ddu(i) = 0
Next
a11 = 0
a12 = 0
a21 = 0
a22 = 0
b11 = 0
b12 = 0
b21 = 0
b22 = 0
a11 = Exp((-1) * eta * wn * h) * (((eta / (1 - (eta * eta)) ^ 0.5) * Sin(wn * h * (1 - (eta * eta)) ^ 0.5)) + Cos(wn * h * (1 - (eta * eta)) ^ 0.5))

a12 = Exp((-1) * eta * wn * h) * Sin(wn * h * (1 - (eta * eta)) ^ 0.5) / (wn * (1 - (eta * eta)) ^ 0.5)

a21 = (-1) * wn * Exp((-1) * eta * wn * h) * wn * Sin(wn * h * (1 - (eta * eta)) ^ 0.5) / ((1 - (eta * eta)) ^ 0.5)
```

```
a22 = (Exp((-1) * eta * wn * h) * Cos(wn * h * (1 - (eta * eta)) ^ 0.5)) - ((eta / (1 - (eta * eta)) ^ 0.5) *
Sin(wn * h * (1 - (eta * eta)) ^ 0.5))

b11 = ((((((2 * eta * eta) - 1) / (wn * wn * h)) + (eta / wn)) * Sin(wn * h * (1 - (eta * eta)) ^ 0.5) / (wn * (1 -
(eta * eta)) ^ 0.5)) + (((((2 * eta * eta) - 1) / (wn * wn * wn * h)) + (1 / (wn * wn))) * Cos(wn * h * (1 - (eta
* eta)) ^ 0.5)) * Exp((-1) * eta * wn * h)) - ((2 * eta) / (wn * wn * wn * h))

b12 = ((((((2 * eta * eta) - 1) / (wn * wn * h))) * Sin(wn * h * (1 - (eta * eta)) ^ 0.5) / (wn * (1 - (eta * eta))
^ 0.5)) + (((((2 * eta * eta) - 1) / (wn * wn * wn * h))) * Cos(wn * h * (1 - (eta * eta)) ^ 0.5)) * (-1) * Exp((-
1) * eta * wn * h)) - (1 / (wn * wn)) + ((2 * eta) / (wn * wn * wn * h))

b21 = (((((((2 * eta * eta) - 1) / (wn * wn * h)) + (eta / wn)) * ((-1) * (eta / ((1 - (eta * eta)) ^ 0.5)) * Sin(wn
* h * (1 - (eta * eta)) ^ 0.5) / (wn * (1 - (eta * eta)) ^ 0.5)) + (Cos(wn * h * (1 - (eta * eta)) ^ 0.5)))) - ((((((2
* eta) - 1) / (wn * wn * wn * h)) + (1 / (wn * wn))) * (wn * ((1 - (eta * eta)) ^ 0.5) * Sin(wn * h * (1 - (eta *
eta)) ^ 0.5)) + (wn * eta * Cos(wn * h * (1 - (eta * eta)) ^ 0.5)))) * Exp((-1) * eta * wn * h)) + ((1) / (wn *
wn * h))

b22 = (((((((2 * eta * eta) - 1) / (wn * wn * h))) * ((-1) * (eta / ((1 - (eta * eta)) ^ 0.5)) * Sin(wn * h * (1 -
(eta * eta)) ^ 0.5) / (wn * (1 - (eta * eta)) ^ 0.5)) + (Cos(wn * h * (1 - (eta * eta)) ^ 0.5)))) - ((((((2 * eta) -
1) / (wn * wn * wn * h))) * (wn * ((1 - (eta * eta)) ^ 0.5) * Sin(wn * h * (1 - (eta * eta)) ^ 0.5)) + (wn * eta
* Cos(wn * h * (1 - (eta * eta)) ^ 0.5)))) * (-1) * Exp((-1) * eta * wn * h)) - ((1) / (wn * wn * h))

For i = 0 To n - 2
   ddu(i + 1) = (a11 * ddu(i)) + (a12 * du(i)) + (b11 * P(i)) + (b12 * P(i + 1))
   du(i) = (a21 * ddu(i)) + (a22 * du(i)) + (b21 * P(i)) + (b22 * P(i + 1))
   acc(i + 1) = (-1) * ((2 * eta * wn * du(i + 1)) + (wn * wn * ddu(i + 1)))
Next
End Function

Private Sub Timer2_Timer()                    //Animate Response
On Error GoTo err8:
curr = 0
Do While (curr < nor)
   If (ctr > n - 1) Then
      anir = 0
      Timer2.Interval = 0
      Form5.Text1.Text = ""
      Exit Sub
   End If
   If (ctr = 0) Then
      Form5.Line (xvalr, yvalr)-(xvalr + Int(ttr * scxr), yvalr - Int(r(0) * scyr))
      x1r = xvalr + Int(ttr * scxr)
      y1r = yvalr - Int(r(0) * scyr)
      ttr = ttr + h
   Else
      Form5.Line (x1r, y1r)-(xvalr + Int(ttr * scxr), yvalr - Int(r(ctr) * scyr))
      If (Abs(maxr) < Abs(minr) And r(ctr) = minr) Then
         Form5.Shape1.Left = xvalr + Int(ttr * scxr) - 75
         Form5.Shape1.Top = yvalr - Int(r(ctr) * scyr) - 75
         Form5.Label14.Left = xvalr + Int(ttr * scxr) + 100
         Form5.Label14.Top = yvalr - Int(r(ctr) * scyr)
         Form5.Label14.Caption = CInt(100 * minr) / 100
         Form5.Shape1.Visible = True
         Form5.Label14.Visible = True
      End If
```

```
    If (Abs(maxr) > Abs(minr) And r(ctr) = maxr) Then
       Form5.Shape1.Left = xvalr + Int(ttr * scxr) - 75
       Form5.Shape1.Top = yvalr - Int(r(ctr) * scyr) - 75
       Form5.Label14.Left = xvalr + Int(ttr * scxr) + 100
       Form5.Label14.Top = yvalr - Int(r(ctr) * scyr)
       Form5.Label14.Caption = CInt(100 * maxr) / 100
       Form5.Shape1.Visible = True
       Form5.Label14.Visible = True
    End If
    x1r = xvalr + Int(ttr * scxr)
    y1r = yvalr - Int(r(ctr) * scyr)
    ttr = ttr + h
  End If
  ctr = ctr + 1
  curr = curr + 1
Loop
Exit Sub
err8:
  Resume Next
End Sub

Private Sub Timer3_Timer()                    //Animate Response Spectra
On Error GoTo err24:
currs = 0
Do While (currs < nors)
  If (ctrs > n - 1) Then
    anirs = 0
    Timer3.Interval = 0
    Form6.Text1.Text = ""
    Exit Sub
  End If
  If (ctrs = 0) Then
    Form6.Line (xvalrs, yvalrs)-(xvalrs + Int(ttrs * scxrs), yvalrs - Int(rs(0) * scyrs))
    x1rs = xvalrs + Int(ttrs * scxrs)
    y1rs = yvalrs - Int(rs(0) * scyrs)
    ttrs = ttrs + tn
  Else
    Form6.Line (x1rs, y1rs)-(xvalrs + Int(ttrs * scxrs), yvalrs - Int(rs(ctrs) * scyrs))
    'If (Abs(maxrs) < Abs(minrs) And rs(ctrs) = minrs) Then
      'Form6.Shape1.Left = xvalrs + Int(ttrs * scxrs) - 75
      'Form6.Shape1.Top = yvalrs - Int(r(ctrs) * scyrs) - 75
      'Form6.Label14.Left = xvalrs + Int(ttrs * scxrs) + 100
      'Form6.Label14.Top = yvalrs - Int(rs(ctrs) * scyrs)
      'Form6.Label14.Caption = CInt(100 * minrs) / 100
      'Form6.Shape1.Visible = True
      'Form6.Label14.Visible = True
    'End If
    If (rs(ctrs) = maxrs) Then
      Form6.Shape1.Left = xvalrs + Int(ttrs * scxrs) - 75
      Form6.Shape1.Top = yvalrs - Int(rs(ctrs) * scyrs) - 75
      Form6.Label14.Left = xvalrs + Int(ttrs * scxrs) + 100
      Form6.Label14.Top = yvalrs - Int(rs(ctrs) * scyrs)
      Form6.Label14.Caption = maxrs 'CInt(100 * maxrs) / 100
      Form6.Shape1.Visible = True
      Form6.Label14.Visible = True
    End If
```

```
      x1rs = xvalrs + Int(ttrs * scxrs)
      y1rs = yvalrs - Int(rs(ctrs) * scyrs)
      ttrs = ttrs + tn
   End If
   ctrs = ctrs + 1
   currs = currs + 1
Loop
Exit Sub
err24:
   Resume Next
End Sub

Public Function mark3()                    //Mark Points in Ground Response
On Error GoTo err11:
Form3.Label5.Caption = CInt(10 * ((xx1 * h) + ((xx2 - xx1) * h / 5))) / 10
Form3.Label6.Caption = CInt(10 * ((xx1 * h) + (2 * (xx2 - xx1) * h / 5))) / 10
Form3.Label7.Caption = CInt(10 * ((xx1 * h) + (3 * (xx2 - xx1) * h / 5))) / 10
Form3.Label8.Caption = CInt(10 * ((xx1 * h) + (4 * (xx2 - xx1) * h / 5))) / 10
Exit Function
err11:
   Resume Next
End Function

Public Function mark5()                    //Mark Points in Response
On Error GoTo err12:
Form5.Label5.Caption = CInt(10 * ((xx1r * h) + ((xx2r - xx1r) * h / 5))) / 10
Form5.Label6.Caption = CInt(10 * ((xx1r * h) + (2 * (xx2r - xx1r) * h / 5))) / 10
Form5.Label7.Caption = CInt(10 * ((xx1r * h) + (3 * (xx2r - xx1r) * h / 5))) / 10
Form5.Label8.Caption = CInt(10 * ((xx1r * h) + (4 * (xx2r - xx1r) * h / 5))) / 10
Exit Function
err12:
   Resume Next
End Function

Public Function mark6()                    //Mark Points in Response Spectrum
On Error GoTo err13:
Form6.Label5.Caption = CInt(10 * ((xx1rs * h) + ((xx2rs - xx1rs) * CDbl(Text10.Text) / 5))) / 10
Form6.Label6.Caption = CInt(10 * ((xx1rs * h) + (2 * (xx2rs - xx1rs) * CDbl(Text10.Text) / 5))) / 10
Form6.Label7.Caption = CInt(10 * ((xx1rs * h) + (3 * (xx2rs - xx1rs) * CDbl(Text10.Text) / 5))) / 10
Form6.Label8.Caption = CInt(10 * ((xx1rs * h) + (4 * (xx2rs - xx1rs) * CDbl(Text10.Text) / 5))) / 10
Exit Function
err13:
   Resume Next
End Function
```