

# Stability Analysis of Vibration Absorbers

By

**Ashraf Uddin Ahmed**

A thesis

Submitted to the

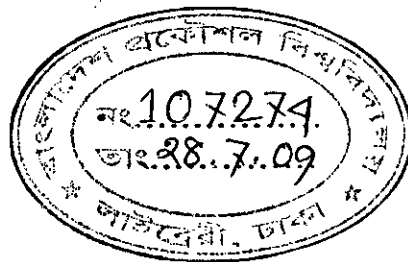
Department of Mechanical Engineering

in partial fulfillment of the requirements for the degree

of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

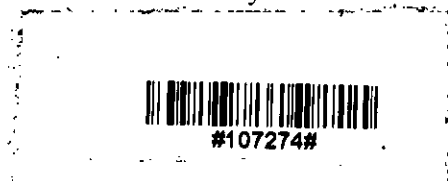
Department of Mechanical Engineering



BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY

Dhaka-1000, Bangladesh

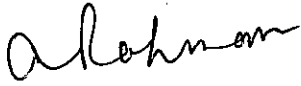
July 2009



# CERTIFICATE OF APPROVAL

This thesis titled, “**Stability Analysis of Vibration Absorbers**”, submitted by **Ashraf Uddin Ahmed**, Roll no: 100710056 F, Session: October 2007 has been accepted as satisfactory in partial fulfillment of the requirement for the Degree of MASTER OF SCIENCE IN MECHANICAL ENGINEERING on 22<sup>nd</sup> July, 2009.

## BOARD OF EXAMINERS



---

**Dr. M. Ashiqur Rahman**  
Professor  
Department of Mechanical Engineering  
BUET, Dhaka-1000, Bangladesh.

Chairman  
(Supervisor)



---

**Dr. Md. Maksud Helali**  
Professor  
Department of Mechanical Engineering  
BUET, Dhaka-1000, Bangladesh.


Member



---

**Dr. Abu Rayhan Md. Ali**  
Professor  
Department of Mechanical Engineering  
BUET, Dhaka-1000, Bangladesh.

Member  
(Ex-Officio)



---

**Dr. Muhammad Fazli Ilahi**  
Professor (Ex- VC, IUT, OIC)  
House No. 13, Road No.1, Sector 11,  
Uttara Model Town, Dhaka.

Member  
(External)

**To my parents**

## Contents

	<b>Page No</b>
<b>Title Page</b>	i
<b>Board of Examiners</b>	ii
<b>Dedication</b>	iii
<b>Contents</b>	iv
<b>List of Symbols and Abbreviations</b>	vi
<b>Acknowledgement</b>	viii
<b>Abstract</b>	ix
<b>Chapter 1</b>	<b>Introduction</b>
1.1	Degrees of Freedom 2
1.2	Concept of Stability 3
1.3	Undamped Vibration Absorber 3
1.4	Shock Absorber 4
1.5	Nonlinear Springs and Dampers 5
1.6	Objective of this Investigation 6
1.7	Outline of Methodology 7
<b>Chapter 2</b>	<b>Literature Review</b>
	Literature Review 8
<b>Chapter 3</b>	<b>Governing Equation</b>
3.1	Mathematical Models and Governing Equations 11
3.2	Boundary Conditions 18
<b>Chapter 4</b>	<b>Methods of Solution</b>
4.1	Multisegment Integration Technique 20
<b>Chapter 5</b>	<b>Results and Discussion</b>
5.1	Stability Analysis of Tuned Vibration Absorber 26
5.2	Stability Analysis of Untuned Vibration Absorbers 30
5.2.1	Stability Analysis of Untuned Vibration Absorber (Case 6) Solved as Initial Value & Boundary Value Problem 30
5.2.1.1	Stability Analysis of Untuned Vibration Absorber (Case 6) Solved as 30

	Initial Value Problem	
5.2.1.2	Stability Analysis of Untuned Vibration Absorber (Case 6) Solved as Boundary Value Problem	31
5.2.2	Stability Analysis of Untuned Vibration Absorber Solved as Boundary Value Problem for Data Set # 1	31
5.2.3	Stability Analysis of Untuned Vibration Absorber Solved as Boundary Value Problem for Data Set # 2	33
<b>Chapter 6</b>	<b>Conclusions and Recommendations</b>	
6.1	Conclusions for Tuned Vibration Absorbers	37
6.2	Conclusions for Untuned Vibration Absorbers	38
6.3	Recommendations for Future Works	39
<b>References</b>		40
<b>Tables</b>		43
<b>Figures</b>		49
<b>Appendix-A</b>		
A.1	Extension for 3 DOFS	82
<b>Appendix-B</b>		
B.1	Some Results for 20s Vibration	89
<b>Appendix-C</b>	<b>Programme Code</b>	
C.1	Code for Nondimensional Displacement with Time for 2 DOFS	95
C.2	Code for Nondimensional Displacement with Forcing Frequency for 2 DOFS	109
C.3	Code for Nondimensional Displacement with Time for 3 DOFS	121
C.4	Code for Nondimensional Displacement with Forcing Frequency for 3 DOFS	138

## List of Symbols & Abbreviations

$a$	= Initial time reference
$b$	= Final time reference
$c$	= Damping coefficient (N-s/m)
$c_1, c_2$	= Main mass and absorber mass damping coefficients (N-s/m)
$c_3$	= Third mass damping coefficient (N-s/m)
$c'$	= Damping nonlinearity index (N-s/m <sup>3</sup> )
$c'_1, c'_2$	= Main mass and absorber mass damping nonlinearity indexes (N-s/m <sup>3</sup> )
$c'_3$	= Third mass damping nonlinearity indexes (N-s/m <sup>3</sup> )
$d$	= $k_1 * x_1 / f$ : Nondimensional displacement for main mass
$e$	= $k_1 * x_2 / f$ : Nondimensional displacement for absorber mass
$f$	= Amplitude of the applied force (N)
$F$	= $f \sin(\omega t)$ (N)
$g$	= $k_1 * x_3 / f$ : Nondimensional displacement for 3 <sup>rd</sup> mass of the 3 DOFS
$k$	= Spring constant (N/m)
$k_1, k_2$	= Main mass and absorber mass spring constants (N/m)
$k_3$	= Third mass spring constant (N/m)
$k'$	= Spring nonlinearity index (N/m <sup>3</sup> )
$k'_1, k'_2$	= Main mass and absorber mass spring nonlinearity indexes (N/m <sup>3</sup> )
$k'_3$	= Third mass spring nonlinearity index (N/m <sup>3</sup> )
$m_1, m_2$	= Main mass and absorber mass (kg)
$m_3$	= Third mass of 3 DOFS (kg)
$r$	= Frequency ratio: $\frac{\omega}{\sqrt{\frac{k_1}{m_1}}}$
$r_1$	= Value of $r$ at first natural frequency of the system
$r_2$	= Value of $r$ at second natural frequency of the system
$t$	= Time (s)
$x_1$	= $x$
$x_1, x_2$	= Main mass and absorber mass deflections (m)
$x_3$	= Third mass deflection for 3 DOFS (m)
$x_1, x_2$	= $y_1, y_3$ (m)

$\dot{x}$	=	$v$
$\dot{x}_1, \dot{x}_2$	=	Main mass and absorber mass velocities (m/s)
$\dot{x}_1, \dot{x}_2$	=	$y_2, y_4$ (m/s)
$\zeta$	=	Damping ratio: $\frac{c_1}{2\sqrt{m_1 k_1}}$
$\zeta_0$	=	Optimum damping ratio : $\frac{\mu}{\sqrt{2(1+\mu)(2+\mu)}}$
$\mu$	=	Mass ratio: $m_2/m_1$
$\omega$	=	Forcing frequency (rad/s)
$\omega_1, \omega_2$	=	Natural frequencies of the main mass and absorber masses
DOF	=	Degrees of Freedom
DOFS	=	Degrees of Freedom System
MDOF	=	Multiple Degrees of Freedom
MDOFS	=	Multiple Degrees of Freedom System
NTMD	=	Nonlinear Tuned Mass Damper
ODE	=	Ordinary Differential Equation
SDOF	=	Single Degrees of Freedom
Tuned Absorber	=	2 DOFS without damping
Untuned Absorber	=	2 DOFS with damping
Spring force	=	$kx \pm k'x^3$
Damping force	=	$c\dot{x} \pm c'\dot{x}x^2$
Hard spring	=	Nonlinearity index ( $k'$ ) is positive
Soft spring	=	Nonlinearity index ( $k'$ ) is negative
Hard damper	=	Nonlinearity index ( $c'$ ) is positive
Soft damper	=	Nonlinearity index ( $c'$ ) is negative
Linear spring	=	Nonlinearity index $k' = 0.0$
Linear damper	=	Nonlinearity index $c' = 0.0$

## **Acknowledgements**

The author wishes to express his deep gratitude and indebtedness to Dr. M. Ashiqur Rahman, Professor, Department of Mechanical Engineering, Bangladesh University of Engineering & Technology (BUET), Dhaka for his supervision, guidance, invaluable suggestion, and constructive criticism throughout this investigation and for patiently reading the manuscript and suggesting the improvements.

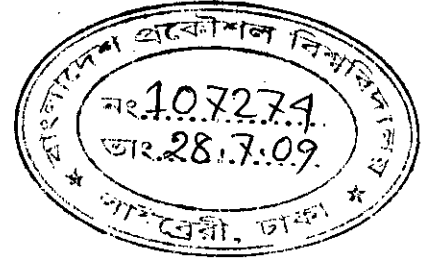


## Abstract

Nonlinear dynamics of a two degrees-of-freedom (DOF) tuned and damped (untuned) vibration absorber systems using nonlinear springs and dampers are studied as a boundary value problem. As far as tuned absorber is concerned, five different combinations of linear and nonlinear springs have been comprehensively analyzed. For the different cases, a comparative study is made varying the forcing frequency. Another comparison is for response versus time for different spring types at three important forcing frequencies: the tuned frequency and two resonant frequencies. Analysis shows that the response of the system is changed because of the spring nonlinearity; the change is different for different cases. Accordingly, an initially stable absorber may become unstable with time and vice versa. Similar investigation is made on untuned vibration absorbers, for 16 different cases varying the spring and damper characteristics. Analysis shows that higher nonlinearity terms make the system more unstable. Change in response is more evident near the frequency ratio of unity. Numerical simulation shows that the systems exhibit quasi periodic motion and instability as system's amplitude increases with time in prescribed boundary conditions. After analyzing 2 DOF system analysis is then made for a linear 3 DOF system.

# Chapter 1

## Introduction



All systems possessing mass and elasticity are capable of free vibration, or vibration that takes place in the absence of external excitation. Of primary interest for such a system is its natural frequency of vibration. Damping in moderate amounts has little influence on the natural frequency and may be neglected in its calculation. The system can then be considered to be conservative, and the principle of the conservation of energy offers another approach to the calculation of the natural frequency. The effect of damping is mainly evident in the diminishing of the vibration amplitude with time. Normal mode vibrations are free undamped vibrations that depend only on the mass and stiffness of the system and how they are distributed. When vibrating at one of these normal modes, all points in the system undergo simple harmonic motion that passes through their equilibrium positions simultaneously. To initiate a normal mode vibration, the system must be given specific initial conditions corresponding to its normal mode. When excitation frequency coincides with one of the natural frequencies of the system, a condition of resonance is encountered with large amplitudes limited only by the damping. Again damping is generally omitted except when its concern is of importance in limiting the amplitude of vibration or in examining the rate of decay of the free oscillation.

Practically vibration problems become nonlinear in nature as amplitude of oscillation becomes large [Kalnins and Dym (1976), Thomson (1981)]. The problem becomes more involved as springs and dampers do not actually behave linearly in vibration problems [Mikhlin and Reshetnikova (2005), Thomson (1981) and Zhu et al. (2004)]. But nonlinear problems, usually having no closed form solutions, are always challenges for practicing engineers. Superposition principle cannot be applied and therefore different mathematical techniques are still developing and being used to solve such problems. Any such numerical technique which makes the computation faster and yield reliable results under any circumstance would be much desirable. For example, Mikhlin and Reshetnikova (2005) studied the nonlinear two-degrees-of-freedom (2DOF) system under consideration consists of a linear oscillator with a relatively big mass which is an approximation of some continuous elastic system, and an essentially nonlinear oscillator with a relatively small mass which is an

absorber of the main linear system vibrations. They analyzed the free and forced vibrations of the system. Zhu et al. (2004) studied nonlinear response of 2DOF vibration system with nonlinear damping and nonlinear spring. They showed the phase plane, change of displacement with forcing frequency. Their study was based on the method of initial value problem. By numerical integration, periodic motions, quasiperiodic motions and chaotic motions of the system were discussed by tracing the bifurcation diagram. Natsiavas (1992) applied the method of averaging to investigate the steady state oscillations and stability of non-linear dynamic vibration absorbers. He pointed out that proper selection of the system parameters would result in substantial improvements of non-linear absorbers and avoid dangerous effects that are likely to occur due to the presence of the non-linearities. Natsiavas (1993) studied the steady state response for a class of strongly non-linear multiple-degree-of-freedom oscillators, where the vibration absorber is modelled as a mass with linear damping and restoring force. A couple of terms used in vibration problems are discussed below:

### **1.1 Degrees of Freedom (DOF):**

A simple definition of "degrees of freedom" is - the number of coordinates that it takes to uniquely specify the position of a system. It is an independent displacement or rotation that a system may exhibit. A degree-of-freedom for a system is analogous to an independent variable for a mathematical function. All system degrees-of-freedom must be specified to fully characterize the system at any given time.

Degrees of freedom (DOF) are the set of independent displacements and/or rotations that specify completely the displaced or deformed position and orientation of the body or system. This is a fundamental concept relating to systems of moving bodies in mechanical engineering, aeronautical engineering, robotics, structural engineering, etc.

The free particle undergoing general motion in space will have three degrees of freedom, and a rigid body will have six degrees of freedom, i.e., three components of position and three angles defining its orientation. Furthermore, a continuous elastic body will require an infinite number of coordinates (three for each point on the body) to describe its motion; hence its degrees of freedom must be infinite. However, in many cases, parts of such bodies may be assumed to be rigid, and the system may be considered to be dynamically equivalent to one having finite degrees of freedom. In fact, a surprisingly large number of vibration

problems can be treated with sufficient accuracy by reducing the system to one having a few degrees of freedom.

## **1.2 Concept of Stability:**

The concept of stability of equilibrium is a strongly intuitive one. If at any level of external cause (in the form of displacements, velocity, force etc.), a structure can sustain a small disturbance from its equilibrium condition, then the structure is said to be in stable equilibrium at that level of external cause. It should be noted that sustaining the disturbance means the structure would oscillate with small amplitude about its equilibrium position. On the other hand, if the structure does not go back to its original position or vibrate with ever increasing amplitude due to the disturbance, then the structure is said to be in an unstable equilibrium state at that level of external cause. The energy method, based on the Lagrange-Dirichlet theorem can be used to analyze the stability of structures. It states that, the equilibrium states are defined by the states of minimum potential energy of the structure and the stability of the equilibrium states is determined by the relative minimum value of its potential energy. A close assessment of the critical load for simple mechanical stability models reveals that the system maintains its state of equilibrium states as long as the work done due to internal resisting forces is greater than that due to the external load for any disturbance from the equilibrium position. In other words, it is the balance between the potential energy due to the internal resisting forces, called internal strain energy or simply strain energy, and the potential energy due to the external force.

## **1.3 Undamped Vibration Absorber:**

Vibration absorber system is a spring-body system, which is added to the structure; the parameters of the absorber are chosen so that the amplitude of the vibration of the structure is greatly reduced, or even eliminated, at a frequency that is usually chosen to be at the original troublesome resonance.

If a single degree of freedom system or mode of a multi-degree of freedom system is excited into resonance, large amplitudes of vibration result with accompanying high dynamic stresses and noise and fatigue problems. In most mechanical systems, this is not acceptable.

If neither the excitation frequency nor the natural frequency can conveniently be altered, this resonance condition can often be successfully controlled by adding a further single degree of freedom system, which is known as the absorber [Breads (1996)].

Accurate tuning of the frequency of the absorber results in induced inertia forces of the absorber mass that counteract the forces applied to the primary system and less work is done on this system. Hence, the normal practical function of the absorber is to reduce resonant oscillations of the primary systems (even though in theory it could be used as inertia balancer at any frequency, provided it is tuned with respect to forcing frequency). While the vibration amplitudes of the primary system can thus be suppressed to a large extent, large displacement amplitudes must be accepted in absorber system [Bachmann (1995)].

#### **1.4 Shock Absorber (Untuned vibration absorber):**

A shock absorber (or damper in technical use) is a mechanical device designed to smooth out or damp shock impulse, and dissipate kinetic energy. Shock absorbers must absorb or dissipate energy. One design consideration, when designing or choosing a shock absorber is where that energy will go. In most dashpots, energy is converted to heat inside the viscous fluid. In hydraulic cylinders, the hydraulic fluid will heat up, while in air cylinders, the hot air is usually exhausted to the atmosphere. In other types of dashpots, such as electromagnetic ones, the dissipated energy can be stored and used later. In smaller terms shock absorbers help to cushion cars on uneven roads.

Shock absorbers are important parts of automobile and motorcycle suspensions, aircraft landing gear, and the supports for many industrial machines. Large shock absorbers have also been used in structural engineering to reduce the susceptibility of structures to earthquake damage and resonance. A transverse mounted shock absorber, called a yaw damper, helps keep railcars from swaying excessively from side to side and are important in passenger railroads, commuter rail and rapid transit systems because they prevent railcars from damaging station platforms.

In a vehicle, it reduces the effect of traveling over rough ground, leading to improved ride quality. Without shock absorbers, the vehicle would have a bouncing ride, as energy is stored in the spring and then released to the vehicle, possibly exceeding the allowed range of suspension movement.

Spring-based shock absorbers commonly use coil springs or leaf springs, though torsion bars can be used in torsional shocks as well. Ideal springs alone, however, are not shock absorbers as springs only store and do not dissipate or absorb energy.

Optimum damping ratio is the ratio of damping constant to critical damping, for which system vibrates with minimum peak. Optimum damping ratio is largely dependent on the mass ratio of the shock absorber. Optimum damping ratio increases with the increase of mass ratio of the system. Relation for optimum damping is obtained by differentiating the nondimensional displacement of the system with damping ratio & equating the final equation to zero.

### 1.5 Nonlinear Springs and Dampers:

In general nonlinear vibrations are not harmonic, and their frequencies vary with amplitude. For example, if the magnitude of the forcing frequency is doubled the response of a nonlinear system is not necessarily doubled. Superposition principle cannot be applied to solve such type of problems. One important type of nonlinearity arises when the restoring force of a spring is not proportional to its deformation. A spring is nonlinear if the force exerted by the spring is a nonlinear function of the displacement. For nonlinear spring,

$$\text{Spring force} = kx \pm k'x^3$$

Where,  $k$  and  $k'$  are spring constants,  $x$  is the displacement of the spring,  $F(x)$  is the force from spring. If positive (+) sign is used the spring is called hard. If negative (-) sign is used the spring is called soft.

The static load-displacement curve for hard spring shows the slope increases as the load increases. Similarly the load – displacement curve for a soft spring shows that the slope decreases as the load increases. Similarly for nonlinear dampers,

$$\text{Damper force} = c\dot{x} \pm c'\dot{x}^2$$

Where,  $c$  and  $c'$  are damping constants,  $\dot{x}$  is the velocity of the moving body. Hard and soft dampers follow similar relations like hard and soft springs respectively [Timoshenko (1974)].

## 1.6 Objectives of this Investigation:

Most numerical studies regarding nonlinear vibration of structures, particularly involving multi degrees of freedom system (MDOFS), have been carried out in the form of initial value problems: all the boundary conditions, termed as system's responses (displacement, velocity etc.), were specified at an initial time reference, followed by numerical integration of the governing differential equation. Such type of analysis involves simultaneous solution of a system of nonlinear equations where the number of equations to be solved is determined by order of the governing equations. For example, for a 2<sup>nd</sup> order equation, each time a 2x 2 system of equation needs to be solved and the mostly used method is Newton-Raphson. Obviously, the problem becomes much more involved if all the boundary conditions are not specified at the same initial time reference, that is, some conditions are also specified at the final time reference. This type of problem needs to simultaneously solve a large number of nonlinear equations that depends on the number of intermediate grid points in between the two time references. Though, Newton-Raphson method can be used to solve that large number of equations, there are chances of non-convergence of solutions.

Present work aims to solve both boundary and initial value problems for any vibratory system. In most of the previous studies of absorbers [Mikhlin and Reshetnikova (2005), Natsiavas (1992 & 1993), Wang (1985) & Thomson (1981)], stability of the system was studied by the method of perturbation. But a simple and direct method, like that of multisegment integration technique [Kalnins and Lestingi (1967)], that helps to directly visualize the system's response with time, would be very useful, in particular for the present study, when a boundary value problem is dealt with. Therefore, objectives of this study can be described as below:

- a. At first to develop a generalized computer code for nonlinear vibration analysis of a multiple degrees of freedom system, and later to use this code for analyzing the 2DOFS having nonlinear springs with and without nonlinear damping specially, tuned and untuned vibration absorbers. Computer coding will be done using Turbo C.
- b. To study the nonlinear dynamic behavior varying the nonlinearity of the springs that is, separately considering hard and soft springs. Similarly, damping coefficient can also be varied to incorporate different types of linear or nonlinear damping.

- c. To study the stability of the tuned and untuned absorber system.
- d. To study the effect of changing boundary conditions on the system's response.
- e. To find response of the system for different mass ratio and spring constant ratio for the 2DOFS.
- f. Soundness of the code can be checked by comparing the available standard result, for the case of linear absorber, given in Thomson (1981).

The effect of different types of nonlinearities on the stability of the system can be also studied for the tuned and untuned vibration absorbers that are both 2DOFS. In future, the developed computer code can be used to study response of the system having higher DOF.

### **1.7 Outline of Methodology:**

Multisegment method of integration developed by Kalnins and Lestingi (1967) will be used to solve the coupled nonlinear differential equations as boundary value problems. Specialty of this method is that the given interval of the independent variable is divided into finite number of segments. Next initial value integrations are performed over each segment followed by a solution of a system of matrix equations to ensure continuity of the dependent variables at all the nodal points. Steps are repeated until continuity of the dependent variables at the nodal points is achieved. More detail of this method is given in chapter 4.



# Chapter 2

## Literature Review

In the domain of mechanical vibration research, dynamic absorbers have extensive applications in reducing vibrations of machinery Thomson (1981) & Zhu et al (2004). Zhu et al. (2004) extensively studied nonlinear response of two degrees of freedom (2DOF) vibration system with nonlinear damping and nonlinear springs. Recently, Mikhlin and Reshetnikova (2005) studied the nonlinear 2DOF system having a linear oscillator with a relatively big mass which is an approximation of some continuous elastic system, and an essentially nonlinear oscillator with a relatively small mass which is an absorber of the main linear system vibrations. They analyzed the free and forced vibrations of the system. Vakakis and Paipetis (1986) investigated the effect of a viscously damped dynamic absorber on an undamped multiple degrees of freedom system (MDOFS). Soom (1983) and Jordanov (1988) studied the optimal parameter design of linear and non-linear dynamic vibration absorbers for damped primary systems. The presence of the non-linearities introduces dangerous instabilities, which in some cases may result in amplification rather than reduction of the vibration amplitudes (Rice 1986 and Shaw et al. 1989). Natsiavas (1992) applied the method of averaging to investigate the steady state oscillations and stability of non-linear dynamic vibration absorbers. Oueini et al. (1998 and 1999) exploited the saturation phenomenon in devising an active vibration suppression technique.

Practically vibration problems become nonlinear in nature as amplitude of oscillation becomes large [Thomson (1981)]. Nonlinear problems, usually having no closed form solutions, are always challenges for practicing engineers. Two widely used techniques are perturbation and iteration method [Thomson (1981) and Zhu et al. (2004)] for the cases of nonlinear vibrations. Superposition principle cannot be applied and therefore different mathematical techniques are being tried to solve such problems. Any such numerical technique which makes the computation faster and yield reliable results under any practically possible boundary conditions (in terms of displacement, velocity etc., of the vibrating bodies), especially for MDOFS, would be much desirable.

The energy transfer to nonlinear normal mode is caused by sub-harmonic resonance, which is possible because of the nonlinear oscillator existence. In papers by Manevitch et al. (2002) and McFarland et al. (2003), theoretical investigation and some experimental verification on the use of nonlinear localization for reducing the transmitted vibrations in structures subjected to transient base motions have been presented. In particular, the experimental assembly, containing the main linear subsystem and the nonlinear absorber, is described by Vakakis et al. (2002).

The simplest type of mount is considered as a single degree of freedom linear spring-mass dashpot. Nakhaie et al. (2003) used the Root Mean Square of absolute acceleration and relative displacement to find the optimal damping ratio and natural frequency of the isolator. They have also obtained the optimal value for damping ratio in order to minimize the absolute acceleration for a step input.

Roberson (1952) and Arnold (1955) considered vibration absorbers with nonlinear cubic spring and found that a softening-type nonlinear spring improves the performance of the absorber. Hunt and Nissen (1982) were the first to implement a practical nonlinear damped absorber, and constructed a softening-type spring device as an assembly of steel Belleville washers. However, nonlinear absorbers may exhibit undesirable secondary resonance.

Shekhar et al. (1998) considered a single stage shock isolator comprising a parallel combination of a cubic nonlinearity in spring and damper. Combining straightforward perturbation method and Laplace transform they have determined the transient response of the system. Three types of input base excitations were considered: the rounded step, the rounded pulse and the oscillatory step. Although the solution is valid for limited range of nonlinearities, it was shown that the nonlinearity in the damping rather than in the stiffness has a more pronounced effect on performance of a shock isolator. The presence of a nonlinear velocity dependent damping term with a positive coefficient was found to have detrimental effects on the isolator performance. Shekhar et al. (1999) have considered different alternatives to improve the performance of an isolator having a nonlinear cubic damping over and above the usual viscous damping. Since they concluded, nonlinearity in the isolator stiffness does not have any appreciable effect on its performance, the stiffness of the isolator is assumed to be constant.

The classical problem of damped vibration absorber that consists of a mass, spring and a viscous damper attached to an undamped single degree of freedom (DOF) system of which the mass is subject to harmonic forcing, has a well-known solution, Den Hartog (1956). If damping is added to the absorber, the vibration amplitude of the main mass cannot be made zero at the forcing frequency but the sensitivity of the system to variations in the forcing frequency decreases. Also the vibration amplitude of the absorber mass decreases considerably with a damped absorber.

Vibration absorbers for multi degree of freedom systems have also been proposed. Wang et al. (1985), presented an approach to suppress vibrations in undamped multi DOF systems. Design of absorbers for beams is discussed in Juang (1984) and Jacquot (1978). Absorber design for general vibrating systems was considered in Wang et al. (1983).

Alexander et al. (2009) explored the performance of a nonlinear tuned mass damper (NTMD), which is modeled as a two degree of freedom system with a cubic nonlinearity. This nonlinearity is physically derived from a geometric configuration of two pairs of springs. The springs in one pair rotate as they extend, which results in a hardening spring stiffness. The other pair provides a linear stiffness term. They discovered a family of detached resonance curves for vanishing linear spring stiffness, a feature that was missed in an earlier study. These detached resonance response curves seem to be a weakness of the nonlinear tuned mass damper (NTMD) when used as a passive device, because they essentially restore a main resonance peak.

Present work aims to solve both boundary and initial value problems for any vibratory system having MDOF. A simple and direct method, like that of multisegment integration technique, that helps to directly visualize the system's response with time, would be very useful, in particular for the present study, when a boundary value problem is dealt with.

Present investigation is to study the nonlinear dynamic behavior of vibration system absorber system having nonlinear spring with and without nonlinear damping but extensively varying the spring type (hard and soft) for a wide range of forcing frequency. The solutions must be obtained solving highly nonlinear equations that are coupled. Both steady and unsteady state solutions are also incorporated. Above all, the problem can be solved when the two boundary conditions are specified at two different times.

# Chapter 3

## Governing Equations

Governing equations of the system are derived from the free-body diagram given in Figure 1, considering nonlinear spring, nonlinear damper and external forces applied on both the main and absorber masses. After rearranging and necessary transformation 2<sup>nd</sup> order differential equations are converted to first order differential equations. After necessary partial differentiation with respect to initial conditions of the transformed variables, field equations for Multisegment Integration Technique are formed.

### 3.1 Mathematical Models and Governing Equations

Description of the proposed model for the 2DOF absorber system is given below. Fig. 1 shows the 2 DOFS while Tables 1 – 2 show the different cases absorbers and other related parameters. Fig.1 shows the arrangement of masses, springs and dampers in the vibration system, and following Fig. 1,

$$\text{Spring force for the 1<sup>st</sup> spring} = k_1 x_1 + k'_1 x_1^3 \quad \dots\dots (1)$$

$$\text{Spring force for the 2<sup>nd</sup> spring} = k_2 (x_1 - x_2) + k'_2 (x_1 - x_2)^3 \quad \dots\dots\dots (2)$$

$$\text{Damping force for the 1<sup>st</sup> damper} = c_1 \dot{x}_1 + c'_1 \dot{x}_1 x_1^2 \quad \dots\dots\dots (3)$$

$$\text{Damping force for the 2<sup>nd</sup> damper} = c_2 (\dot{x}_1 - \dot{x}_2) + c'_2 (\dot{x}_1 - \dot{x}_2)(x_1 - x_2)^2 \quad \dots\dots\dots (4)$$

The equations of motion are as follows for the main mass and the absorber mass, respectively,

$$m_1 \ddot{x}_1 + (k_1 x_1 + k'_1 x_1^3) + (c_1 \dot{x}_1 + c'_1 \dot{x}_1 x_1^2) + \{k_2 (x_1 - x_2) + k'_2 (x_1 - x_2)^3\} + \{c_2 (\dot{x}_1 - \dot{x}_2) + c'_2 (\dot{x}_1 - \dot{x}_2)(x_1 - x_2)^2\} = F$$

..... (5)

$$m_2 \ddot{x}_2 - \{k_2(x_1 - x_2) + k'_2(x_1 - x_2)^3\} - \{c_2(\dot{x}_1 - \dot{x}_2) + c'_2(\dot{x}_1 - \dot{x}_2)(x_1 - x_2)^2\} = 0$$

..... (6)

For transformations, let  $x_1 = y_1$  and  $x_2 = y_3$

$$\frac{dx_1}{dt} = \dot{x}_1 = y_2 \quad \frac{dx_2}{dt} = \dot{x}_2 = y_4$$

With those transformations, Equations 5 and 6 become,

$$m_1 \frac{dy_2}{dt} + (k_1 y_1 + k'_1 y_1^3) + (c_1 y_2 + c'_1 y_2 y_1^2) + \{k_2(y_1 - y_3) + k'_2(y_1 - y_3)^3\} + \{c_2(y_2 - y_4) + c'_2(y_2 - y_4)(y_1 - y_3)^2\} = F$$

..... (7)

$$m_2 \frac{dy_4}{dt} - \{k_2(y_1 - y_3) + k'_2(y_1 - y_3)^3\} - \{c_2(y_2 - y_4) + c'_2(y_2 - y_4)(y_1 - y_3)^2\} = 0$$

..... (8)

Rearrangement of Equation (7) & (8) gives,

$$\frac{dy_2}{dt} = \frac{1}{m_1} \left[ F - (k_1 y_1 + k'_1 y_1^3) - (c_1 y_2 + c'_1 y_2 y_1^2) - \{k_2(y_1 - y_3) + k'_2(y_1 - y_3)^3\} \right] - \frac{1}{m_2} \left[ -\{c_2(y_2 - y_4) + c'_2(y_2 - y_4)(y_1 - y_3)^2\} \right]$$

..... (9)

$$\frac{dy_4}{dt} = \frac{1}{m_2} \left[ \left\{ k_2(y_1 - y_3) + k'_2(y_1 - y_3)^3 \right\} + \left\{ c_2(y_2 - y_4) + c'_2(y_2 - y_4)(y_1 - y_3)^2 \right\} \right]$$

..... (10)

The governing Equations (9) and (10) can now be rewritten as a set of four nonlinear first order ordinary differential equation (ODE) as follows:

$$\frac{dy_1}{dt} = y_2 \text{ ..... (11)}$$

$$\frac{dy_2}{dt} = \frac{1}{m_1} \left[ F - (k_1 y_1 + k'_1 y_1^3) - (c_1 y_2 + c'_1 y_2 y_1^2) - \left\{ k_2(y_1 - y_3) + k'_2(y_1 - y_3)^3 \right\} \right] - \left\{ c_2(y_2 - y_4) + c'_2(y_2 - y_4)(y_1 - y_3)^2 \right\}$$

.....: (12)

$$\frac{dy_3}{dt} = y_4 \text{ ..... (13)}$$

$$\frac{dy_4}{dt} = \frac{1}{m_2} \left[ \left\{ k_2(y_1 - y_3) + k'_2(y_1 - y_3)^3 \right\} + \left\{ c_2(y_2 - y_4) + c'_2(y_2 - y_4)(y_1 - y_3)^2 \right\} \right]$$

..... (14)

The additional fields Equations, needed for multisegment method of integration (chapter 4), are derived now from Equations 11-14. This is done by differentiating both sides of Equations 11-14, partially w.r.t.  $y(a)$ . For example at first,

$\frac{\partial}{\partial y_1(a)}$  of Equation (11) gives

$$\frac{\partial}{\partial y_1(a)} \left\{ \frac{dy_1}{dt} \right\} = \frac{\partial}{\partial y_1(a)} \{y_2\}$$

or,

$$\frac{d}{dt} \left\{ \frac{\partial y_1(t)}{\partial y_1(a)} \right\} = \frac{\partial y_2(t)}{\partial y_1(a)}$$

And, finally, using the symbol  $Y$  for the partial derivative term of  $y(t)$  w.r.t.  $y(a)$ , we get the additional field equations that are actually the governing differential Equations of  $Y$ , as follows:

$$\frac{d}{dt}(Y_{11}) = Y_{21} \dots\dots\dots (15)$$

Now  $\frac{\partial}{\partial y_1(a)}$  of Equation (12)

$$\frac{\partial}{\partial y_1(a)} \left\{ \frac{dy_2}{dt} \right\} = \frac{\partial}{\partial y_1(a)} \left\{ \frac{1}{m_1} \left[ \begin{array}{l} F - (k_1 y_1 + k'_1 y_1^3) - (c_1 y_2 + c'_1 y_2 y_1^2) \\ - \{k_2 (y_1 - y_3) + k'_2 (y_1 - y_3)^3\} \\ - \{c_2 (y_2 - y_4) + c'_2 (y_2 - y_4)(y_1 - y_3)^2\} \end{array} \right] \right\}$$

or,

$$\frac{d(Y_{21})}{dt} = -\frac{1}{m_1} \left[ \begin{array}{l} (k_1 Y_{11} + 3k'_1 y_1^2 Y_{11}) + (c_1 Y_{21} + c'_1 y_1^2 Y_{21} + 2c'_1 y_1 y_2 Y_{11}) \\ + \{k_2 (Y_{11} - Y_{31}) + 3k'_2 (y_1 - y_3)^2 (Y_{11} - Y_{31})\} \\ + \{c_2 (Y_{21} - Y_{41}) + c'_2 (y_1 - y_3)^2 (Y_{21} - Y_{41}) + 2c'_2 (y_1 - y_3)(y_2 - y_4)(Y_{11} - Y_{31})\} \end{array} \right]$$

..... (16)

Next,  $\frac{\partial}{\partial y_1(a)}$  of Equation (13)

$$\frac{\partial}{\partial y_1(a)} \left\{ \frac{dy_3}{dt} \right\} = \frac{\partial}{\partial y_1(a)} \{y_4\}$$

or,

$$\frac{d}{dt} \left[ \frac{\partial y_3(t)}{\partial y_1(a)} \right] = \frac{\partial y_4(t)}{\partial y_1(a)}$$

or,

$$\frac{d}{dt}(Y_{31}) = Y_{41} \dots\dots\dots (17)$$

Finally,  $\frac{\partial}{\partial y_1(a)}$  of Equation (14)

$$\frac{\partial}{\partial y_1(a)} \left\{ \frac{dy_4}{dt} \right\} = \frac{\partial}{\partial y_1(a)} \left\{ \frac{1}{m_2} \left[ \left\{ k_2(y_1 - y_3) + k'_2(y_1 - y_3)^3 \right\} + \left\{ c_2(y_2 - y_4) + c'_2(y_2 - y_4)(y_1 - y_3)^2 \right\} \right] \right\}$$

or,

$$\frac{d(Y_{41})}{dt} = \frac{1}{m_2} \left[ \left\{ k_2(Y_{11} - Y_{31}) + 3k'_2(y_1 - y_3)^2(Y_{11} - Y_{31}) \right\} + \left\{ c_2(Y_{21} - Y_{41}) + c'_2(y_1 - y_3)^2(Y_{21} - Y_{41}) + 2c'_2(y_1 - y_3)(y_2 - y_4)(Y_{11} - Y_{31}) \right\} \right]$$

..... (18)



Similarly  $\frac{\partial}{\partial y_2(a)}$  of Equations (11), (12), (13), (14) will give

$$\frac{d}{dt}(Y_{12}) = Y_{22} \dots\dots\dots (19)$$

$$\frac{d(Y_{22})}{dt} = -\frac{1}{m_1} \left[ \begin{aligned} &(k_1 Y_{12} + 3k'_1 y_1^2 Y_{12}) + (c_1 Y_{22} + c'_1 y_1^2 Y_{22} + 2c'_1 y_1 y_2 Y_{12}) \\ &+ \{k_2 (Y_{12} - Y_{32}) + 3k'_2 (y_1 - y_3)^2 (Y_{12} - Y_{32})\} \\ &+ \{c_2 (Y_{22} - Y_{42}) + c'_2 (y_1 - y_3)^2 (Y_{22} - Y_{42}) + 2c'_2 (y_1 - y_3)(y_2 - y_4)(Y_{12} - Y_{32})\} \end{aligned} \right] \dots\dots\dots (20)$$

$$\frac{d}{dt}(Y_{32}) = Y_{42} \dots\dots\dots (21)$$

$$\frac{d(Y_{42})}{dt} = \frac{1}{m_2} \left[ \begin{aligned} &\{k_2 (Y_{12} - Y_{32}) + 3k'_2 (y_1 - y_3)^2 (Y_{12} - Y_{32})\} \\ &+ \{c_2 (Y_{22} - Y_{42}) + c'_2 (y_1 - y_3)^2 (Y_{22} - Y_{42}) + 2c'_2 (y_1 - y_3)(y_2 - y_4)(Y_{12} - Y_{32})\} \end{aligned} \right] \dots\dots\dots (22)$$

It can be seen that every column of the governing equations of  $Y$  has similar form.

Thus,  $\frac{\partial}{\partial y_3(a)}$  of Equations (11), (12), (13), (14) will give

$$\frac{d}{dt}(Y_{13}) = Y_{23} \dots\dots\dots (23)$$

$$\frac{d(Y_{23})}{dt} = -\frac{1}{m_1} \left[ \begin{aligned} &(k_1 Y_{13} + 3k'_1 y_1^2 Y_{13}) + (c_1 Y_{23} + c'_1 y_1^2 Y_{23} + 2c'_1 y_1 y_2 Y_{13}) \\ &+ \{k_2 (Y_{13} - Y_{33}) + 3k'_2 (y_1 - y_3)^2 (Y_{13} - Y_{33})\} \\ &+ \{c_2 (Y_{23} - Y_{43}) + c'_2 (y_1 - y_3)^2 (Y_{23} - Y_{43}) + 2c'_2 (y_1 - y_3)(y_2 - y_4)(Y_{13} - Y_{33})\} \end{aligned} \right]$$

..... (24)

$$\frac{d}{dt}(Y_{33}) = Y_{43} \dots\dots\dots (25)$$

$$\frac{d(Y_{43})}{dt} = \frac{1}{m_2} \left[ \left\{ k_2(Y_{13} - Y_{33}) + 3k'_2(y_1 - y_3)^2(Y_{13} - Y_{33}) \right\} + \left\{ c_2(Y_{23} - Y_{43}) + c'_2(y_1 - y_3)^2(Y_{23} - Y_{43}) + 2c'_2(y_1 - y_3)(y_2 - y_4)(Y_{13} - Y_{33}) \right\} \right]$$

..... (26)

Finally,  $\frac{\partial}{\partial y_4(a)}$  of Equations (11), (12), (13), (14) will give

$$\frac{d}{dt}(Y_{14}) = Y_{24} \dots\dots\dots (27)$$

$$\frac{d(Y_{24})}{dt} = -\frac{1}{m_1} \left[ \left( k_1 Y_{14} + 3k'_1 y_1^2 Y_{14} \right) + \left( c_1 Y_{24} + c'_1 y_1^2 Y_{24} + 2c'_1 y_1 y_2 Y_{14} \right) + \left\{ k_2(Y_{14} - Y_{34}) + 3k'_2(y_1 - y_3)^2(Y_{14} - Y_{34}) \right\} + \left\{ c_2(Y_{24} - Y_{44}) + c'_2(y_1 - y_3)^2(Y_{24} - Y_{44}) + 2c'_2(y_1 - y_3)(y_2 - y_4)(Y_{14} - Y_{34}) \right\} \right]$$

..... (28)

$$\frac{d}{dt}(Y_{34}) = Y_{44} \dots\dots\dots (29)$$

$$\frac{d(Y_{44})}{dt} = \frac{1}{m_2} \left[ \left\{ k_2(Y_{14} - Y_{34}) + 3k'_2(y_1 - y_3)^2(Y_{14} - Y_{34}) \right\} + \left\{ c_2(Y_{24} - Y_{44}) + c'_2(y_1 - y_3)^2(Y_{24} - Y_{44}) + 2c'_2(y_1 - y_3)(y_2 - y_4)(Y_{14} - Y_{34}) \right\} \right]$$

..... (30)

These governing Equations (11 – 30) can be solved when the boundary conditions are specified. Equations of boundary conditions are described in the next section.

### 3.2 Boundary Conditions:

According to the multisegment method of integration, the boundary conditions for this boundary value problem are arranged in the following matrix form,

$$Ay(a) + By(b) = C \dots\dots (31)$$

The boundary condition were chosen arbitrarily for this analysis since the devised program is capable to calculate the values of  $y$  for any specified boundary condition which is the primary objective of this analysis. For demonstrating the method of solution, a set of arbitrarily chosen data, used for the boundary value problem analysis, are as below.

$$y_1(a) = 0.05\text{m}; y_3(a) = -0.07\text{m}; y_2(b) = 0.06\text{m/s}; y_4(b) = -0.06\text{m/s}.$$

The above boundary conditions are used for both tuned and untuned vibration absorbers. Fig. 2 specifies that it leads to a boundary value problem. Therefore, the matrices for tuned and untuned absorbers of Equation 31 must be as follows:

$$[y(a)] = \begin{bmatrix} y_1(a) \\ y_2(a) \\ y_3(a) \\ y_4(a) \end{bmatrix}$$

$$[y(b)] = \begin{bmatrix} y_1(b) \\ y_2(b) \\ y_3(b) \\ y_4(b) \end{bmatrix}$$

$$[A] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$[B] = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$[C] = \begin{bmatrix} 0.05 \\ 0.06 \\ -0.07 \\ -0.06 \end{bmatrix}$$

Only for untuned absorbers, initial value analysis has been performed. The chosen boundary conditions for initial value problem:

$$y_1(a) = 0.0\text{m}; y_2(a) = 0.0\text{m/s}; y_3(a) = 0.0\text{m}; y_4(a) = 0.0\text{m/s}.$$

After deriving the governing Equations (11 – 30) and the boundary condition (Equation 31) multisegment integration technique is used to solve those equations as a boundary value problem.

# Chapter 4

## Method of Solution

As mentioned in chapter 1 vibration problem would become quite involved if all the boundary conditions are not specified at the same initial time reference, that is, some conditions are also specified at the final time reference. This type of problem needs to simultaneously solve a large number of nonlinear equations that depends on the number of intermediate grid points in between the two time references. Though, Newton-Raphson method can be used to solve that large number of equations, there are chances of non-convergence of solutions. Therefore Multi-segment Integration Technique developed by Kalnins and Lestingi (1967) has been used in the present study to solve the equations derived in chapter 3.

### 4.1 Multisegment Integration Technique

At first the  $m^{\text{th}}$  order ordinary differential equation (ODE) is reduced to ' $m$ ' first order ODE. Then the scheme of multisegment method of integration of a system of  $m$  first order ordinary differential equations is as follows:

$$\frac{dy(x)}{dx} = F(x, y^1(x), y^2(x), \dots, y^m(x)) \quad \dots \dots \dots (32)$$

in the interval  $(x_1 < x < x_{M+1})$  consists of

- a. the division of the given interval into  $M$  segments;
- b.  $(m+1)$  initial value integrations over each segment;
- c. solution of a system of matrix equations to ensure continuity of the dependent variables at the nodal points;
- d. repetition of (b) and (c) until continuity of the independent variables at the nodal points is achieved.

In Eq. (32) the symbol  $y(x)$  denotes column matrix whose elements are  $m$  dependent variables, denoted by  $y_j(x)(j=1,2,3 \dots, m)$ ;  $F$  represents  $m$  functions arranged in a column matrix form; and  $x$  is the independent variable. Here for convenience the first  $m/2$  elements of  $y(x_1)$  and the last  $m/2$  elements of  $y(x_{M+1})$  are prescribed by the boundary conditions.

If at the initial point  $x_i$  of the segment  $S_i$  a set of values  $y(x_i)$  is prescribed for the variables of Eqs. (32) then the variables at any  $x$  within  $S_i$  can be expressed as

$$y(x) = T[y^1(x_i), y^2(x_i), \dots, y^m(x_i)] \dots \dots \dots (33)$$

where the function  $T$  is uniquely dependent on  $x$  and the system of equations. From Eqs. (33) the expressions for the small changes  $\delta y(x)$  can be expressed, to a first approximation, by the following linear equations:

$$\delta y(x) = Y_i(x) \delta y(x_i) \dots \dots \dots (34)$$

where,

$$Y_i(x) = \begin{bmatrix} \frac{\partial y^1(x)}{\partial y^1(x_i)} & \frac{\partial y^1(x)}{\partial y^2(x_i)} & \dots & \frac{\partial y^1(x)}{\partial y^m(x_i)} \\ \frac{\partial y^2(x)}{\partial y^1(x_i)} & \frac{\partial y^2(x)}{\partial y^2(x_i)} & \dots & \frac{\partial y^2(x)}{\partial y^m(x_i)} \\ \dots & \dots & \dots & \dots \\ \frac{\partial y^m(x)}{\partial y^1(x_i)} & \frac{\partial y^m(x)}{\partial y^2(x_i)} & \dots & \frac{\partial y^m(x)}{\partial y^m(x_i)} \end{bmatrix} \dots \dots \dots (35)$$

Expressing Eqs. (34) in finite difference form and evaluating them at  $x = x_{i+1}$ ,

$$y'(x_{i+1}) - y(x_{i+1}) = Y_i(x_{i+1})[y'(x_i) - y(x_i)] \quad \dots\dots\dots (36)$$

where  $y'$  denotes a trial solution state and  $y$  denotes an iterated solution state based on the condition of continuity of the variables at the nodal points. Eq. (36) is rearranged as

$$Y_i(x_{i+1})y(x_i) - y(x_{i+1}) = -Z_i(x_{i+1}) \quad \dots\dots\dots (37)$$

where,  $Z_i(x_{i+1}) = y'(x_{i+1}) - Y_i(x_{i+1})y'(x_i)$

In order to determine the coefficients  $Y_i(x)$  in Eqs. (37) in the  $j$ th column of  $Y_i(x)$  can be regarded as a set of new variables, which is solution of an initial value problem governed within each segment by a linear system of first order differential equations, obtained from Eqs. (32) by differentiation with respect to  $y^j(x_i)$  in the form

$$\frac{d}{dx} \left[ \frac{\partial y(x)}{\partial y^j(x_i)} \right] = \frac{\partial}{\partial y^j(x_i)} \left\{ F[x, y^1(x), y^2(x), \dots, y^m(x)] \right\} \quad \dots\dots\dots (38)$$

thus the columns of the matrix  $Y_i(x)$  are defined as the solutions of  $m$  initial value problems governed in  $S_i$  by (38), with  $j=1, 2, \dots, m$  where the initial values, in view of Eqs. (34), are given by

$$Y_i(x_i) = I \quad \dots\dots\dots (39)$$

where  $I$  denotes the  $(m,m)$  unit matrix. to obtain the iterated solution  $y(x_i)$  Eqs. (37) are rewritten as a partitioned matrix product of the form

$$\begin{bmatrix} y_1(x_{i+1}) \\ \dots \\ y_2(x_{i+1}) \end{bmatrix} = \begin{bmatrix} Y_i^1(x_{i+1}) & \vdots & Y_i^2(x_{i+1}) \\ \dots & \dots & \dots \\ Y_i^3(x_{i+1}) & \vdots & Y_i^4(x_{i+1}) \end{bmatrix} \begin{bmatrix} y_1(x_i) \\ \dots \\ y_2(x_i) \end{bmatrix} + \begin{bmatrix} Z_i^1(x_{i+1}) \\ \dots \\ Z_i^2(x_{i+1}) \end{bmatrix}$$

so that the known boundary conditions are separated from the unknowns and, therefore, turns into a pair of equations given by

$$\begin{aligned} Y_i^1(x_{i+1})y_1(x_i) + Y_i^2(x_{i+1})y_2(x_i) - y_1(x_{i+1}) &= -Z_i^1(x_{i+1}) \\ Y_i^3(x_{i+1})y_1(x_i) + Y_i^4(x_{i+1})y_2(x_i) - y_1(x_{i+1}) &= -Z_i^2(x_{i+1}) \end{aligned} \quad \dots \dots \dots (40)$$

The result is a simultaneous system of  $2M$  linear matrix equations, in which the known coefficients  $Y_i^j(x_{i+1})$  and  $Z_i^j(x_{i+1})$  are  $(m/2, m/2)$  and  $(m/2, 1)$  matrices, respectively, and the unknown,  $y_j(x_i)$  are  $(m/2, 1)$  matrices. Since  $y_1(x_1)$  and  $y_2(x_{M+1})$  are known, there are exactly  $2m$  unknowns:  $y_1(x_i)$ , with  $i=2,3,\dots,M+1$ , and  $y_2(x_i)$ , with  $i=1,2,\dots,M$ .

By means of Gaussian elimination, the system of equations (40) is first brought to the form

$$\begin{aligned} E_i y_2(x_i) - y_1(x_{i+1}) &= A_i \\ C_i y_1(x_{i+1}) - y_2(x_{i+1}) &= B_i \end{aligned} \quad \dots \dots \dots (41)$$

for  $i=1,2,\dots,M$ . Using the notations  $Z_i^j$  and  $Y_i^j$  in place of symbols  $Z_i^j(x_{i+1})$  and  $Y_i^j(x_{i+1})$ , the  $(m/2, m/2)$  matrices  $E_i$  and  $C_i$  in the Eqs. (41) are defined by



$$E_1 = Y_1^2, \quad C_1 = Y_1^4 (Y_1^2)^{-1}$$

$$\text{and } E_i = Y_i^2 + Y_i^1 C_{i-1}^{-1} \quad C_i = (Y_i^4 + Y_i^3 C_{i-1}^2) E_i^{-1}$$

for  $i=2,3,\dots,M$ .

The  $(m/2, 1)$  matrices  $A_i$  and  $B_i$  are given by

$$A_1 = -Z_1^1 - Y_1^1 y_1(x_1),$$

$$B_1 = -Z_1^2 - Y_1^2 y_1(x_1) - Y_1^4 E_1^{-1} A_1,$$

$$A_i = -Z_i^1 - Y_i^1 C_{i-1}^{-1} B_{i-1},$$

$$b_i = -Z_i^2 - Y_i^3 C_{i-1}^{-1} B_{i-1} - (Y_i^4 + Y_i^3 C_{i-1}^{-1}) E_i^{-1} A_i,$$

for  $i=2,3,\dots,M$ .

then the unknowns of (40) are obtained by

$$y_1(x_{M+1}) = C_{M-1} [B_M - y_2(x_{M+1})],$$

$$y_2(x_M) = E_{M-1} [y_1(x_{M+1}) + A_M],$$

$$\text{and } y_1(x_{M-i+1}) = C_{M-i}^{-1} [y_2(x_{M-i+1}) + N_{M-i}],$$

$$y_2(x_{M-i}) = E_{M-i}^{-1} [y_1(x_{M-i+1}) + A_{M-i}],$$

for  $i=2,3,\dots,M-1$ .

Assuming  $y(x_i)$  as the next trial solution  $y'(x_i)$  the process is repeated until the integration results of Eqs. (32) at  $x_{i+1}$ , as obtained from the integrations in segment  $S_i$  with the initial values  $y(x_i)$ , match with the elements of  $y(x_{i+1})$  as obtained from (37) and also with the boundary conditions at  $x_{M+1}$ . But it is worth mentioning that number of segment, that is,  $M$  has been kept to one to get the results and these are also reliable as discussed in the next chapter.

# Chapter 5

## Results and Discussion

Results are obtained for tuned absorbers (Cases 1 – 5 in Table 1) and untuned absorbers (Cases 6 – 21 in Table 2). Chosen boundary condition and different parameters of interest and chosen boundary conditions are shown in Tables 3 – 5 and Figs. 1 – 2.

Solutions of the boundary value problem with any arbitrarily chosen boundary conditions are possible by the present method. For example, in Table 4 it could be all zeros in 2<sup>nd</sup> – 4<sup>th</sup> columns. These results are not shown here for brevity. Results of tuned absorbers are obtained only for boundary value problem analysis. But for untuned absorbers both initial and boundary problems are solved. Moreover, two data sets (set # 1 & set # 2), given in Table 3 are used to generate results for untuned absorbers. These two sets of data are used to see the effect nonlinearities present in the system on its response. Again analysis of stability is made for different nonlinearities of the system. Characteristics of the responses of untuned absorbers for data set # 1 and data set # 2 are given in Table 7 – 8. Three different frequency ratios are considered for the stability analysis of untuned absorbers. These frequency ratios are  $r=0.3162$ , 1.0 and 4.744.  $r=0.3162$  represents the response of the system at lower forcing frequency where as  $r=4.744$  represents the system response at higher forcing frequency. At  $r=1.0$  system vibrates with high amplitude. It is found that in some cases system is unstable at  $r=1.0$ . The results are described now sequentially.

### 5.1 Stability Analysis of Tuned Vibration Absorbers

A few results obtained numerically by the described method in chapter 4, are first shown in Table 6 for tuned frequency (frequency ratio,  $r=1$ ). As discussed, Equation 31 defines the boundary conditions while Fig. 2 shows it graphically. Table 6 shows the unknown responses, that is, final displacements for  $y_1$  and  $y_3$  and initial velocities for  $y_2$  and  $y_4$ , for the main mass and absorber mass, respectively, for the different cases 1 to 5. As seen,  $y_1(b)$  and  $y_2(a)$  are -0.0791m and -0.9944m/s, respectively for case 1. Spring nonlinearity has little effect on  $y_1(b)$ ; with reference to case 1,  $y_1(b)$  for four other cases does not change remarkably but  $y_2(a)$  is 1.408% and 1.407% lower for case 2 and case 4 and 1.458% and 1.489% higher, respectively, for case 3 and case 5.

For the absorber mass,  $y_3(b)$  and  $y_4(a)$  are 0.0222m and -1.6667m/s, respectively for case 1. Spring nonlinearity has notable effect on both the values of  $y_3(b)$  and  $y_4(a)$ . For example, with reference to case 1,  $y_3(b)$  is 15.315% and 14.865% lower for case 2 and case 4; but 14.865% and 15.315% higher for case 3 and case 5. Again,  $y_4(a)$  is 3.216% and 3.084% higher for case 2 and case 4; but 3.318% and 3.258% lower for case 3 and case 5, respectively.

Figs. 3(a) and (b), for case 1, show the response of the linear 2 DOFS with the change of forcing frequency. This analysis was made intentionally to compare the exact results for tuned absorbers given in fig. 3(c) [Thompson (1981)]. These results of Thompson (1981) are of course, for steady state vibrations but present study includes all terms. Figs. 3(a) and 3(b), for the main mass and the absorber, respectively, prove the reliability of the code as these give very similar curves given in fig. 3(c) [Thompson (1981)]. As seen in Fig. 3(a-b), deflection tends to a small value (zero for steady state vibration as in Thompson (1981)) at the tuned frequency ( $r=1$ ), while the system response becomes boundless at the two natural frequencies corresponding to  $r_1 = 0.8$  and  $r_2=1.25$ .

Having analyzed  $d - r$  curves for a linear absorber ( Figs. 3(a), (b)), it would be now easy to comprehend the effect of spring nonlinearity on the system's response from Figs. 4(a) and (b) that represent the  $d-r$  curves for case 2. From Figs. 3 and 4, a general observation that can be made is that the non-dimensional deflection of absorber mass at  $r = 0.8$  and 1.25 is always greater than that of main mass.  $d - r$  and  $e - r$  curves of the 2 DOFS for other nonlinear Cases 3-5 (that are not shown here for the sake of brevity) are similar to that of Figs. 4.

Fig. 5 allows the comparison of different cases of spring combination from  $d$  versus  $r$  curves at  $t=50$ s. As seen  $d$  is small in the regions  $0 < r < 0.65$ , in the vicinity of the tuned frequency  $r=1$  and for high frequency for  $r > 1.4$ . But  $d$  is too high at the two resonant frequencies. For all the cases 1 – 5, the first resonant frequency  $r_1=0.8$  seems to be more dangerous than the second one ( $r_2=1.25$ ), in terms of non-dimensional displacements. It is also interesting to see from this figure that absolute non-dimensional displacements at  $r = 0.8$  is the highest for case 5. Case 1 and case 2 show the second and third highest displacements, respectively at  $r_1=0.8$ . While,  $d= -200.0152$  and  $-172.433$  for Cases 3 and 4, respectively at

$r_1=0.8$ . At  $r=1.25$ , case 4 shows the highest deflection. Case 2 and case 5 show second and third highest displacements respectively at  $r_2=1.25$ .

Non-dimensional deflections with time at different frequencies are shown in Figs. 6-8, 10-16. To avoid clumsiness, the entire data is sorted; one at the interval of fifty is shown only out of the entire 2500 data. Therefore, some of these Figures 6-8, 10-16 may have a higher peak in the interval  $t=0$ -50 seconds.

For case 1, Fig. 6 shows non-dimensional deflection of main mass with time at the tuned frequency. As seen,  $d$  is too small at this frequency which will be evident while discussing  $d-t$  curves for other frequencies. Since  $d$  changes almost negligibly (range of  $d$  varies from -4.67 to 4.55) during the entire period of analysis, it represents stable solutions. Similar  $d-t$  curves are presented for different cases 2-5 in Fig. 7, at the tuned frequency.  $d$  varies from 4.73 to 4.63 during the entire period of analysis. Therefore, like case 1, all other cases with spring nonlinearity, have similar stable responses at the tuned frequency as shown by Fig. 7.

Unlike Figs. 6, 7 there is clear increase of amplitude with time when  $r_1=0.8$ , corresponds to the first natural frequency for case 1 as shown in Fig. 8.  $d$  increases gradually from its initial peak value of 60.08 to -208.4 for the time period. Such boundless increase of  $d$  with time indicates that the system is unstable at this forcing frequency.

As discussed, the vibration absorber system for case 1 with forcing frequency ratio 0.8, shown in Fig. 8, vibrates with ever increasing large amplitudes with time showing divergence. It is also interesting to see the same instability phenomenon in terms of phase plane of case 1 in Fig. 9. In Fig. 9, smaller ellipses with solid line represent the change of velocity with deflection for the first two seconds and the dotted ellipses represent the same for the last two seconds of vibration. During the first two seconds of vibration, amplitude of the system was high (1.32m) with low velocity (15m/s). For the last two seconds, amplitude and velocity of the system become triple clearly indicating the instability of the system.

As  $r_2 = 1.25$  corresponds to the second natural frequency of the absorber system, the system vibrates with high amplitudes (first two peaks are  $d=115.94$  and 108.5) showing instability at the beginning, as shown in Fig. 10, for case 1. Interestingly, with time, the responses decrease; near  $t=50$ s, the amplitudes are -48.65 and 43.85, respectively. Though the

amplitudes are still high, but the trend shows gradual convergence with time. Thus it can be predicted that this linear system (case 1) approaches stability with time from an initial unstable state, exactly opposite to that for  $r_1 = 0.8$  (Figs. 8-9).

Fig. 11 and Fig. 12 show the vibration for  $r = 0.8$  and  $1.25$ , respectively for case 2. Fig. 11 shows that for  $r_1 = 0.8$ , system vibrates with high amplitudes (peaks of  $d$  vary from  $-252.16$  to  $258.95$ ) for the entire period. Similarly, Fig. 12 shows that for  $r_2 = 1.25$ , system still vibrates with high amplitudes (peaks  $d$  vary from  $-125.79$  to  $118.62$ ) for the entire period. Comparing with the previous case of Fig. 8-10, it can be said that for case 2, the absorber system remains violent for the entire time period.

All other nonlinear systems (cases 3-5) give high amplitude of vibration for  $r = 0.8$  and  $1.25$  and the shape of the graphs differs for different cases. Some representative characteristics are shown in Figs. 13 – 15 for  $r_1 = 0.8$  only.

For case 3,  $d$  varies from  $-166.2$  to  $171.39$  as shown in Fig. 13. Though smaller peaks are observed within the above range ( $-166.2$  to  $171.39$ ), it is obvious that for the entire time period the system remains resonant.

For case 4,  $d$  varies from  $-159.45$  to  $155.11$  at  $r_1 = 0.8$  as shown in Fig. 14. As seen from the figure, amplitude decreases with time for  $t = 0 - 25$ s. Minimum amplitude observed from the figure is  $76.85$ . From  $t = 25 - 50$ s amplitude of the system increases again, resulting an unstable system at  $r_1 = 0.8$ .

As seen from Fig. 15, for case 5 that the maximum peak amplitudes vary from  $-92.38$  to  $112.26$ , indicating that for the entire time period this system also remains resonant. Moreover, there is a sudden divergence trend as  $d$  becomes  $221.0$  when time  $t$  approaches  $50$  seconds. As the velocity at  $t = 50$ s is defined (Table 4), so the system's deflects with  $d = 221.0$  at  $t = 50$ s.

For each particular forcing frequency, absorber mass gives the similar graph to that of main mass but the important difference is that absorber mass gives larger amplitude vibration than main mass vibration as shown in Fig. 16 that shows absorber mass deflection at  $r_1 = 0.8$  for case 5.

## 5.2 Stability Analysis of Untuned Vibration Absorbers

### 5.2.1 Stability Analysis of Untuned Vibration Absorber (Case 6) Solved as Initial & Boundary Value Problems

Case 6 (spring with main mass and damper with absorber mass only), although is a two degrees of freedom system but it actually responses like a single degrees of freedom system (SDOF) when vibrates [Thomson (1981)]. If  $\zeta = 0.0$  (zero damping) we have an undamped SDOF with resonant frequency  $\omega_1 = \sqrt{\frac{k_1}{m_1}}$ , ( $r=r_1=1$ ). A plot of  $d - r$  will approach infinity at  $r=1.0$ . If  $\zeta$  is infinitely large, the damper mass and the spring mass will move together as a single mass, and again we have an undamped SDOF system, but with lower natural frequency of  $\sqrt{\frac{k_1}{m_1 + m_2}}$ .

#### 5.2.1.1 Stability Analysis of Untuned Vibration Absorber (Case 6) Solved as Initial Value Problems

Figs. 17 (a), (b) show the absolute non-dimensional displacement ( $|d|$ ) versus frequency ratio ( $r$ ) for case 6. This problem is solved as initial value problem. No nonlinearity is taken into consideration. As seen from the figures, peak amplitude is lowest when  $\zeta_0 = 0.288$  which is known as optimum damping ratio of the system. This analysis was made intentionally to compare the exact results for untuned viscous damper given in fig. 17(c) [Thompson (1981)]. These results of Thompson (1981) are, of course, for steady state vibrations but present study includes the transient terms. Figs. 17(a-b) for the main mass, prove the reliability of the code again as these give very similar curves given in Thompson (1981). For fig. 17 the mass ratio ( $\mu$ ) is 1.0 and damping ratios ( $\zeta$ ) are 0.1 and 1.0 for fig. 17(a) and 0.288 and 0.5 for fig. 17(b). From the figures, at low forcing frequency ( $r < 0.3747$ ) all systems (for varying  $\zeta$ ) vibrate with small amplitude ( $d$  near unity). From  $0.3747 < r < 1.1$  amplitude increases to its highest value and again decreases. For  $\zeta=0.1$  and 1, peak amplitude occurs at  $r = 0.95$  and  $0.73$  respectively. Again for  $\zeta=0.288$  and  $0.5$ , peak amplitude occurs at  $r = 0.8$ . Moreover, peak amplitude varies widely with  $\zeta$ . At  $\zeta = 0.1$  and  $1.0$  peak values of amplitude are 4.2 and 5.02, respectively. At  $\zeta = 0.288$  and  $0.5$  peak amplitude are 2.89 and 3.15, respectively.

### 5.2.1.2 Stability Analysis of Untuned Vibration Absorber (Case 6) Solved as Boundary Value Problems

$d - r$  curves for case 6 having  $\mu=1.0$  but solved as boundary value problem are shown in fig. 18. The boundary conditions given in Table 4 were used for this observation. As the boundary conditions were fixing final velocity of the system and damping force is proportional to velocity, eventually the damping force became fixed at that particular boundary ( $t=b$ ). Damper in this case has little effect on the system's final displacement unless optimum  $\zeta = \zeta_0$  is used. As seen from the figure, for  $\zeta=1.0$ , amplitude ( $d$ ) range varies from 6.60 – 8.59, for  $\zeta=0.1$  range of  $d$  varies from 5.0 - 5.7 and for  $\zeta=0.5$   $d$  varies from 3.295 to 5.28. But, for optimum damping ratio ( $\zeta=0.288$ ) system shows similar deflection to that of initial value problem. This also indicates the effect of optimum damping ratio ( $\zeta_0$ ) on the system's response is independent of boundary conditions.

$d_{\max} - \zeta$  relation for varying mass ratio is shown in fig. 19. Each curve in this figure was drawn by taking the peak amplitudes of a particular mass ratio while varying the damping ratio. From fig. 19 the line of a particular  $\mu$  becomes steeper with the decrease of its value. So system with lower mass ratio but with fixed damping ratio gives larger vibration.

### 5.2.2 Stability Analysis of Untuned Vibration Absorber Solved as Boundary Value Problem for Data Set # 1

Summary of stability of the system for data set # 1 is given in table 7. Observing the behavior of case 6 as initial and boundary value problems, analysis is next made on untuned absorber, comprising springs and dampers with both masses of the absorber. Here, two sets of data (set # 1 & set # 2) are taken to see the effect of nonlinearity indices of springs and dampers on the system's response (Table 3). Set # 1 and set # 2 consist of same masses and main spring stiffness. But the changes are made in damping constants and absorber spring stiffness. For set # 1, damping constant used is obtained from the relation of optimum damping ratio ( $\zeta_0$ ) and mass ratio ( $\mu$ ) (Table 3). Again for set # 2, chosen damping constant is larger than that of set # 1. For nonlinearity indices (both springs and dampers), values chosen for set # 1 are much smaller than that of set # 2. For data set # 1, system response for cases 7 – 21 could be seen. Figs. 20 – 29 are the selected figures for untuned absorber parameters of set # 1.



Fig. 20 is for absolute non-dimensional displacement versus frequency ratio for case 7 at  $t = 50$ s. Due to imposed velocity of the system at final condition the damper force also becomes fixed at the end. As seen from the figure, peak amplitude occurs at frequency ratio near unity and the peak value is 3.37. Cases 8 – 21 show similar type of deflection at  $t = 50$ s.

Fig. 21 shows  $d - t$  curve for case 7 having mass ratio, damping ratio and indexes mentioned above, at low forcing frequency ( $r=0.3162$ ). This analysis was made for a time period of 50s at  $r=0.3162$ . Peak values of  $d$  range are 4.714 to 3.047. Other linear and nonlinear combination of springs and dampers (cases 8 – 21) show similar type of deflection at  $r=0.3162$ , but the range of amplitude  $d$  varies.

Fig. 22 shows the  $d - t$  curve for case 7 at  $r = 1.0$ . As seen from the figure, system vibrates with very high amplitude initially and with time amplitude decreases to a small value. For the entire 50s period, amplitude ( $d$ ) varies from 102.09 to 1.60. As seen from the figure, although the system is initially unstable but approaches to stability with time.

Fig. 23 shows  $d - t$  curve for case 7 at  $r=4.744$ . Compared to fig. 22, at high forcing frequency system vibrates with lower amplitude somewhat similar to response of a vibration isolator [Thomson (1981)], and the amplitude decreases continuously ranging from 3.955 – 1.866 (farther stability is achieved). Other linear and nonlinear combination of springs and dampers (cases 8 – 21) show similar type of response at  $r=4.744$  as presented in fig. 23 but the amplitude range varies with the cases.

In fig. 24,  $d - t$  curve is shown for case 8 at  $r = 1.0$ . With this combination of spring - damper system vibrates with continuous high amplitude. As seen from the figure,  $d$  varies in the range from 64.04 to 54.9 remaining violent with time.

$d - t$  curve for case 10 at  $r=1.0$  is shown in fig. 25. Here again system vibrates with decreasing amplitude from an initial unstable state, as in case 7 at  $r=1.0$  (fig. 22) but its approach to stability is much slower than that of case 7. For the entire 50s period of vibration, amplitude range of the system is 104.5 – 40.33.

Fig. 26 shows the  $d - t$  curves for case 11 at  $r = 1.012$ . Case 9 also shows similar type of response at  $r=1.012$ . Solution for this type of spring and damper combination, does not converge with time, resulting an unstable system at  $r=1.0$ . This happens due to soft springs in absorber side with linear damping (case 9, case 11). The absorber spring force becomes

negative due to negative index and this force cannot be diminished by the linear or soft damper. This problem is eliminated when hard damper is used. Another way to solve this type of problem is to reduce the value of spring index of absorber mass. As seen from figure, amplitude  $d$  ranges from 86.53 – 2.09. Here again system approaches to stability with time.

$d - t$  curve for case 12 at  $r=1.0$  is shown in fig. 27. From this figure, it can be said that, although the vibration amplitude is high but it is lower than that of linear damper case. In case of linear damper, system approaches to stability with time but in hard damper case as in fig. 27, system never reaches to its stability, rather vibrates with high amplitude. Range of  $d$  varies from 49.91 to 30.214. Cases 12 – 16 give similar type of response at  $r=1.0$ .

$d - t$  curve for case 16 at  $r=1.0$  is shown in fig. 28. Amplitude ranges from 52.07 – 32.28. Cases 12 – 16 at  $r=1.0$  show similar response.

Fig 29 shows  $d - t$  relation for case 18 at  $r=1.012$ . As seen from figure, range of  $d$  varies from 86.1 – 2.09 approaching to stability with time. As solutions of the system comprising soft dampers (cases 17 – 21) does not converge, system is unstable at  $r=1.0$ . Cases 17, 19 – 21 show similar type of response at  $r=1.012$  but with varying range of amplitude.

### **5.2.3 Stability Analysis of Untuned Vibration Absorber Solved as Boundary Value Problem for Data Set # 2**

Table 8 shows summary of different cases. For Cases 9, 11, 14, 16 & 17 – 21 numerical solution does not converge due to larger spring and damper indices and as a result these cases do not show any convergence even in lower or higher speed ratio. However, results of all the cases 7 – 21 with data set # 1 have already been discussed previously.

Figs. 30 – 46 are obtained for the system having untuned absorber parameters of data set # 2. Dampers, nonlinearity indices of springs and dampers are kept to a high value, given in Table 3 to see their effect on the untuned system vibration. Analysis shows that, cases 9, 11, 14 & 16 – 21 show instability from the very beginning due to high negative spring and damper nonlinear indices as numerical solution diverges for these cases.

Fig. 30 shows non-dimensional displacement versus frequency ratio for both main and absorber mass at  $t = 0.5s$  (in the transient region) for case 7. From the observation, the absorber mass deflection is always greater than that of main mass. Again at  $t=0.5s$  the deflection is high for all forcing frequencies. Although in fig. 30, the system comprises linear spring with linear damper but other cases (8, 10, 12, 13 & 15) show the similar type of deflection at transient period  $t=0.5s$ . In fig. 30,  $d$  varies from 368.94 to 536.53 and  $e$  from 742.15 to 1078.2 for the entire frequency range of study.

Fig. 31 shows the  $d - r$  and  $e - r$  curves for case 7 at  $t=1.0s$ .  $d$  initially increases to a large value as  $r$  approaches to 0.9. Then with  $r > 0.9$  system-response decreases and after  $r = 4.0$  system vibrates in a steady manner. Again,  $e$  increases to maximum amplitude as  $r$  approaches 1.14 and after that  $e$  decreases continuously. Cases 8 – 15 give similar types of  $d - r$  and  $e - r$  curves at  $t=1.0s$  except cases 9, 11, 14.

Fig. 32 again shows the  $d - r$  and  $e - r$  curves for case 7 at  $t=50.0s$ . As seen from the figure, both main mass and absorber mass vibrate with decreasing amplitude for gradually increasing frequency ratio. Cases 8 – 15 (except 9, 11, 14) give similar response with frequency ratio at  $t=50.0s$ .

For case 7, fig. 33 shows the non-dimensional deflection of main mass with time at  $r = 0.3162$ . As seen from the graph  $d$  varies slightly ( $d$  varies from -8.54 to 7.13) during the entire period of analysis.

Fig. 34 shows the non-dimensional deflection of absorber mass with time at  $r = 0.3162$  for case 7. From the figure it is observed that, absorber mass shoots to a very high value ( $e = 862.97$ ) at start and due to the damping effect, the deflection reduces to a lower value ( $e$  varies from 8.43 to 7.41 from  $t = 10 - 50s$ ) over the entire period of analysis.

Fig. 35 and fig. 36 show vibration for case 7 at  $r = 1.0$  for main mass and absorber mass respectively. In fig. 35 main mass of the system vibrates with high amplitudes (first two peaks are  $d = 74.374$  and  $77.2$ ) showing instability at the beginning. Interestingly, with time, the response decreases; near  $t = 50s$  the amplitudes are 4.56 and -5.37 respectively. The amplitudes have become considerably small and the trend shows a gradual convergence with time. Thus, it can be predicted that this linear system approaches stability with time from an

initial unstable state. Again in fig. 36, absorber mass of the system starts from a very high amplitudes and then the amplitude gradually decreases to small values.

At high forcing frequency ( $r = 4.744$ ),  $d - t$  relation for linear springs with linear damping (case 7) are given in fig. 37. In fig. 37 amplitude of vibration decreases gradually (range of  $d$  varies from 7.198 to 5.50).

Fig. 38 shows  $d - t$  curve for case 8 at  $r = 0.3162$ . Here the main mass initially shoots to a high value ( $d = 18.5$ ) and then the amplitude of vibration reduces gradually. After shooting,  $d$  varies from 6.25 to 5.0 during entire 50s vibration showing stability of the system. Case 10 combination shows similar type of response at  $r = 0.3162$ .

The effect of nonlinearity of springs with linear damper on main mass at  $r = 1.0$  is shown in fig. 39 case 8. Although at start amplitude is quite high ( $d = 56.84$ ), it decreases sharply ( $d$  becomes 31.82) and then increases with the time; last two amplitudes are 75.34 and -75.44. Thus, it can be predicted that this nonlinear spring and linear damping system approaches instability with time.

Again the effect of nonlinearity of springs with linear damper on absorber mass at  $r = 1.0$  is shown in fig. 40,  $e - t$  relation for case 10. Here again system starts vibrating from high amplitude and drops sharply, and then increases gradually with time approaching instability.

Fig. 41 shows the similar  $d - r$  and  $e - r$  curves for case 12 at  $t = 0.5$ s (in the transient region). The range of deflection for both main mass and absorber mass is same to that of case 7, except in nonlinear case there is a high shooting at start. For nonlinear spring combination with nonlinear damping (cases 13, 15)  $d - r$  and  $e - r$  give similar curves to that of case 7 at  $t = 0.5$ s. For cases (13 - 16) no shooting occurs initially.

In fig. 42,  $d - t$  curve is drawn for case 12 at  $r = 0.3162$ . The range of  $d$  varies from 2.8464 to -2.4881 for total 50s vibration. Again at  $t = 50$ s a shooting occurs at a comparatively larger value of  $d$  (6.387). Comparing Fig. 42 ( $d - t$  for case 12) with  $d - t$  curve for case 7 at  $r = 0.3162$ , we see that the deflection range is smaller for hard damper than that of linear damping case. Thus, nonlinear damping (hard damping) causes vibration of smaller amplitude and this is the fact for both linear and nonlinear combination of springs.

Fig. 43,  $e - t$  curve, shows the absorber mass deflection with case 12 of previous figure at  $r = 0.3162$ . Here also figure supports the comparison that is made for the main mass. Although absorber mass begins vibration with larger value of  $e$  ( $e=360$  at start), the amplitude of vibration falls to a smaller range and becomes almost stable ( $e$  varies from 2.602 to 2.589 for  $t=10-50s$ ).

Fig. 44 and Fig. 45 show the vibration of case 12 for main mass and absorber mass, respectively at  $r=1.012$ . In fig. 43, 1<sup>st</sup> two amplitudes of vibration are -64.6 and 64.93 respectively. With the passing of time, amplitude decreases and the last two amplitudes ( $d$ ) are to -8.55 and 6.17, approaching to stability from an initial unstable state. Similar trend occurs in fig. 45, where  $e$  at start is -136.34 and gradually decreases to -6.9 and 5.94 respectively.

Fig. 46, shows  $d - t$  curve for case 12 at high forcing frequency ( $r = 4.744$ ). Comparing to case 7 at high forcing frequency, amplitude of vibration for this combination (case 12) is much smaller than that of linear damper combination. Range of  $d$  varies from 1.42 to 1.081. A sudden shooting occurs at the end of the vibration period. Cases 13 and 15 show similar type of displacements as in fig. 46 at  $r=4.744$ . Summary of stability of the system for data set # 2 is given in Table 8.

# Chapter 6

## Conclusions and Recommendations

Following the set objectives of this study, stability in terms of responses of tuned and untuned vibration absorbers with nonlinearities in springs and dampers has been extensively studied in the form of boundary value problems. Soundness of the code has been demonstrated comparing a few results of present analysis with available exact results. Following conclusions can be drawn from the present study:

### 6.1 Conclusions for Tuned Vibration Absorbers:

- When the unknown boundary values, representing the response of the system at the two time references  $t=a$  and  $t=b$ , are numerically calculated, it is found that, spring nonlinearity has little effect on the main mass. For example, with reference to case 1 (linear case),  $y_1(b)$  does not change remarkably for nonlinear cases 2-5. Another boundary value,  $y_2(a)$  is only 1.408% and 1.407% lower for case 2 and case 4 and 1.458% and 1.489% higher, respectively, for case 3 and case 5.
- Spring nonlinearity has an obvious effect on the effect on the absorber mass. For example, with reference to case 1,  $y_3(b)$  is 15.315% and 14.865% lower for case 2 and case 4; but, 14.865% and 15.315% higher for case 3 and case 5. Again,  $y_4(a)$  is 3.216% and 3.084% higher for case 2 and case 4; but 3.318% and 3.258% lower for case 3 and case 5, respectively.
- Like the linear absorber (case 1), all the nonlinear cases 2-5 show more or less violent responses at the two resonant frequencies corresponding to  $r_1 = 0.8$  and  $r_2=1.25$ . While at the tuned frequency  $r =1.0$ , the responses of the system are stable and very low for all the cases 1-5.
- First resonant frequency is more dangerous than the second one for cases 1 – 5 in terms of non-dimensional displacements at  $t=50s$ . Absolute non-dimensional displacements at  $r = 0.8$  is the highest for case 5. Case 1 and case 2 show the second and third highest displacements, respectively at  $r_1=0.8$ .

- For linear springs (case 1) the tuned absorber system becomes more unstable with the time when  $r_1=0.8$  but the same system becomes stable from an unstable state for  $r_2=1.25$ . The nonlinear systems, however, more or less shows violent response at the two resonant frequencies and thus can be considered to remain unstable for the entire time period.

## 6.2 Conclusions for Untuned Vibration Absorbers:

- Effect of optimum damping ratio ( $\zeta_0$ ) on the system's response is independent of boundary conditions. In case of optimum damping untuned absorber vibrates with minimum amplitude. This happens for both initial value and boundary value problems. For example, case 6 with optimum damping ratio ( $\zeta_0=0.288$ ) shows similar deflections for initial value and boundary value problems.
- Mass ratio ( $\mu$ ) plays a significant role on the maximum deflection of untuned absorber. For the same damping ratio, system with larger mass ratio shows smaller peak amplitude. But all the systems with a particular mass ratio show minimum peak at optimum damping ratio.
- Increased nonlinearity in spring and damper makes the system more unstable. For example, systems with data set # 1 show more stability than that for data set # 2 (having higher nonlinearity indexes compared to data set # 1) for different cases at  $r=1.0$ .
- Untuned system with different cases (7 – 21) show similar responses for both lower and higher forcing frequencies. For example, at forcing frequencies  $r=0.3162$  and  $4.744$  all the system with data set # 1 show similar responses. For data set # 2, systems that converge at  $r=0.3162$  and  $4.744$  show similar response also.
- From this study we can conclude that practical springs and dampers should contain smaller nonlinearity indices as systems with lower nonlinearity index approaches to more stability.

- Practically no spring or damper remains linear with ever-increasing response. Therefore this study is particularly useful for cars' shock absorber design and application. As stability of a shock absorber (which can be considered an untuned absorber) can easily change because of damper and spring nonlinearities.
- This study is also useful when specially, when a vibration control scheme becomes a boundary value problem.
- Nonlinear vibration analysis of machinery and equipment in power plants can utilize this study.
- Nonlinear vibration analysis of buildings, bridges and other structures can also utilize this study.

### **6.3 Recommendations for Future Works**

In future, the developed computer code can be used to study response of the similar system having higher DOF. Practically, this method of vibration analysis will provide handy information for active/passive vibration control of structures. The following recommendations can be made for future works from experience gained while achieving the set objectives of this thesis:

1. The present analysis should be extended to observe the effect of vibration of the system with higher degrees of freedom. Nonlinearity in system can also be analyzed for both spring and damper. In Appendix – A, investigation for 3 DOFS with spring nonlinearity is presented. Therefore, systems with more degrees of freedom can be studied.
2. The effect of self-excited force on the system can be studied. Self-excited vibration forces with and without nonlinearity can be added to this present study.
3. Experimental studies can be carried out to verify the results obtained for the tuned and untuned vibration absorbers.



## Reference:

- [1] Alexander N. A., Schilder F., (2009), "Exploring the Performance of a Nonlinear Tuned Mass Damper" *Journal of Sound and Vibration* 319, pp. 445–462.
- [2] Arnold F. R., (1955), "Steady-State Behavior of Systems Provided with Nonlinear Dynamic Vibration Absorbers," *Journal of Applied Mechanics* 22, pp. 487-492.
- [3] Bachmann H., (1995), "Vibration Problems in Structures", Birkhauser.
- [4] Breads F. C., (1996), "Structural Vibration: Analysis and Damping", Arnold.
- [5] Den Hartog J.P., (1956), "Mechanical Vibrations (4<sup>th</sup> Edition)", McGraw Hill.
- [6] Hunt J. B. and Nissen J. C., (1982), "The Broadband Dynamic Absorber," Letter to the Editor, *Journal of Sound and Vibration* 83, pp. 573-578.
- [7] Jacquot R.G., (1978), "Optimal Dynamic Vibration Absorbers for General Beam Systems", *Journal of Sound and Vibration* 60, pp. 535-542.
- [8] Jordanov I.N., Cheshankov B.I., (1988). "Optimal Design of Linear and Non-linear Dynamic Vibration Absorbers", *Journal of Sound and Vibration* 123(1), pp. 157-170.
- [9] Juang J.N., (1984), "Optimal Design of a Passive Vibration Absorber for a Truss Beam", *AIAA Journal of Guidance Control Dynamics* 7, pp. 733-738.
- [10] Kalnins A. and Dym C. L.,(1976), "Vibration Beams, Plates, and Shells", Dowden, Hutchinson & Ross, Inc.
- [11] Kalnins, A., and Lestingi, J. E., (1967). "On Nonlinear Analysis of Elastic Shells of Revolution", *J. Appl. Mech.* 34, pp. 59-64.
- [12] Manevitch L. I., Gendelman O., Musienko A. I., Vakakis A. F., and Bergman L., (2003), "Dynamic interaction of a semi-infinite linear chain of coupled oscillators with a strongly nonlinear end attachment", *Physica D. Nonlinear Phenomena* 178(1-2), pp. 1–18.
- [13] McFarland D. M., Bergman L. A., Vakakis A. F., Manevitch L. I., and Gendelman O., (2002), "Energy pumping into passive nonlinear energy sinks attached to forced linear substructures: analytical and experimental results", 9<sup>th</sup> Conference on Nonlinear Vibrations, Stability, and Dynamics of Structures, Virginia Polytechnic Institute and State University.
- [14] Mikhlin Y. V., Reshetnikova S. N., (2005). "Dynamical Interaction of an Elastic System and Essentially Nonlinear Absorber", *Journal of Sound and Vibration* 283 (1-2), pp. 91-120.

- [15] Nakhaie G., Narimani A., Golnaraghi M. F., Swanson D.A, (2003), "Practical Frequency and Time Optimal Design of Passive Linear Vibration Isolation Mounts," *Vehicle System Dynamics* 39, pp. 437-466.
- [16] Natsiavas S., (1992). "Steady State Oscillations and Stability of Non-linear Dynamic Vibration Absorbers", *Journal of Sound and Vibration* 156 (2), pp. 227-245.
- [17] Natsiavas, S. (1993). "Dynamics of multiple-degree-of-freedom oscillators with colliding components", *Journal of Sound and Vibration* 165 3, pp. 439-453.
- [18] Oueini S.S., Chin C.M., and Nayfeh A.H., (1999), "Dynamics of a Cubic Nonlinear Vibration Absorber", *Nonlinear Dynamics* 20, pp. 283-295.
- [19] Oueini S.S., Nayfeh A.H., and Pratt J.R., (1998), "A Nonlinear Vibration Absorber for Flexible Structures", *Nonlinear Dynamics* 15, pp. 259-282.
- [20] Rice H.J., (1986). "Combinational Instability of the Non-linear Vibration Absorber", *Journal of Sound and Vibration* 108(4), pp. 526-532.
- [21] Roberson R. E., (1952), "Synthesis of a Nonlinear Dynamic Vibration Absorber," *Journal of the Franklin Institute* 254, pp. 205-220.
- [22] Shaw J., Shaw S.W., Haddow A.G., (1989). "On the Response of the Non-linear Vibration Absorber", *Journal of Non-linear Mechanics* 24, pp. 281-293.
- [23] Shekhar N. C., Hatwal H. , Mallik A. K., (1998), " Response of Non-linear Dissipative Shock Isolators," *Journal of Sound and Vibration* 214, pp. 589-603.
- [24] Shekhar N. C., Hatwal H., Mallik A. K., (1999), " Performance of Nonlinear Isolators and Absorbers to Shock Excitation," *Journal of Sound and Vibration* 227, pp. 293-307.
- [25] Soom A., Lee M., (1983). "Optimal Design of Linear and Non-linear Vibration Absorbers for Damped System". *Journal of Vibration, Acoustic Stress, and Reliability in Design* 105, pp. 112-119.
- [26] Thomson, W. T., (1981). "Theory of Vibrations with Applications (2<sup>nd</sup> Edition)", George Allen & Unwin.
- [27] Timoshenko, S., (1974). "Vibration Problems in Engineering (4<sup>th</sup> Edition)", John Willy & Sons.
- [28] Vakakis A.F., and Paipetis S.A., (1986). "The Effect of a Viscously Damped Dynamic Absorber on a Linear Multi Degree of Freedom System", *Journal of Sound and Vibration* 105(1) pp. 45-60.
- [29] Wang B.P., Kitis L., Pilkey D., Palazzolo A. , (1985), "Synthesis of Dynamic Vibration

Absorbers”, ASME Journal of Vibration, Acoustics, Stress and Reliability in Design 107, pp. 161-166.

- [30] Zhu S.J., Zheng Y.F., Fu Y.M., (2004). “Analysis of Non-linear Dynamics of a Two Degree of Freedom Vibration System with Non-linear Damping and Nonlinear Spring”, Journal of Sound and Vibration 271, pp. 15-24.

Table 1: Different cases of tuned absorbers

Case	Combination of springs
Case 1	Both springs linear
Case 2	Both springs hard
Case 3	1 <sup>st</sup> spring: hard 2 <sup>nd</sup> spring: soft
Case 4	1 <sup>st</sup> spring: soft 2 <sup>nd</sup> spring: hard
Case 5	Both springs soft

Table 2: Different cases of untuned vibration absorbers

Case	Combination of Springs	Combination of Dampers	Summary of the Systems
Case 6	Linear spring with main mass only	Linear damper with absorber mass only	$c_1=0.0, k_2=0.0$
Case 7	Both springs linear	Linear dampers	Both linear and nonlinear spring combinations with linear dampers, Cases 7 – 11
Case 8	Both springs hard		
Case 9	1 <sup>st</sup> spring: hard 2 <sup>nd</sup> spring: soft		
Case 10	1 <sup>st</sup> spring: soft 2 <sup>nd</sup> spring: hard		
Case 11	Both springs soft		
Case 12	Both springs linear	Hard dampers	Both linear and nonlinear spring combinations with hard dampers, Cases 12 – 16
Case 13	Both springs hard		
Case 14	1 <sup>st</sup> spring: hard 2 <sup>nd</sup> spring: soft		
Case 15	1 <sup>st</sup> spring: soft 2 <sup>nd</sup> spring: hard		
Case 16	Both springs soft		
Case 17	Both springs linear	Soft dampers	Both linear and nonlinear spring combinations with soft dampers, Cases 17 – 21
Case 18	Both springs hard		
Case 19	1 <sup>st</sup> spring: hard 2 <sup>nd</sup> spring: soft		
Case 20	1 <sup>st</sup> spring: soft 2 <sup>nd</sup> spring: hard		
Case 21	Both springs soft		

Table 3: Chosen Parameters of Tuned & Untuned Vibration Absorbers:

Parameter	Values for the Tuned Absorber	Values for the Untuned Absorber Solved as Initial Value Problem	Values for the untuned absorbers solved as boundary value problem.	
			set # 1	set # 2
$m_1$ (kg)	5.0	100.0	100.0	100.0
$m_2$ (kg)	1.0	100.0	1.0	1.0
$k_1$ (N/m)	1000.0	1000.0	1000.0	1000.0
$k'_1$ (N/m <sup>3</sup> )	$\pm 0.5$	0.0	$\pm 0.5$	$\pm 0.5$
$k_2$ (N/m)	200.0	0.0	10.0	10.0
$k'_2$ (N/m <sup>3</sup> )	$\pm 0.5$	0.0	$\pm 0.005$	$\pm 0.5$
$c_1$ (Ns/m)	0.0	0.0	0.03139	1.0
$c'_1$ (Ns/m <sup>3</sup> )	0.0	0.0	$\pm 0.003$	$\pm 0.01$
$c_2$ (Ns/m)	0.0	63.246, 182.174, 316.23, 632.46	3.139	10.0
$c'_2$ (Ns/m <sup>3</sup> )	0.0	0.0	$\pm 0.03$	$\pm 0.1$
$\sqrt{\frac{k_1}{m_1}}$	14.14	3.162	3.162	3.162
$\mu$	0.2	1.0	0.01	0.01
$\zeta$	0.0	0.1, $\zeta_0 = 0.288$ , 0.5, 1.0	$\zeta_0 = 0.00496$	0.0158
$f$ (N)	20.0	20.0	20.0	20.0
$\omega_1$ (rad/s)	11.132	3.162	3.006	3.006
$\omega_2$ (rad/s)	16.675	-	3.326	3.326

$$\text{Damping ratio, } \zeta = \frac{c_1}{2\sqrt{m_1 k_1}}$$

$$\text{Optimum damping ratio } \zeta_0 = \frac{\mu}{\sqrt{2(1+\mu)(2+\mu)}}$$

Table 4: Chosen boundary conditions of boundary value problem for tuned & untuned vibration absorbers

Parameter	Values for Tuned absorbers	Values for Untuned absorbers	
		set # 1	set # 2
$y_1(t=0.0s)$ (m)	0.05	0.05	0.05
$y_2(t=50.0s)$ (m/s)	0.06	0.06	0.06
$y_3(t=0.0s)$ (m)	-0.07	-0.07	-0.07
$y_4(t=50.0s)$ (m/s)	-0.06	-0.06	-0.06

Table 5: Chosen boundary conditions of initial value problem for untuned vibration absorbers

Parameter	Values for Untuned absorbers
$y_1(t=0.0s)$ (m)	0.0
$y_2(t=0.0s)$ (m/s)	0.0
$y_3(t=0.0s)$ (m)	0.0
$y_4(t=0.0s)$ (m/s)	0.0

Table 6: Different cases of absorbers and calculated boundary values at the tuned frequency ( $r=1$ ).

[Given boundary conditions:  $y_1(t=0s) = 0.05$  m,  $y_2(t=50s) = 0.06$  m/s,  $y_3(t=0s) = -0.07$  m,  $y_4(t=50s) = -0.06$  m/s]

Case	Combination of springs	Calculated boundary values			
		Main mass		absorber mass	
		$y_1$ (b) (m)	$y_2$ (a) (m/s)	$y_3$ (b) (m)	$y_4$ (a) (m/s)
Case 1	Both springs linear	-0.0791	-0.9944	0.0222	-1.6667
Case 2	Both springs hard	-0.0793	-0.9804	0.0188	-1.7203
Case 3	1 <sup>st</sup> spring: hard 2 <sup>nd</sup> spring: soft	-0.0790	-1.0089	0.0255	-1.6144
Case 4	1 <sup>st</sup> spring: soft 2 <sup>nd</sup> spring: hard	-0.0792	-0.9806	0.0189	-1.7181
Case 5	Both springs soft	-0.0790	-1.0092	0.0256	-1.6124

Table 7: Characteristics of the Responses of Untuned Absorbers for Data Set # 1

Cases	Characteristics of the Responses at		
	$r=0.3162$	$r=1.0$	$r=4.744$
7	approaches farther stability from an initially stable state	Approaches stability from an initially unstable state	approaches farther stability from an initially stable state
8		Remains unstable with continuous high amplitude	
9		Unstable from the very beginning if $r=1.0$ . But approaches stability from an initially unstable state at $r=1.012$	
10		Approaches stability from an initially unstable state	
11		Unstable from the very beginning if $r=1.0$ . But approaches stability from an initially unstable state at $r=1.012$	
12		Remains unstable with increasing amplitude	
13			
14			
15			
16			
17		Unstable from the very beginning if $r=1.0$ . But approaches stability from an initially unstable state at $r=1.012$	
18			
19			
20			
21			



Table 8: Characteristics of the Responses of Untuned Absorbers for Data Set # 2

Cases	Characteristics of the Responses at		
	$r=0.3162$	$r=1.0$	$r=4.744$
7	Stable	Approaches stability from an initially unstable state	approaches farther stability from an initially stable state
8		Remains unstable with increasing amplitude	
9	Unstable from the very beginning		
10	Stable	Remains unstable with increasing amplitude	approaches farther stability from an initially stable state
11	Unstable from the very beginning		
12	Stable	Unstable from the very beginning if $r=1.0$ . But approaches stability from an initially unstable state at $r=1.012$	Stable
13			
14	Unstable from the very beginning		
15	Stable	Unstable from the very beginning if $r=1.0$ . But approaches stability from an initially unstable state at $r=1.012$	Stable
16	Unstable from the very beginning		
17			
18			
19			
20			
21			

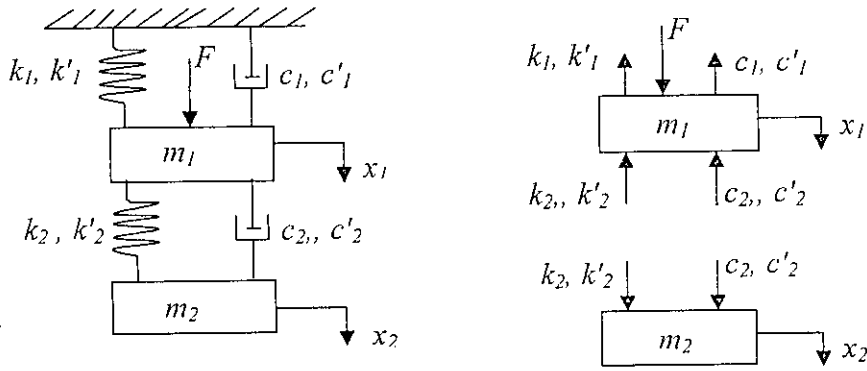


Fig. 1 Arrangement of masses, springs and dampers for TDOF vibration system.

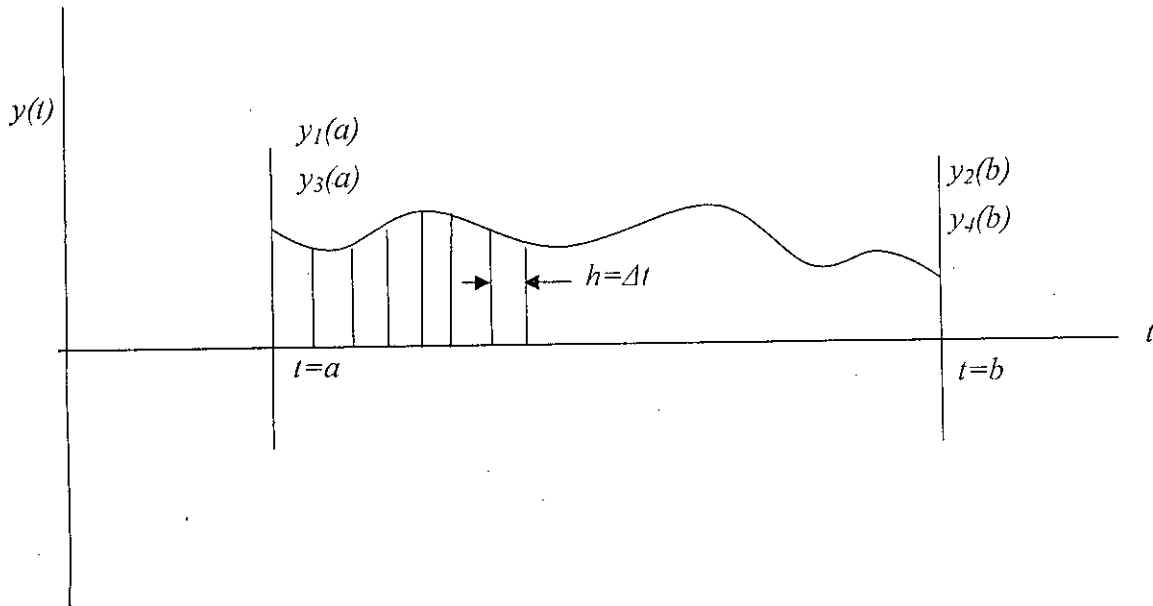
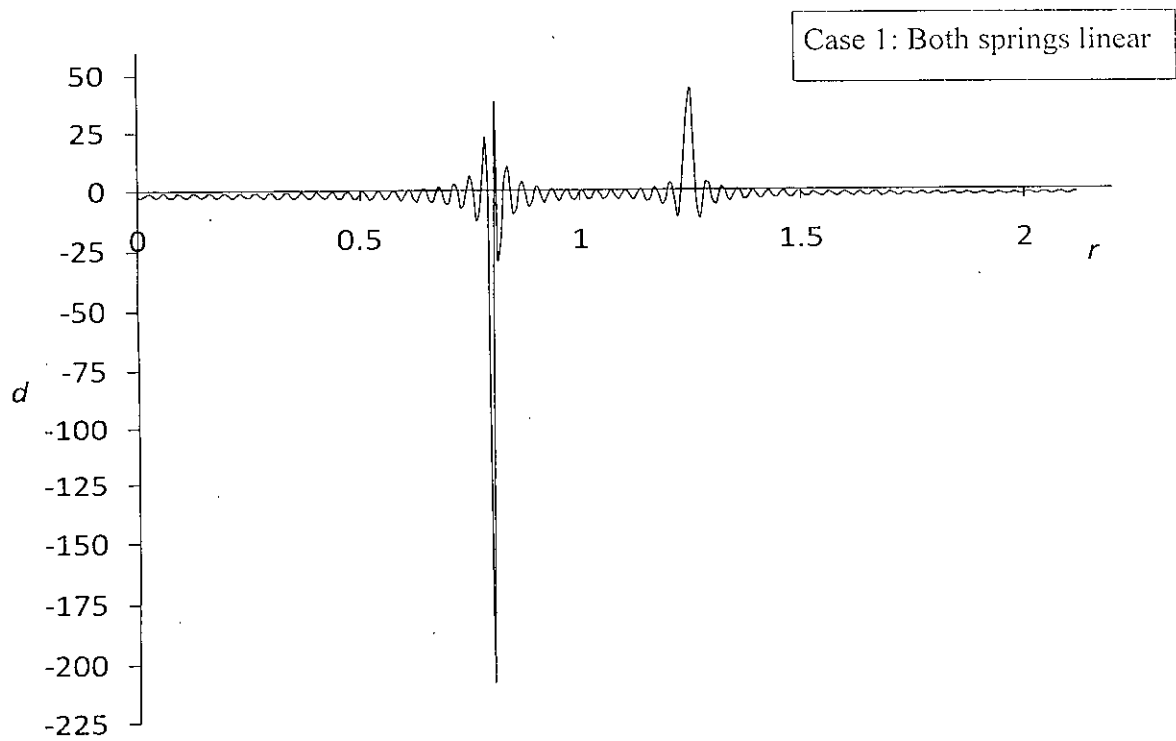
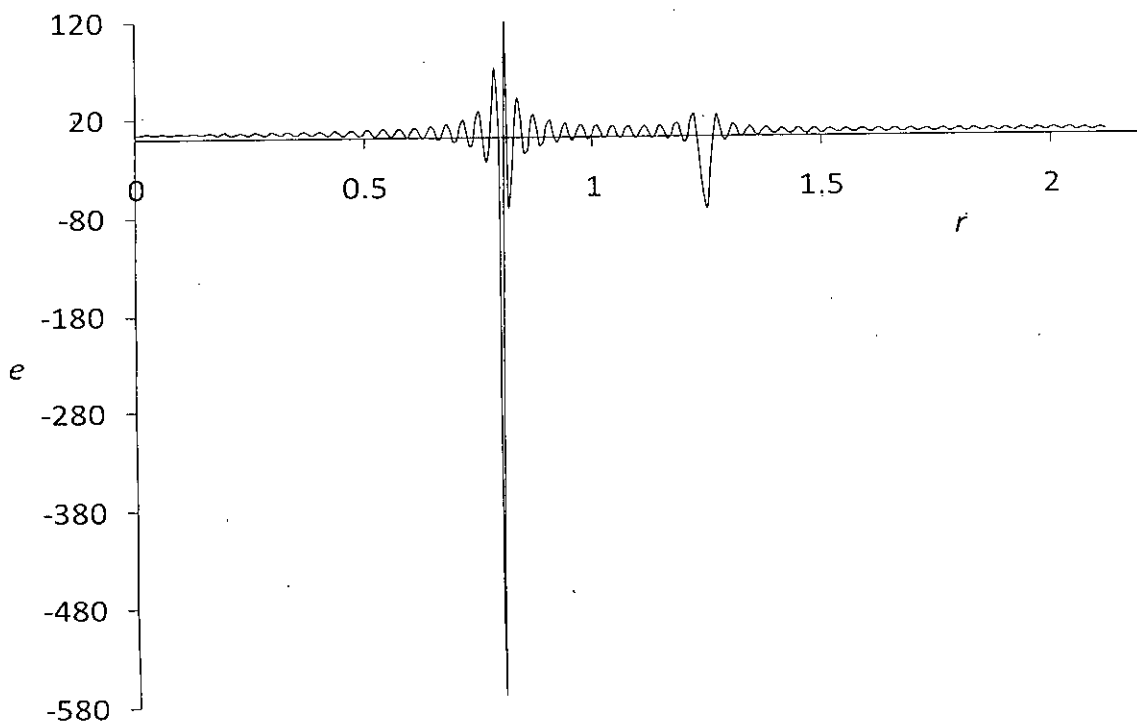


Fig. 2 Prescribed boundary conditions at  $t(a)$  and  $t(b)$ .

FIGURES FOR  
TUNED ABSORBERS  
(FIGS. 3 – 16)



(a)



(b)

Fig. 3 Non-dimensional displacement - frequency ratio for case 1 at  $t = 50s$  : (a) main mass (b) absorber mass

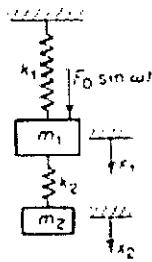


Figure 5.6-1. Vibration absorber.

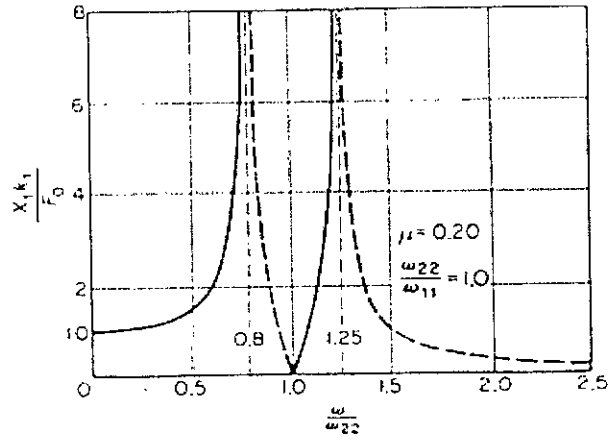
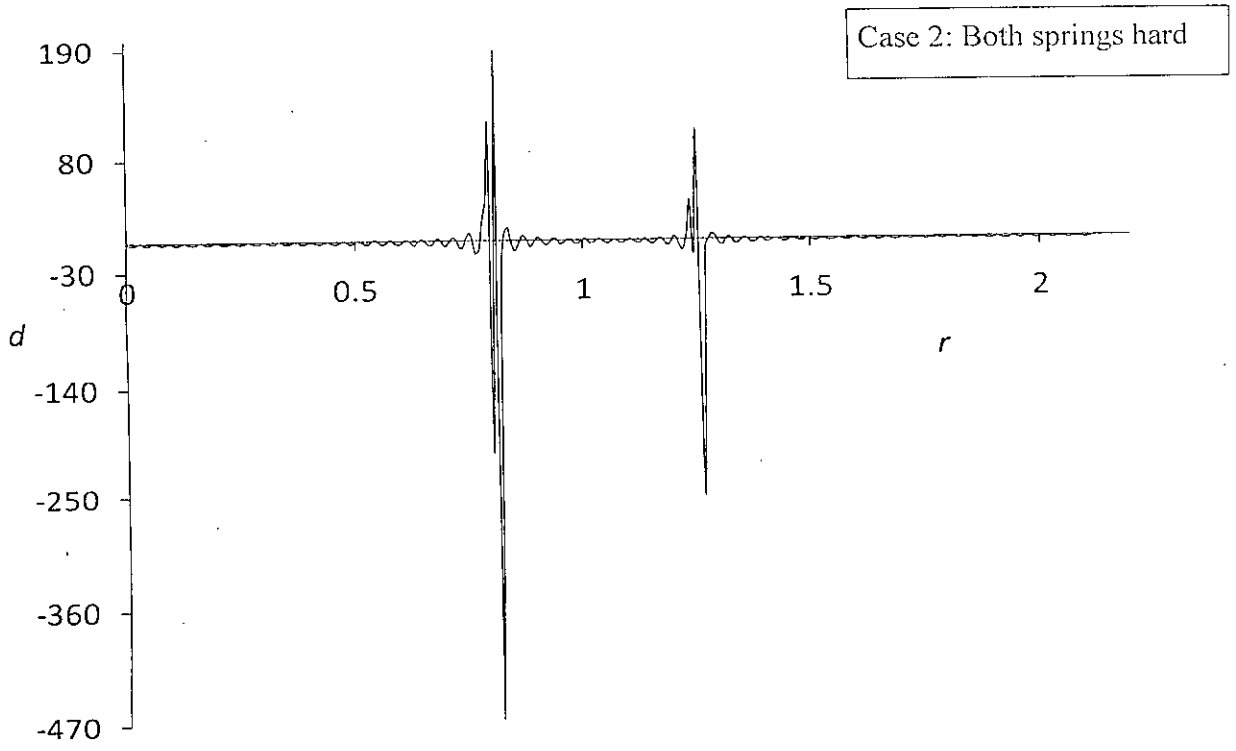


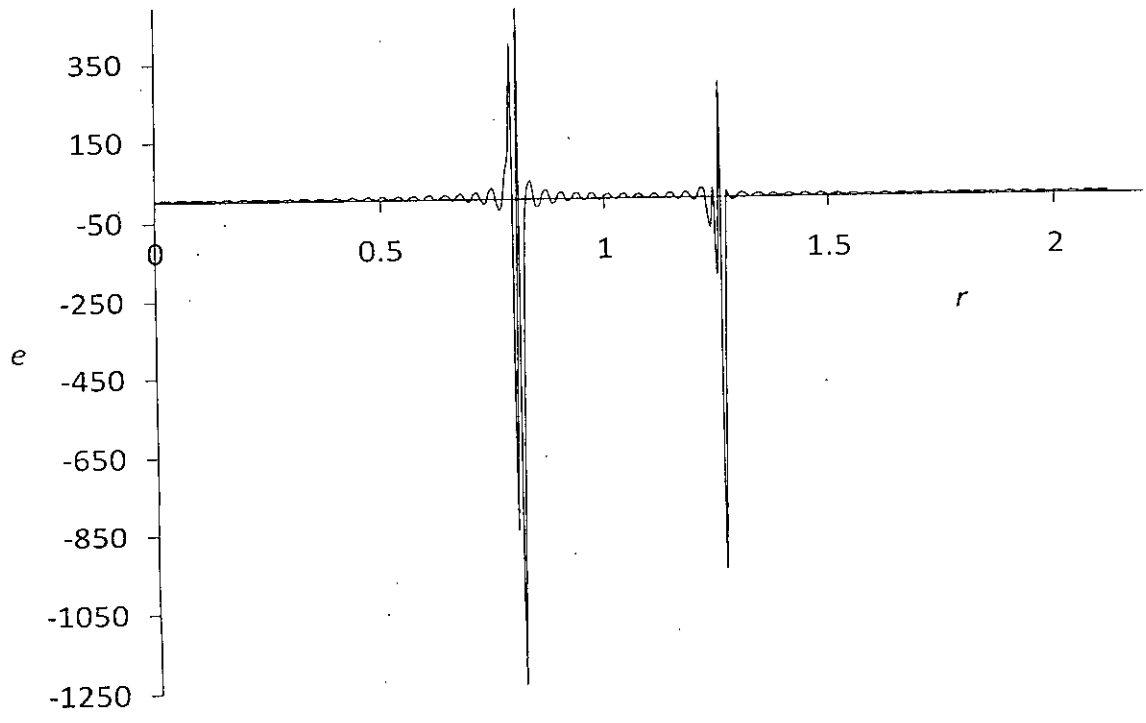
Figure 5.6-2. Response vs. frequency.

(c)

Fig. 3 (c) Exact solution given in Thomson (1981) for tuned absorbers.



(a)



(b)

Fig. 4 Non-dimensional displacement - frequency ratio for case 2 at  $t=50s$  : (a) main mass (b) absorber mass.

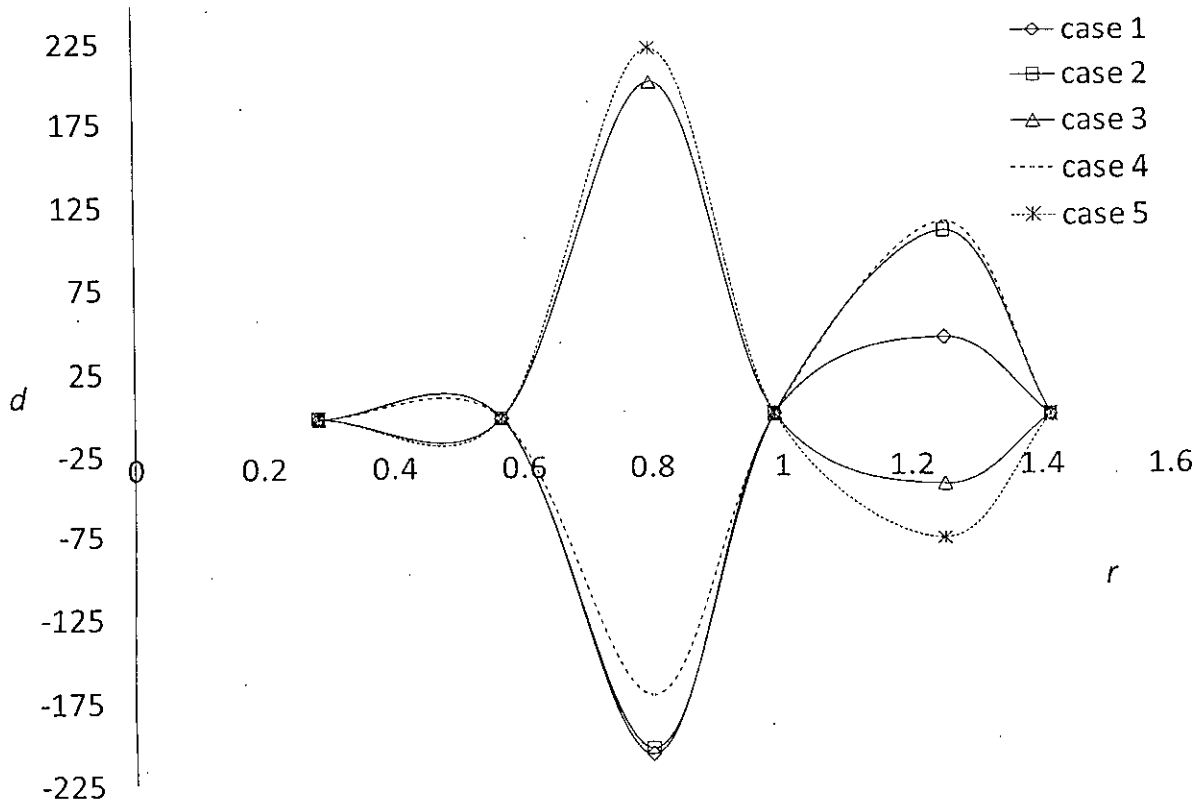


Fig. 5 Comparison of non-dimensional displacement - frequency ratio for all cases together at  $t=50s$ .

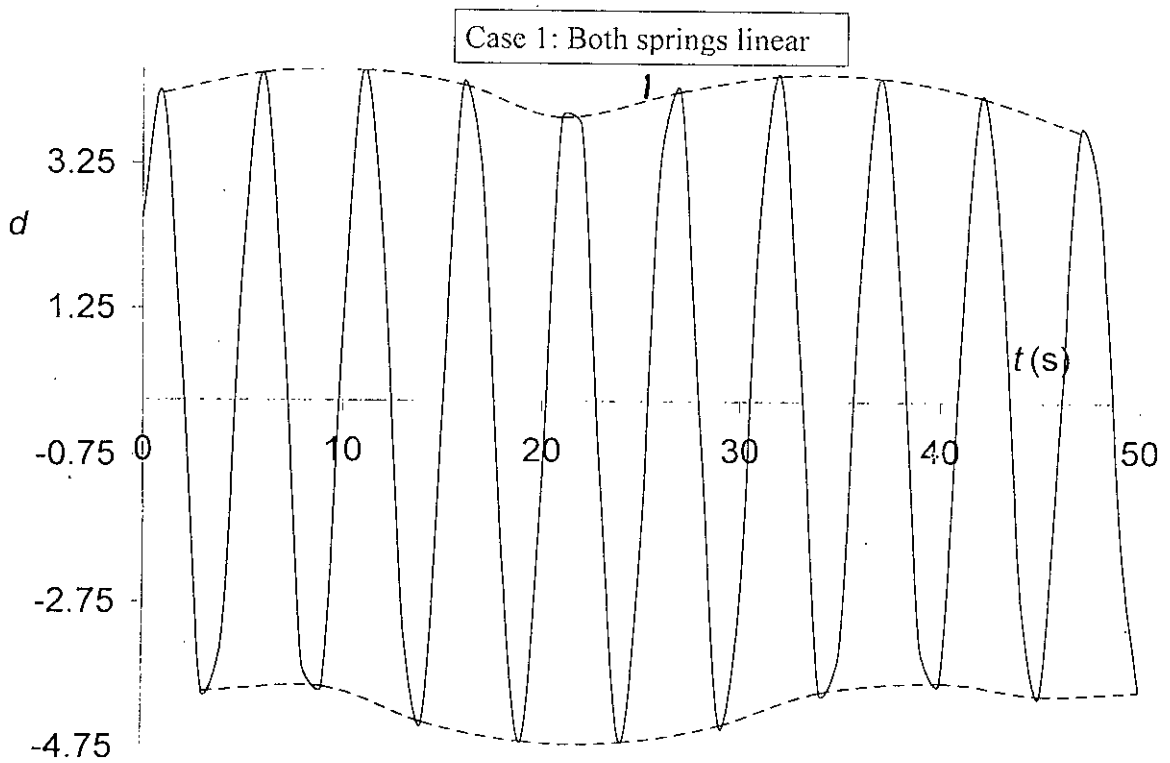


Fig. 6 Non-dimensional displacement - time for case 1 (both springs linear) at the tuned frequency ( $r = 1.0$ ).

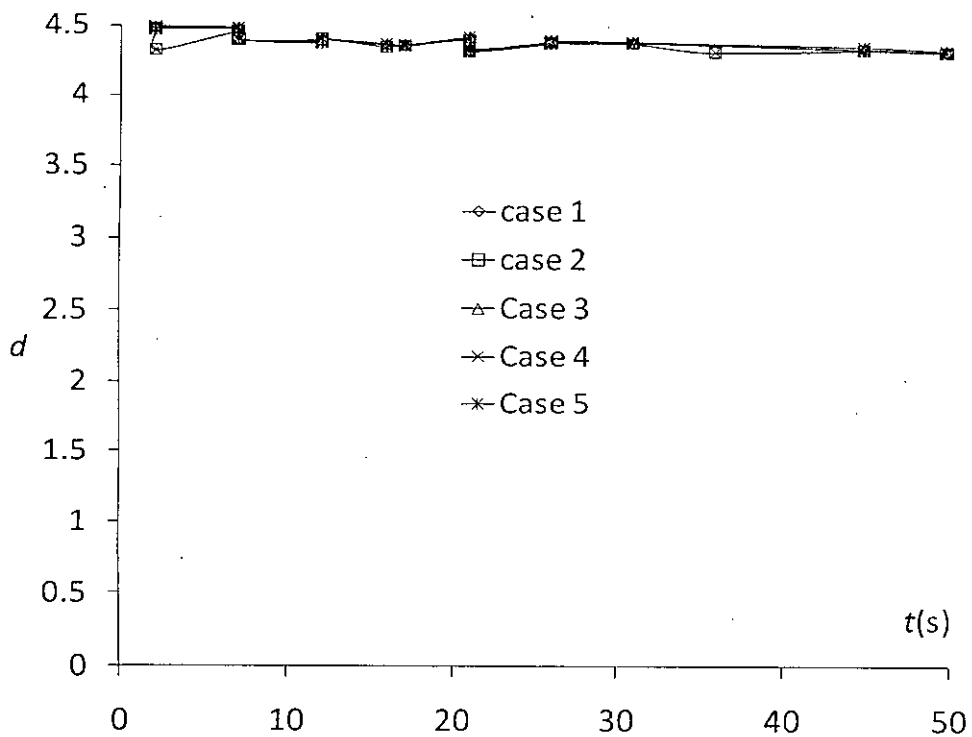


Fig. 7 Non-dimensional displacement - time for all combination of springs at tuned frequency ( $r=1.0$ ).



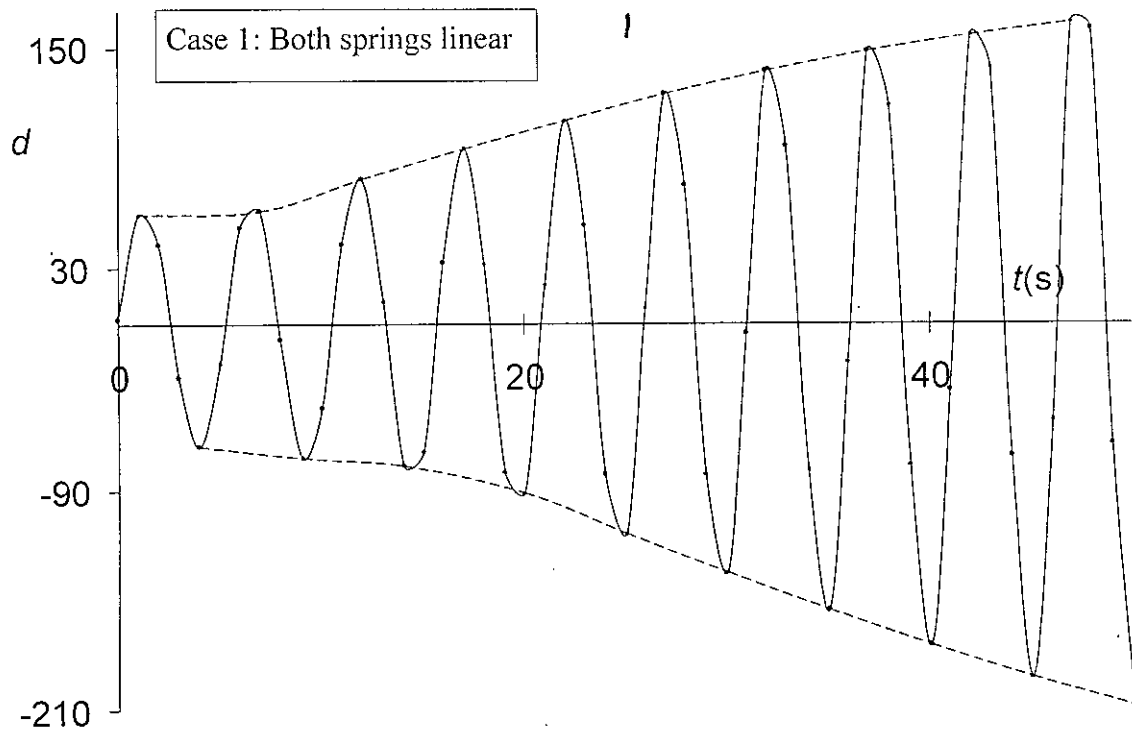


Fig. 8 Non-dimensional displacement - time for case 1 at  $r_1 = 0.8$ .

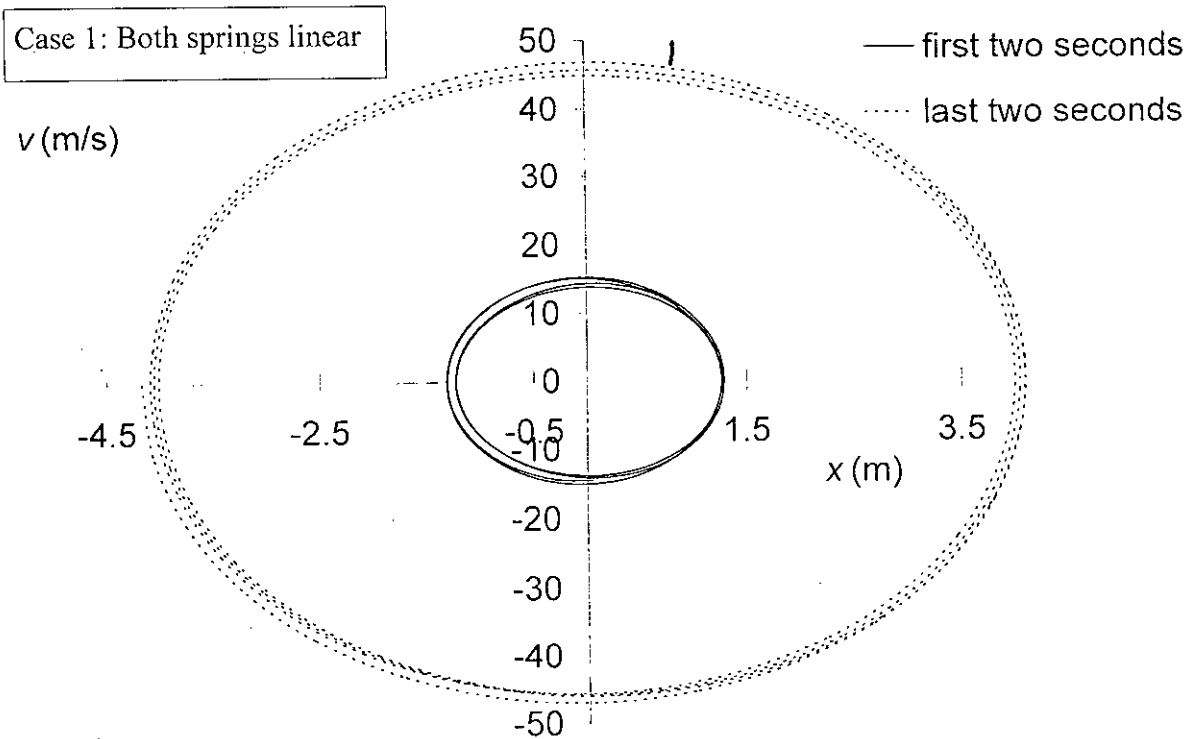


Fig. 9 Main mass velocity - displacement (phase plane) for case 1,  $r_1 = 0.8$ ; for the first 2 and the last 2 seconds.

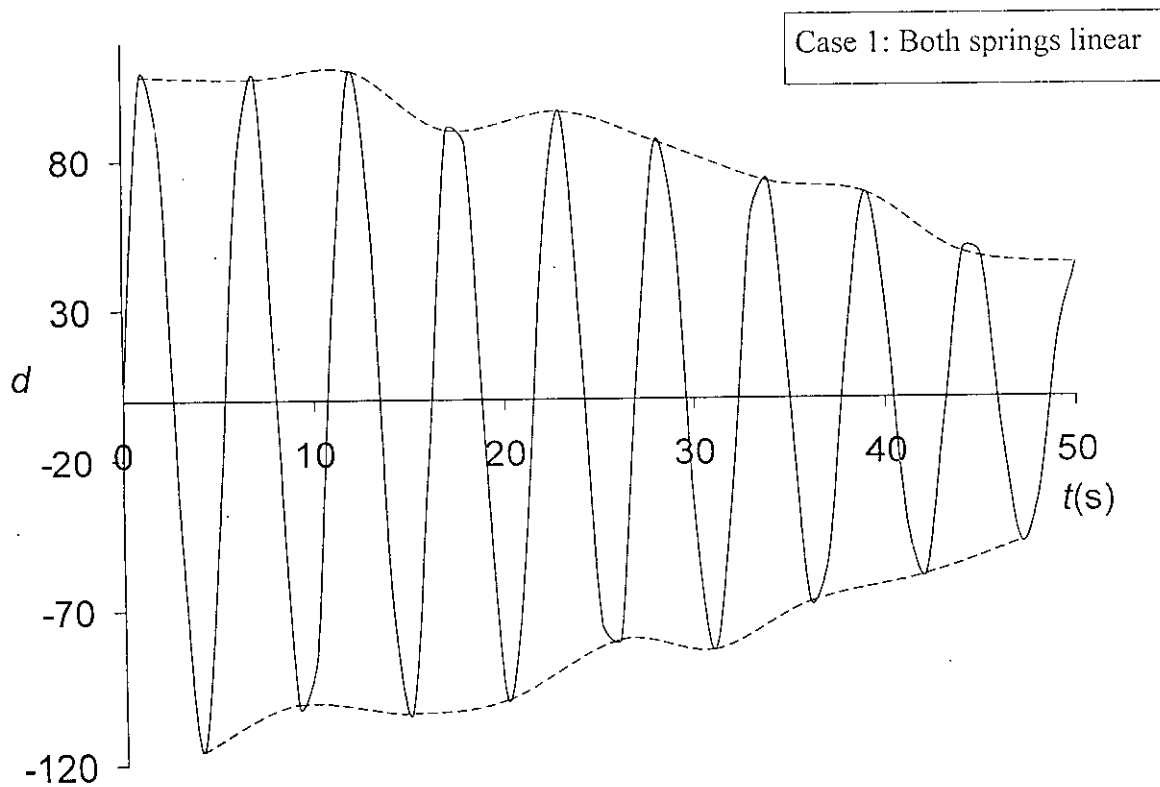


Fig. 10 Non-dimensional displacement - time for case 1 at  $r_2 = 1.25$

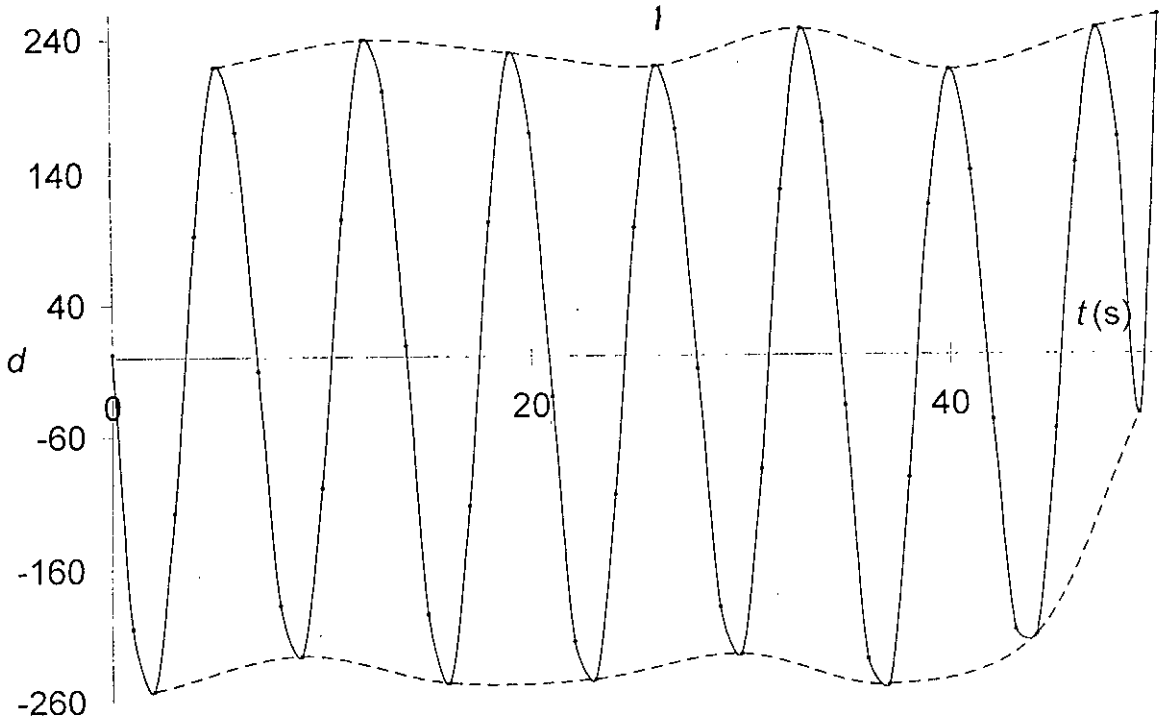


Fig. 11 Non-dimensional displacement - time for case 2 (both springs hard) at  $r_1 = 0.8$

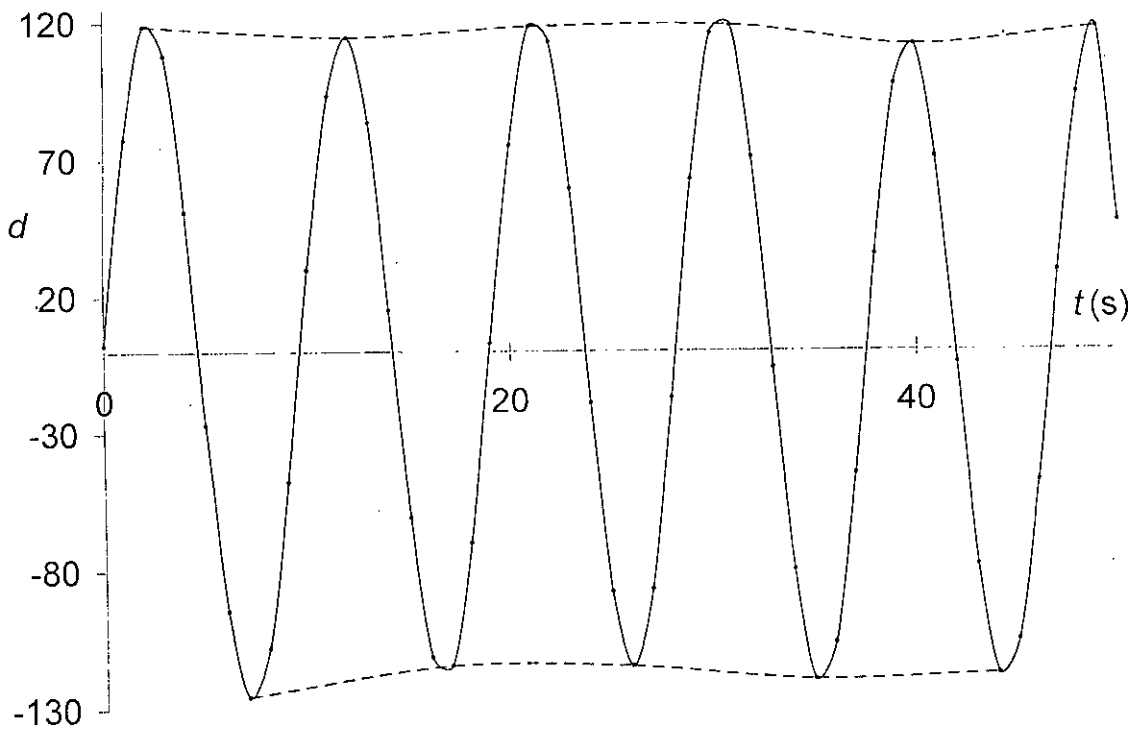


Fig. 12 Non-dimensional displacement - time ( $t$ ) for case 2 (both springs hard) at  $r_2 = 1.25$ .

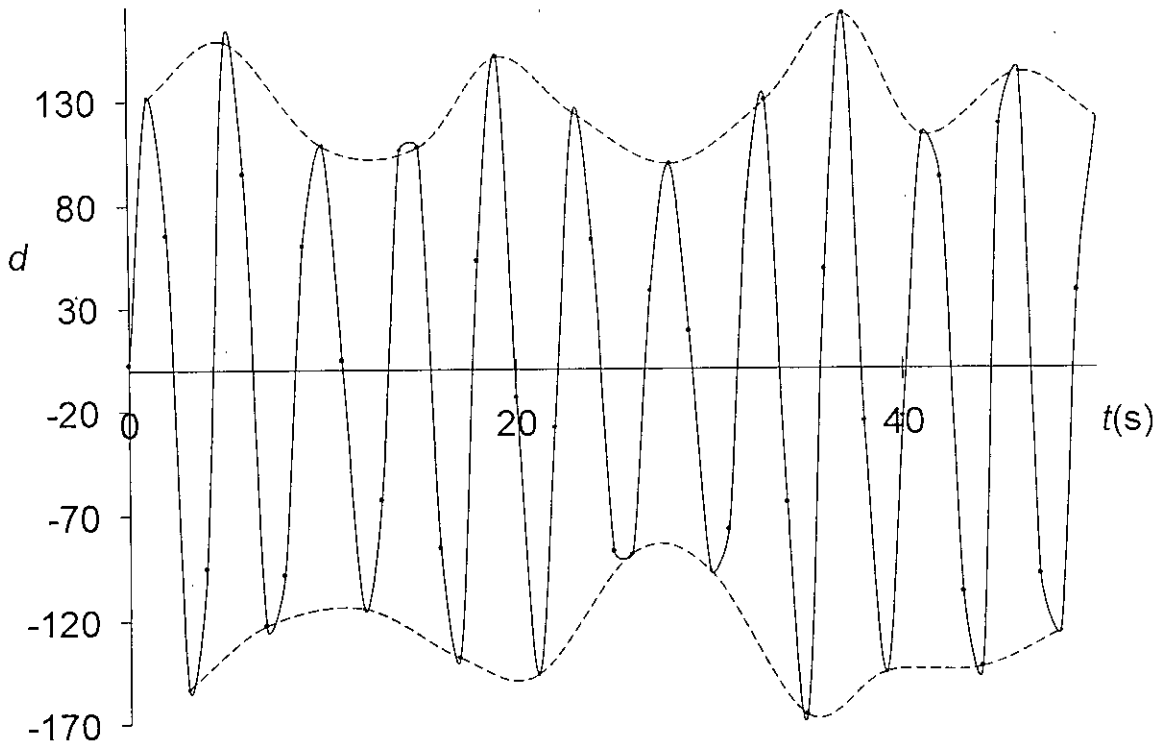


Fig. 13 Non-dimensional displacement - time for case 3 (1<sup>st</sup> spring hard & 2<sup>nd</sup> spring soft) at  $r_1 = 0.8$ .

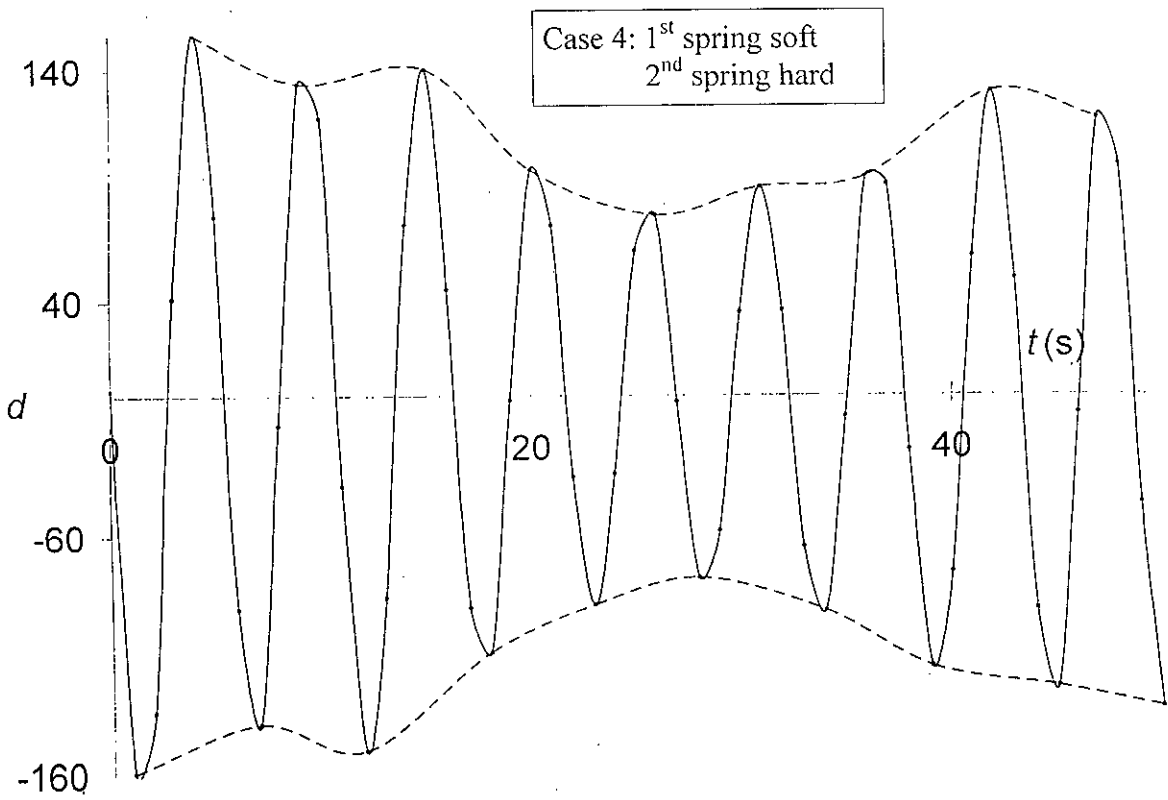


Fig. 14 Non-dimensional displacement - time for case 4 at  $r_1 = 0.8$

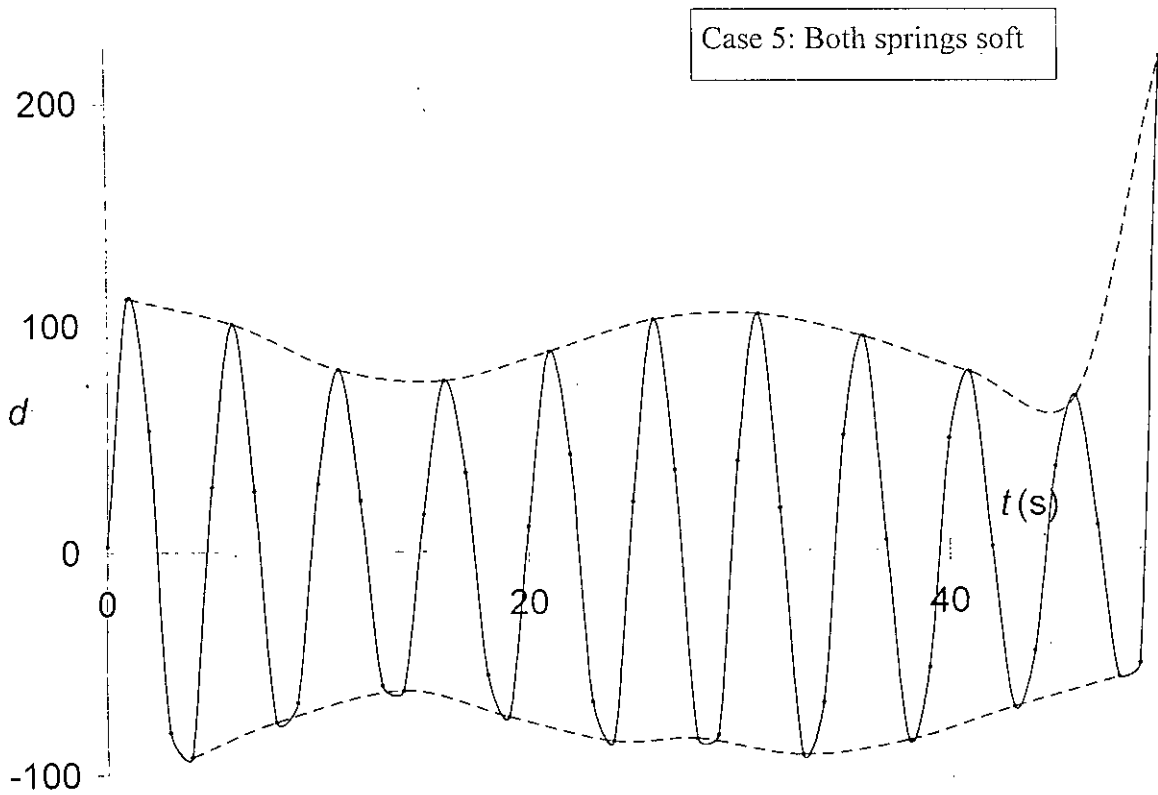


Fig. 15 Non-dimensional displacement of - time for case 5 at  $r_1 = 0.8$

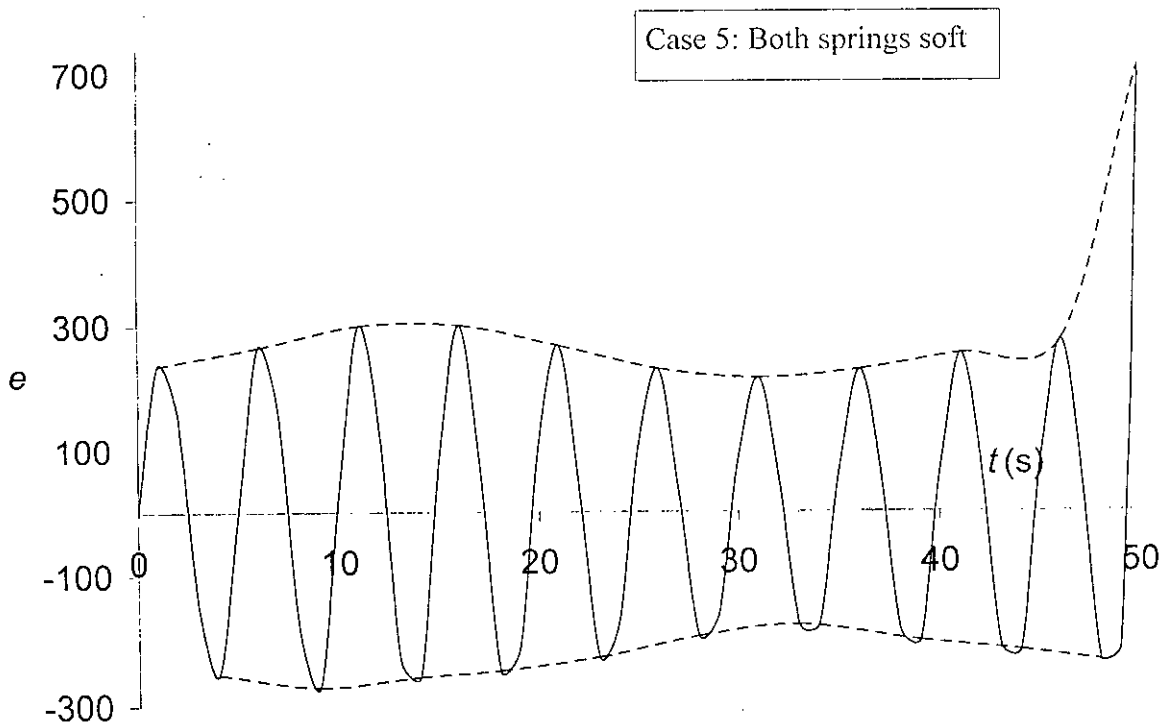


Fig. 16 Non-dimensional displacement of absorber mass - time for case 5 at  $r_1 = 0.8$ .

FIGURES FOR  
UNTUNED ABSORBERS (CASE 6)  
SOLVED AS INITIAL AND BOUNDARY  
VALUE PROBLEMS (FIGS. 17 – 19)

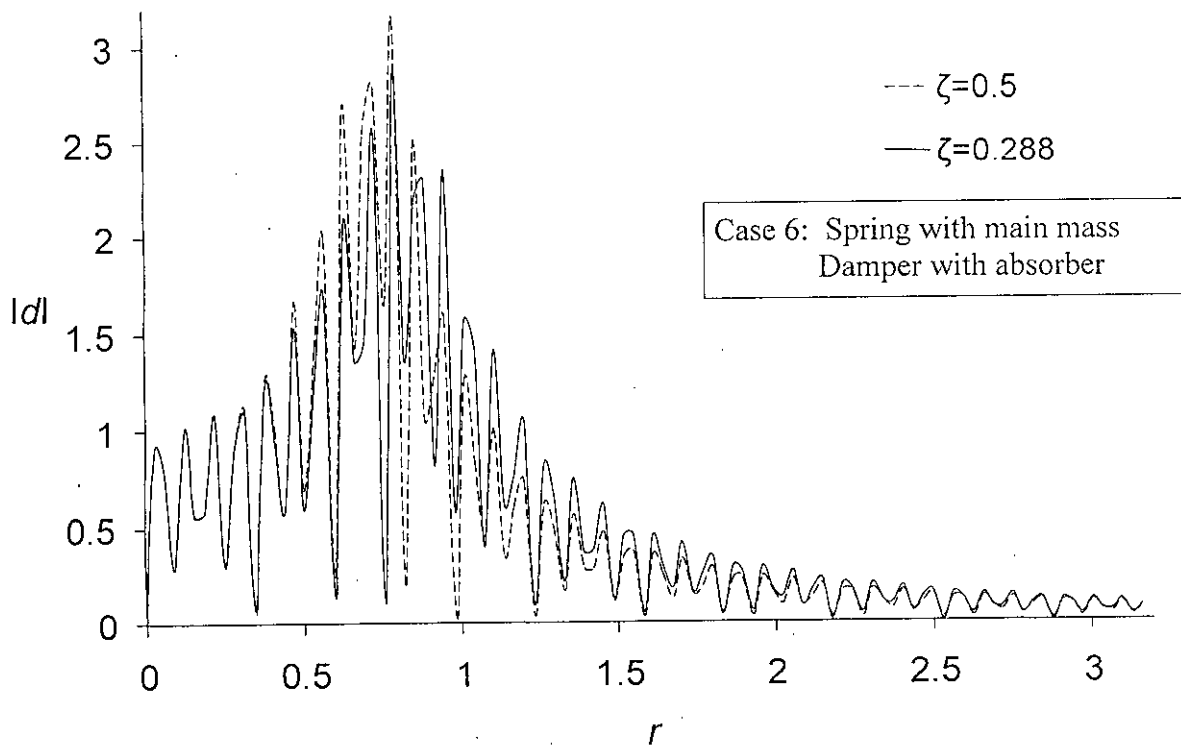
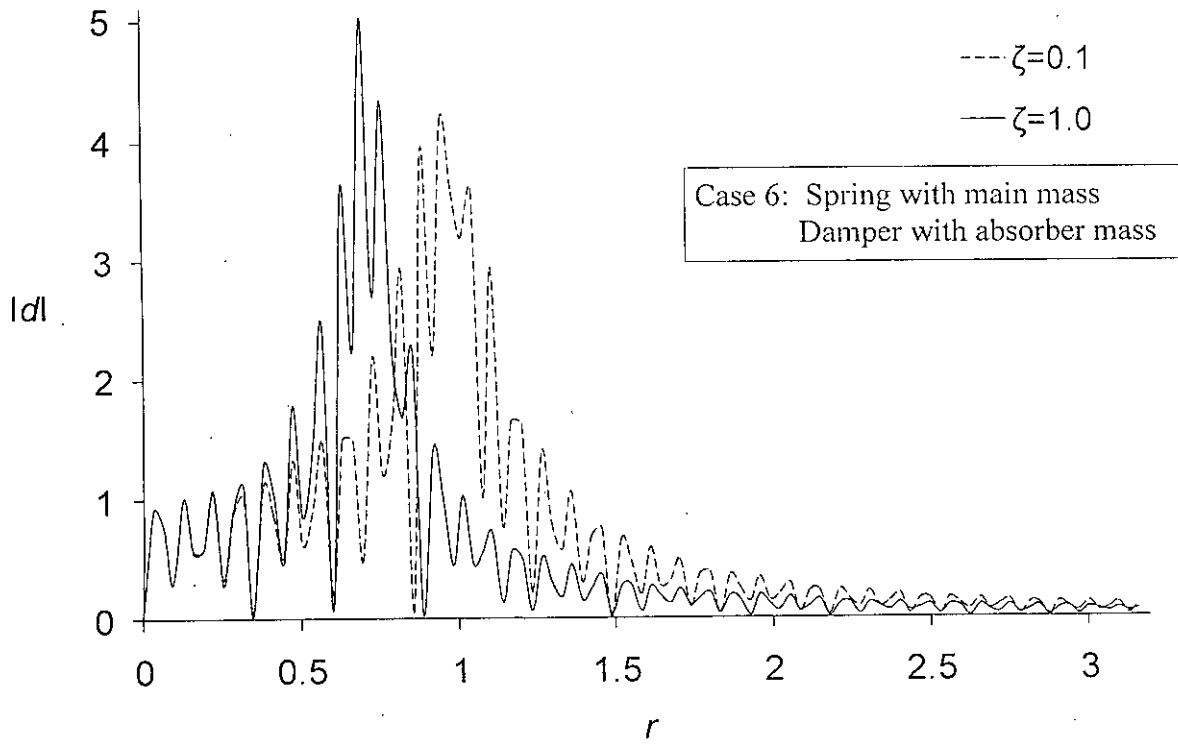


Fig 17.  $|d| - r$  for case 6 at  $t=20s$  having  $\mu=1.0$  and solved as initial value problem (a)  $\zeta=0.1, 1.0$ . (b)  $\zeta=0.288, 0.5$ .

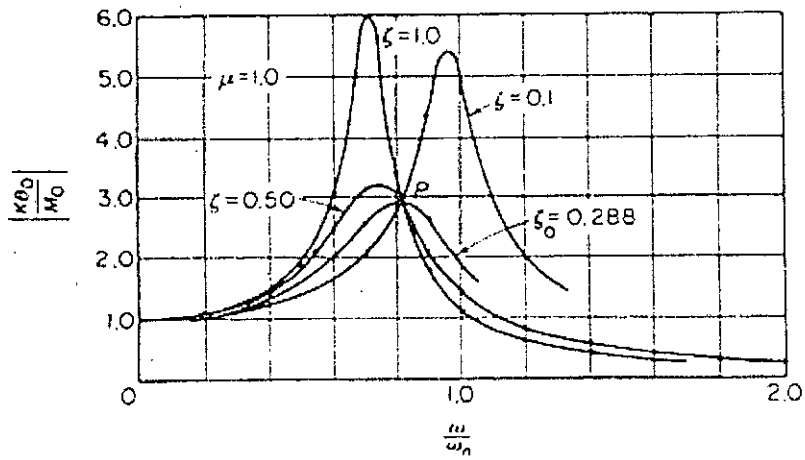


Figure 5.8-3. Response of an untuned viscous damper (all curves pass through  $P$ ).

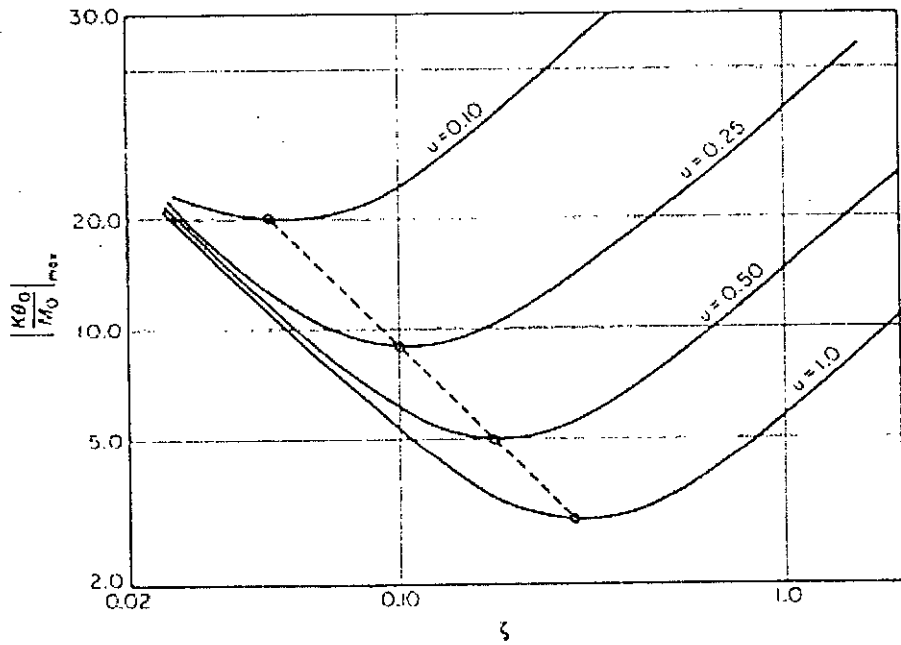


Figure 5.8-4.

107279

(c)

Fig 17 (c) Exact solution given in Thomson (1981) for untuned dampers / absorbers.



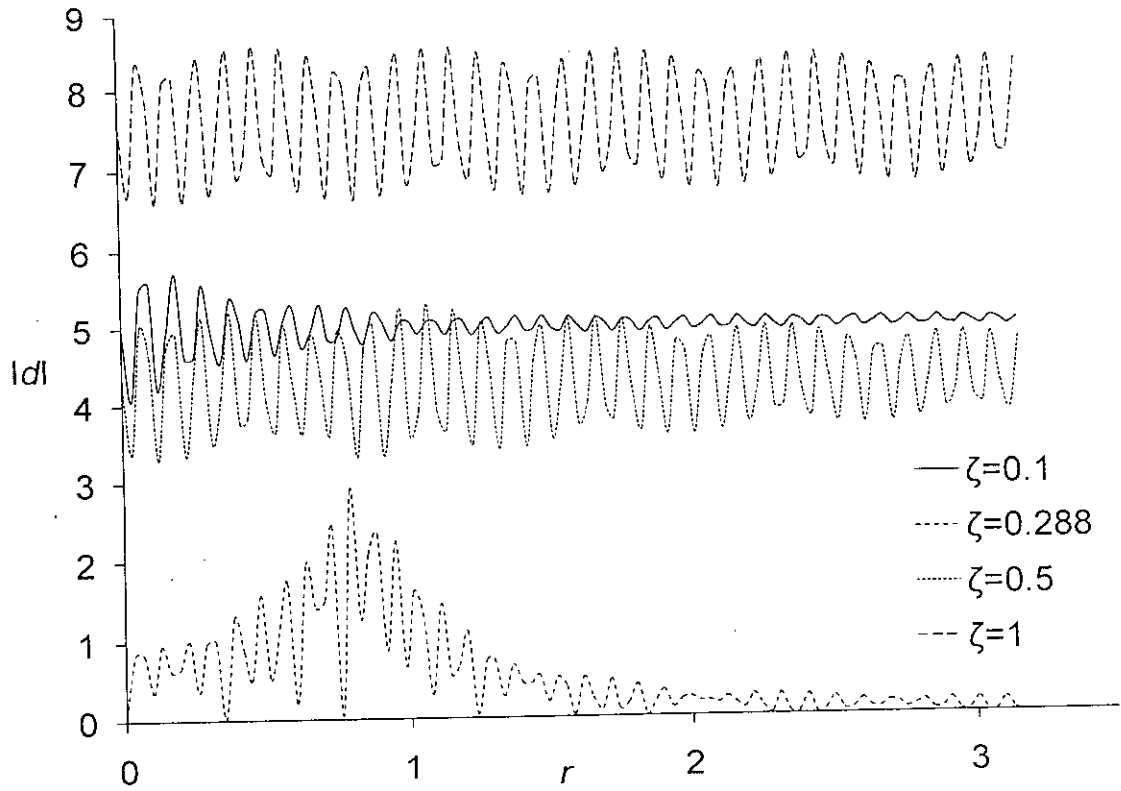


Fig. 18  $|d| - r$  for case 6 (spring with main mass & damper with absorber mass) at  $t=20s$  having  $\mu=1.0$  and solved as boundary value problem.

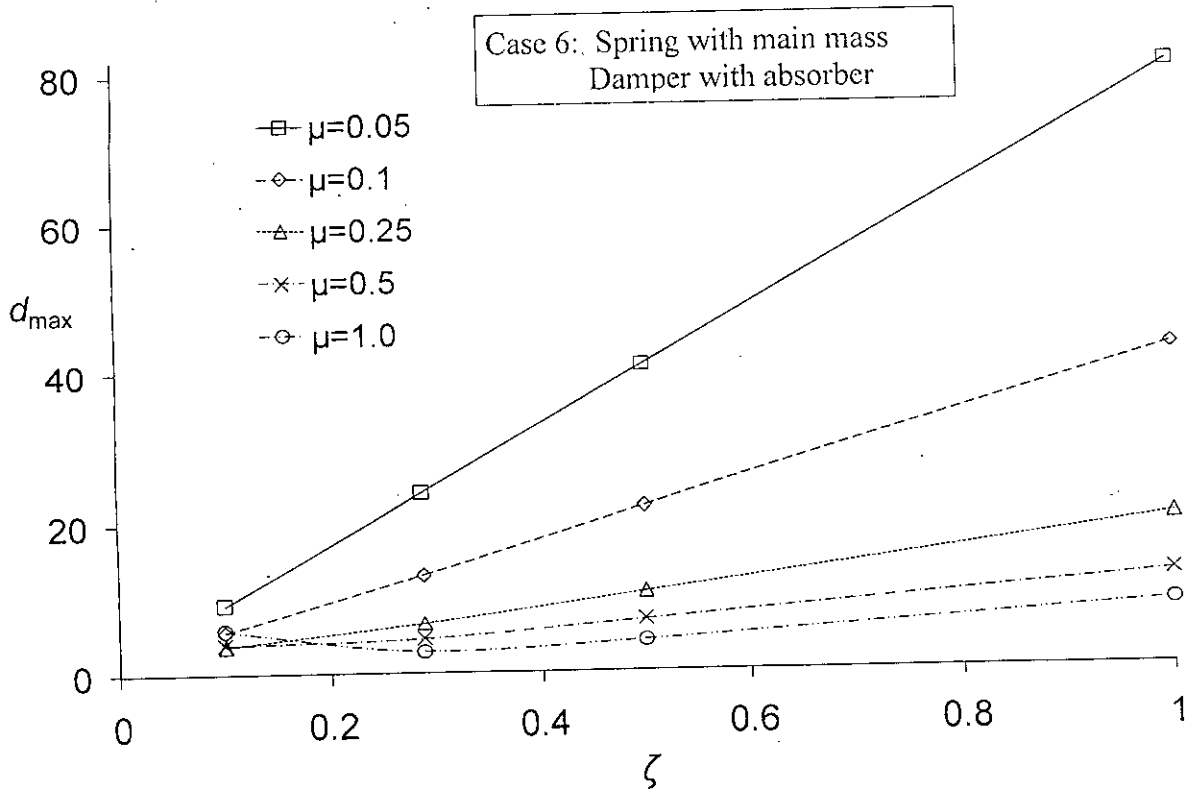


Fig. 19  $d_{\max} - \zeta$  with varying mass ratio ( $\mu$ ) for case 6 and solved as boundary value problem at  $t=20s$ .

FIGURES FOR  
UNTUNED ABSORBERS  
USING DATA SET # 1 (FIGS. 20 – 29)

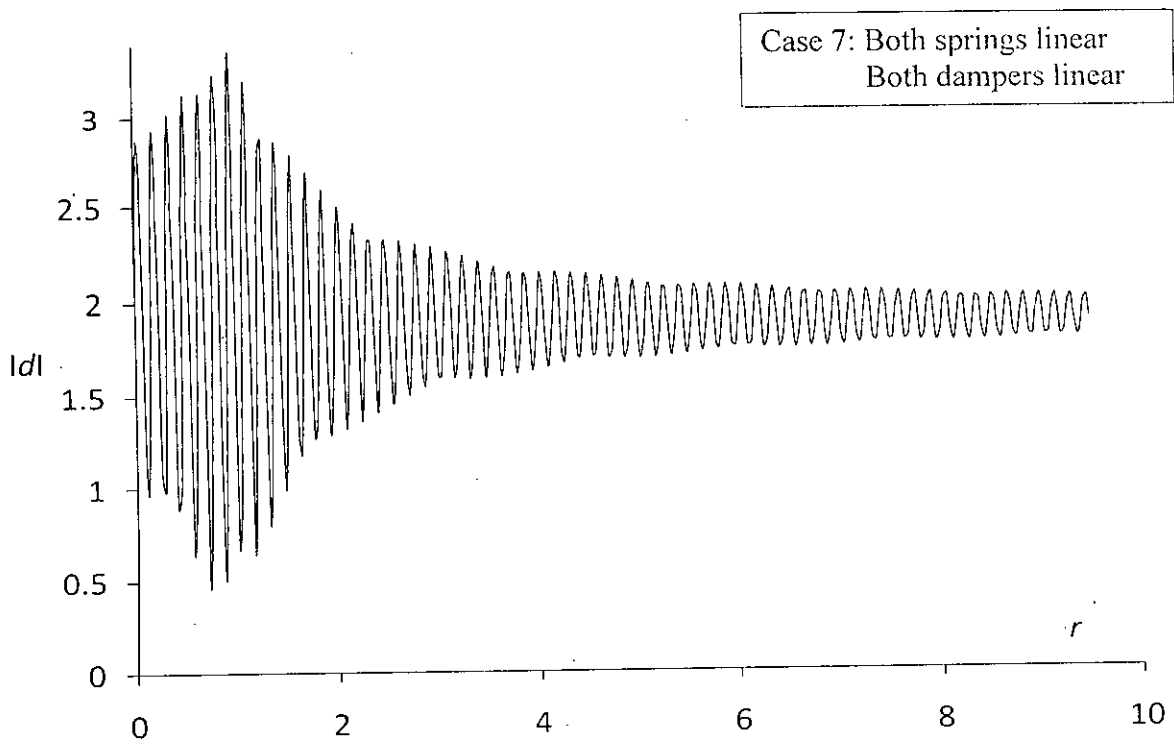


Fig. 20  $|d| - r$  for case 7 having  $\mu=0.01$  and  $\zeta=0.00496$  at  $t=50s$  and solved as boundary value problem.

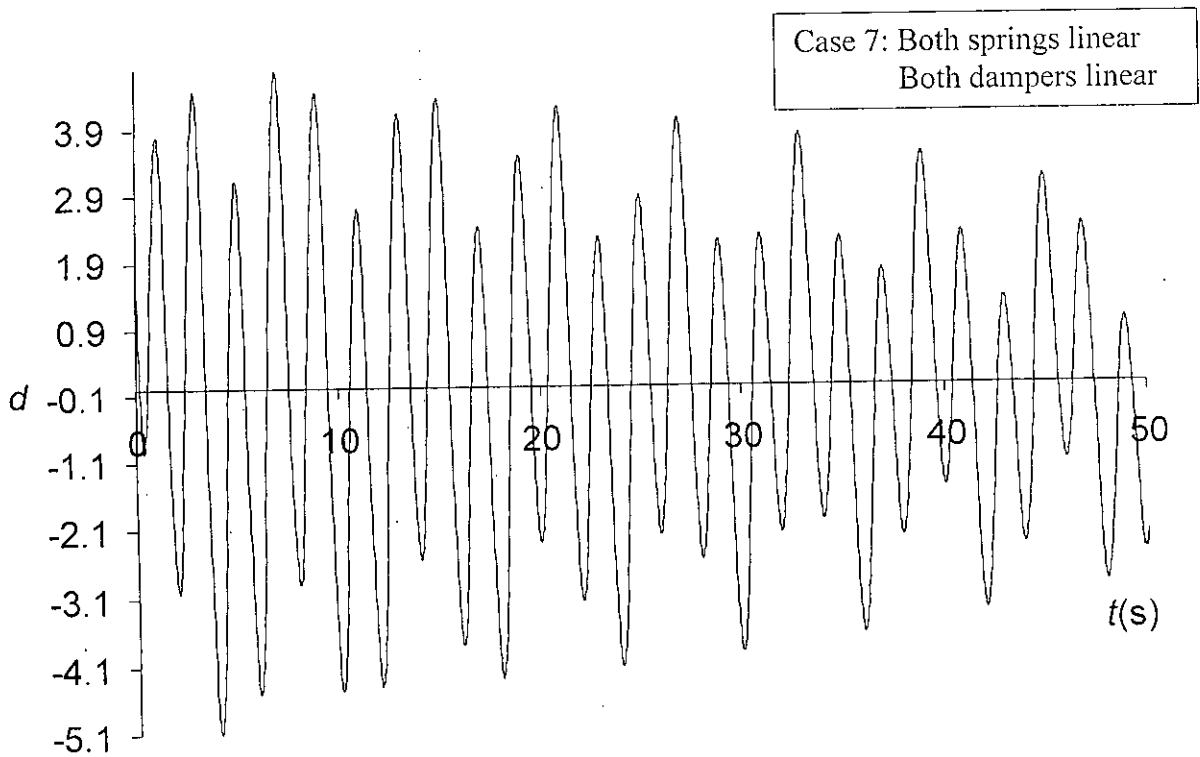


Fig. 21  $d - t$  for case 7 in the system with  $\mu=0.01$  and  $\zeta=0.00496$  at  $r=0.3162$ .

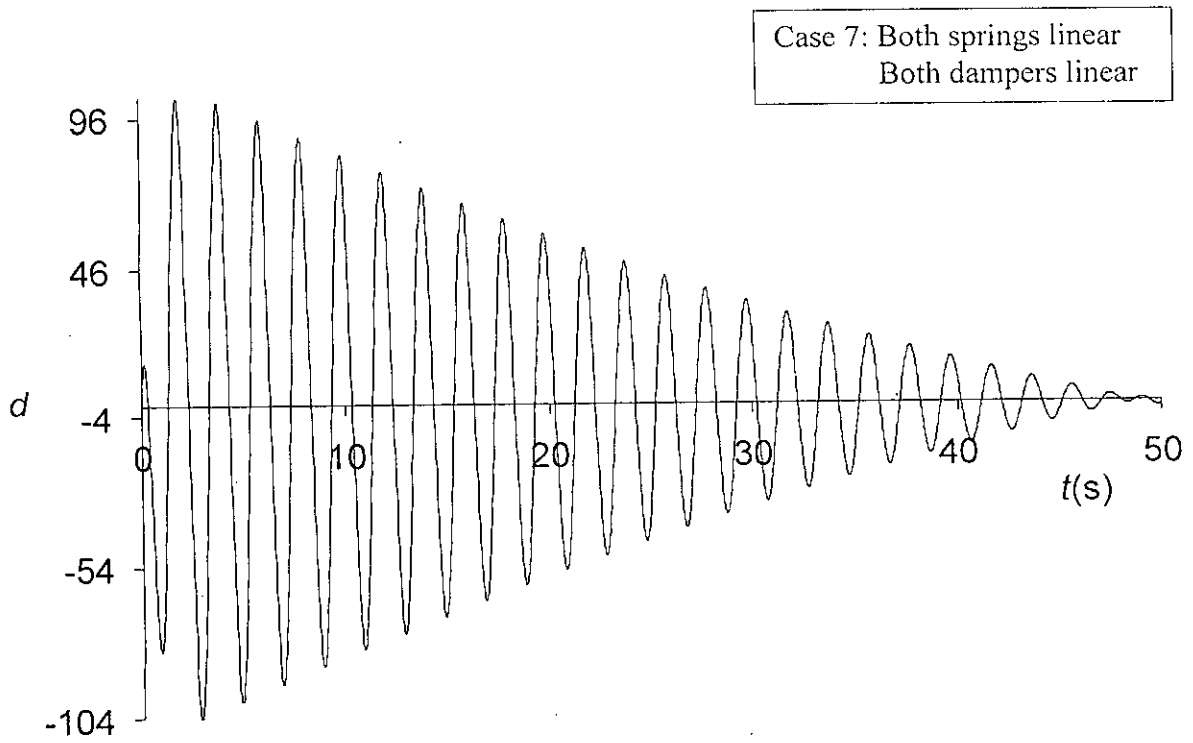


Fig. 22  $d - t$  for case 7 with  $\mu=0.01$ ,  $\zeta=0.00496$  and  $r=1.0$ .

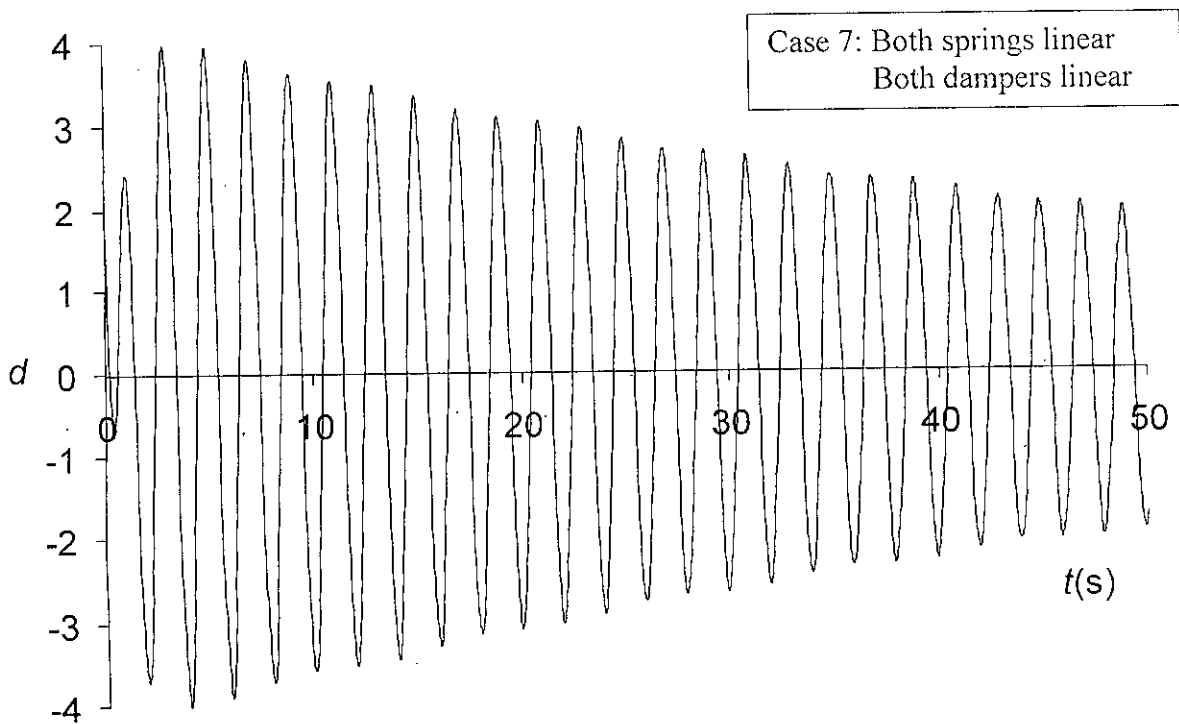


Fig. 23  $d - t$  for case 7 with  $\mu=0.01$ ,  $\zeta=0.00496$  and  $r=4.744$ .

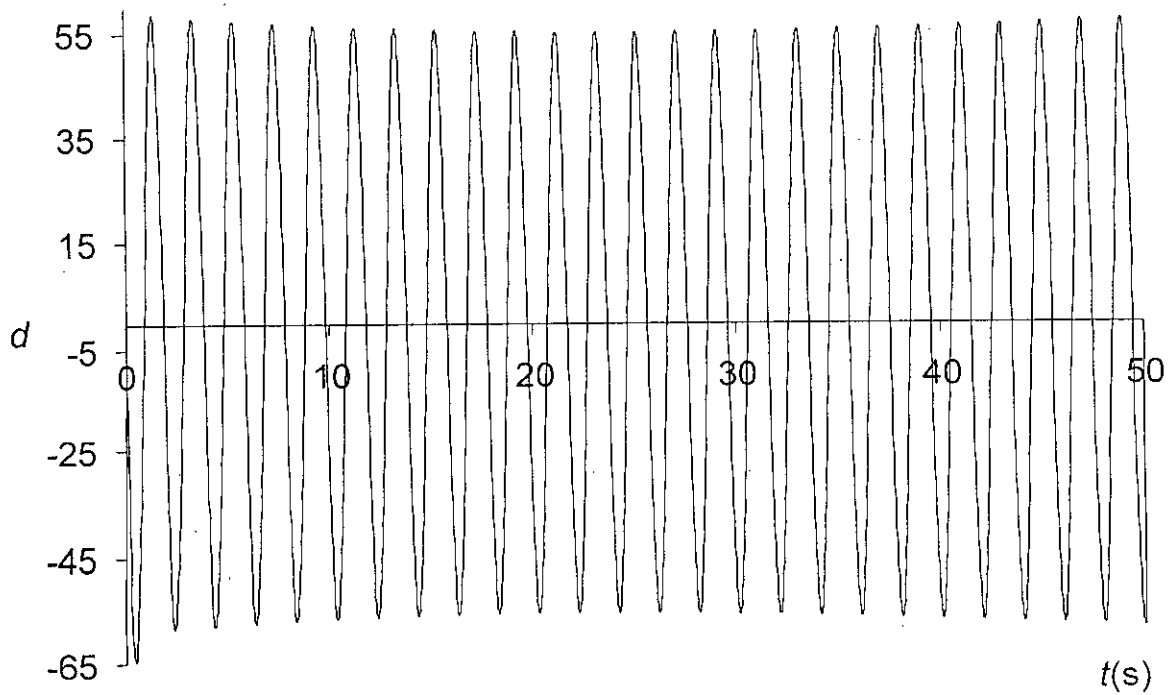


Fig. 24  $d - t$  for case 8 (Both springs hard & Linear dampers) with  $\mu=0.01$ ,  $\zeta=0.00496$  and  $r=1.0$ .

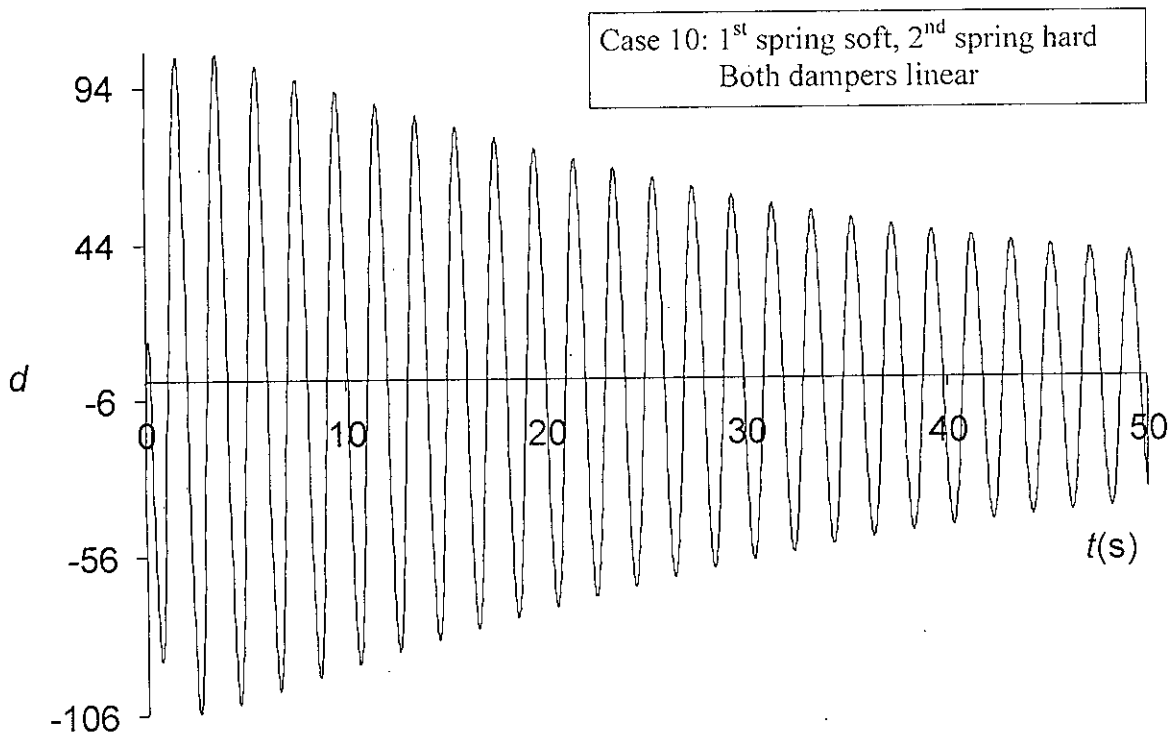


Fig. 25  $d - t$  for case 10 with  $\mu=0.01$ ,  $\zeta=0.00496$  and  $r=1.0$ .

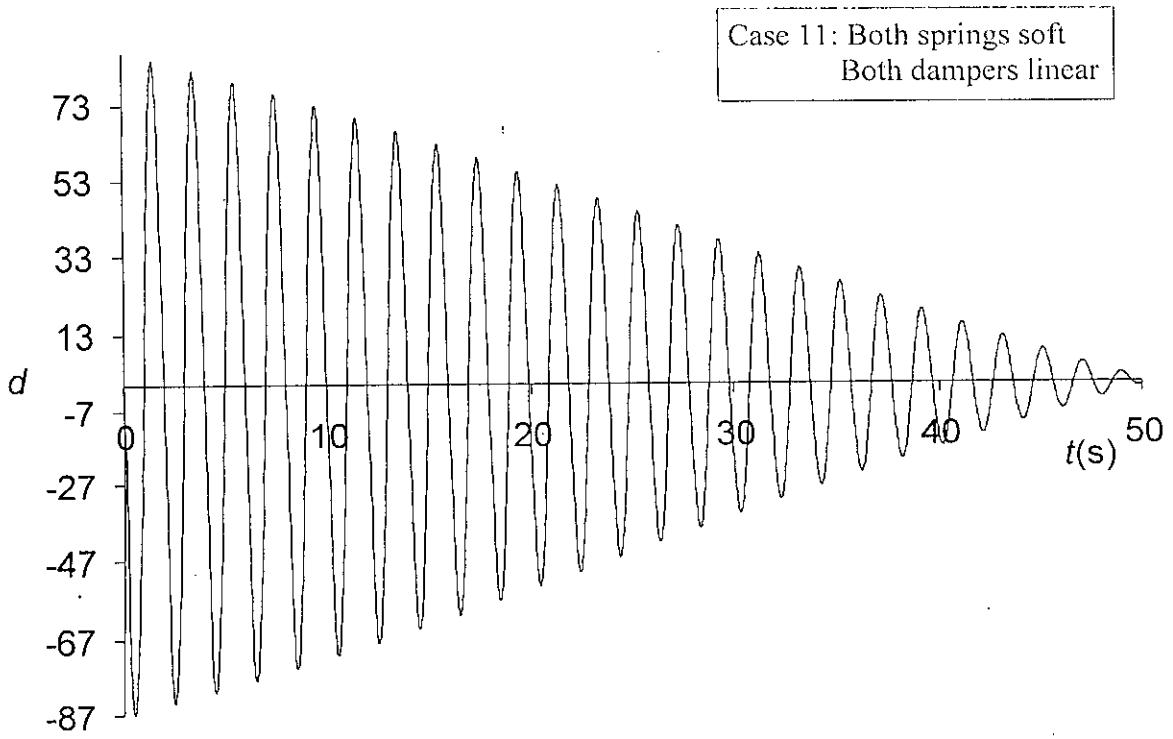


Fig. 26  $d-t$  for case 11 with  $\mu=0.01$ ,  $\zeta=0.00496$  and  $r=1.012$ .

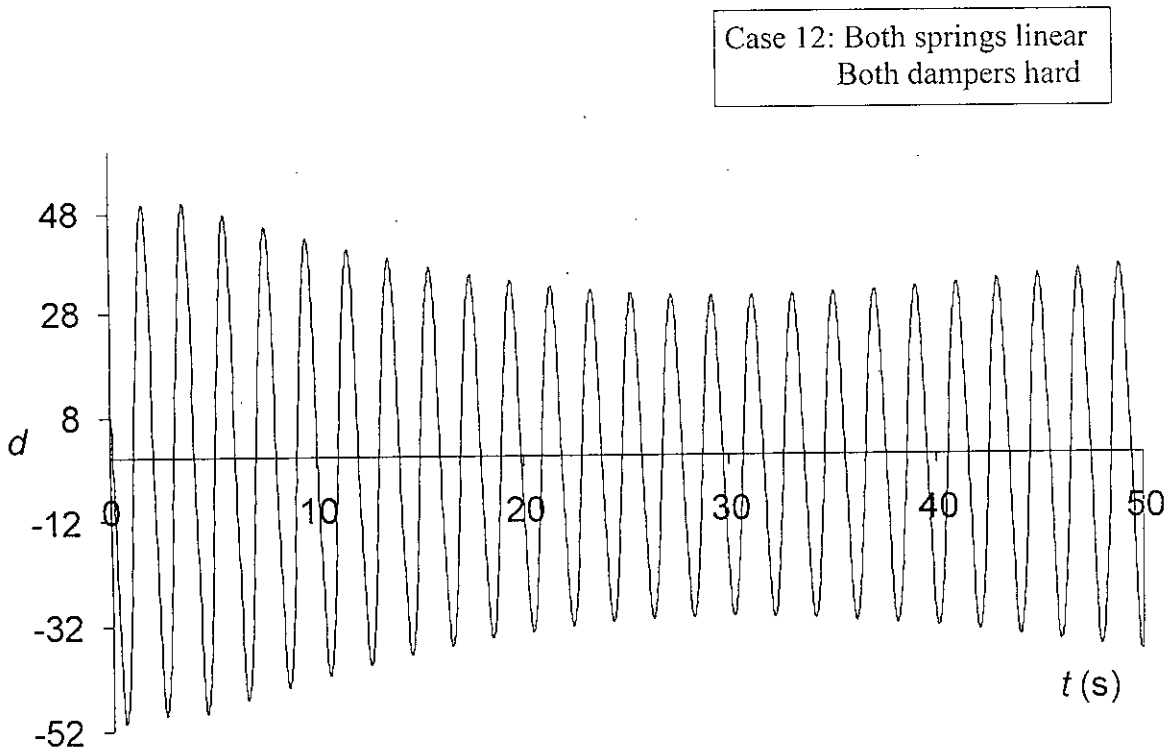


Fig. 27  $d-t$  for case 12 with  $\mu=0.01$ ,  $\zeta=0.00496$  and  $r=1.0$ .

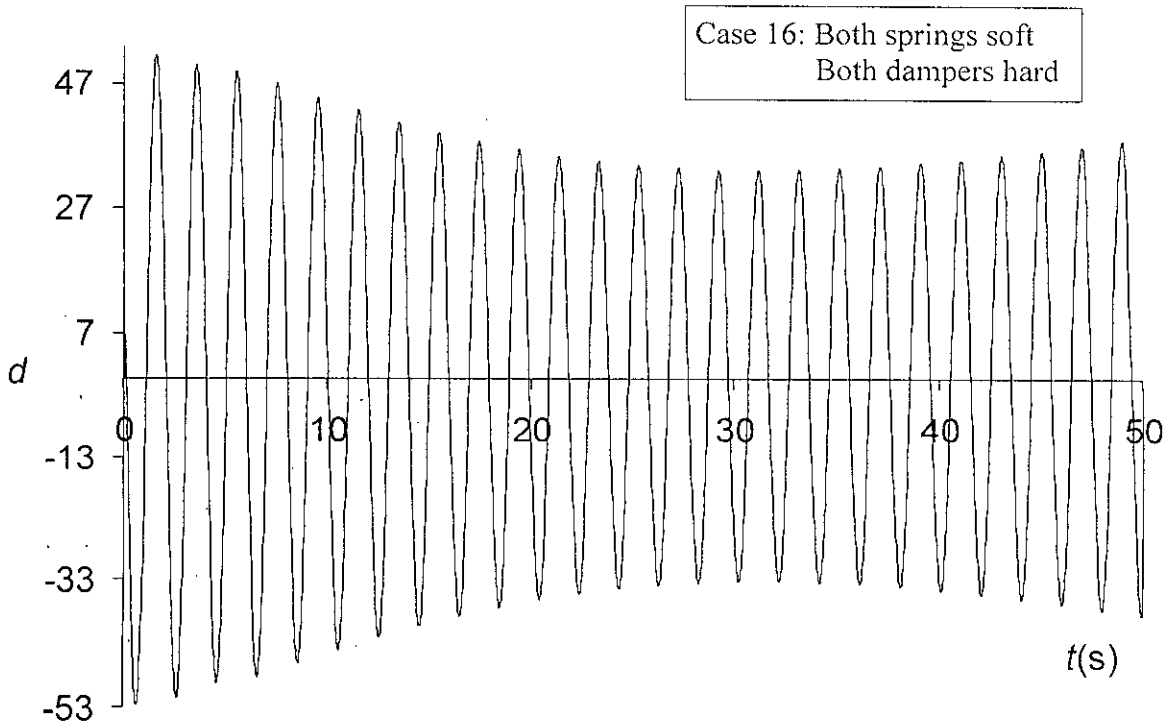


Fig. 28  $d - t$  for case 16 with  $\mu=0.01$ ,  $\zeta=0.00496$  and  $r=1.0$ .

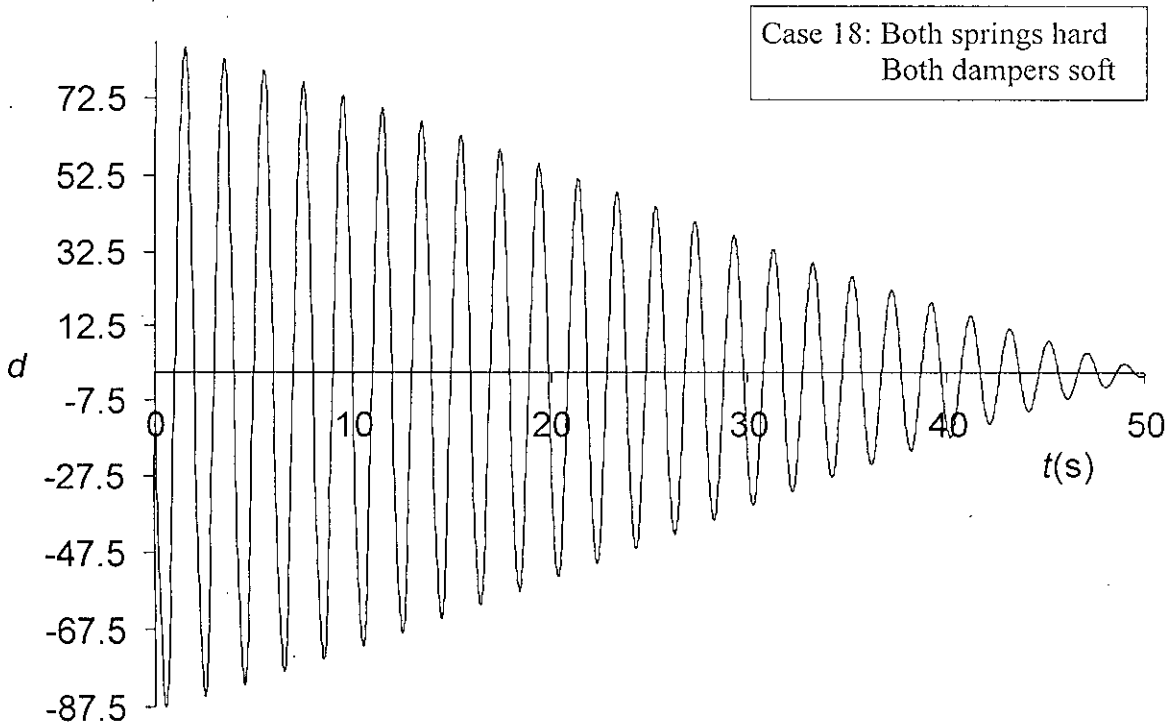


Fig. 29  $d - t$  for case 18 with  $\mu=0.01$ ,  $\zeta=0.00496$  and  $r=1.012$ .

FIGURES FOR  
UNTUNED ABSORBERS  
USING DATA SET # 2 (FIGS. 30 – 46)



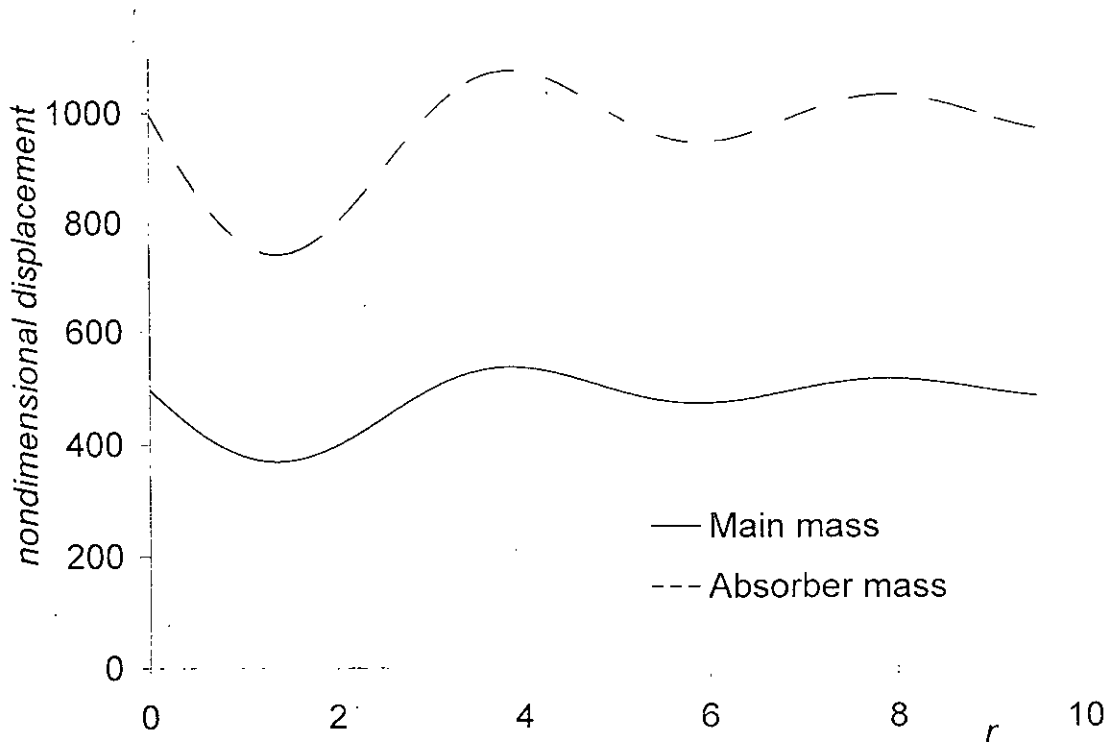


Fig. 30 Non-dimensional displacement vs. frequency ratio at  $t=0.5s$  and  $\zeta=0.0158$  for case 7 (both springs linear & both dampers linear).

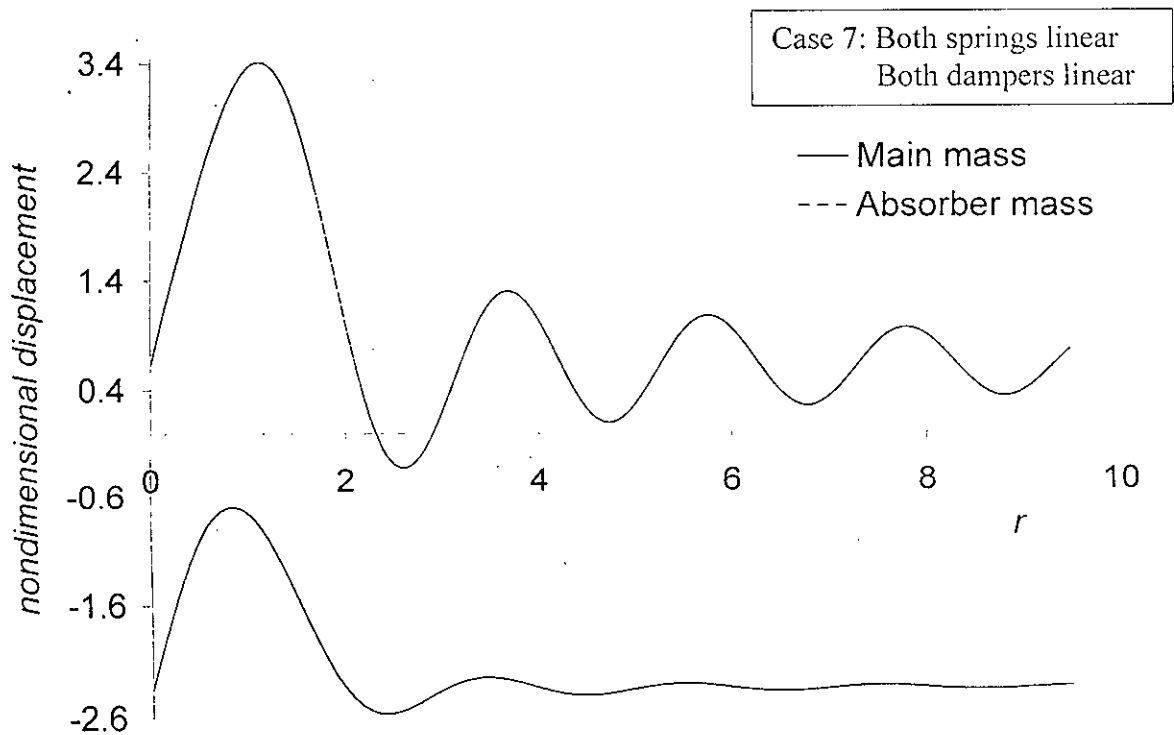


Fig. 31 Non-dimensional displacement vs. frequency ratio at  $t=1.0s$  and  $\zeta=0.0158$  for case 7.

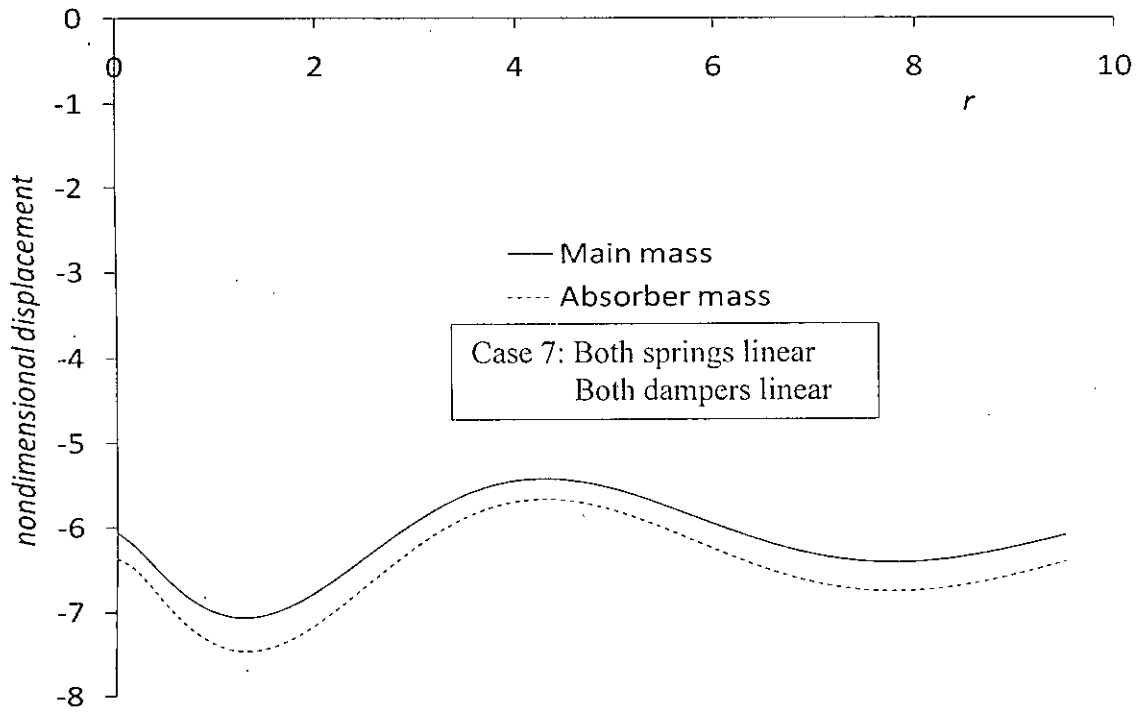


Fig. 32 Non-dimensional displacement vs. frequency ratio at  $t=20.0s$  and  $\zeta=0.0158$  for case 7.

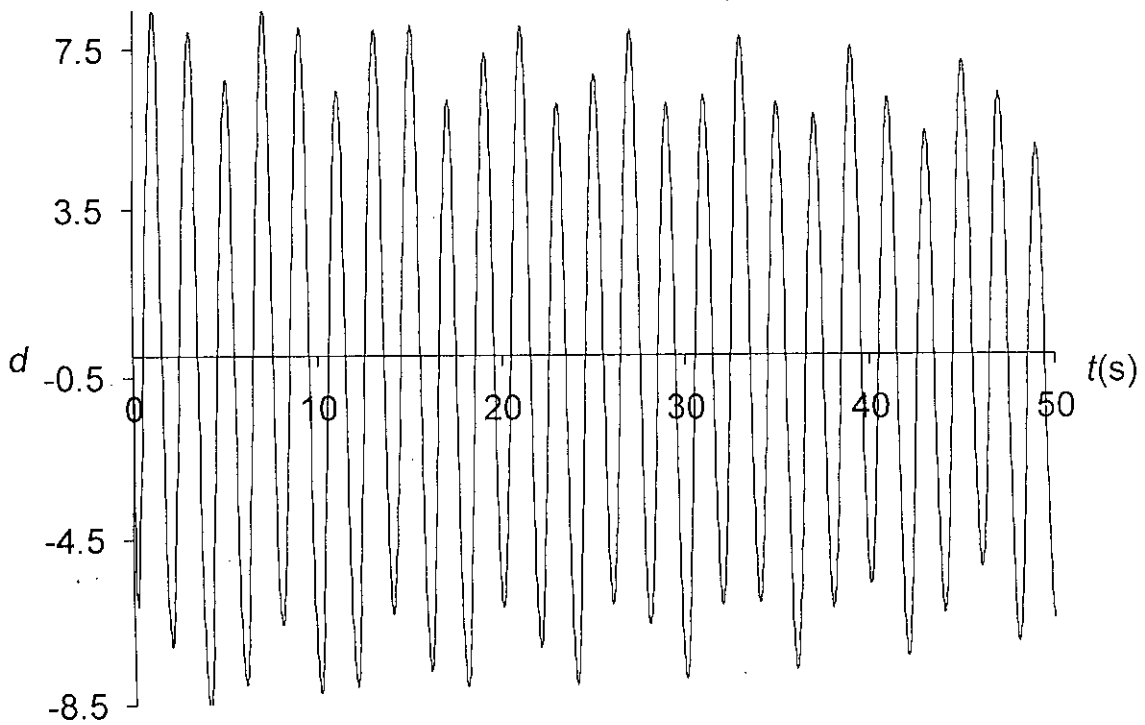


Fig. 33  $d-t$  at  $r=0.3162$  and  $\zeta=0.0158$  for case 7 (both springs linear & both dampers linear).

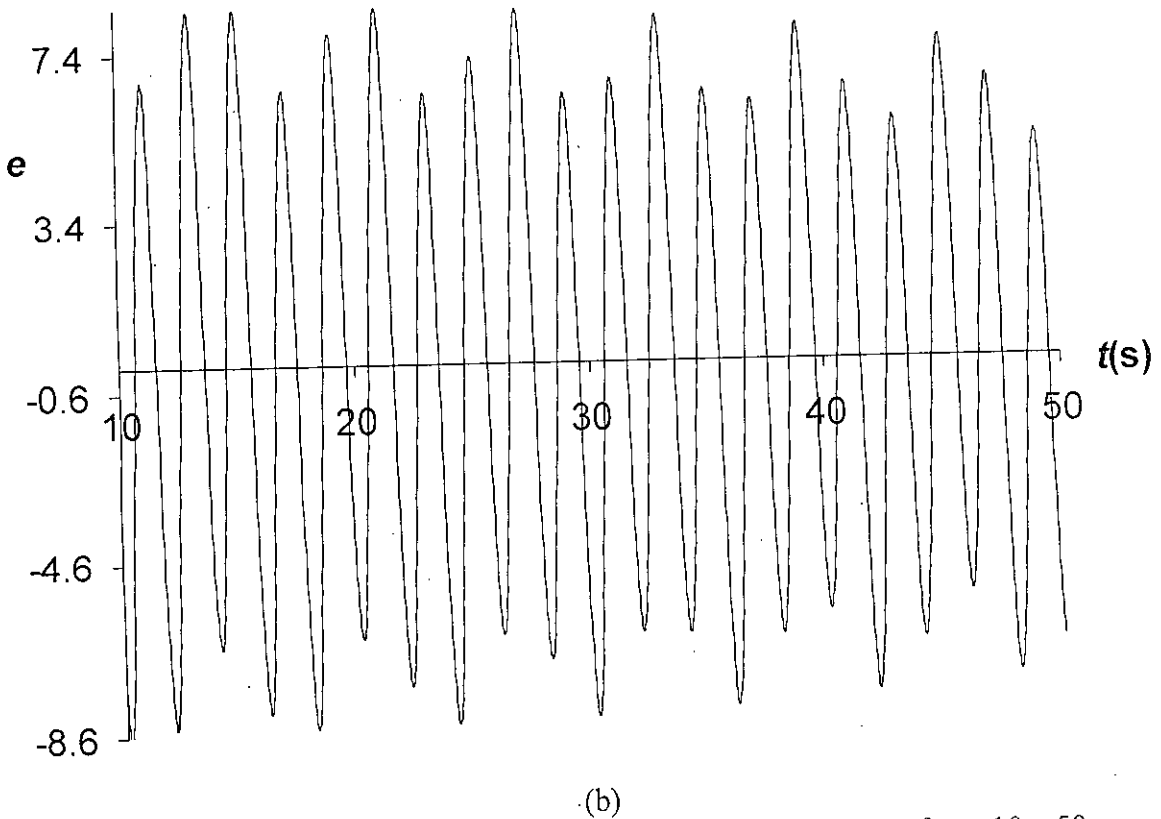
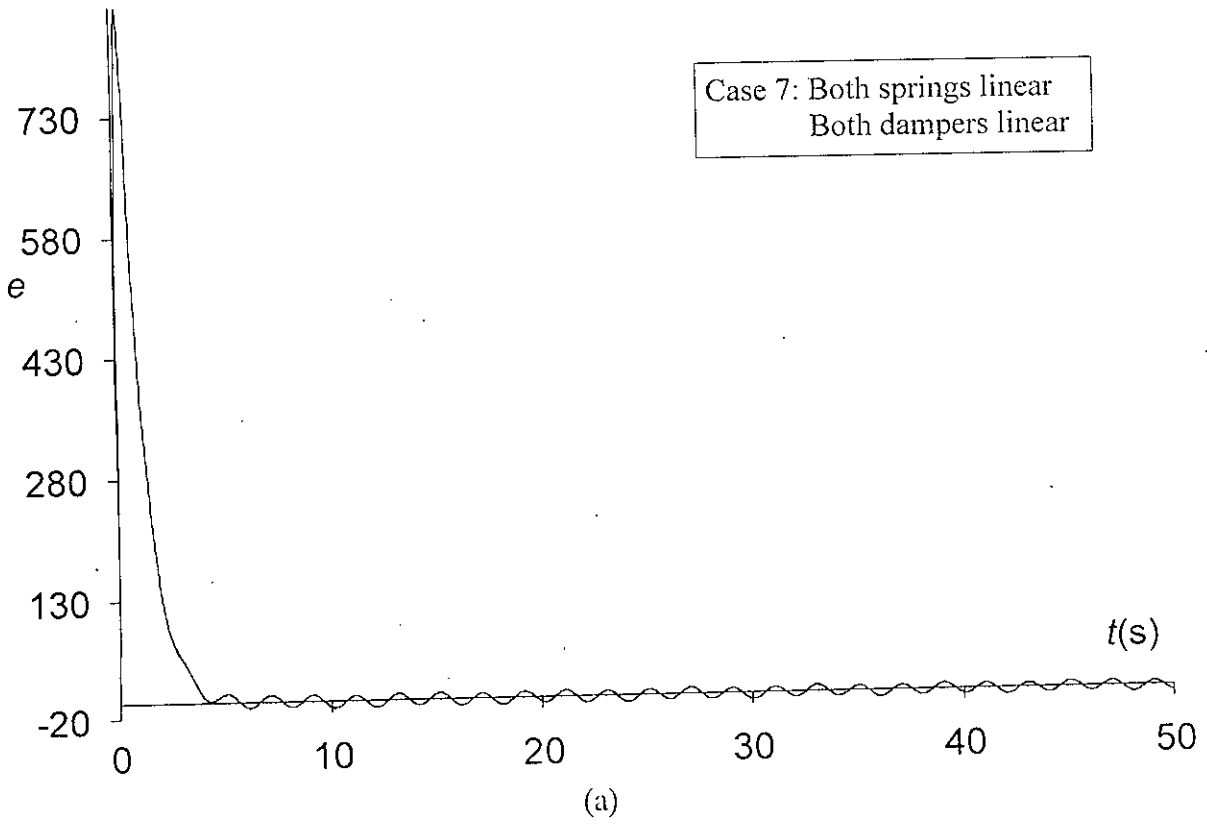


Fig 34 (a)  $e - t$  at  $r = 0.3162$  and  $\zeta = 0.0158$  for case 7 (b) enlarged view for  $t = 10 - 50$  s.

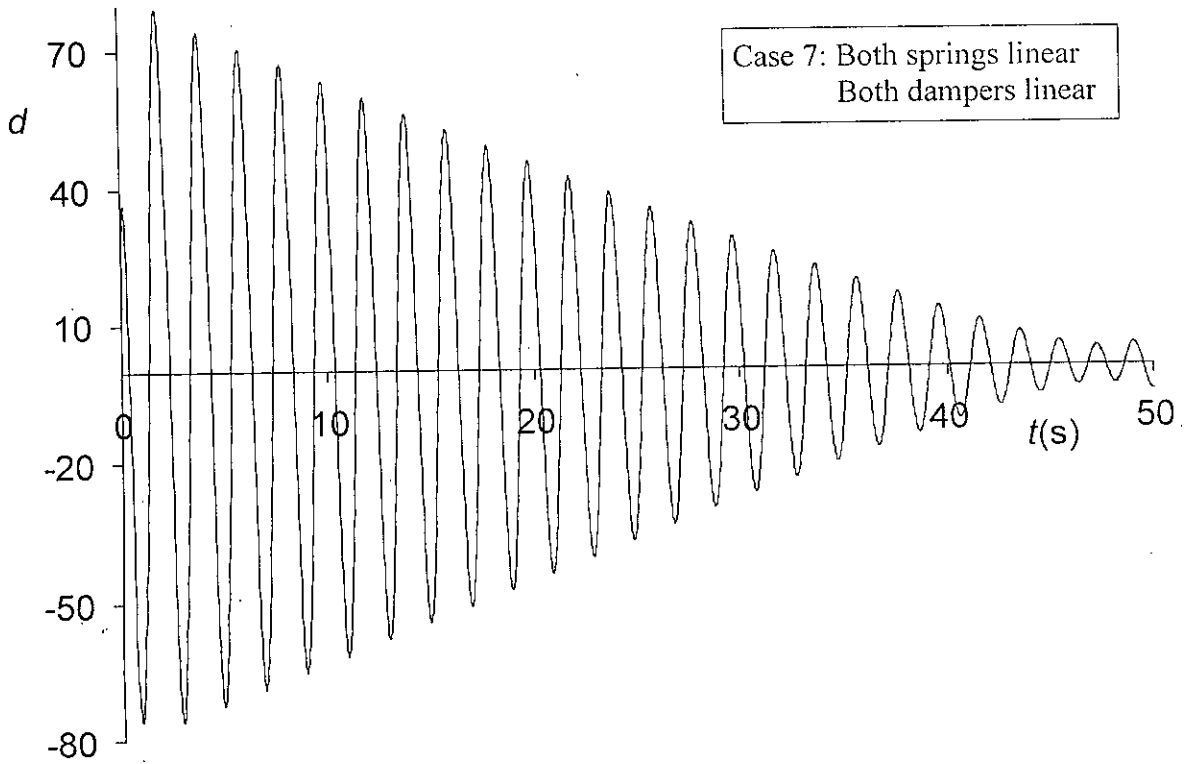


Fig. 35  $d - t$  at  $r = 1.0$  and  $\zeta = 0.0158$  for case 7.

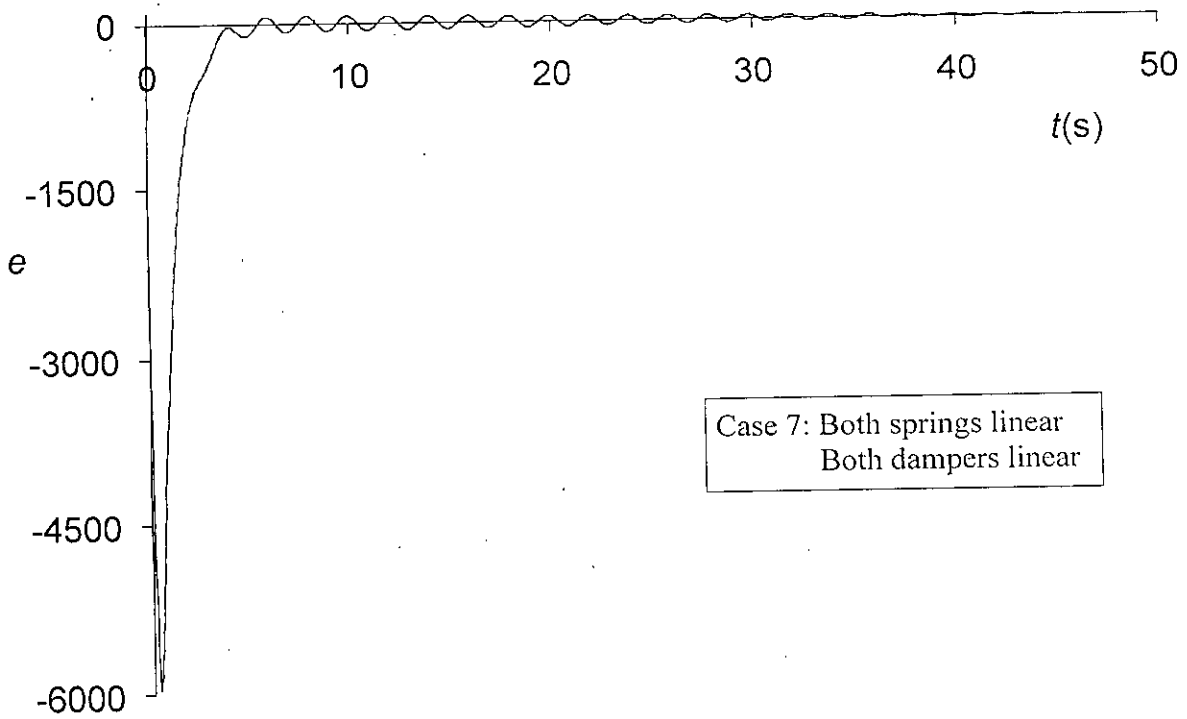


Fig. 36  $e - t$  at  $r = 1.0$  and  $\zeta = 0.0158$  for case 7.

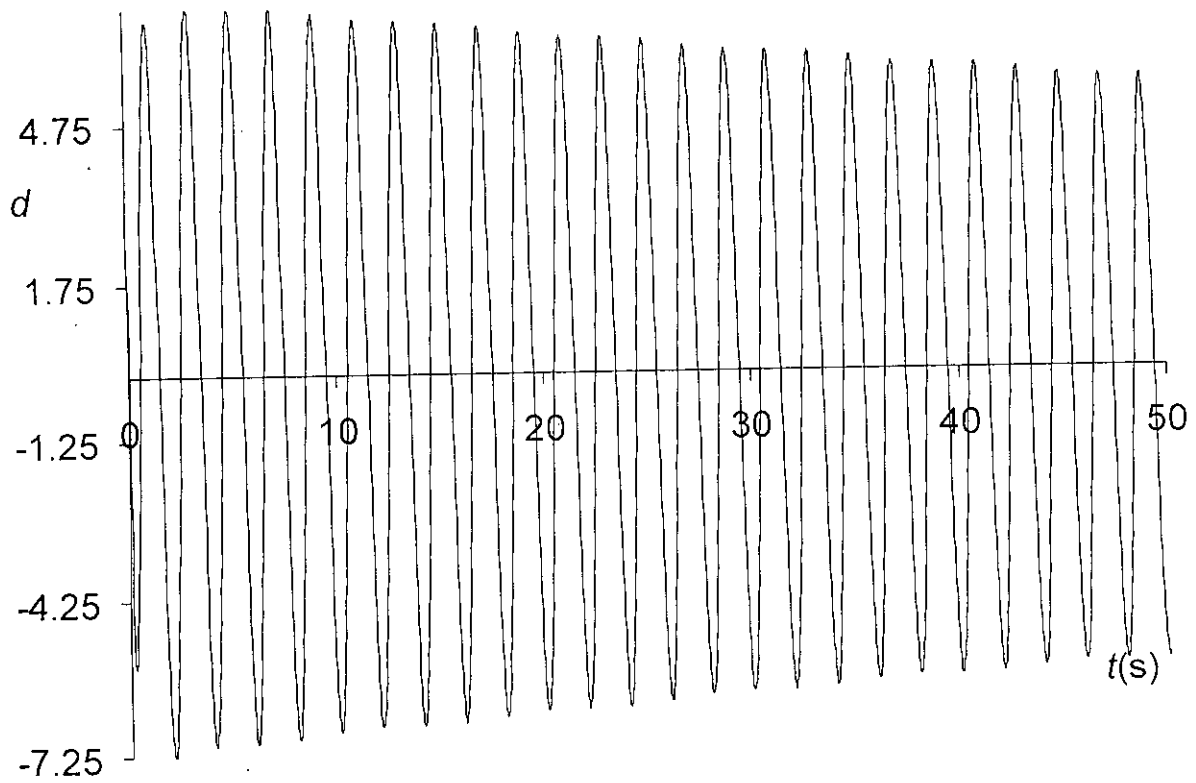


Fig. 37  $d-t$  at  $r = 4.744$  and  $\zeta = 0.0158$  for case 7 (both springs linear and both dampers linear).

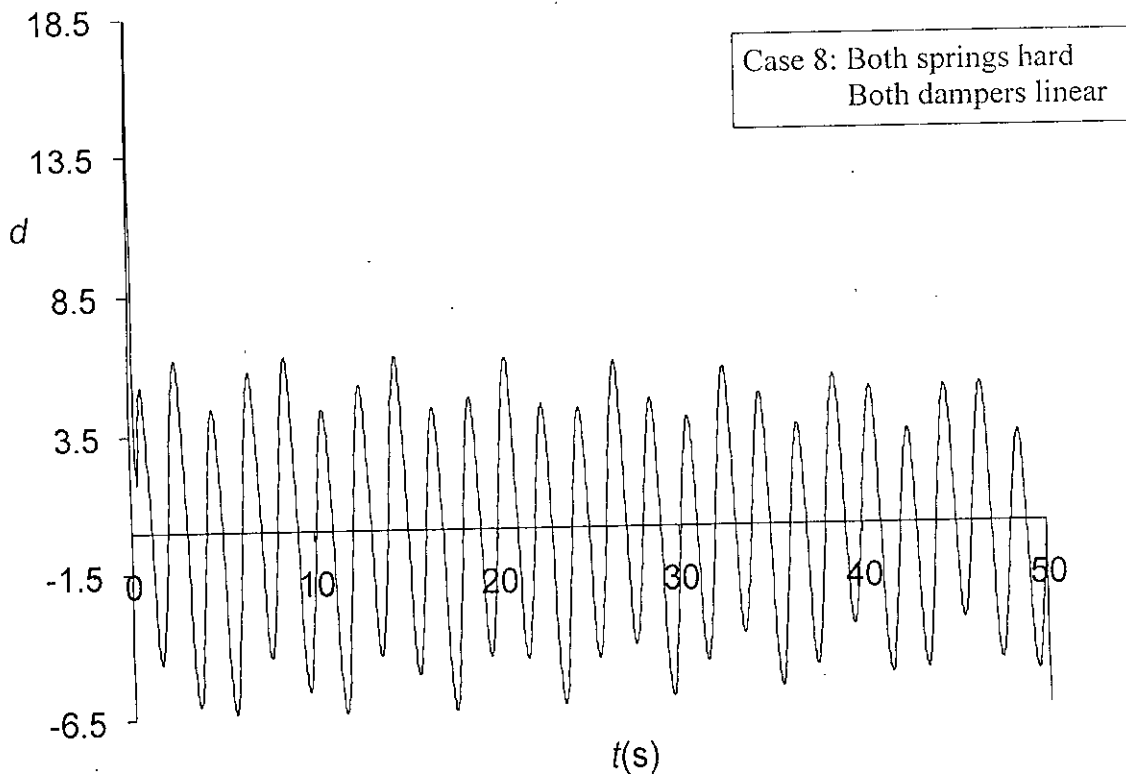


Fig. 38  $d-t$  for at  $r = 0.3162$  and  $\zeta = 0.0158$  for case 8.

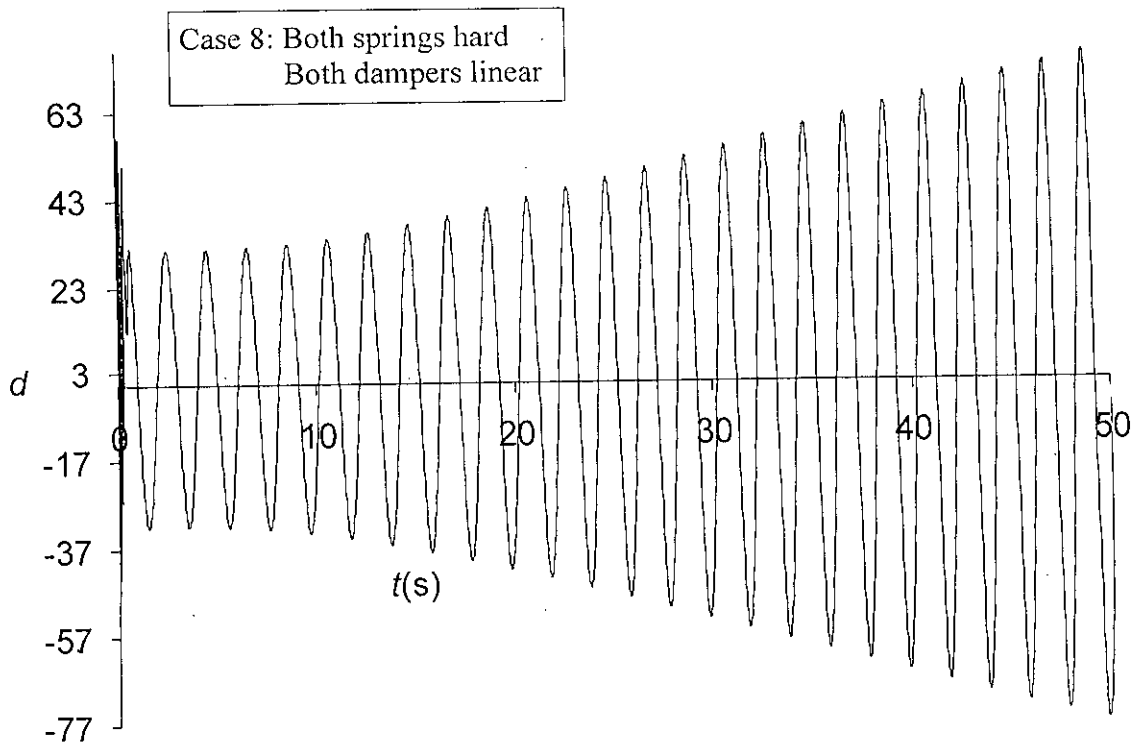


Fig. 39  $d - t$  at  $r=1.0$  and  $\zeta=0.0158$  for case 8.

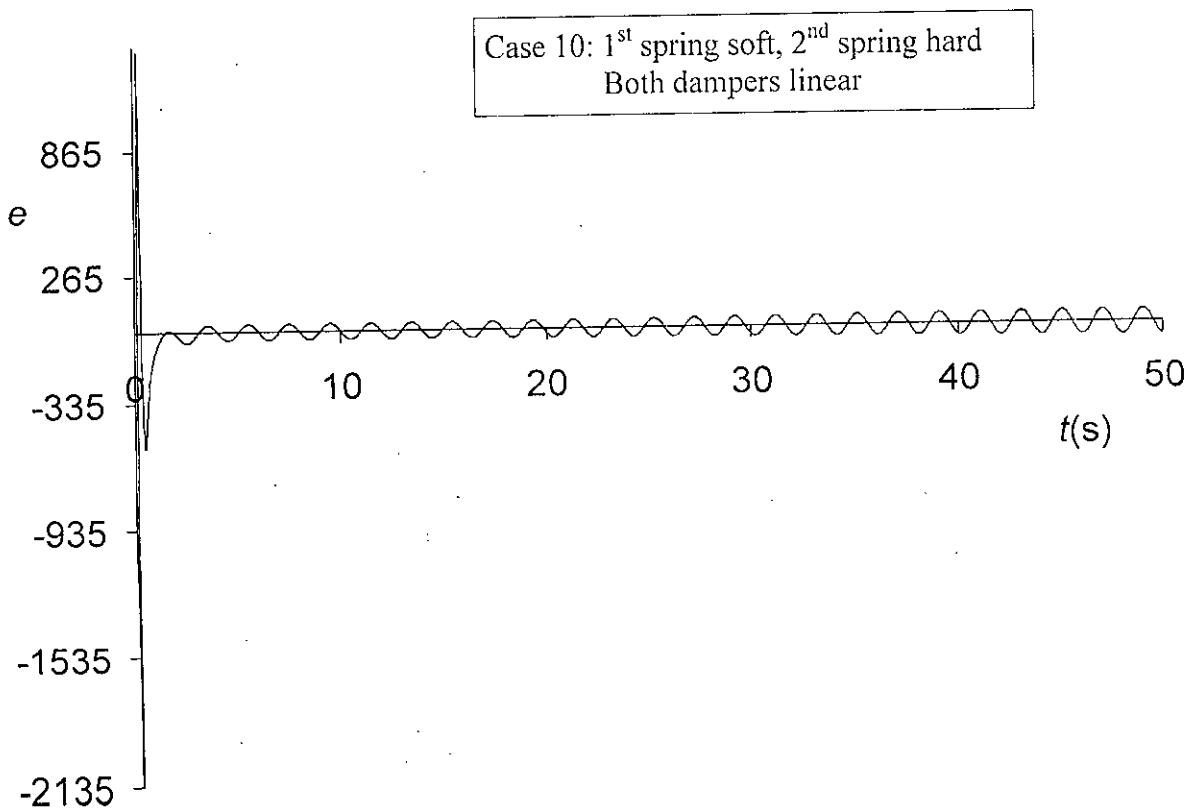


Fig. 40  $e - t$  at  $r=1.0$  and  $\zeta=0.0158$  for case 10.

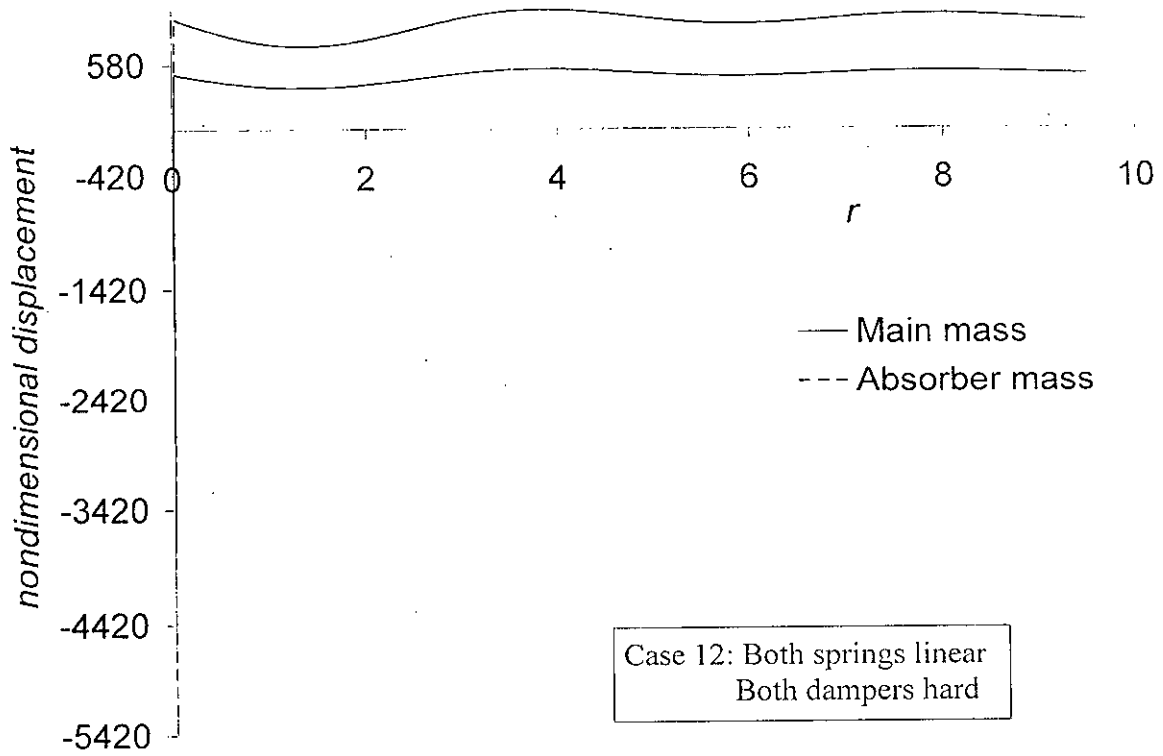


Fig. 41 Non-dimensional displacement vs. frequency ratio at  $t=0.5s$  and  $\zeta=0.0158$  for case 12.

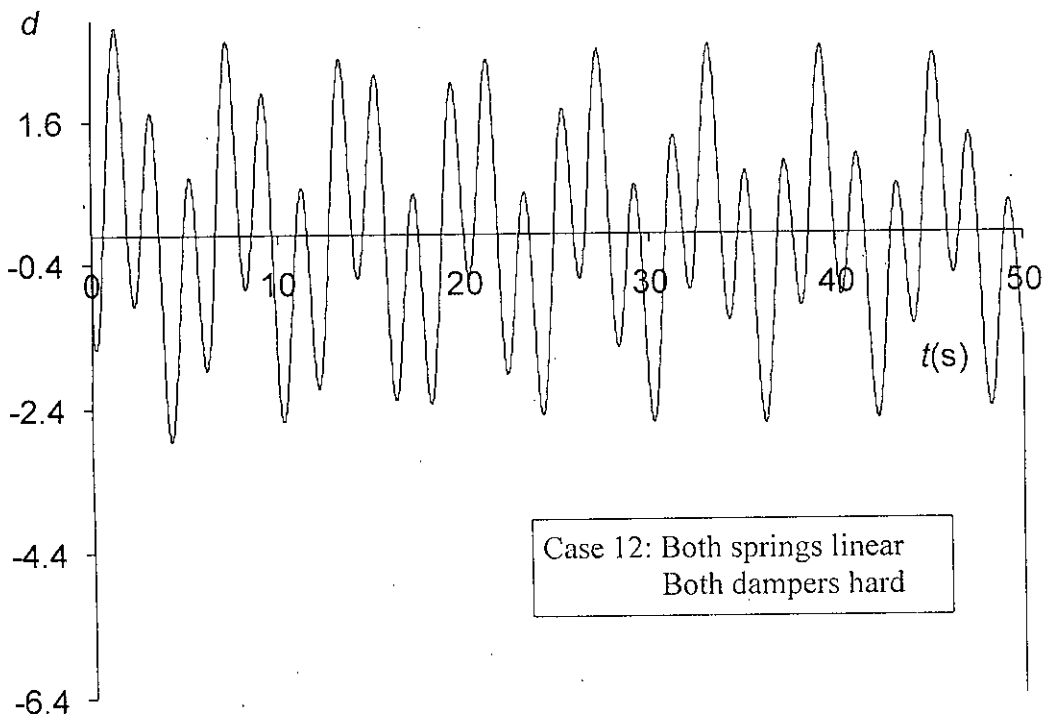
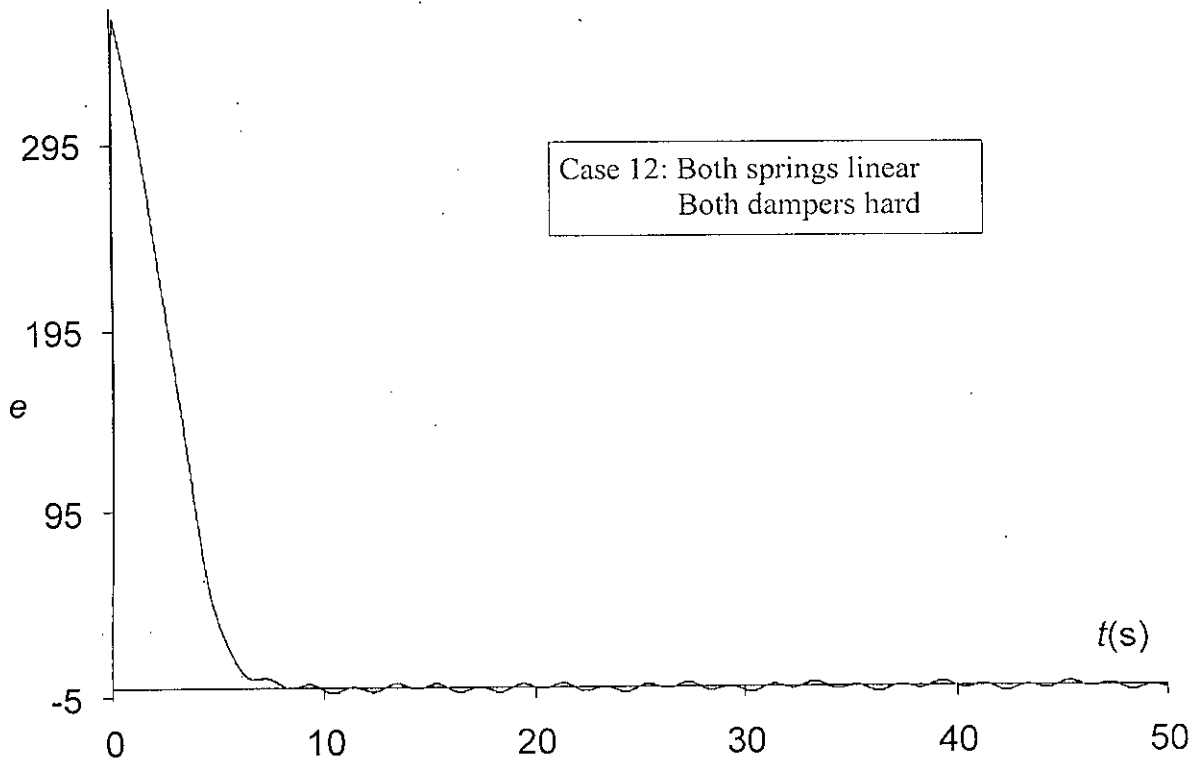
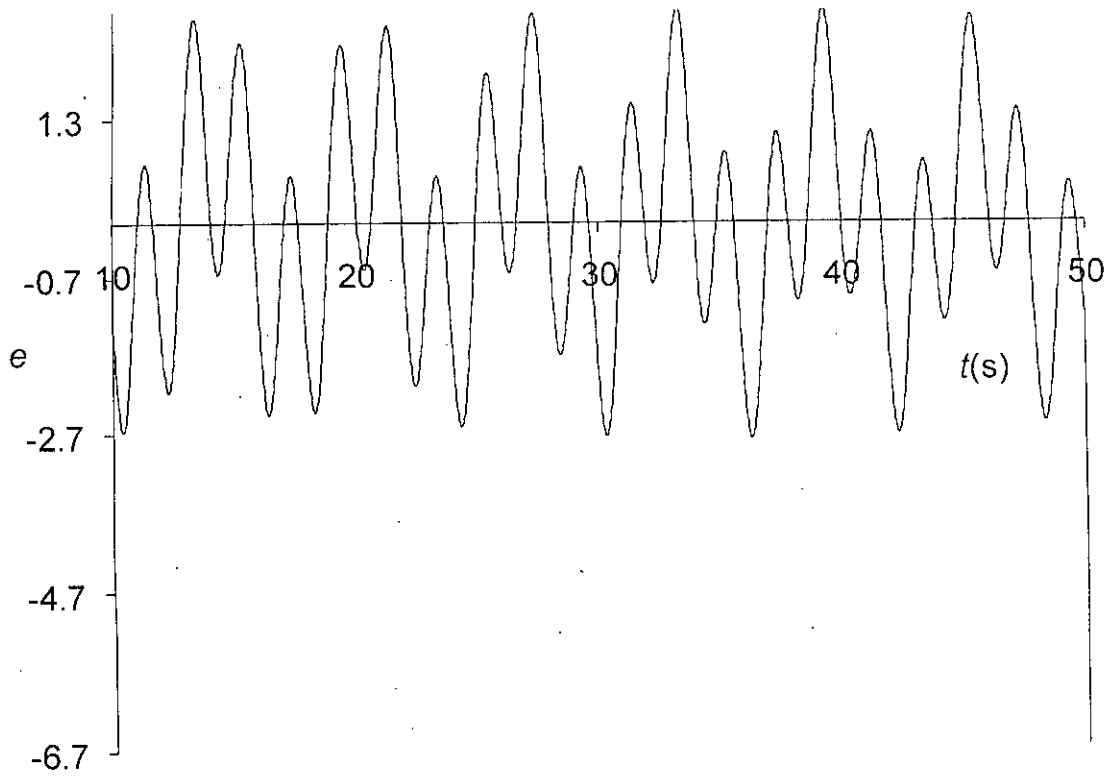


Fig. 42  $d-t$  at  $r=0.3162$  and  $\zeta=0.0158$  for case 12.



(a)



(b)

Fig. 43 (a)  $e - t$  at  $r = 0.3162$  and  $\zeta = 0.0158$  for case 12 (b) enlarged view for  $t = 10 - 50$ s.



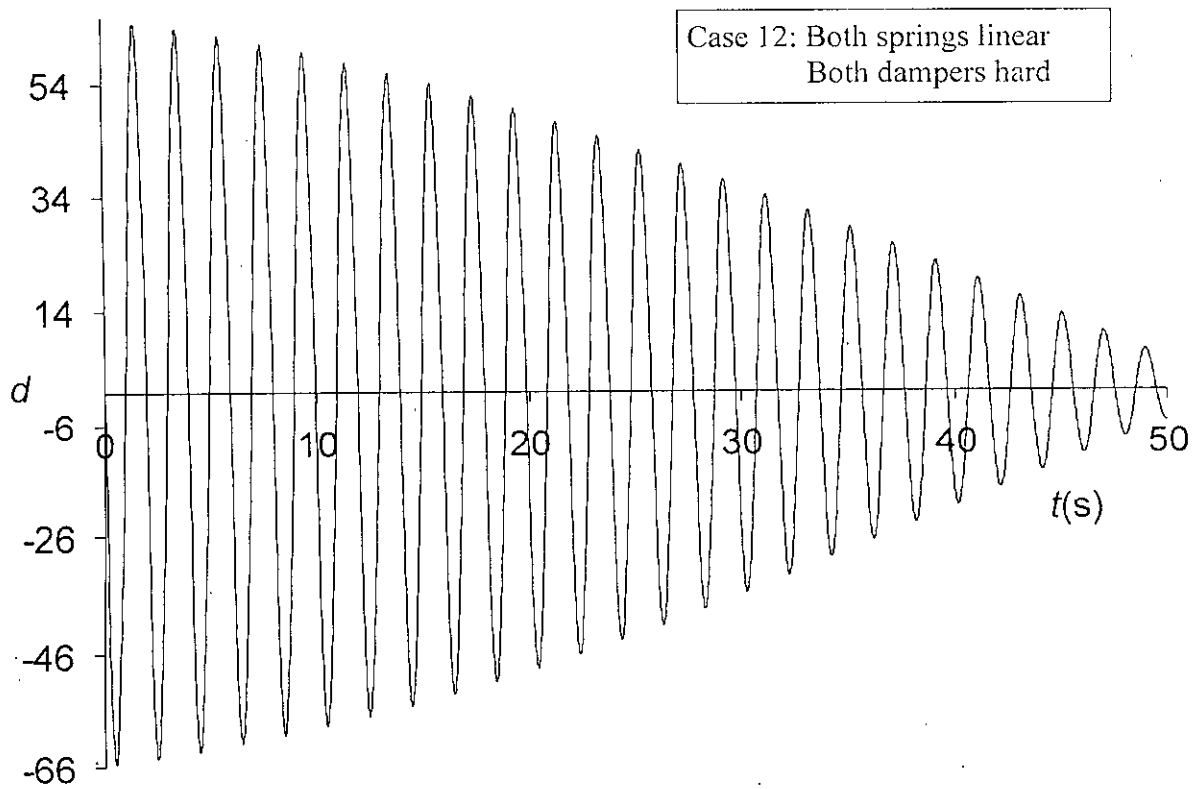


Fig. 44  $d - t$  at  $r=1.012$  and  $\zeta=0.0158$  for case 12.

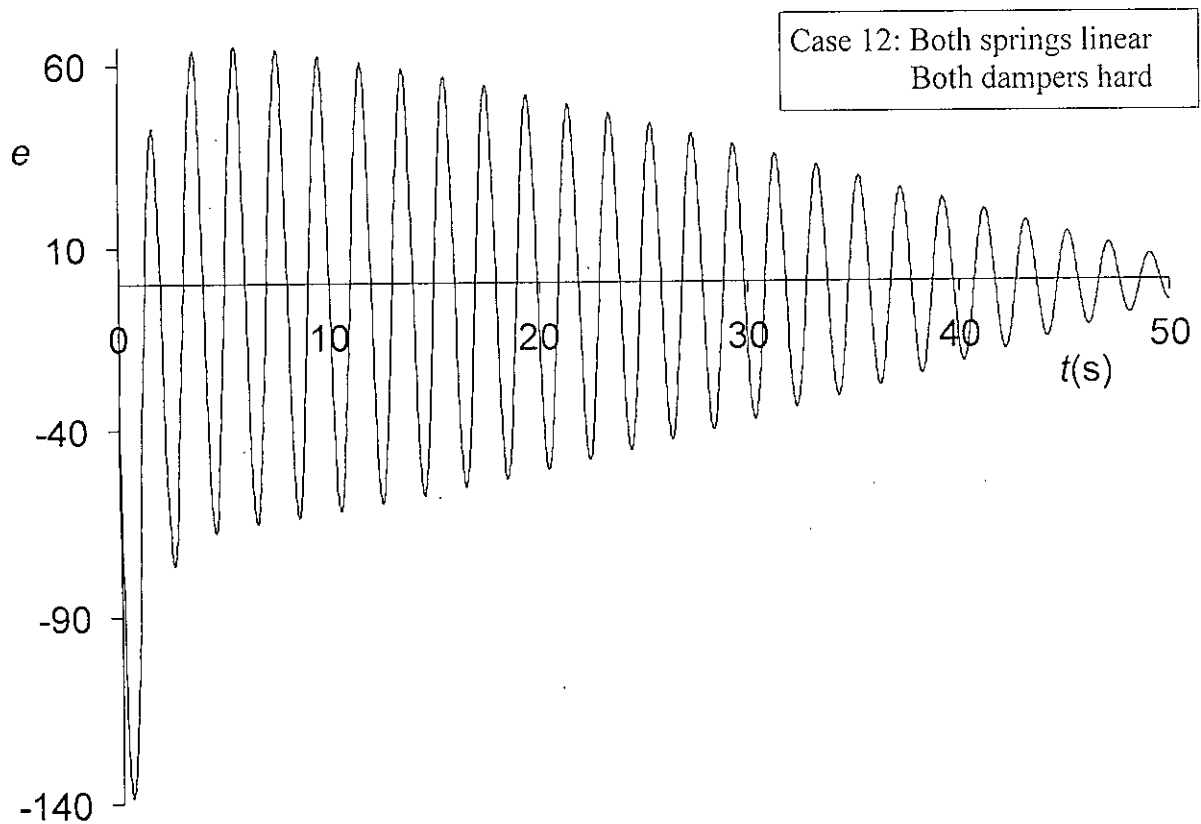


Fig. 45  $e - t$  at  $r=1.012$  and  $\zeta=0.0158$  for case 12.

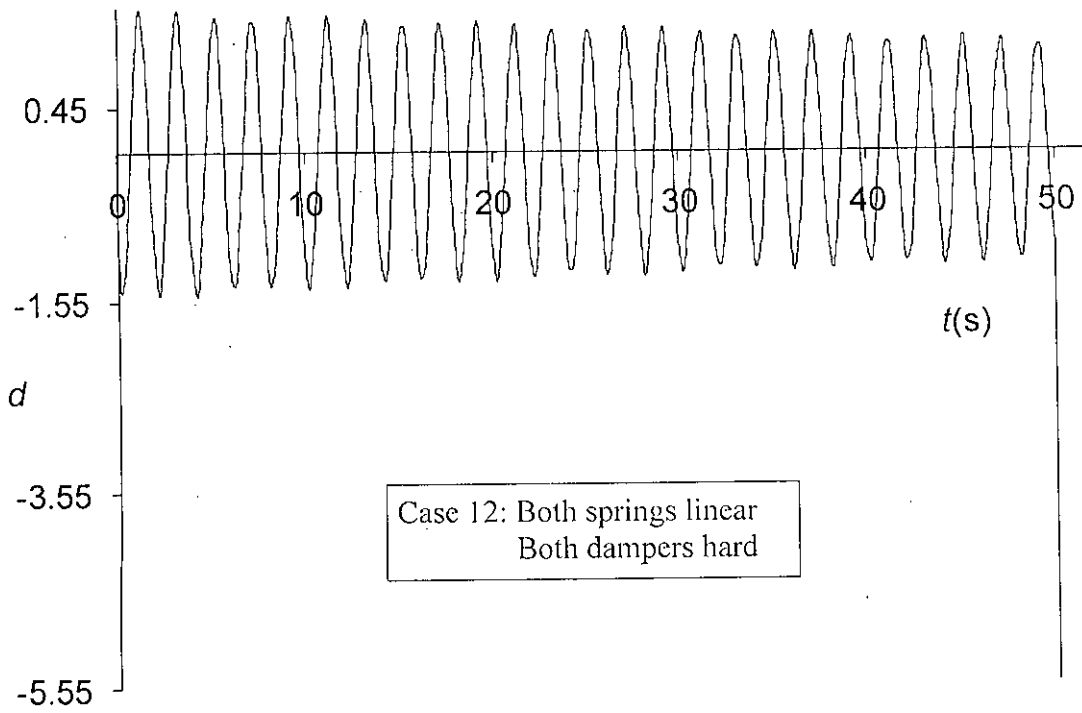


Fig. 46  $d - t$  at  $r = 4.744$  and  $\zeta = 0.0158$  for case 12.

# Appendix – A

## A.1 Extension for 3 DOFS

Analyzing 2 DOF tuned absorber and damped system with nonlinearity, coding was extended for 3DOFS. Fig. A1 shows the arrangements of the system masses. Here also, result is obtained for non-dimensional displacement with respect to frequency ratio and non-dimensional displacement with respect to time. For 3DOFS analysis no damping is taken under consideration. Parameters chosen for this analysis are given in the table A1:

Table A1: Chosen Parameters of 3 DOFS:

Parameter	Values for tuned absorber
$m_1$ (kg)	20
$m_2$ (kg)	20
$m_3$ (kg)	20
$k_1$ (N/m)	50
$k'_1$ (N/m <sup>3</sup> )	0.05
$k_2$ (N/m)	50
$k'_2$ (N/m <sup>3</sup> )	0.05
$k_3$ (N/m)	50
$k'_3$ (N/m <sup>3</sup> )	0.05
$c_1$ (Ns/m)	0.0
$c'_1$ (Ns/m <sup>3</sup> )	0.0
$c_2$ (Ns/m)	0.0
$c'_2$ (Ns/m <sup>3</sup> )	0.0
$c_3$ (Ns/m)	0.0
$c'_3$ (Ns/m <sup>3</sup> )	0.0
$\omega_1$ (rad/s)	1.58
$f$ (N)	20

Again the boundary conditions taken for this analysis are given in Table A2:

Table A2: Chosen boundary value for 3 DOFS.

Parameters	Boundary Values
$y_1(a)$ (m)	0.05
$y_2(b)$ (m/s)	0.06
$y_3(a)$ (m)	-0.07
$y_4(b)$ (m/s)	-0.06
$y_5(a)$ (m)	0.0
$y_6(b)$ (m/s)	0.0

Non dimensional displacement versus frequency ratio for 1<sup>st</sup>, 2<sup>nd</sup> and 3<sup>rd</sup> mass of undamped 3 DOFS are shown in fig. A2(a), (b) and (c) respectively. All the springs in the system are considered linear. As seen from the figure, system shows large amplitude at three forcing frequency. At frequency ratios ( $r$ ) 0.7, 2.08 and 2.97 resonance occurs in the system as a result of which each mass of the system vibrates with larger amplitude. Highest amplitude occurs in first mass is at second resonant frequency amounting  $d$  equals to 16.98. Similarly for second and third mass, highest amplitude occurs at 3<sup>rd</sup> and 2<sup>nd</sup> resonance frequency respectively. Again peak amplitudes at these frequencies are 13.84 and 14.97 for 2<sup>nd</sup> and 3<sup>rd</sup> mass respectively. Again nonlinearity of the system is analyzed only for the springs. In the nonlinearity analysis, all three springs are considered either hard or soft at a time. Obtained figures of the nonlinear system are similar to fig. A2. Resonance occurs in the same  $r$  values as that occurs in linear system. The basic difference is the peak amplitudes of the system. For soft springs, peak occurs in 1<sup>st</sup> mass ( $d_{\max}=15.97$ ) and at second resonance frequency. Again for hard springs peak occurs in 1st mass ( $d_{\max}=16.73$ ) and it occurs also in second resonant frequency.

Fig A3(a), (b), and (c) show the non-dimensional deflection versus time for 1<sup>st</sup>, 2<sup>nd</sup> and 3<sup>rd</sup> masses of 3 DOFS, respectively. From the figures, amplitude for all masses of the system increases continuously with entire 50s time. 1<sup>st</sup> mass of the system shows  $d_{\max}$  of 1.264 starting from a very small value. Again 2<sup>nd</sup> and 3<sup>rd</sup> masses also shows similar trend like 1<sup>st</sup> mass. In cases of 2<sup>nd</sup> and 3<sup>rd</sup> masses  $e_{\max}$  and  $g_{\max}$  are 1.584 and 0.886 respectively. Again fig. A3 represents the non-dimensional displacements versus time for nonlinear vibration system. For hard springs,  $d_{\max}$ ,  $e_{\max}$ , and  $g_{\max}$  are 0.568, 0.72 and 0.429 respectively and this maximum amplitude occurs near the end of 50s period. Soft spring's  $d_{\max}$ ,  $e_{\max}$  and  $g_{\max}$  are 0.785, 0.876 and 0.55 respectively. Displacement versus time curve shows that the system tends to vibrate with larger amplitude with time approaching to instability.

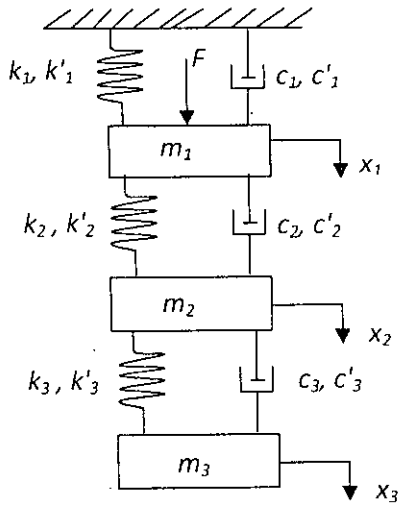
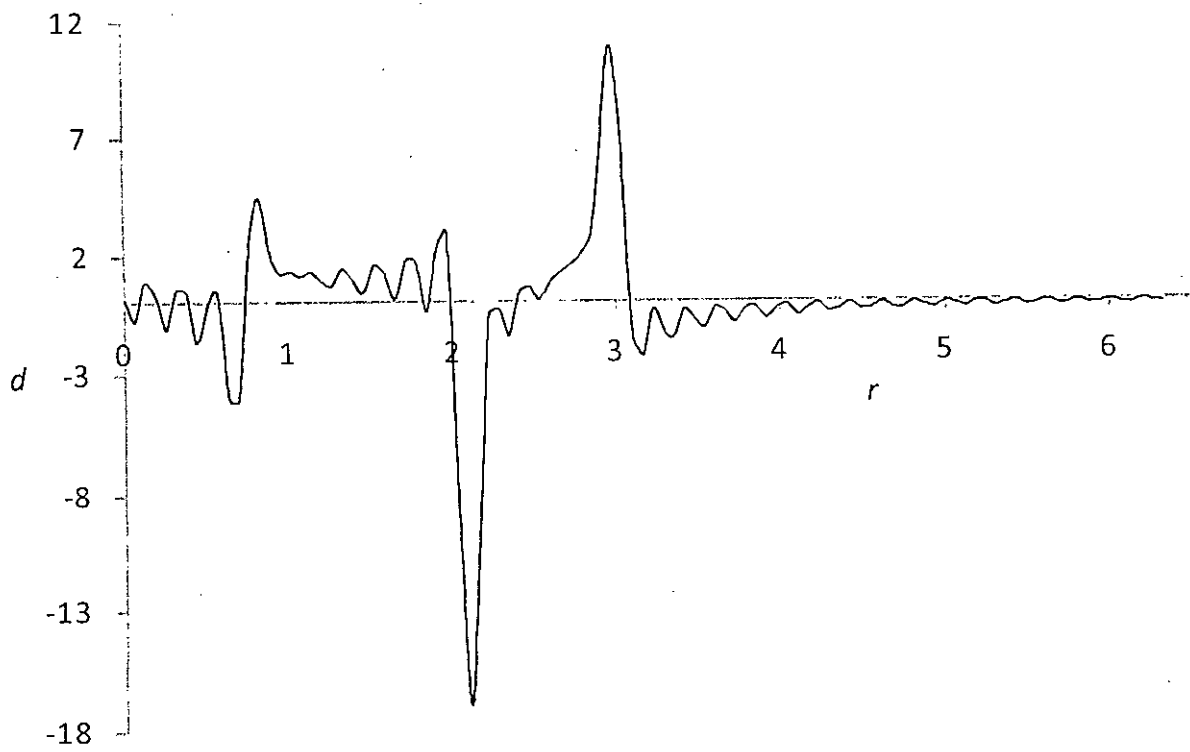
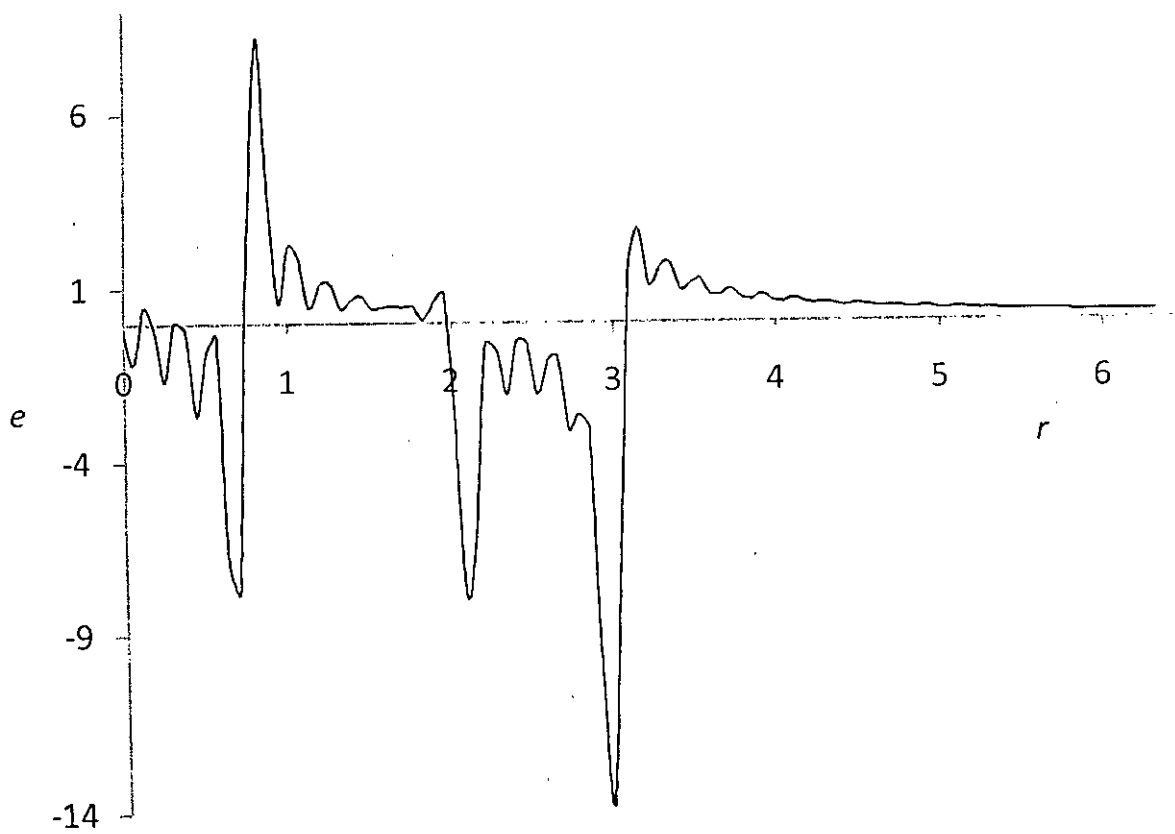


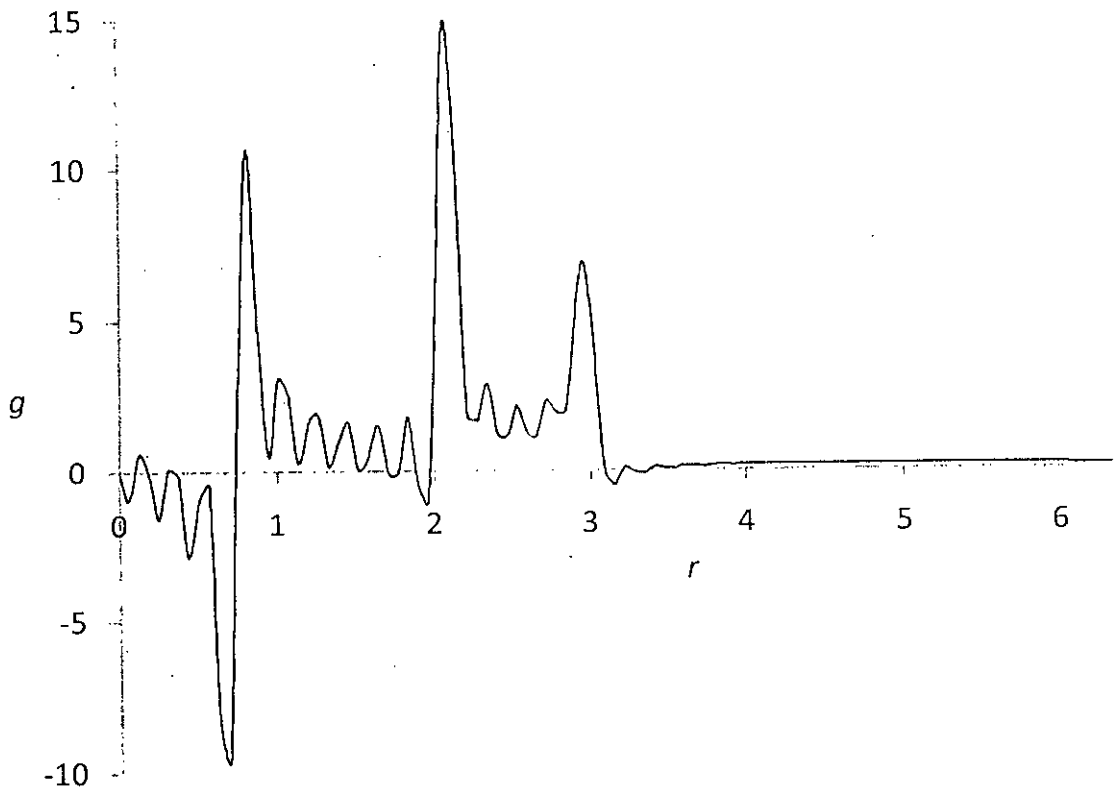
Fig. A1 Arrangement of masses, springs and dampers for 3-DOF vibration system.



(a)

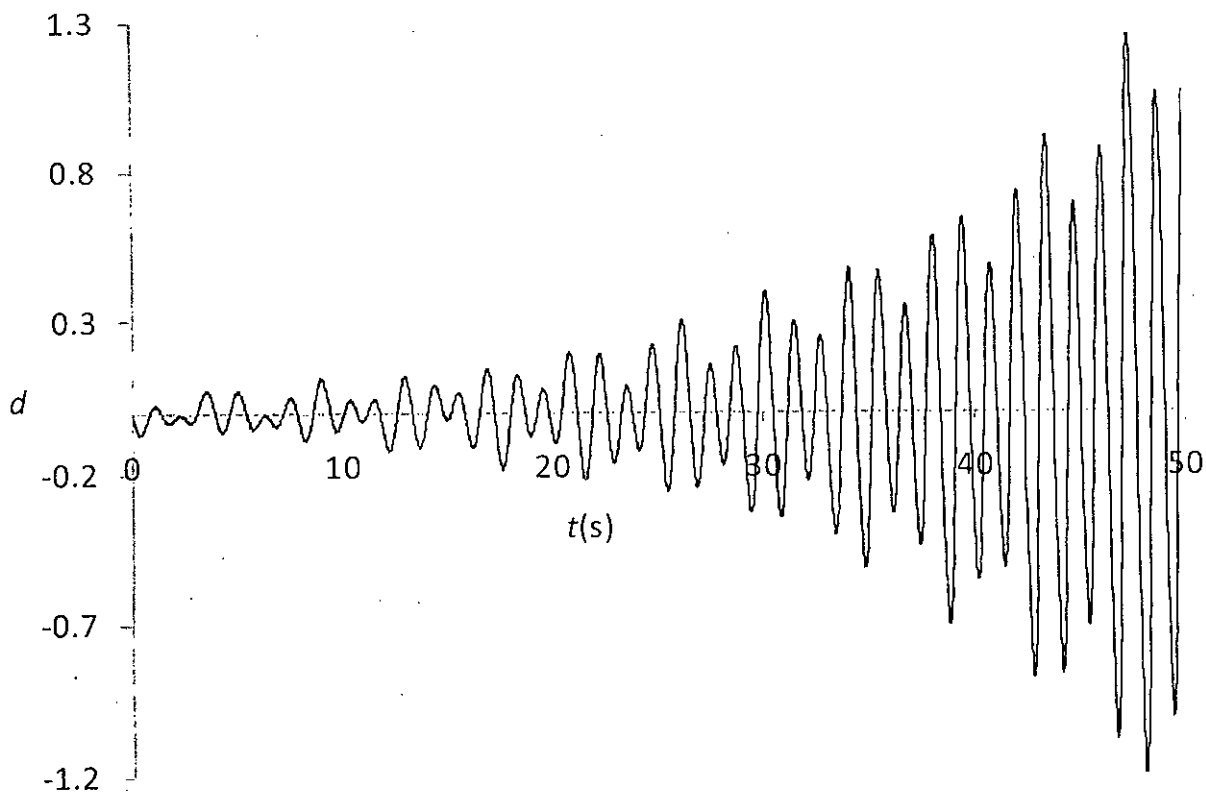


(b)

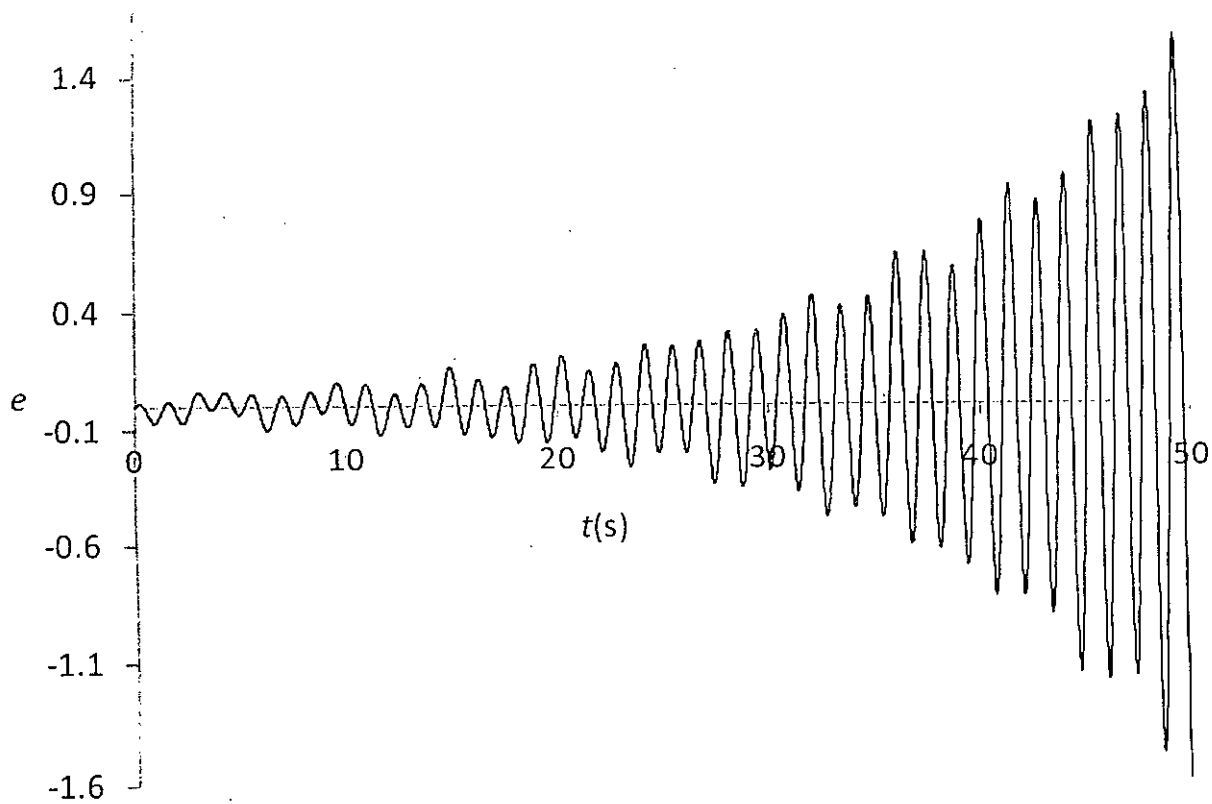


(c)

Fig. A2 Non-dimensional displacements versus frequency ratio for 3 DOF system. (a) 1<sup>st</sup> mass displacement (b) 2<sup>nd</sup> mass displacement and (c) 3<sup>rd</sup> mass displacement

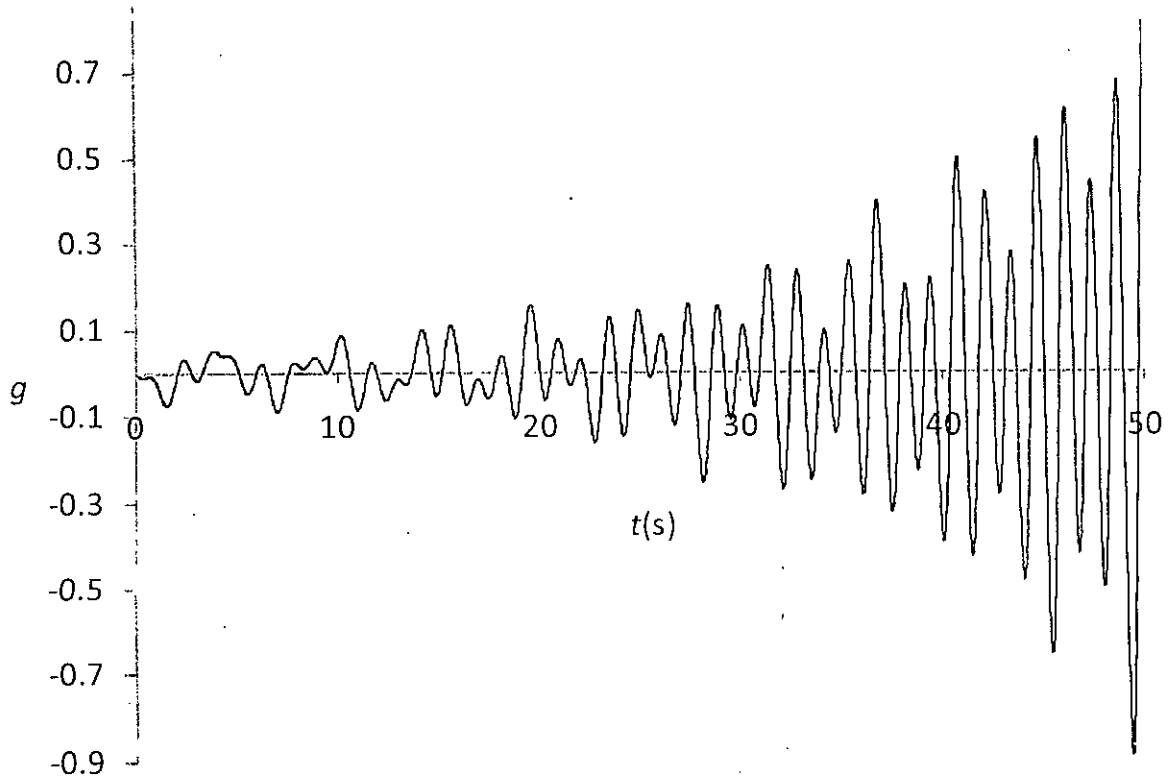


(a)





(b)



(c)

Fig. A3 Non-dimensional displacements versus time for 3 DOF system. (a) 1<sup>st</sup> mass displacement (b) 2<sup>nd</sup> mass displacement and (c) 3<sup>rd</sup> mass displacement

## Appendix – B

### B.1 Some Results for 20s Vibration

In this chapter some  $d - r$  curves for different tuned and untuned absorber system is discussed for  $t = 20s$  response. For tuned analysis at 20s, parameters used are given in Table 3. For untuned absorber analysis at 20s, parameters are taken from Table 3 for Data Set # 1.

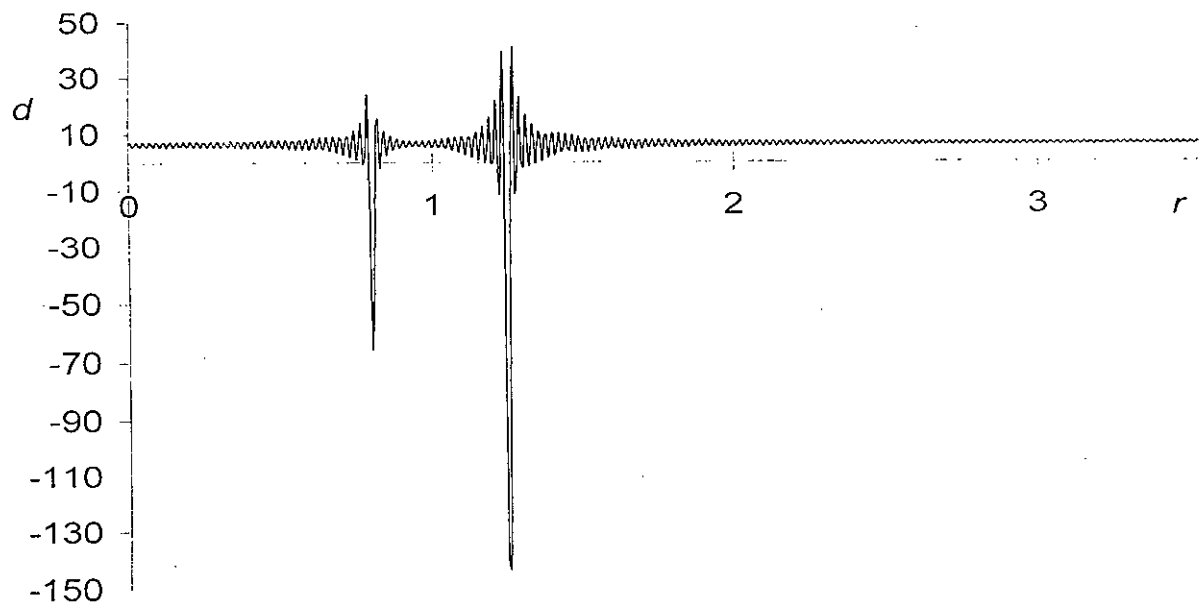
Figs. B1(a) and (b), for case 1, show the response of the linear TDOFS with the change of forcing frequency. This analysis was made intentionally to compare the exact results for tuned absorbers given in fig. 3(c) [Thompson (1981)]. These results of Thompson (1981) are of course, for steady state vibrations but present study includes all terms. Figs. B1(a) and B1(b), for the main mass and the absorber, respectively, prove the reliability of the code as these give very similar curves given in fig. 3(c) [Thompson (1981)]. As seen in Fig. B1, deflection tends to a small value (zero for steady state vibration as in Thompson (1981)) at the tuned frequency ( $r=1$ ), while the system response becomes boundless at the two natural frequencies corresponding to  $r_1 = 0.8$  and  $r_2=1.25$ .

Having analyzing  $d - r$  curves for a linear absorber ( Figs. B1(a), (b)), it would be now easy to comprehend the effect of spring nonlinearity on the system's response from Figs. B2(a) and (b) that represent the  $d - r$  curves for case 2. From Figs. B1 and B2, a general observation that can be made is that the non-dimensional deflection of absorber mass at  $r = 0.8$  and  $1.25$  is always greater than that of main mass.  $d - r$  and  $e - r$  curves of the 2 DOFS for other nonlinear Cases 3 – 5 are similar to that of Figs. B2, that are not shown here for the sake of brevity.

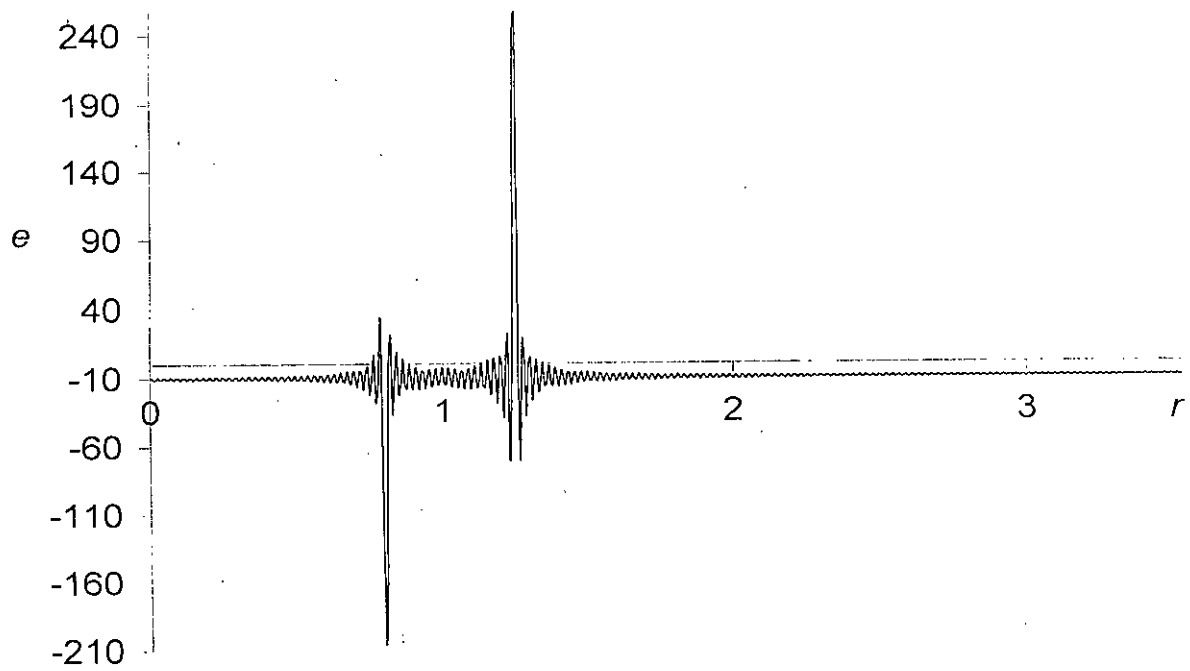
Fig. B3 allows the comparison of different cases of spring combination from  $d$  versus  $r$  curves at  $t=20s$ . As seen  $d$  is small in the regions  $0 < r < 0.65$ , in the vicinity of the tuned frequency  $r=1$  and for high frequency for  $r > 1.4$ . But  $d$  is too high at the two resonant frequencies. For all the cases 1, 2 and 4, the second resonant frequency  $r_2=1.25$  seems to be more dangerous than the first one ( $r_1=0.8$ ), in terms of non-dimensional displacements. While the first natural frequency is more dangerous than the second one for cases 3 and 5. It is also interesting to see from this figure that absolute non-dimensional displacements at  $r_1 = 0.8$  and  $r_2=1.25$  are the highest for case 1. Case 4 and case 2 show the second and third highest

displacements, respectively at  $r_2=1.25$ . While,  $d= -64.96$  and  $-64.9$  for Cases 3 and 5, respectively at  $r_1=0.8$ .

Fig. B4 is for absolute non-dimensional displacement versus frequency ratio for case 7. Due to imposed velocity of the system at final condition the damper force also become fixed at the end. As seen from the figure, peak amplitude occurs at frequency ratio near unity and the peak value is 3.382.

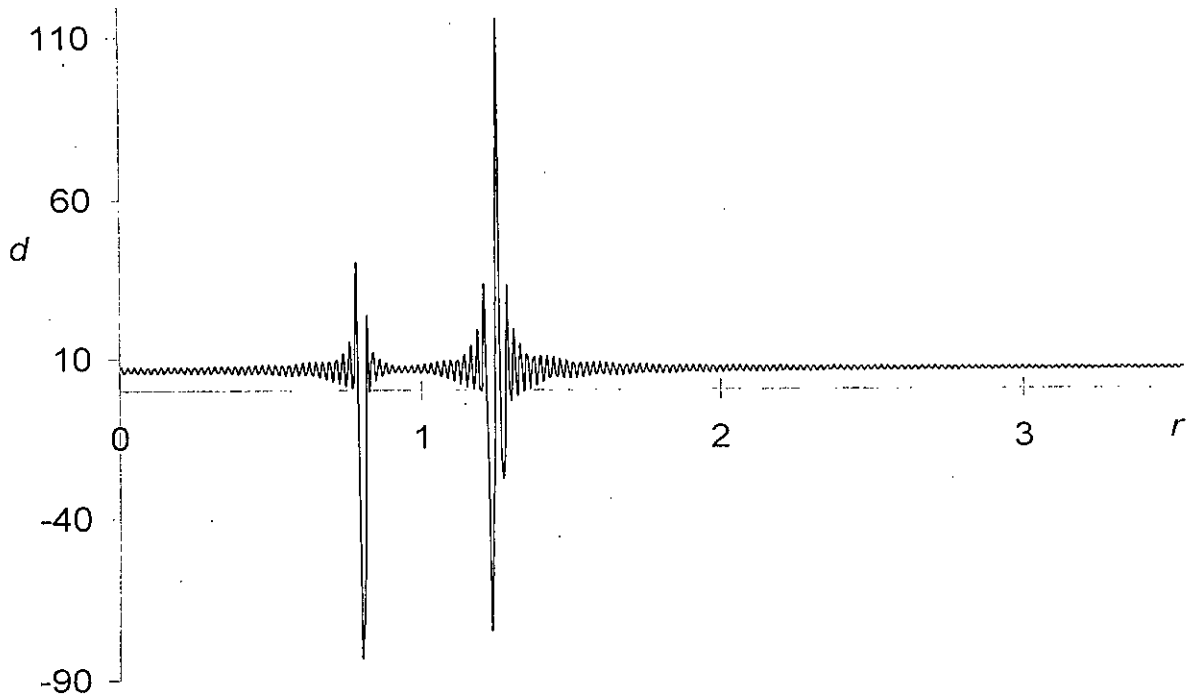


(a)

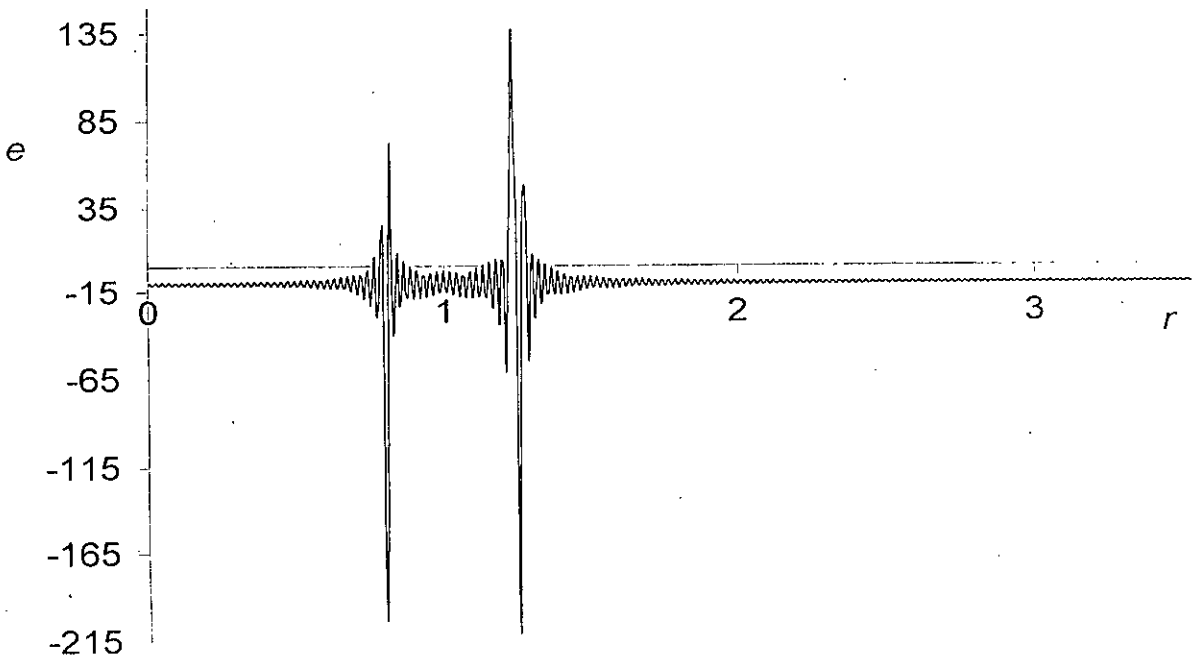


(b)

Fig. B1 Non-dimensional displacement - frequency ratio for case 1 at  $t=20s$ : (a) main mass (b) absorber mass.



(a)



(b)

Fig. B2 Non-dimensional displacement - frequency ratio for case 2 at  $t=20s$ : (a) main mass  
(b) absorber mass.

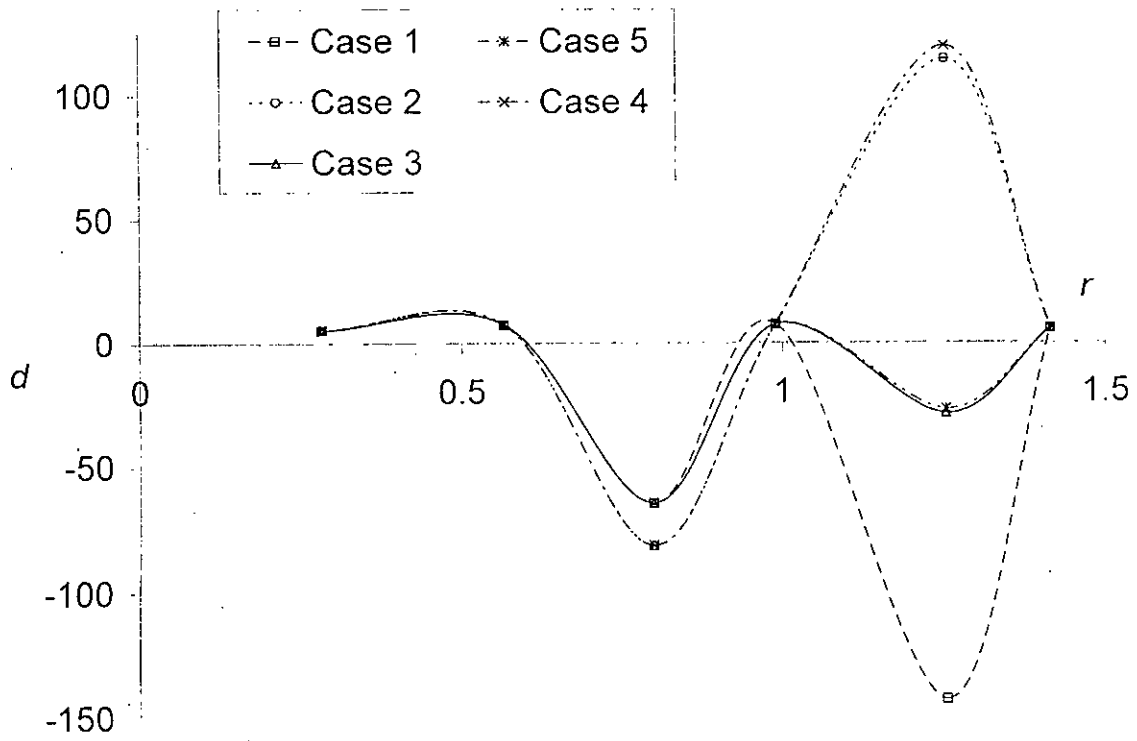


Fig. B3 Comparison of non-dimensional displacement - frequency ratio for all cases together at  $t=20s$ .

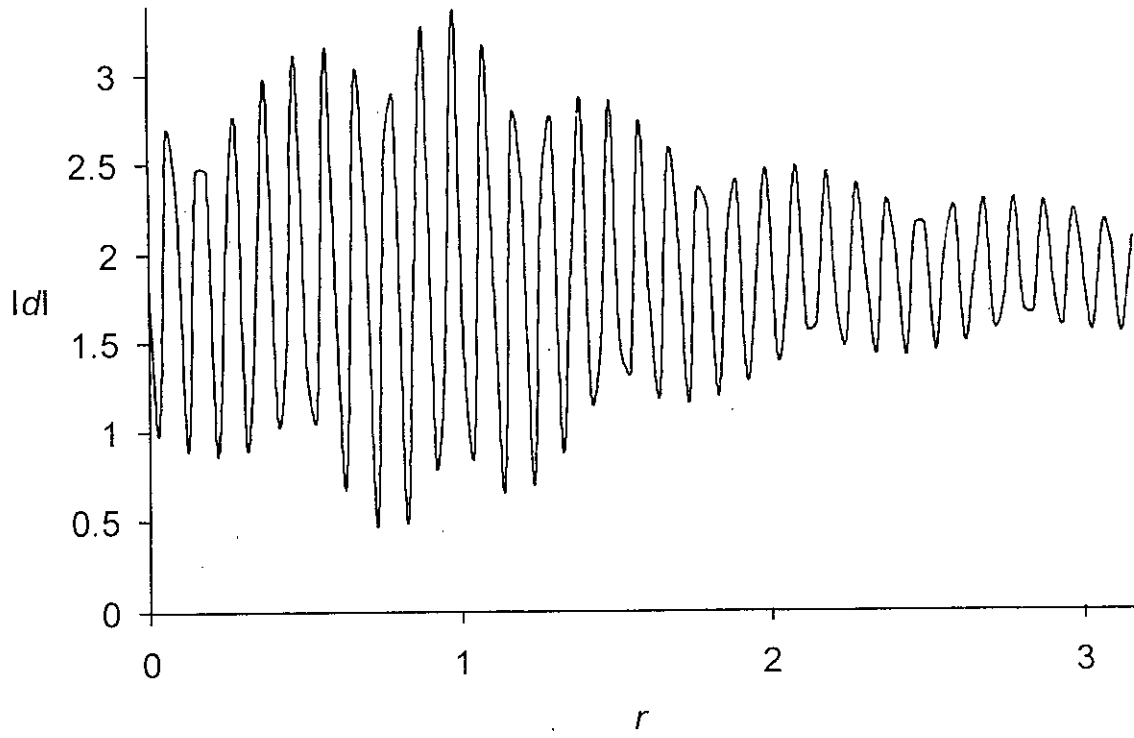


Fig. B4  $|d| - r$  for case 7 having  $\mu=0.01$  and  $\zeta=0.00496$  at  $t=20$ s and solved as boundary value problem.

# Appendix – C

## Programme Code

### C.1 Code for Nondimensional Displacement with Time for 2 DOFS

```
# include <stdio.h>
# include <conio.h>
# include <math.h>
# include <stdlib.h>
# define n 4
# define ma1 100.0
# define ma2 1.0
# define k_1 1000
# define k_1p 0.0
# define k_2 10.0
# define k_2p 0.0
# define c1 0.0
# define c1p 0.0
# define c2 0.0
# define c2p 0.0
# define f1 20.0
# define f2 0.0
# define tb 50.0
# define h 0.02
# define w 2.86
# define zero 0.0

/* mass m in kg : spring constant k in N/m : damping constant c in Ns/m : force amplitude f in N: angular frequency
w in rad/s : tb or t in second(s) */

double fn1(double t, double y1, double y2, double y3, double y4, double w1)
{
double z=0.0;
z= (f1 *sin(w*t)-(k_1*y1+k_1p*pow(y1,3))-(k_2*(y1-y3)+k_2p*pow((y1-y3),3)) - (c1*y2+c1p*y2*pow(y1,2)) -
(c2*(y2-y4)+c2p*(y2-y4)*pow((y1-y3),2)) )/ma1 ;
return z;
}
double fn2(double t, double y1, double y2, double y3, double y4, double w1)
{
double z=0.0;
z= (k_2*(y1-y3)+k_2p*pow((y1-y3),3) + (c2*(y2-y4)+c2p*(y2-y4)*pow((y1-y3),2)) )/ma2 ;
return z;
}
float FN1(float t, float Y0, float Y1, float Y2, float Y3, float y1, float y2, float y3, float y4)
{
```



```

float z=0.0;
z= -(k_1*Y0+k_1p*3*pow(y1,2)*Y0+k_2*(Y0-Y2)+3*k_2p*pow((y1-y3),2)*(Y0-Y2)+ c1*Y1+
c1p*Y1*pow(y1,2)+ 2*c1p*y2*y1*Y0+ c2*(Y1-Y3)+ c2p*pow((y1-y3),2)*(Y1-Y3)+ 2*c2p*(y1-y3)*(y2-
y4)*(Y0-Y2) )/ma1;
return z;
}
float FN2(float t,float Y0,float Y1, float Y2, float Y3, float y1, float y2, float y3, float y4)
{
float z=0.0;
z= (k_2*(Y0-Y2)+k_2p*3*pow((y1-y3),2)*(Y0-Y2)+ c2*(Y1-Y3)+ c2p*pow((y1-y3),2)*(Y1-Y3)+ 2*c2p*(y1-
y3)*(y2-y4)*(Y0-Y2))/ma2;
return z;
}
void main()
{
int i,j,n1,l,b,k,u,acc,freq;
double yt[n];
float Y1[n][n]={{1.0},
{0.0, 1.0},{0.0,0.0,1.0},{0.0,0.0,0.0,1.0}};
float n2,t,y1new=0.0,y1old=1.0, convgnc1,
y2new=0.0,y2old=1.0,convgnc2,y3new=0.0,y3old=1.0,convgnc3,y4new=0.0,y4old=1.0,convgnc4;
double k1[n][n],K[n][n],L[n][n],M[n][n],N[n][n],Y[n][n];
double k2[n][n]={{0.0},{0.0},{0.0},{0.0}};
float m_1[n],m_2[n][n],m_3[n];
float l1[n],q[n][n],r[n],ya[n],yb[n], Yb[n][n];
float
a[n][n],a1[n][n],a2[n][n],a3[n][n],a4[n][n],m1[n][n],m2[n][n],m3[n][n],m4[n],ab1[n][n],ab2[n][n],ab3[n][n],inverse[
n][n],test[n][n];
float p1,p2,p3,p4,sum,sum_s,freq1;
float yta[n], ytb[n];
float ytini[n]={0.0,0.0,0.0,0.0};
float sum4,sum3,sum2,sum1;
FILE *fp;
float l[n][n]= {{1.0,0.0,0.0,0.0},
{0.0,1.0,0.0,0.0},
{0.0,0.0,1.0,0.0},
{0.0,0.0,0.0,1.0}};

float A[n][n]={{01.0,0.0,0.0,0.0},
{0.0,0.0,0.0,0.0},
{0.0,0.0,01.0,0.0},
{0.0,0.0,0.0,0.0}};

float B[n][n]={{0.0,0.0,0.0,0.0},
{0.0,01.0,0.0,0.0},
{0.0,0.0,0.0,0.0},
{0.0,0.0,0.0,01.0}};

```

```

float C[n]={0.05,0.06,-0.07,-0.06};

t= zero;
n2=((tb-t)/h);
n1=n2;
fp=fopen("d:\result\dabsr.xls","w");
clrscr();
for (b=0;b<n;b++)
{yt[b]=ytini[b];}
convgnc1=fabs(y1new-y1old);
convgnc2=fabs(y2new-y2old);
convgnc3=fabs(y3new-y3old);
convgnc4=fabs(y4new-y4old);

while (0.0001<convgnc1 && 0.0001<convgnc2 && 0.0001<convgnc3 && 0.0001<convgnc4)
{
yt[0]=0.05;

yt[2]=0.07;
for (b=0;b<n;b++)
{yta[b]=ytini[b];}
for (b=0;b<n;b++)
{ytb[b]=ytini[b];}
for (b=0;b<n;b++)
{r[b]=ytini[b];}
for (b=0;b<n;b++)
{l1[b]=ytini[b];}

for (k=0;k<n;k++)
{ for (b=0;b<n;b++)
{Yb[k][b]=k2[k][b];}}
for (k=0;k<n;k++)
{ for (b=0;b<n;b++)
{m2[k][b]=k2[k][b];}}
for (k=0;k<n;k++)
{ for (b=0;b<n;b++)
{q[k][b]=k2[k][b];}}
for (b=0;b<n;b++)
{yta[b]=yt[b];}
sum1=0.0;
sum2=0.0;
sum3=0.0;
sum4=0.0;
sum=0.0;
sum_s=0.0;
p1=0.0;
p2=0.0;

```

```

p3=0.0;
p4=0.0;
t=0.0;
for (k=0;k<n;k++)
{ for (b=0;b<n;b++)
{k1[k][b]=k2[k][b];}}
for (k=0;k<n;k++)
{ for (b=0;b<n;b++)
{K[k][b]=k2[k][b];}}
for (k=0;k<n;k++)
{ for (b=0;b<n;b++)
{L[k][b]=k2[k][b];}}
for (k=0;k<n;k++)
{ for (b=0;b<n;b++)
{M[k][b]=k2[k][b];}}
for (k=0;k<n;k++)
{ for (b=0;b<n;b++)
{N[k][b]=k2[k][b];}}
for (k=0;k<n;k++)
{ for (b=0;b<n;b++)
{Y[k][b]=Y1[k][b];}}

for (i=0;i<n1;i++)
{
k1[0][0]=h * yt[1];
k1[0][1]=h * fn1(t, yt[0],yt[1], yt[2], yt[3], w);
k1[0][2]=h * yt[3];
k1[0][3]=h * fn2(t, yt[0],yt[1], yt[2], yt[3], w);
k1[1][0]=h * (yt[1]+k1[0][1]/2.0);
k1[1][1]=h * fn1(t+h/2.0, yt[0]+k1[0][0]/2.0, yt[1]+k1[0][1]/2.0, yt[2]+k1[0][2]/2.0, yt[3]+k1[0][3]/2.0, w);
k1[1][2]=h * (yt[3]+k1[0][3]/2.0);
k1[1][3]=h * fn2(t+h/2.0, yt[0]+k1[0][0]/2.0, yt[1]+k1[0][1]/2.0, yt[2]+k1[0][2]/2.0, yt[3]+k1[0][3]/2.0, w);
k1[2][0]=h * (yt[1]+k1[1][1]/2.0);
k1[2][1]=h * fn1(t+h/2.0, yt[0]+k1[1][0]/2.0, yt[1]+k1[1][1]/2.0, yt[2]+k1[1][2]/2.0, yt[3]+k1[1][3]/2.0, w);
k1[2][2]=h * (yt[3]+k1[1][3]/2.0);
k1[2][3]=h * fn2(t+h/2.0, yt[0]+k1[1][0]/2.0, yt[1]+k1[1][1]/2.0, yt[2]+k1[1][2]/2.0, yt[3]+k1[1][3]/2.0, w);
k1[3][0]=h * (yt[1]+k1[2][1]);
k1[3][1]=h * fn1(t+h, yt[0]+k1[2][0], yt[1]+k1[2][1], yt[2]+k1[2][2], yt[3]+k1[2][3], w);
k1[3][2]=h * (yt[3]+k1[2][3]);
k1[3][3]=h * fn2(t+h, yt[0]+k1[2][0], yt[1]+k1[2][1], yt[2]+k1[2][2], yt[3]+k1[2][3], w);
K[0][0]=h * Y[1][0];
K[0][1]=h * FN1(t, Y[0][0], Y[1][0], Y[2][0], Y[3][0], yt[0], yt[1], yt[2], yt[3]);
K[0][2]=h * Y[3][0];
K[0][3]=h * FN2(t, Y[0][0], Y[1][0], Y[2][0], Y[3][0], yt[0], yt[1], yt[2], yt[3]);
K[1][0]=h * (Y[1][0]+K[0][1]/2.0);
K[1][1]=h * FN1(t+h/2.0, Y[0][0]+K[0][0]/2.0, Y[1][0]+K[0][1]/2.0, Y[2][0]+K[0][2]/2.0, Y[3][0]+K[0][3]/2.0,
yt[0], yt[1], yt[2], yt[3]);
K[1][2]=h * (Y[3][0]+K[0][3]/2.0);

```

```

K[1][3]=h * FN2(t+h/2.0, Y[0][0]+K[0][0]/2.0, Y[1][0]+K[0][1]/2.0, Y[2][0]+K[0][2]/2.0, Y[3][0]+K[0][3]/2.0,
yt[0], yt[1], yt[2], yt[3] );
K[2][0]=h * (Y[1][0]+K[1][1]/2.0);
K[2][1]=h * FN1(t+h/2.0, Y[0][0]+K[1][0]/2.0, Y[1][0]+K[1][1]/2.0, Y[2][0]+K[1][2]/2.0, Y[3][0]+K[1][3]/2.0,
yt[0], yt[1], yt[2], yt[3] );
K[2][2]=h * (Y[3][0]+K[1][3]/2.0);
K[2][3]=h * FN2(t+h/2.0, Y[0][0]+K[1][0]/2.0, Y[1][0]+K[1][1]/2.0, Y[2][0]+K[1][2]/2.0, Y[3][0]+K[1][3]/2.0,
yt[0], yt[1], yt[2], yt[3] );
K[3][0]=h * (Y[1][0]+K[2][1]);
K[3][1]=h * FN1(t+h, Y[0][0]+K[2][0], Y[1][0]+K[2][1], Y[2][0]+K[2][2], Y[3][0]+K[2][3], yt[0], yt[1], yt[2],
yt[3] );
K[3][2]=h * (Y[3][0]+K[2][3]);
K[3][3]=h * FN2(t+h, Y[0][0]+K[2][0], Y[1][0]+K[2][1], Y[2][0]+K[2][2], Y[3][0]+K[2][3], yt[0], yt[1], yt[2],
yt[3] );
for(l=0;l<n;l++)
{Y[l][0]=Y[l][0]+ (K[0][l]+ 2.0*K[1][l]+ 2.0*K[2][l]+ K[3][l])/6.0 ;}
L[0][0]=h * Y[1][1];
L[0][1]=h * FN1(t, Y[0][1],Y[1][1], Y[2][1], Y[3][1], yt[0], yt[1], yt[2], yt[3] );
L[0][2]=h * Y[3][1];
L[0][3]=h * FN2(t, Y[0][1],Y[1][1], Y[2][1], Y[3][1], yt[0], yt[1], yt[2], yt[3] );

L[1][0]=h * (Y[1][1]+L[0][1]/2.0);
L[1][1]=h * FN1(t+h/2.0, Y[0][1]+L[0][0]/2.0, Y[1][1]+L[0][1]/2.0, Y[2][1]+L[0][2]/2.0, Y[3][1]+L[0][3]/2.0,
yt[0], yt[1], yt[2], yt[3] );
L[1][2]=h * (Y[3][1]+L[0][3]/2.0);
L[1][3]=h * FN2(t+h/2.0, Y[0][1]+L[0][0]/2.0, Y[1][1]+L[0][1]/2.0, Y[2][1]+L[0][2]/2.0, Y[3][1]+L[0][3]/2.0,
yt[0], yt[1], yt[2], yt[3] );

L[2][0]=h * (Y[1][1]+L[1][1]/2.0);
L[2][1]=h * FN1(t+h/2.0, Y[0][1]+L[1][0]/2.0, Y[1][1]+L[1][1]/2.0, Y[2][1]+L[1][2]/2.0, Y[3][1]+L[1][3]/2.0,
yt[0], yt[1], yt[2], yt[3] );
L[2][2]=h * (Y[3][1]+L[1][3]/2.0);
L[2][3]=h * FN2(t+h/2.0, Y[0][1]+L[1][0]/2.0, Y[1][1]+L[1][1]/2.0, Y[2][1]+L[1][2]/2.0, Y[3][1]+L[1][3]/2.0,
yt[0], yt[1], yt[2], yt[3] );

L[3][0]=h * (Y[1][1]+L[2][1]);
L[3][1]=h * FN1(t+h, Y[0][1]+L[2][0], Y[1][1]+L[2][1], Y[2][1]+L[2][2], Y[3][1]+L[2][3], yt[0], yt[1], yt[2], yt[3]
);
L[3][2]=h * (Y[3][1]+L[2][3]);
L[3][3]=h * FN2(t+h, Y[0][1]+L[2][0], Y[1][1]+L[2][1], Y[2][1]+L[2][2], Y[3][1]+L[2][3], yt[0], yt[1], yt[2], yt[3]
);
for(l=0;l<n;l++)
{Y[l][1]=Y[l][1]+ (L[0][l]+ 2.0*L[1][l]+ 2.0*L[2][l]+ L[3][l])/6.0 ;}

M[0][0]=h * Y[1][2];
M[0][1]=h * FN1(t, Y[0][2],Y[1][2], Y[2][2], Y[3][2], yt[0], yt[1], yt[2], yt[3] );
M[0][2]=h * Y[3][2];
M[0][3]=h * FN2(t, Y[0][2],Y[1][2], Y[2][2], Y[3][2], yt[0], yt[1], yt[2], yt[3] );

```

$M[1][0]=h * (Y[1][2]+M[0][1]/2.0) ;$   
 $M[1][1]=h * FN1(t+h/2.0, Y[0][2]+M[0][0]/2.0, Y[1][2]+M[0][1]/2.0, Y[2][2]+M[0][2]/2.0, Y[3][2]+M[0][3]/2.0, yt[0], yt[1], yt[2], yt[3]) ;$   
 $M[1][2]=h * (Y[3][2]+M[0][3]/2.0) ;$   
 $M[1][3]=h * FN2(t+h/2.0, Y[0][2]+M[0][0]/2.0, Y[1][2]+M[0][1]/2.0, Y[2][2]+M[0][2]/2.0, Y[3][2]+M[0][3]/2.0, yt[0], yt[1], yt[2], yt[3]) ;$

$M[2][0]=h * (Y[1][2]+M[1][1]/2.0) ;$   
 $M[2][1]=h * FN1(t+h/2.0, Y[0][2]+M[1][0]/2.0, Y[1][2]+M[1][1]/2.0, Y[2][2]+M[1][2]/2.0, Y[3][2]+M[1][3]/2.0, yt[0], yt[1], yt[2], yt[3]) ;$   
 $M[2][2]=h * (Y[3][2]+M[1][3]/2.0) ;$   
 $M[2][3]=h * FN2(t+h/2.0, Y[0][2]+M[1][0]/2.0, Y[1][2]+M[1][1]/2.0, Y[2][2]+M[1][2]/2.0, Y[3][2]+M[1][3]/2.0, yt[0], yt[1], yt[2], yt[3]) ;$

$M[3][0]=h * (Y[1][2]+M[2][1]) ;$   
 $M[3][1]=h * FN1(t+h, Y[0][2]+M[2][0], Y[1][2]+M[2][1], Y[2][2]+M[2][2], Y[3][2]+M[2][3], yt[0], yt[1], yt[2], yt[3]) ;$   
 $M[3][2]=h * (Y[3][2]+M[2][3]) ;$   
 $M[3][3]=h * FN2(t+h, Y[0][2]+M[2][0], Y[1][2]+M[2][1], Y[2][2]+M[2][2], Y[3][2]+M[2][3], yt[0], yt[1], yt[2], yt[3]) ;$

$for(i=0; i<n; i++)$   
 $\{Y[i][2]=Y[i][2]+ (M[0][i]+ 2.0*M[1][i]+ 2.0*M[2][i]+ M[3][i])/6.0 ;\}$

$N[0][0]=h * Y[1][3];$   
 $N[0][1]=h * FN1(t, Y[0][3], Y[1][3], Y[2][3], Y[3][3], yt[0], yt[1], yt[2], yt[3]) ;$   
 $N[0][2]=h * Y[3][3];$   
 $N[0][3]=h * FN2(t, Y[0][3], Y[1][3], Y[2][3], Y[3][3], yt[0], yt[1], yt[2], yt[3]) ;$

$N[1][0]=h * (Y[1][3]+N[0][1]/2.0);$   
 $N[1][1]=h * FN1(t+h/2.0, Y[0][3]+N[0][0]/2.0, Y[1][3]+N[0][1]/2.0, Y[2][3]+N[0][2]/2.0, Y[3][3]+N[0][3]/2.0, yt[0], yt[1], yt[2], yt[3]) ;$   
 $N[1][2]=h * (Y[3][3]+N[0][3]/2.0);$   
 $N[1][3]=h * FN2(t+h/2.0, Y[0][3]+N[0][0]/2.0, Y[1][3]+N[0][1]/2.0, Y[2][3]+N[0][2]/2.0, Y[3][3]+N[0][3]/2.0, yt[0], yt[1], yt[2], yt[3]) ;$

$N[2][0]=h * (Y[1][3]+N[1][1]/2.0);$   
 $N[2][1]=h * FN1(t+h/2.0, Y[0][3]+N[1][0]/2.0, Y[1][3]+N[1][1]/2.0, Y[2][3]+N[1][2]/2.0, Y[3][3]+N[1][3]/2.0, yt[0], yt[1], yt[2], yt[3]) ;$   
 $N[2][2]=h * (Y[3][3]+N[1][3]/2.0);$   
 $N[2][3]=h * FN2(t+h/2.0, Y[0][3]+N[1][0]/2.0, Y[1][3]+N[1][1]/2.0, Y[2][3]+N[1][2]/2.0, Y[3][3]+N[1][3]/2.0, yt[0], yt[1], yt[2], yt[3]) ;$

$N[3][0]=h * (Y[1][3]+N[2][1]) ;$   
 $N[3][1]=h * FN1(t+h, Y[0][3]+N[2][0], Y[1][3]+N[2][1], Y[2][3]+N[2][2], Y[3][3]+N[2][3], yt[0], yt[1], yt[2], yt[3]) ;$   
 $N[3][2]=h * (Y[3][3]+N[2][3]) ;$

```
N[3][3]=h * FN2(t+h, Y[0][3]+N[2][0], Y[1][3]+N[2][1], Y[2][3]+N[2][2], Y[3][3]+N[2][3], yt[0], yt[1], yt[2],
yt[3] );
```

```
for(l=0;l<n;l++)
{Y[l][3]=Y[l][3]+(N[0][l]+ 2.0*N[1][l]+ 2.0*N[2][l]+ N[3][l])/6.0 ;}
t=t+h;
for (j=0;j<n;j++)
{ yt[j]= yt[j] + (k1[0][j]+ 2.0*k1[1][j]+ 2.0*k1[2][j]+ k1[3][j])/6.0 ; }
}
for (i=0;i<n;i++)
{ytb[i]=yt[i];}
for (i=0;i<n;i++)
{for (j=0;j<n;j++)
{Yb[i][j]=Y[i][j];}}
```

```
for(i=0;i<n;i++)
{ for (j=0;j<n;j++)
{sum=0.0;
for (k=0;k<n;k++)
{sum= sum + Yb[i][k]*yta[k];
m_1[i]= sum ;
}}}
```

```
for(i=0;i<n;i++)
{ for (j=0;j<n;j++)
{sum=0.0;
sum= sum + m_1[i]-ytb[i];
l1[i]= sum ;
}}}
```

```
for(j=0;j<n;j++)
{ for (i=0;i<n;i++)
{sum=0.0;
for (k=0;k<n;k++)
{sum= sum + B[i][k]*Yb[k][j];
m_2[i][j]= sum ;
}}}
```

```
for(i=0;i<n;i++)
{ for (j=0;j<n;j++)
{sum=0.0;
for (k=0;k<n;k++)
{sum= sum + B[i][k]*l1[k];
m_3[i]= sum ;
}}}
```

```
for(i=0;i<n;i++)
{ for (j=0;j<n;j++)
{sum=0.0;
```

```

sum= sum + A[i][j]+m_2[i][j];
q[i][j]= sum ;
}}

for(i=0;i<n;i++)
{ for (j=0;j<n;j++)
{sum=0.0;
sum= sum + C[i]+m_3[i];
r[i]= sum ;
}}
for(i=0;i<n;i++)
{for(j=0;j<n;j++)
{a[i][j]=q[i][j];
}}

for(i=0;j<n;i++)
{for(j=0;j<n;j++)
{a1[i][j]=a[i][j];
}}

for (i=0;i<n;i++)
{sum1=sum1+a1[i][i];
}
p1=sum1/1;

for (i=0;i<n;i++)
{ for (j=0;j<n;j++)
{m1[i][j]=p1*a1[i][j];}}

for (i=0;i<n;i++)
{ for (j=0;j<n;j++)
{ sum_s=0.0;
sum_s=sum_s+a1[i][j]-m1[i][j];
ab1[i][j]=sum_s;}}

for(i=0;i<n;i++)
{ for (j=0;j<n;j++)
{sum_s=0.0;
for (k=0;k<n;k++)
{sum_s= sum_s + a[i][k]*ab1[k][j];
a2[i][j]= sum_s ;
}}}

for (i=0;i<n;i++)
{sum2=sum2+a2[i][i];
}
p2=sum2/2;
for (i=0;j<n;i++)

```

```

{ for (j=0;j<n;j++)
{m2[i][j]=p2*I[i][j];}}
for (i=0;i<n;i++)
{ for (j=0;j<n;j++)
{ sum_s=0.0;
sum_s=sum_s+a2[i][j]-m2[i][j];
ab2[i][j]=sum_s;}}
for(i=0;i<n;i++)
{ for (j=0;j<n;j++)
{sum_s=0.0;
for (k=0;k<n;k++)
{sum_s= sum_s + a[i][k]*ab2[k][j];
a3[i][j]= sum_s ;
}}}
for (i=0;i<n;i++)
{sum3=sum3+a3[i][i];
}
p3=sum3/3;
for (i=0;i<n;i++)
{ for (j=0;j<n;j++)
{m3[i][j]=p3*I[i][j];}}
for (i=0;i<n;i++)
{ for (j=0;j<n;j++)
{ sum_s=0.0;
sum_s=sum_s+a3[i][j]-m3[i][j];
ab3[i][j]=sum_s;}}

for(i=0;i<n;i++)
{ for (j=0;j<n;j++)
{sum_s=0.0;
for (k=0;k<n;k++)
{sum_s= sum_s + a[i][k]*ab3[k][j];
a4[i][j]= sum_s ;
}}}
for (i=0;i<n;i++)
{ sum4= sum4+ a4[i][i];}

p4= sum4/ 4 ;

for (i=0;i<n;i++)
{ for (j=0;j<n;j++)
{inverse[i][j]=ab3[i][j]/p4;}}

for(i=0;i<n;i++)
{ for (j=0;j<n;j++)
{sum_s=0.0;
for (k=0;k<n;k++)
{sum_s= sum_s + inverse[i][k]*q[k][j];

```



```

test[i][j]= sum_s ;
}}}

for(i=0;i<n;i++)
{ for (j=0;j<n;j++)
{sum_s=0.0;
for (k=0;k<n;k++)
{sum_s= sum_s + inverse[i][k]*r[k];
ya[i]= sum_s ;
}}}

for(i=0;i<n;i++)
{ for (j=0;j<n;j++)
{sum_s=0.0;
for (k=0;k<n;k++)
{sum_s= sum_s + Yb[i][k]*ya[k];
m4[i]= sum_s ;
}}}
y1old=yb[0];
y2old=yb[1];
y3old=yb[2];
y4old=yb[3];
for (i=0;i<n;i++)
{ for (j=0;j<n;j++)
{ sum_s=0.0;
sum_s=sum_s+m4[i]-11[i];
yb[i]=sum_s;}}

y1new=yb[0] ;
y2new=yb[1] ;
y3new=yb[2] ;
y4new=yb[3] ;
for (b=0;b<n;b++)
{yt[b]=ya[b];}

convgnc1= fabs(y1new-y1old) ;
convgnc2= fabs(y2new-y2old) ;
convgnc3= fabs(y3new-y3old) ;
convgnc4= fabs(y4new-y4old) ;

} /* for acc loop */ /* while loop end */

for (b=0;b<n;b++)
{yt[b]=ya[b];}
t=0.0;

for (k=0;k<n;k++)
{ for (b=0;b<n;b++)

```

```

    {k1[k][b]=k2[k][b];}
for (k=0;k<n;k++)
    { for (b=0;b<n;b++)
      {K[k][b]=k2[k][b];}}
for (k=0;k<n;k++)
    { for (b=0;b<n;b++)
      {L[k][b]=k2[k][b];}}
for (k=0;k<n;k++)
    { for (b=0;b<n;b++)
      {M[k][b]=k2[k][b];}}
for (k=0;k<n;k++)
    { for (b=0;b<n;b++)
      {N[k][b]=k2[k][b];}}
for (k=0;k<n;k++)
    { for (b=0;b<n;b++)
      {Y[k][b]=Y1[k][b];}}

fputs (" t(s)\t k_1/f1 *y1(x1)(m)\t y2(m/s)\t k_1/f1 *y3(x2)(m)\t y4(m/s) \n",fp);
fprintf(fp," %ft %ft %ft %ft %f\n ",t,k_1/f1*yt[0],yt[1],k_1/f1*yt[2],yt[3]);
for (i=0;i<(n1-1);i++)
    {
k1[0][0]=h * yt[1];
k1[0][1]=h * fn1(t, yt[0],yt[1], yt[2], yt[3], w);
k1[0][2]=h * yt[3];
k1[0][3]=h * fn2(t, yt[0],yt[1], yt[2], yt[3], w);

k1[1][0]=h * (yt[1]+k1[0][1]/2.0);
k1[1][1]=h * fn1(t+h/2.0, yt[0]+k1[0][0]/2.0, yt[1]+k1[0][1]/2.0, yt[2]+k1[0][2]/2.0, yt[3]+k1[0][3]/2.0, w) ;
k1[1][2]=h * (yt[3]+k1[0][3]/2.0);
k1[1][3]=h * fn2(t+h/2.0, yt[0]+k1[0][0]/2.0, yt[1]+k1[0][1]/2.0, yt[2]+k1[0][2]/2.0, yt[3]+k1[0][3]/2.0, w) ;

k1[2][0]=h * (yt[1]+k1[1][1]/2.0);
k1[2][1]=h * fn1(t+h/2.0, yt[0]+k1[1][0]/2.0, yt[1]+k1[1][1]/2.0, yt[2]+k1[1][2]/2.0, yt[3]+k1[1][3]/2.0, w) ;
k1[2][2]=h * (yt[3]+k1[1][3]/2.0);
k1[2][3]=h * fn2(t+h/2.0, yt[0]+k1[1][0]/2.0, yt[1]+k1[1][1]/2.0, yt[2]+k1[1][2]/2.0, yt[3]+k1[1][3]/2.0, w) ;

k1[3][0]=h * (yt[1]+k1[2][1]);
k1[3][1]=h * fn1(t+h, yt[0]+k1[2][0], yt[1]+k1[2][1], yt[2]+k1[2][2], yt[3]+k1[2][3], w) ;
k1[3][2]=h * (yt[3]+k1[2][3]);
k1[3][3]=h * fn2(t+h, yt[0]+k1[2][0], yt[1]+k1[2][1], yt[2]+k1[2][2], yt[3]+k1[2][3], w) ;

K[0][0]=h * Y[1][0] ;
K[0][1]=h * FN1(t, Y[0][0], Y[1][0], Y[2][0], Y[3][0], yt[0], yt[1], yt[2], yt[3] ) ;
K[0][2]=h * Y[3][0] ;
K[0][3]=h * FN2(t, Y[0][0], Y[1][0], Y[2][0], Y[3][0], yt[0], yt[1], yt[2], yt[3] ) ;

K[1][0]=h * (Y[1][0]+K[0][1]/2.0) ;

```

$K[1][1]=h * FN1(t+h/2.0, Y[0][0]+K[0][0]/2.0, Y[1][0]+K[0][1]/2.0, Y[2][0]+K[0][2]/2.0, Y[3][0]+K[0][3]/2.0, yt[0], yt[1], yt[2], yt[3] ) ;$   
 $K[1][2]=h * (Y[3][0]+K[0][3]/2.0) ;$   
 $K[1][3]=h * FN2(t+h/2.0, Y[0][0]+K[0][0]/2.0, Y[1][0]+K[0][1]/2.0, Y[2][0]+K[0][2]/2.0, Y[3][0]+K[0][3]/2.0, yt[0], yt[1], yt[2], yt[3] ) ;$

$K[2][0]=h * (Y[1][0]+K[1][1]/2.0) ;$   
 $K[2][1]=h * FN1(t+h/2.0, Y[0][0]+K[1][0]/2.0, Y[1][0]+K[1][1]/2.0, Y[2][0]+K[1][2]/2.0, Y[3][0]+K[1][3]/2.0, yt[0], yt[1], yt[2], yt[3] ) ;$   
 $K[2][2]=h * (Y[3][0]+K[1][3]/2.0) ;$   
 $K[2][3]=h * FN2(t+h/2.0, Y[0][0]+K[1][0]/2.0, Y[1][0]+K[1][1]/2.0, Y[2][0]+K[1][2]/2.0, Y[3][0]+K[1][3]/2.0, yt[0], yt[1], yt[2], yt[3] ) ;$

$K[3][0]=h * (Y[1][0]+K[2][1]) ;$   
 $K[3][1]=h * FN1(t+h, Y[0][0]+K[2][0], Y[1][0]+K[2][1], Y[2][0]+K[2][2], Y[3][0]+K[2][3], yt[0], yt[1], yt[2], yt[3] ) ;$   
 $K[3][2]=h * (Y[3][0]+K[2][3]) ;$   
 $K[3][3]=h * FN2(t+h, Y[0][0]+K[2][0], Y[1][0]+K[2][1], Y[2][0]+K[2][2], Y[3][0]+K[2][3], yt[0], yt[1], yt[2], yt[3] ) ;$

$for(i=0; i<n; i++)$   
 $\{ Y[i][0]=Y[i][0] + (K[0][i] + 2.0*K[1][i] + 2.0*K[2][i] + K[3][i])/6.0 ; \}$

$L[0][0]=h * Y[1][1] ;$   
 $L[0][1]=h * FN1(t, Y[0][1], Y[1][1], Y[2][1], Y[3][1], yt[0], yt[1], yt[2], yt[3] ) ;$   
 $L[0][2]=h * Y[3][1] ;$   
 $L[0][3]=h * FN2(t, Y[0][1], Y[1][1], Y[2][1], Y[3][1], yt[0], yt[1], yt[2], yt[3] ) ;$

$L[1][0]=h * (Y[1][1]+L[0][1]/2.0) ;$   
 $L[1][1]=h * FN1(t+h/2.0, Y[0][1]+L[0][0]/2.0, Y[1][1]+L[0][1]/2.0, Y[2][1]+L[0][2]/2.0, Y[3][1]+L[0][3]/2.0, yt[0], yt[1], yt[2], yt[3] ) ;$   
 $L[1][2]=h * (Y[3][1]+L[0][3]/2.0) ;$   
 $L[1][3]=h * FN2(t+h/2.0, Y[0][1]+L[0][0]/2.0, Y[1][1]+L[0][1]/2.0, Y[2][1]+L[0][2]/2.0, Y[3][1]+L[0][3]/2.0, yt[0], yt[1], yt[2], yt[3] ) ;$

$L[2][0]=h * (Y[1][1]+L[1][1]/2.0) ;$   
 $L[2][1]=h * FN1(t+h/2.0, Y[0][1]+L[1][0]/2.0, Y[1][1]+L[1][1]/2.0, Y[2][1]+L[1][2]/2.0, Y[3][1]+L[1][3]/2.0, yt[0], yt[1], yt[2], yt[3] ) ;$   
 $L[2][2]=h * (Y[3][1]+L[1][3]/2.0) ;$   
 $L[2][3]=h * FN2(t+h/2.0, Y[0][1]+L[1][0]/2.0, Y[1][1]+L[1][1]/2.0, Y[2][1]+L[1][2]/2.0, Y[3][1]+L[1][3]/2.0, yt[0], yt[1], yt[2], yt[3] ) ;$

$L[3][0]=h * (Y[1][1]+L[2][1]) ;$   
 $L[3][1]=h * FN1(t+h, Y[0][1]+L[2][0], Y[1][1]+L[2][1], Y[2][1]+L[2][2], Y[3][1]+L[2][3], yt[0], yt[1], yt[2], yt[3] ) ;$   
 $L[3][2]=h * (Y[3][1]+L[2][3]) ;$   
 $L[3][3]=h * FN2(t+h, Y[0][1]+L[2][0], Y[1][1]+L[2][1], Y[2][1]+L[2][2], Y[3][1]+L[2][3], yt[0], yt[1], yt[2], yt[3] ) ;$

```

for(l=0;l<n;l++)
{Y[l][1]=Y[l][1]+ (L[0][l]+ 2.0*L[1][l]+ 2.0*L[2][l]+ L[3][l])/6.0 ;}

M[0][0]=h * Y[1][2] ;
M[0][1]=h * FN1(t, Y[0][2],Y[1][2], Y[2][2], Y[3][2], yt[0], yt[1], yt[2], yt[3] ) ;
M[0][2]=h * Y[3][2] ;
M[0][3]=h * FN2(t, Y[0][2],Y[1][2], Y[2][2], Y[3][2], yt[0], yt[1], yt[2], yt[3] ) ;

M[1][0]=h * (Y[1][2]+M[0][1]/2.0 ) ;
M[1][1]=h * FN1(t+h/2.0, Y[0][2]+M[0][0]/2.0, Y[1][2]+M[0][1]/2.0, Y[2][2]+M[0][2]/2.0, Y[3][2]+M[0][3]/2.0,
yt[0], yt[1], yt[2], yt[3] ) ;
M[1][2]=h * (Y[3][2]+M[0][3]/2.0 ) ;
M[1][3]=h * FN2(t+h/2.0, Y[0][2]+M[0][0]/2.0, Y[1][2]+M[0][1]/2.0, Y[2][2]+M[0][2]/2.0, Y[3][2]+M[0][3]/2.0,
yt[0], yt[1], yt[2], yt[3] ) ;

M[2][0]=h * (Y[1][2]+M[1][1]/2.0) ;
M[2][1]=h * FN1(t+h/2.0, Y[0][2]+M[1][0]/2.0, Y[1][2]+M[1][1]/2.0, Y[2][2]+M[1][2]/2.0, Y[3][2]+M[1][3]/2.0,
yt[0], yt[1], yt[2], yt[3] ) ;
M[2][2]=h * (Y[3][2]+M[1][3]/2.0) ;
M[2][3]=h * FN2(t+h/2.0, Y[0][2]+M[1][0]/2.0, Y[1][2]+M[1][1]/2.0, Y[2][2]+M[1][2]/2.0, Y[3][2]+M[1][3]/2.0,
yt[0], yt[1], yt[2], yt[3] ) ;

M[3][0]=h * (Y[1][2]+M[2][1] ) ;
M[3][1]=h * FN1(t+h, Y[0][2]+M[2][0], Y[1][2]+M[2][1], Y[2][2]+M[2][2], Y[3][2]+M[2][3], yt[0], yt[1], yt[2],
yt[3] ) ;
M[3][2]=h * (Y[3][2]+M[2][3] ) ;
M[3][3]=h * FN2(t+h, Y[0][2]+M[2][0], Y[1][2]+M[2][1], Y[2][2]+M[2][2], Y[3][2]+M[2][3], yt[0], yt[1], yt[2],
yt[3] ) ;

for(l=0;l<n;l++)
{Y[l][2]=Y[l][2]+ (M[0][l]+ 2.0*M[1][l]+ 2.0*M[2][l]+ M[3][l])/6.0 ;}

N[0][0]=h * Y[1][3];
N[0][1]=h * FN1(t, Y[0][3], Y[1][3], Y[2][3], Y[3][3], yt[0], yt[1], yt[2], yt[3] ) ;
N[0][2]=h * Y[3][3];
N[0][3]=h * FN2(t, Y[0][3], Y[1][3], Y[2][3], Y[3][3], yt[0], yt[1], yt[2], yt[3] ) ;

N[1][0]=h * (Y[1][3]+N[0][1]/2.0);
N[1][1]=h * FN1(t+h/2.0, Y[0][3]+N[0][0]/2.0, Y[1][3]+N[0][1]/2.0, Y[2][3]+N[0][2]/2.0, Y[3][3]+N[0][3]/2.0,
yt[0], yt[1], yt[2], yt[3] ) ;
N[1][2]=h * (Y[3][3]+N[0][3]/2.0);
N[1][3]=h * FN2(t+h/2.0, Y[0][3]+N[0][0]/2.0, Y[1][3]+N[0][1]/2.0, Y[2][3]+N[0][2]/2.0, Y[3][3]+N[0][3]/2.0,
yt[0], yt[1], yt[2], yt[3] ) ;

N[2][0]=h * (Y[1][3]+N[1][1]/2.0);
N[2][1]=h * FN1(t+h/2.0, Y[0][3]+N[1][0]/2.0, Y[1][3]+N[1][1]/2.0, Y[2][3]+N[1][2]/2.0, Y[3][3]+N[1][3]/2.0,
yt[0], yt[1], yt[2], yt[3] ) ;

```

```

N[2][2]=h * (Y[3][3]+N[1][3]/2.0);
N[2][3]=h * FN2(t+h/2.0, Y[0][3]+N[1][0]/2.0, Y[1][3]+N[1][1]/2.0, Y[2][3]+N[1][2]/2.0, Y[3][3]+N[1][3]/2.0,
yt[0], yt[1], yt[2], yt[3] );

N[3][0]=h * (Y[1][3]+N[2][1] );
N[3][1]=h * FN1(t+h, Y[0][3]+N[2][0], Y[1][3]+N[2][1], Y[2][3]+N[2][2], Y[3][3]+N[2][3], yt[0], yt[1], yt[2],
yt[3] );
N[3][2]=h * (Y[3][3]+N[2][3] );
N[3][3]=h * FN2(t+h, Y[0][3]+N[2][0], Y[1][3]+N[2][1], Y[2][3]+N[2][2], Y[3][3]+N[2][3], yt[0], yt[1], yt[2],
yt[3] );

for(l=0;l<n;l++)
{Y[l][3]=Y[l][3]+ (N[0][l]+ 2.0*N[1][l]+ 2.0*N[2][l]+ N[3][l])/6.0 ;}

t=t+h;

for (j=0;j<n;j++)
{ yt[j]= yt[j] + (k1[0][j]+ 2.0*k1[1][j]+ 2.0*k1[2][j]+ k1[3][j])/6.0 ; }
printf("\n at t=%f y1=%ft y3=%ft \n",t,yt[0],yt[2]);
fprintf(fp," %ft %ft %ft %ft %f\n ",t,k_1/f1*yt[0],yt[1],k_1/f1*yt[2],yt[3]);
}
fprintf(fp," %ft %ft %ft %ft %f\n ",t+h,k_1/f1*yb[0],yb[1],k_1/f1*yb[2],yb[3]);
fclose(fp);

getch();
}

```

## C. 2 Code for Nondimensional Displacement with Forcing Frequency for 2 DOFS

```

#include <stdio.h>
#include <conio.h>
#include <math.h>
#define n 4
#define ma1 100
#define ma2 1.0
#define k_1 1000
#define k_1p 0.5
#define k_2 10
#define k_2p 0.5
#define c1 0.10
#define c1p 0.05
#define c2 10.0
#define c2p 0.5
#define f1 20.0
#define f2 0.0
#define tb 20.0
#define h 0.02
#define wf 30.0
#define wi 0
#define dw 0.1

/* mass m in kg : spring constant k in N/m : damping constant c in Ns/m : force amplitude f in N: angular frequency
w in rad/s : tb or t in second(s) */
double fn1(double t, double y1, double y2, double y3, double y4, double w1)
{
double z=0.0;
z= (f1*sin(w1*t)-(k_1*y1+k_1p*pow(y1,3))-(k_2*(y1-y3)+k_2p*pow((y1-y3),3))- (c1*y2+c1p*y2*pow(y1,2)) -
(c2*(y2-y4)+c2p*(y2-y4)*pow((y1-y3),2)) )/ma1 ; ;
return z;
}

double fn2(double t, double y1, double y2, double y3, double y4, double w1)
{
double z=0.0;
z= (k_2*(y1-y3)+k_2p*pow((y1-y3),3)+ (c2*(y2-y4)+c2p*(y2-y4)*pow((y1-y3),2)) )/ma2 ;
return z;
}

float FN1(float t, float Y0, float Y1, float Y2, float Y3, float y1, float y2, float y3, float y4)
{
float z=0.0;
z= -(k_1*Y0+k_1p*3*pow(y1,2)*Y0+k_2*(Y0-Y2)+3*k_2p*pow((y1-y3),2)*(Y0-Y2)+ c1*Y1+
c1p*Y1*pow(y1,2)+ 2*c1p*y2*y1*Y0+ c2*(Y1-Y3)+ c2p*pow((y1-y3),2)*(Y1-Y3)+ 2*c2p*(y1-y3)*(y2-
y4)*(Y0-Y2) )/ma1 ;

```

```

return z;
}
float FN2(float t,float Y0,float Y1, float Y2, float Y3, float y1, float y2, float y3, float y4)
{
float z=0.0;
z= (k_2*(Y0-Y2)+k_2p*3*pow((y1-y3),2)*(Y0-Y2)+ c2*(Y1-Y3)+ c2p*pow((y1-y3),2)*(Y1-Y3)+ 2*c2p*(y1-
y3)*(y2-y4)*(Y0-Y2) )/ma2 ;
return z;
}
void main()
{
int i,j,n1,l,b,k,u,acc,freq;
double yt[n];
float Y1[n][n]={{1.0},
                {0.0, 1.0},{0.0,0.0,1.0},{0.0,0.0,0.0,1.0}};
float n2,nf,t=0.0,w=0.0,y1new=0.0,y1old=1.0, convgnc1,
y2new=0.0,y2old=1.0,convgnc2,y3new=0.0,y3old=1.0,convgnc3,y4new=0.0,y4old=1.0,convgnc4;
double k1[n][n],K[n][n],L[n][n],M[n][n],N[n][n],Y[n][n];
double k2[n][n]={{0.0},{0.0},{0.0},{0.0}};
float m_1[n],m_2[n][n],m_3[n];
float l1[n],q[n][n],r[n],ya[n],yb[n], Yb[n][n];
float
a[n][n],a1[n][n],a2[n][n],a3[n][n],a4[n][n],m1[n][n],m2[n][n],m3[n][n],m4[n],ab1[n][n],ab2[n][n],ab3[n][n],inverse[
n][n],test[n][n];
float p1,p2,p3,p4,sum,sum_s,freq1;
float yta[n], ytb[n];
float ytini[n]={0.0,0.0,0.0,0.0};
float sum4,sum3,sum2,sum1;
FILE *fp;

float I[n][n]= {{1.0,0.0,0.0,0.0},
                {0.0,1.0,0.0,0.0},
                {0.0,0.0,1.0,0.0},
                {0.0,0.0,0.0,1.0}} ;

float A[n][n]={{1.0,0.0,0.0,0.0},
               {0.0,0.0,0.0,0.0},
               {0.0,0.0,1.0,0.0},
               {0.0,0.0,0.0,0.0}};

float B[n][n]={{0.0,0.0,0.0,0.0},
               {0.0,0.1,0.0,0.0},
               {0.0,0.0,0.0,0.0},
               {0.0,0.0,0.0,0.1}};

float C[n]={0.05,0.06,-0.07,-0.06};
n2=((tb-t)/h);
n1=n2;
w=wj;

```

```

freq1=(wf-w)/dw;
freq=freq1;
nf=pow((k_1/mal),0.5);
fp=fopen("d:\\result\\dabsrw.xls","w");

clrscr();

fputs ("t(s)\t w/w1\t k1/f1*y1\t k1/f1*y3(m) \n",fp);

for(u=0;u<=freq;u++)
{ y1new=0.0;
y2new=0.0;
y3new=0.0;
y4new= 0.0;
y1old=1.0;
y2old=1.0;
y3old=1.0;
y4old=1.0;
convgnc1=fabs(y1new-y1old);
convgnc2=fabs(y2new-y2old);
convgnc3=fabs(y3new-y3old);
convgnc4=fabs(y4new-y4old);

for (b=0;b<n;b++)
{yt[b]=ytini[b];}
for (b=0;b<n;b++)
{yta[b]=ytini[b];}
for (b=0;b<n;b++)
{ytb[b]=ytini[b];}
for (b=0;b<n;b++)
{r[b]=ytini[b];}
for (b=0;b<n;b++)
{l1[b]=ytini[b];}

for (k=0;k<n;k++)
{ for (b=0;b<n;b++)
{Yb[k][b]=k2[k][b];}}
for (k=0;k<n;k++)
{ for (b=0;b<n;b++)
{m2[k][b]=k2[k][b];}}
for (k=0;k<n;k++)
{ for (b=0;b<n;b++)
{q[k][b]=k2[k][b];}}

while (0.0001<convgnc1 && 0.0001<convgnc2 && 0.0001<convgnc3 && 0.0001<convgnc4)
{
yt[0]=0.05;
/* yt[1]=-0.724; */

```



```

yt[2]=0.07;
/* yt[3]=0.0183; */

    for (b=0;b<n;b++)
    {yta[b]=ytini[b];}
    for (b=0;b<n;b++)
    {ytb[b]=ytini[b];}
    for (b=0;b<n;b++)
    {r[b]=ytini[b];}
    for (b=0;b<n;b++)
    {l1[b]=ytini[b];}

    for (k=0;k<n;k++)
    { for (b=0;b<n;b++)
    {Yb[k][b]=k2[k][b];}}
    for (k=0;k<n;k++)
    { for (b=0;b<n;b++)
    {m2[k][b]=k2[k][b];}}
    for (k=0;k<n;k++)
    { for (b=0;b<n;b++)
    {q[k][b]=k2[k][b];}}

    for (b=0;b<n;b++)
    {yta[b]=yt[b];}
sum1=0.0;
sum2=0.0;
sum3=0.0;
sum4=0.0;
sum=0.0;
sum_s=0.0;
p1=0.0;
p2=0.0;
p3=0.0;
p4=0.0;
t=0.0;
    for (k=0;k<n;k++)
    { for (b=0;b<n;b++)
    {k1[k][b]=k2[k][b];}}
    for (k=0;k<n;k++)
    { for (b=0;b<n;b++)
    {K[k][b]=k2[k][b];}}
    for (k=0;k<n;k++)
    { for (b=0;b<n;b++)
    {L[k][b]=k2[k][b];}}
    for (k=0;k<n;k++)
    { for (b=0;b<n;b++)
    {M[k][b]=k2[k][b];}}
    for (k=0;k<n;k++)

```

```

{ for (b=0;b<n;b++)
  {N[k][b]=k2[k][b];}}
for (k=0;k<n;k++)
  { for (b=0;b<n;b++)
    {Y[k][b]=Y1[k][b];}}

for (i=0;i<n1;i++)
  {
k1[0][0]=h * yt[1];
k1[0][1]=h * fn1(t, yt[0],yt[1], yt[2], yt[3], w);
k1[0][2]=h * yt[3];
k1[0][3]=h * fn2(t, yt[0],yt[1], yt[2], yt[3], w);

k1[1][0]=h * (yt[1]+k1[0][1]/2.0);
k1[1][1]=h * fn1(t+h/2.0, yt[0]+k1[0][0]/2.0, yt[1]+k1[0][1]/2.0, yt[2]+k1[0][2]/2.0, yt[3]+k1[0][3]/2.0, w) ;
k1[1][2]=h * (yt[3]+k1[0][3]/2.0);
k1[1][3]=h * fn2(t+h/2.0, yt[0]+k1[0][0]/2.0, yt[1]+k1[0][1]/2.0, yt[2]+k1[0][2]/2.0, yt[3]+k1[0][3]/2.0, w) ;

k1[2][0]=h * (yt[1]+k1[1][1]/2.0);
k1[2][1]=h * fn1(t+h/2.0, yt[0]+k1[1][0]/2.0, yt[1]+k1[1][1]/2.0, yt[2]+k1[1][2]/2.0, yt[3]+k1[1][3]/2.0, w) ;
k1[2][2]=h * (yt[3]+k1[1][3]/2.0);
k1[2][3]=h * fn2(t+h/2.0, yt[0]+k1[1][0]/2.0, yt[1]+k1[1][1]/2.0, yt[2]+k1[1][2]/2.0, yt[3]+k1[1][3]/2.0, w) ;

k1[3][0]=h * (yt[1]+k1[2][1]);
k1[3][1]=h * fn1(t+h, yt[0]+k1[2][0], yt[1]+k1[2][1], yt[2]+k1[2][2], yt[3]+k1[2][3], w) ;
k1[3][2]=h * (yt[3]+k1[2][3]);
k1[3][3]=h * fn2(t+h, yt[0]+k1[2][0], yt[1]+k1[2][1], yt[2]+k1[2][2], yt[3]+k1[2][3], w) ;

K[0][0]=h * Y[1][0] ;
K[0][1]=h * FN1(t, Y[0][0], Y[1][0], Y[2][0], Y[3][0], yt[0], yt[1], yt[2], yt[3] ) ;
K[0][2]=h * Y[3][0] ;
K[0][3]=h * FN2(t, Y[0][0], Y[1][0], Y[2][0], Y[3][0], yt[0], yt[1], yt[2], yt[3] ) ;

K[1][0]=h * (Y[1][0]+K[0][1]/2.0) ;
K[1][1]=h * FN1(t+h/2.0, Y[0][0]+K[0][0]/2.0, Y[1][0]+K[0][1]/2.0, Y[2][0]+K[0][2]/2.0, Y[3][0]+K[0][3]/2.0,
yt[0], yt[1], yt[2], yt[3] ) ;
K[1][2]=h * (Y[3][0]+K[0][3]/2.0) ;
K[1][3]=h * FN2(t+h/2.0, Y[0][0]+K[0][0]/2.0, Y[1][0]+K[0][1]/2.0, Y[2][0]+K[0][2]/2.0, Y[3][0]+K[0][3]/2.0,
yt[0], yt[1], yt[2], yt[3] ) ;

K[2][0]=h * (Y[1][0]+K[1][1]/2.0) ;
K[2][1]=h * FN1(t+h/2.0, Y[0][0]+K[1][0]/2.0, Y[1][0]+K[1][1]/2.0, Y[2][0]+K[1][2]/2.0, Y[3][0]+K[1][3]/2.0,
yt[0], yt[1], yt[2], yt[3] ) ;
K[2][2]=h * (Y[3][0]+K[1][3]/2.0) ;
K[2][3]=h * FN2(t+h/2.0, Y[0][0]+K[1][0]/2.0, Y[1][0]+K[1][1]/2.0, Y[2][0]+K[1][2]/2.0, Y[3][0]+K[1][3]/2.0,
yt[0], yt[1], yt[2], yt[3] ) ;

K[3][0]=h * (Y[1][0]+K[2][1]) ;

```

```

K[3][1]=h * FN1(t+h, Y[0][0]+K[2][0], Y[1][0]+K[2][1], Y[2][0]+K[2][2], Y[3][0]+K[2][3], yt[0], yt[1], yt[2],
yt[3] );
K[3][2]=h * (Y[3][0]+K[2][3] );
K[3][3]=h * FN2(t+h, Y[0][0]+K[2][0], Y[1][0]+K[2][1], Y[2][0]+K[2][2], Y[3][0]+K[2][3], yt[0], yt[1], yt[2],
yt[3] );

for(l=0;l<n;l++)
{Y[l][0]=Y[l][0]+ (K[0][l]+ 2.0*K[1][l]+ 2.0*K[2][l]+ K[3][l])/6.0 ;}

L[0][0]=h * Y[1][1];
L[0][1]=h * FN1(t, Y[0][1],Y[1][1], Y[2][1], Y[3][1], yt[0], yt[1], yt[2], yt[3] );
L[0][2]=h * Y[3][1];
L[0][3]=h * FN2(t, Y[0][1],Y[1][1], Y[2][1], Y[3][1], yt[0], yt[1], yt[2], yt[3] );

L[1][0]=h * (Y[1][1]+L[0][1]/2.0 );
L[1][1]=h * FN1(t+h/2.0, Y[0][1]+L[0][0]/2.0, Y[1][1]+L[0][1]/2.0, Y[2][1]+L[0][2]/2.0, Y[3][1]+L[0][3]/2.0,
yt[0], yt[1], yt[2], yt[3] );
L[1][2]=h * (Y[3][1]+L[0][3]/2.0 );
L[1][3]=h * FN2(t+h/2.0, Y[0][1]+L[0][0]/2.0, Y[1][1]+L[0][1]/2.0, Y[2][1]+L[0][2]/2.0, Y[3][1]+L[0][3]/2.0,
yt[0], yt[1], yt[2], yt[3] );

L[2][0]=h * (Y[1][1]+L[1][1]/2.0 );
L[2][1]=h * FN1(t+h/2.0, Y[0][1]+L[1][0]/2.0, Y[1][1]+L[1][1]/2.0, Y[2][1]+L[1][2]/2.0, Y[3][1]+L[1][3]/2.0,
yt[0], yt[1], yt[2], yt[3] );
L[2][2]=h * (Y[3][1]+L[1][3]/2.0 );
L[2][3]=h * FN2(t+h/2.0, Y[0][1]+L[1][0]/2.0, Y[1][1]+L[1][1]/2.0, Y[2][1]+L[1][2]/2.0, Y[3][1]+L[1][3]/2.0,
yt[0], yt[1], yt[2], yt[3] );

L[3][0]=h * (Y[1][1]+L[2][1] );
L[3][1]=h * FN1(t+h, Y[0][1]+L[2][0], Y[1][1]+L[2][1], Y[2][1]+L[2][2], Y[3][1]+L[2][3], yt[0], yt[1], yt[2], yt[3]
);
L[3][2]=h * (Y[3][1]+L[2][3] );
L[3][3]=h * FN2(t+h, Y[0][1]+L[2][0], Y[1][1]+L[2][1], Y[2][1]+L[2][2], Y[3][1]+L[2][3], yt[0], yt[1], yt[2], yt[3]
);

for(l=0;l<n;l++)
{Y[l][1]=Y[l][1]+ (L[0][l]+ 2.0*L[1][l]+ 2.0*L[2][l]+ L[3][l])/6.0 ;}

M[0][0]=h * Y[1][2];
M[0][1]=h * FN1(t, Y[0][2],Y[1][2], Y[2][2], Y[3][2], yt[0], yt[1], yt[2], yt[3] );
M[0][2]=h * Y[3][2];
M[0][3]=h * FN2(t, Y[0][2],Y[1][2], Y[2][2], Y[3][2], yt[0], yt[1], yt[2], yt[3] );

M[1][0]=h * (Y[1][2]+M[0][1]/2.0 );
M[1][1]=h * FN1(t+h/2.0, Y[0][2]+M[0][0]/2.0, Y[1][2]+M[0][1]/2.0, Y[2][2]+M[0][2]/2.0, Y[3][2]+M[0][3]/2.0,
yt[0], yt[1], yt[2], yt[3] );
M[1][2]=h * (Y[3][2]+M[0][3]/2.0 );

```

M[1][3]=h \* FN2(t+h/2.0, Y[0][2]+M[0][0]/2.0, Y[1][2]+M[0][1]/2.0, Y[2][2]+M[0][2]/2.0, Y[3][2]+M[0][3]/2.0, yt[0], yt[1], yt[2], yt[3] );

M[2][0]=h \* (Y[1][2]+M[1][1]/2.0);  
M[2][1]=h \* FN1(t+h/2.0, Y[0][2]+M[1][0]/2.0, Y[1][2]+M[1][1]/2.0, Y[2][2]+M[1][2]/2.0, Y[3][2]+M[1][3]/2.0, yt[0], yt[1], yt[2], yt[3] );  
M[2][2]=h \* (Y[3][2]+M[1][3]/2.0);  
M[2][3]=h \* FN2(t+h/2.0, Y[0][2]+M[1][0]/2.0, Y[1][2]+M[1][1]/2.0, Y[2][2]+M[1][2]/2.0, Y[3][2]+M[1][3]/2.0, yt[0], yt[1], yt[2], yt[3] );

M[3][0]=h \* (Y[1][2]+M[2][1] );  
M[3][1]=h \* FN1(t+h, Y[0][2]+M[2][0], Y[1][2]+M[2][1], Y[2][2]+M[2][2], Y[3][2]+M[2][3], yt[0], yt[1], yt[2], yt[3] );  
M[3][2]=h \* (Y[3][2]+M[2][3] );  
M[3][3]=h \* FN2(t+h, Y[0][2]+M[2][0], Y[1][2]+M[2][1], Y[2][2]+M[2][2], Y[3][2]+M[2][3], yt[0], yt[1], yt[2], yt[3] );

for(i=0;i<n;i++)  
{Y[1][2]=Y[1][2]+ (M[0][1]+ 2.0\*M[1][1]+ 2.0\*M[2][1]+ M[3][1])/6.0 ;}

N[0][0]=h \* Y[1][3];  
N[0][1]=h \* FN1(t, Y[0][3], Y[1][3], Y[2][3], Y[3][3], yt[0], yt[1], yt[2], yt[3] );  
N[0][2]=h \* Y[3][3];  
N[0][3]=h \* FN2(t, Y[0][3], Y[1][3], Y[2][3], Y[3][3], yt[0], yt[1], yt[2], yt[3] );

N[1][0]=h \* (Y[1][3]+N[0][1]/2.0);  
N[1][1]=h \* FN1(t+h/2.0, Y[0][3]+N[0][0]/2.0, Y[1][3]+N[0][1]/2.0, Y[2][3]+N[0][2]/2.0, Y[3][3]+N[0][3]/2.0, yt[0], yt[1], yt[2], yt[3] );  
N[1][2]=h \* (Y[3][3]+N[0][3]/2.0);  
N[1][3]=h \* FN2(t+h/2.0, Y[0][3]+N[0][0]/2.0, Y[1][3]+N[0][1]/2.0, Y[2][3]+N[0][2]/2.0, Y[3][3]+N[0][3]/2.0, yt[0], yt[1], yt[2], yt[3] );

N[2][0]=h \* (Y[1][3]+N[1][1]/2.0);  
N[2][1]=h \* FN1(t+h/2.0, Y[0][3]+N[1][0]/2.0, Y[1][3]+N[1][1]/2.0, Y[2][3]+N[1][2]/2.0, Y[3][3]+N[1][3]/2.0, yt[0], yt[1], yt[2], yt[3] );  
N[2][2]=h \* (Y[3][3]+N[1][3]/2.0);  
N[2][3]=h \* FN2(t+h/2.0, Y[0][3]+N[1][0]/2.0, Y[1][3]+N[1][1]/2.0, Y[2][3]+N[1][2]/2.0, Y[3][3]+N[1][3]/2.0, yt[0], yt[1], yt[2], yt[3] );

N[3][0]=h \* (Y[1][3]+N[2][1] );  
N[3][1]=h \* FN1(t+h, Y[0][3]+N[2][0], Y[1][3]+N[2][1], Y[2][3]+N[2][2], Y[3][3]+N[2][3], yt[0], yt[1], yt[2], yt[3] );  
N[3][2]=h \* (Y[3][3]+N[2][3] );  
N[3][3]=h \* FN2(t+h, Y[0][3]+N[2][0], Y[1][3]+N[2][1], Y[2][3]+N[2][2], Y[3][3]+N[2][3], yt[0], yt[1], yt[2], yt[3] );

for(i=0;i<n;i++)  
{Y[1][3]=Y[1][3]+ (N[0][1]+ 2.0\*N[1][1]+ 2.0\*N[2][1]+ N[3][1])/6.0 ;}

```

t=t+h;
for (j=0;j<n;j++)
{ yt[j]=yt[j] + (k1[0][j]+ 2.0*k1[1][j]+ 2.0*k1[2][j]+ k1[3][j])/6.0 ; }
}
for (i=0;i<n;i++)
{ytb[i]=yt[i];}
for (i=0;i<n;i++)
{for (j=0;j<n;j++)
{Yb[i][j]=Y[i][j];}}

for(i=0;i<n;i++)
{ for (j=0;j<n;j++)
{sum=0.0;
for (k=0;k<n;k++)
{sum= sum + Yb[i][k]*yta[k];
m_1[j]= sum ;
}}}

for(i=0;i<n;i++)
{ for (j=0;j<n;j++)
{sum=0.0;
sum= sum + m_1[i]-ytb[i];
l1[i]= sum ;
}}

for(i=0;j<n;i++)
{ for (j=0;j<n;j++)
{sum=0.0;
for (k=0;k<n;k++)
{sum= sum + B[i][k]*Yb[k][j];
m_2[i][j]= sum ;
}}}

for(i=0;i<n;i++)
{ for (j=0;j<n;j++)
{sum=0.0;
for (k=0;k<n;k++)
{sum= sum + B[i][k]*l1[k];
m_3[i]= sum ;
}}}

for(i=0;i<n;i++)
{ for (j=0;j<n;j++)
{sum=0.0;
sum= sum + A[i][j]+m_2[i][j];
q[i][j]= sum ;
}}

```

```

for(i=0;i<n;i++)
{ for (j=0;j<n;j++)
  {sum=0.0;
  sum= sum + C[i]+m_3[i];
  r[i]= sum ;
  }}

```

```

for(i=0;i<n;i++)
  {for(j=0;j<n;j++)
  {a[i][j]=q[i][j];
  }}

```

```

for(i=0;i<n;i++)
  {for(j=0;j<n;j++)
  {a1[i][j]=a[i][j];
  }}

```

```

for (i=0;i<n;i++)
{sum1=sum1+a1[i][i];
}
p1=sum1/1;

```

```

for (i=0;i<n;i++)
  { for (j=0;j<n;j++)
  {m1[i][j]=p1*I[i][j];}}

```

```

for (i=0;i<n;i++)
  { for (j=0;j<n;j++)
  { sum_s=0.0;
  sum_s=sum_s+a1[i][j]-m1[i][j];
  ab1[i][j]=sum_s;}}

```

```

  for(i=0;i<n;i++)
  { for (j=0;j<n;j++)
  {sum_s=0.0;
  for (k=0;k<n;k++)
  {sum_s= sum_s + a[i][k]*ab1[k][j];
  a2[i][j]= sum_s ;
  }}}

```

```

for (i=0;i<n;i++)
{sum2=sum2+a2[i][i];
}
p2=sum2/2;

```

```

for (i=0;i<n;i++)
  { for (j=0;j<n;j++)

```

```

{m2[i][j]=p2*I[i][j];}

for (i=0;i<n;i++)
{ for (j=0;j<n;j++)
{ sum_s=0.0;
sum_s=sum_s+a2[i][j]-m2[i][j];
ab2[i][j]=sum_s;}}
  for(i=0;i<n;i++)
{ for (j=0;j<n;j++)
{sum_s=0.0;
for (k=0;k<n;k++)
{sum_s= sum_s + a[i][k]*ab2[k][j];
a3[i][j]= sum_s ;
}}}
for (i=0;i<n;i++)
{sum3=sum3+a3[i][i];
}
p3=sum3/3;

for (i=0;i<n;i++)
{ for (j=0;j<n;j++)
{m3[i][j]=p3*I[i][j];}}

for (i=0;i<n;i++)
{ for (j=0;j<n;j++)
{ sum_s=0.0;
sum_s=sum_s+a3[i][j]-m3[i][j];
ab3[i][j]=sum_s;}}

  for(i=0;i<n;i++)
{ for (j=0;j<n;j++)
{sum_s=0.0;
for (k=0;k<n;k++)
{sum_s= sum_s + a[i][k]*ab3[k][j];
a4[i][j]= sum_s ;
}}}
for (i=0;i<n;i++)
{ sum4= sum4+ a4[i][i];}

p4= sum4/ 4 ;

for (i=0;i<n;i++)
{ for (j=0;j<n;j++)
{inverse[i][j]=ab3[i][j]/p4;}}

for(i=0;i<n;i++)
{ for (j=0;j<n;j++)
{sum_s=0.0;

```

```

for (k=0;k<n;k++)
{sum_s= sum_s + inverse[i][k]*q[k][j];
test[i][j]= sum_s ;
}}}
for(i=0;i<n;i++)
{ for (j=0;j<n;j++)
{sum_s=0.0;
for (k=0;k<n;k++)
{sum_s= sum_s + inverse[i][k]*r[k];
ya[i]= sum_s ;
}}}
for(i=0;i<n;i++)
{ for (j=0;j<n;j++)
{sum_s=0.0;
for (k=0;k<n;k++)
{sum_s= sum_s + Yb[i][k]*ya[k];
m4[i]= sum_s ;
}}}
y1old=yb[0];
y2old=yb[1];
y3old=yb[2];
y4old=yb[3];
for (i=0;i<n;i++)
{ for (j=0;j<n;j++)
{ sum_s=0.0;
sum_s=sum_s+ m4[i]-l[i];
yb[i]=sum_s;}}
y1new=yb[0] ;
y2new=yb[1] ;
y3new=yb[2] ;
y4new=yb[3] ;

for (b=0;b<n;b++)
{yt[b]=ya[b];}

convgnc1= fabs(y1new-y1old) ;
convgnc2= fabs(y2new-y2old) ;
convgnc3= fabs(y3new-y3old) ;
convgnc4= fabs(y4new-y4old) ;

} /* for acc loop */

printf("\n at t=%f w=%f y1=%f y3=%f\n",t,w,yb[0],yb[2]);
fprintf(fp," %ft %ft %ft %f \n ",t,w/nf,k_1/f!*yb[0],k_1/f!*yb[2]);

w=w+dw;
}
fclose(fp);

```



```
getch();  
}
```

### C.3 Code for Nondimensional Displacement with Time for 3 DOFS

```
# include <stdio.h>
# include <conio.h>
# include <math.h>
# include <stdlib.h>
# define n 6
# define ma1 20.0
# define ma2 20.0
# define ma3 20.0
# define k_1 50.0
# define k_1p -0.05
# define k_2 50.0
# define k_2p -0.05
# define k_3 50.0
# define k_3p -0.05
# define c1 0.0
# define c1p 0.0
# define c2 0.0
# define c2p 0.0
# define c3 0.0
# define c3p 0.0
# define f1 20.0
# define f2 0.0
# define f3 0.0
# define tb 50.0
# define h 0.02
# define w 50.0
# define zero 0.0

/* mass m in kg : spring constant k in N/m : damping constant c in Ns/m : force amplitude f in N: angular frequency
w in rad/s : tb or t in second(s) */

double fn1(double t ,double y1,double y2, double y3, double y4, double y5, double y6, double w1)
{
double z=0.0;
z= (-k_1*y1-k_1p*pow(y1,3)-k_2*y1+k_2*y3-k_2p*pow(y1-y3,3)-c1*y2-c1p*y2*pow(y1,2)-c2*y2+c2*y4-
c2p*(y2-y4)*pow(y1-y3,2)-f1*sin(w*t))/ma1 ;
return z;
}

double fn2(double t, double y1,double y2, double y3, double y4, double y5, double y6, double w1)
{
double z=0.0;
```

```

z= (-k_3*(y3-y5)+k_3p*pow(y3-y5,2)+c3*(y4-y6)+c3p*(y4-y6)*pow(y2-y5,2)-k_2*(y1-y3)-k_2p*pow(y1-y3,3)-
c2*(y2-y4)-c2p*(y2-y4)*pow(y1-y3,2)-f2*sin(w*t)) /ma2 ;
return z;
}

```

```

double fn3(double t, double y1, double y2, double y3, double y4, double y5, double y6, double w1)
{
double z=0.0;
z= (-k_3*(y3-y5)-k_3p*pow(y3-y5,3)-c3*(y4-y6)-c3p*(y4-y6)*pow(y3-y5,2)-f3*sin(w*t)) /ma3 ;
return z;
}

```

```

float FN1(float t, float Y0, float Y1, float Y2, float Y3, float Y4, float Y5, float y1, float y2, float y3, float y4, float
y5, float y6)
{
float z=0.0;
z= (-k_1*Y0-k_1p*3*pow(y1,2)*Y0-k_2*(Y0-Y2)-k_2p*(Y0-Y2)*3*pow(y1-y3,2)-c1*Y1-c1p*Y1*pow
(y1,2)-c1p*2*y1*y2*Y0-c2*(Y1-Y3)-c2p*pow(y1-y3,2)*(Y1-Y3)-c2p*2*(y2-y4)*(y1-y3)*(Y0-Y2)) /ma1;
return z;
}

```

```

float FN2(float t, float Y0, float Y1, float Y2, float Y3, float Y4, float Y5, float y1, float y2, float y3, float y4, float
y5, float y6)
{
float z=0.0;
z= (-k_3*(Y2-Y4)+k_3p*2*(y3-y5)*(Y2-Y4)+c3*(Y3-Y5)+2*c3p*(y2-y5)*(y4-y6)*(Y1-Y4)+c3p*pow(y2-
y5,2)*(Y3-Y5)-k_2*(Y0-Y2)-k_2p*3*pow(y1-y3,2)*(Y0-Y2)-c2*(Y1-Y3)-c2p*pow(y1-y3,2)*(Y1-Y3)-
c2p*2*(y2-y4)*(y1-y3)*(Y0-Y2)) /ma2;
return z;
}

```

```

float FN3(float t, float Y0, float Y1, float Y2, float Y3, float Y4, float Y5, float y1, float y2, float y3, float y4, float
y5, float y6)
{
float z=0.0;
z= (-k_3*(Y2-Y4)-k_3p*3*pow(y3-y5,2)*(Y2-Y4)-c3*(Y3-Y5)-c3p*pow(y3-y5,2)*(Y3-Y5)-c3p*(y4-
y6)*2*(y3-y5)*(Y2-Y4)) /ma3;
return z;
}

```

```

void main()
{
int i,j,n1,l,b,k,u,acc,freq,z;
double yt[n];
float Y1[n][n] ={{1.0},
{0.0, 1.0},{0.0,0.0,1.0},{0.0,0.0,0.0,1.0},{0.0,0.0,0.0,0.0,1.0},{0.0,0.0,0.0,0.0,0.0,1.0}};

```

```

float n2,t,y1new=0.0,y1old=1.0, convgnc1,
y2new=0.0,y2old=1.0,convgnc2,y3new=0.0,y3old=1.0,convgnc3,y4new=0.0,y4old=1.0,convgnc4,y5new=0.0,
y5old=1.0, convgnc5, y6new=0.0, y6old=1.0, convgnc6;
double k1[n][n],K[n][n],L[n][n],M[n][n],N[n][n],Y[n][n];
double k2[n][n]={0.0},{0.0},{0.0},{0.0},{0.0},{0.0}};
float m_1[n],m_2[n][n],m_3[n],m_4[n][n];
float l1[n],q[n][n],r[n],ya[n],yb[n], Yb[n][n];
float
a[n][n],a1[n][n],a2[n][n],a3[n][n],a4[n][n],a5[n][n],a6[n][n],m1[n][n],m2[n][n],m3[n][n],m4[n],m5[n][n],ab1[n][n],a
b2[n][n],ab3[n][n],ab4[n][n],ab5[n][n],inverse[n][n],tcst[n][n];
float p1,p2,p3,p4,p5,p6,sum,sum_s,freq1;
float yta[n], ytb[n];
float ytini[n]={0.0,0.0,0.0,0.0,0.0,0.0};
float sum6,sum5,sum4,sum3,sum2,sum1;
FILE *fp;

float I[n][n]= {{ 1.0,0.0,0.0,0.0,0.0,0.0},
                {0.0,1.0,0.0,0.0,0.0,0.0},
                {0.0,0.0,1.0,0.0,0.0,0.0},
                {0.0,0.0,0.0,1.0,0.0,0.0},
                {0.0,0.0,0.0,0.0,1.0,0.0},
                {0.0,0.0,0.0,0.0,0.0,1.0}};

float A[n][n]={ {0.1,0.0,0.0,0.0,0.0,0.0},
                {0.0,0.0,0.0,0.0,0.0,0.0},
                {0.0,0.0,0.1,0.0,0.0,0.0},
                {0.0,0.0,0.0,0.0,0.0,0.0},
                {0.0,0.0,0.0,0.0,0.1,0.0},
                {0.0,0.0,0.0,0.0,0.0,0.0}};

float B[n][n]={ {0.0,0.0,0.0,0.0,0.0,0.0},
                {0.0,0.1,0.0,0.0,0.0,0.1},
                {0.0,0.0,0.0,0.0,0.0,0.0},
                {0.0,0.0,0.0,0.1,0.0,0.0},
                {0.0,0.0,0.0,0.0,0.0,0.0},
                {0.0,0.0,0.0,0.0,0.0,0.1}};

float C[n]={0.05,0.06,-0.07,-0.06,0.0,0.0};

t= zero;
n2=((tb-t)/h);
n1=n2;
/*w=0.0;
freq1=(wf-w)/dw;
freq=freq1;
*/
fp=fopen("g:\\result\\3dof.xls","w");

```

```

clrscr();
for (b=0;b<n;b++)
{yt[b]=ytini[b];}
convgnc1=fabs(y1new-y1old);
convgnc2=fabs(y2new-y2old);
convgnc3=fabs(y3new-y3old);
convgnc4=fabs(y4new-y4old);
convgnc5=fabs(y5new-y5old);
convgnc6=fabs(y6new-y6old);

while (0.0001<convgnc1 && 0.0001<convgnc2 && 0.0001<convgnc3 && 0.0001<convgnc4 &&
0.0001<convgnc5 && 0.0001<convgnc6)
{
yt[0]=0.05;
/* yt[1]=-0.724; */
yt[2]=0.07;
/* yt[3]=0.0183; */
/* printf("\n\n convergence1=%f ",convgnc1);
printf("\n\n convergence2=%f ",convgnc2);
printf("\n\n convergence3=%f ",convgnc3);
printf("\n\n convergence4=%f ",convgnc4);
*/
for (b=0;b<n;b++)
{yta[b]=ytini[b];}
for (b=0;b<n;b++)
{ytb[b]=ytini[b];}
for (b=0;b<n;b++)
{r[b]=ytini[b];}
for (b=0;b<n;b++)
{l1[b]=ytini[b];}

for (k=0;k<n;k++)
{ for (b=0;b<n;b++)
{Yb[k][b]=k2[k][b];}}
for (k=0;k<n;k++)
{ for (b=0;b<n;b++)
{m2[k][b]=k2[k][b];}}
for (k=0;k<n;k++)
{ for (b=0;b<n;b++)
{q[k][b]=k2[k][b];}}

for (b=0;b<n;b++)
{yta[b]=yt[b];}
/* printf("\n\n at acc=%d yta= %f\t%f\t%f\t%f",acc,yta[0][0],yta[1][0],yta[2][0],yta[3][0]);

```

```

printf("\n\n value of yt matrix");
for (l=0;l<n;l++)
{printf(" \n %f ",yt[l]);}
*/
sum1=0.0;
sum2=0.0;
sum3=0.0;
sum4=0.0;
sum=0.0;
sum_s=0.0;
p1=0.0;
p2=0.0;
p3=0.0;
p4=0.0;
p5=0.0;
p6=0.0;
t=0.0;
for (k=0;k<n;k++)
{ for (b=0;b<n;b++)
{k1[k][b]=k2[k][b];}}
for (k=0;k<n;k++)
{ for (b=0;b<n;b++)
{K[k][b]=k2[k][b];}}
for (k=0;k<n;k++)
{ for (b=0;b<n;b++)
{L[k][b]=k2[k][b];}}
for (k=0;k<n;k++)
{ for (b=0;b<n;b++)
{M[k][b]=k2[k][b];}}
for (k=0;k<n;k++)
{ for (b=0;b<n;b++)
{N[k][b]=k2[k][b];}}
for (k=0;k<n;k++)
{ for (b=0;b<n;b++)
{Y[k][b]=Y1[k][b];}}

for (i=0;i<n1;i++)
{
k1[0][0]=h * yt[1];
k1[0][1]=h * fn1(t, yt[0],yt[1], yt[2], yt[3], yt[4], yt[5], w);
k1[0][2]=h * yt[3];
k1[0][3]=h * fn2(t, yt[0],yt[1], yt[2], yt[3], yt[4], yt[5], w);
k1[0][4]=h * yt[5];
k1[0][5]=h * fn3(t, yt[0],yt[1], yt[2], yt[3], yt[4], yt[5], w);

k1[1][0]=h * (yt[1]+k1[0][1]/2.0);
k1[1][1]=h * fn1(t+h/2.0, yt[0]+k1[0][0]/2.0, yt[1]+k1[0][1]/2.0, yt[2]+k1[0][2]/2.0, yt[3]+k1[0][3]/2.0,
yt[4]+k1[0][4]/2.0, yt[5]+k1[0][5]/2.0, w) ;

```

$k1[1][2]=h * (yt[3]+k1[0][3]/2.0);$   
 $k1[1][3]=h * fn2(t+h/2.0, yt[0]+k1[0][0]/2.0, yt[1]+k1[0][1]/2.0, yt[2]+k1[0][2]/2.0, yt[3]+k1[0][3]/2.0,$   
 $yt[4]+k1[0][4]/2.0, yt[5]+k1[0][5]/2.0, w);$   
 $k1[1][4]=h * (yt[5]+k1[0][5]/2.0);$   
 $k1[1][5]=h * fn3(t+h/2.0, yt[0]+k1[0][0]/2.0, yt[1]+k1[0][1]/2.0, yt[2]+k1[0][2]/2.0, yt[3]+k1[0][3]/2.0,$   
 $yt[4]+k1[0][4]/2.0, yt[5]+k1[0][5]/2.0, w);$

$k1[2][0]=h * (yt[1]+k1[1][1]/2.0);$   
 $k1[2][1]=h * fn1(t+h/2.0, yt[0]+k1[1][0]/2.0, yt[1]+k1[1][1]/2.0, yt[2]+k1[1][2]/2.0, yt[3]+k1[1][3]/2.0,$   
 $yt[4]+k1[1][4]/2.0, yt[5]+k1[1][5]/2.0, w);$   
 $k1[2][2]=h * (yt[3]+k1[1][3]/2.0);$   
 $k1[2][3]=h * fn2(t+h/2.0, yt[0]+k1[1][0]/2.0, yt[1]+k1[1][1]/2.0, yt[2]+k1[1][2]/2.0, yt[3]+k1[1][3]/2.0,$   
 $yt[4]+k1[1][4]/2.0, yt[5]+k1[1][5]/2.0, w);$   
 $k1[2][4]=h * (yt[5]+k1[1][5]/2.0);$   
 $k1[2][5]=h * fn3(t+h/2.0, yt[0]+k1[1][0]/2.0, yt[1]+k1[1][1]/2.0, yt[2]+k1[1][2]/2.0, yt[3]+k1[1][3]/2.0,$   
 $yt[4]+k1[1][4]/2.0, yt[5]+k1[1][5]/2.0, w);$

$k1[3][0]=h * (yt[1]+k1[2][1]/2.0);$   
 $k1[3][1]=h * fn1(t+h/2.0, yt[0]+k1[2][0]/2.0, yt[1]+k1[2][1]/2.0, yt[2]+k1[2][2]/2.0, yt[3]+k1[2][3]/2.0,$   
 $yt[4]+k1[2][4]/2.0, yt[5]+k1[2][5]/2.0, w);$   
 $k1[3][2]=h * (yt[3]+k1[2][3]);$   
 $k1[3][3]=h * fn2(t+h/2.0, yt[0]+k1[2][0]/2.0, yt[1]+k1[2][1]/2.0, yt[2]+k1[2][2]/2.0, yt[3]+k1[2][3]/2.0,$   
 $yt[4]+k1[2][4]/2.0, yt[5]+k1[2][5]/2.0, w);$   
 $k1[3][4]=h * (yt[5]+k1[2][5]/2.0);$   
 $k1[3][5]=h * fn3(t+h/2.0, yt[0]+k1[2][0]/2.0, yt[1]+k1[2][1]/2.0, yt[2]+k1[2][2]/2.0, yt[3]+k1[2][3]/2.0,$   
 $yt[4]+k1[2][4]/2.0, yt[5]+k1[2][5]/2.0, w);$

$k1[4][0]=h * (yt[1]+k1[3][1]/2.0);$   
 $k1[4][1]=h * fn1(t+h/2.0, yt[0]+k1[3][0]/2.0, yt[1]+k1[3][1]/2.0, yt[2]+k1[3][2]/2.0, yt[3]+k1[3][3]/2.0,$   
 $yt[4]+k1[3][4]/2.0, yt[5]+k1[3][5]/2.0, w);$   
 $k1[4][2]=h * (yt[3]+k1[3][3]);$   
 $k1[4][3]=h * fn2(t+h/2.0, yt[0]+k1[3][0]/2.0, yt[1]+k1[3][1]/2.0, yt[2]+k1[3][2]/2.0, yt[3]+k1[3][3]/2.0,$   
 $yt[4]+k1[3][4]/2.0, yt[5]+k1[3][5]/2.0, w);$   
 $k1[4][4]=h * (yt[5]+k1[3][5]/2.0);$   
 $k1[4][5]=h * fn3(t+h/2.0, yt[0]+k1[3][0]/2.0, yt[1]+k1[3][1]/2.0, yt[2]+k1[3][2]/2.0, yt[3]+k1[3][3]/2.0,$   
 $yt[4]+k1[3][4]/2.0, yt[5]+k1[3][5]/2.0, w);$

$k1[5][0]=h * (yt[1]+k1[4][1]);$   
 $k1[5][1]=h * fn1(t+h, yt[0]+k1[4][0], yt[1]+k1[4][1], yt[2]+k1[4][2], yt[3]+k1[4][3], yt[4]+k1[4][4], yt[5]+k1[4][5],$   
 $w);$   
 $k1[5][2]=h * (yt[3]+k1[4][3]);$   
 $k1[5][3]=h * fn2(t+h, yt[0]+k1[4][0], yt[1]+k1[4][1], yt[2]+k1[4][2], yt[3]+k1[4][3], yt[4]+k1[4][4], yt[5]+k1[4][5],$   
 $w);$   
 $k1[5][4]=h * (yt[5]+k1[4][5]);$   
 $k1[5][5]=h * fn3(t+h, yt[0]+k1[4][0], yt[1]+k1[4][1], yt[2]+k1[4][2], yt[3]+k1[4][3], yt[4]+k1[4][4], yt[5]+k1[4][5],$   
 $w);$

for(z=0;z<n;z++)





```

K[5][0]=h * (Y[1][z]+K[4][1]) ;
K[5][1]=h * FN1(t+h, Y[0][z]+K[4][0], Y[1][z]+K[4][1], Y[2][z]+K[4][2], Y[3][z]+K[4][3], Y[4][z]+K[4][4],
Y[5][z]+K[4][5], yt[0], yt[1], yt[2], yt[3], yt[4], yt[5]) ;
K[5][2]=h * (Y[3][z]+K[3][3]/2.0) ;
K[5][3]=h * FN2(t+h, Y[0][z]+K[4][0], Y[1][z]+K[4][1], Y[2][z]+K[4][2], Y[3][z]+K[4][3], Y[4][z]+K[4][4],
Y[5][z]+K[4][5], yt[0], yt[1], yt[2], yt[3], yt[4], yt[5]) ;
K[5][4]=h * (Y[5][z]+K[4][5]/2.0) ;
K[5][5]=h * FN3(t+h, Y[0][z]+K[4][0], Y[1][z]+K[4][1], Y[2][z]+K[4][2], Y[3][z]+K[4][3], Y[4][z]+K[4][4],
Y[5][z]+K[4][5], yt[0], yt[1], yt[2], yt[3], yt[4], yt[5]) ;

for(l=0;l<n;l++)
{ Y[l][z]=Y[l][z]+ (K[0][l]+ 2.0*K[1][l]+ 2.0*K[2][l]+ 2.0*K[3][l]+ 2.0*K[4][l]+ K[5][l])/6.0 ;}

}

t=t+h;

for (j=0;j<n;j++)
{ yt[j]= yt[j] + (k1[0][j]+ 2.0*k1[1][j]+ 2.0*k1[2][j]+ 2.0* k1[3][j]+ 2.0*k1[4][j]+ k1[5][j])/6.0 ; }
/* printf("\n at t=%f and h=%f\n y1=%ft y2=%ft y3=%ft y4=%f\n",t,h,yt[0],yt[1],yt[2],yt[3]);
*/
}

for (i=0;i<n;i++)
{ytb[i]=yt[i];}
/* printf("\n\n value of ytb matrix");
for (l=0;l<n;l++)
{printf (" \n %ft %ft %ft %f",ytb[l][0],ytb[l][1],ytb[l][2],ytb[l][3]);}
*/
for (i=0;i<n;i++)
{for (j=0;j<n;j++)
{Yb[i][j]=Y[i][j];}}

for(i=0;i<n;i++)
{ for (j=0;j<n;j++)
{sum=0.0;
for (k=0;k<n;k++)
{sum= sum + Yb[i][k]*yta[k];
m_1[i]= sum ;
}}}

for(i=0;i<n;i++)
{ for (j=0;j<n;j++)
{sum=0.0;
sum= sum + m_1[i]-ytb[i];
l1[j]= sum ;
}
}

```

```

}}

/* printf("\n\n 11 matrix is");
for (i=0;i<n;i++)
{printf("\n\n %ft %ft %ft %ft",11[i][0],11[i][1],11[i][2],11[i][3]);*/

for(i=0;i<n;i++)
{ for (j=0;j<n;j++)
{sum=0.0;
for (k=0;k<n;k++)
{sum= sum + B[i][k]*Yb[k][j];
m_2[i][j]= sum ;
}}}

for(i=0;i<n;i++)
{ for (j=0;j<n;j++)
{sum=0.0;
for (k=0;k<n;k++)
{sum= sum + B[i][k]*11[k];
m_3[i]= sum ;
}}}

for(i=0;i<n;i++)
{ for (j=0;j<n;j++)
{sum=0.0;
sum= sum + A[i][j]+m_2[i][j];
q[i][j]= sum ;
}}

for(i=0;i<n;i++)
{ for (j=0;j<n;j++)
{sum=0.0;
sum= sum + C[i]+m_3[i];
r[i]= sum ;
}}
/* printf("\n\n q matrix is\n");
for(i=0;i<n;i++)
{
printf("\n\n %ft %ft %ft %ft ",q[i][0],q[i][1],q[i][2],q[i][3]);
}
printf("\n\n r matrix is\n");
for(i=0;i<n;i++)
{
printf("\n\n %ft %ft %ft %ft ",r[i][0],r[i][1],r[i][2],r[i][3]);
} */

for(i=0;i<n;i++)

```

```

{for(j=0;j<n;j++)
{a[i][j]=q[i][j];
}}
/* printf("\nvalue of a mat");
for (i=0;i<n;i++)
{printf("\n\n %ft %ft %ft %ft",a[i][0],a[i][1],a[i][2],a[i][3]); }
*/
for(i=0;i<n;i++)
{for(j=0;j<n;j++)
{a1[i][j]=a[i][j];
}}

for (i=0;i<n;i++)
{sum1=sum1+a1[i][i];
}
p1=sum1/1;

for (i=0;i<n;i++)
{ for (j=0;j<n;j++)
{m1[i][j]=p1*I[i][j];}}

for (i=0;i<n;i++)
{ for (j=0;j<n;j++)
{ sum_s=0.0;
sum_s=sum_s+a1[i][j]-m1[i][j];
ab1[i][j]=sum_s;}}

for(i=0;i<n;i++)
{ for (j=0;j<n;j++)
{sum_s=0.0;
for (k=0;k<n;k++)
{sum_s= sum_s + a[i][k]*ab1[k][j];
a2[i][j]= sum_s ;
}}}
/* printf("\nvalue of a2 mat");
for (i=0;i<n;i++)
{printf("\n\n %ft %ft %ft %ft",a2[i][0],a2[i][1],a2[i][2],a2[i][3]);}
*/
for (i=0;i<n;i++)
{sum2=sum2+a2[i][i];
}
p2=sum2/2;

for (i=0;i<n;i++)
{ for (j=0;j<n;j++)
{m2[i][j]=p2*I[i][j];}}

for (i=0;i<n;i++)

```

```

{ for (j=0;j<n;j++)
{ sum_s=0.0;
sum_s=sum_s+a2[i][j]-m2[i][j];
ab2[i][j]=sum_s;}}

    for(i=0;i<n;i++)
{ for (j=0;j<n;j++)
{sum_s=0.0;
for (k=0;k<n;k++)
{sum_s= sum_s + a[i][k]*ab2[k][j];
a3[i][j]= sum_s ;
}}}
/* printf("\nvalue of a3 mat");
for (i=0;i<n;i++)
{printf("\n\n %f\t %f\t %f\t %f\t",a3[i][0],a3[i][1],a3[i][2],a3[i][3]);}
*/
for (i=0;i<n;i++)
{sum3=sum3+a3[i][i];
}
p3=sum3/3;

for (i=0;i<n;i++)
{ for (j=0;j<n;j++)
{m3[i][j]=p3*I[i][j];} }

for (i=0;i<n;i++)
{ for (j=0;j<n;j++)
{ sum_s=0.0;
sum_s=sum_s+a3[i][j]-m3[i][j];
ab3[i][j]=sum_s;}}
/* printf("\nvalue of ab3 mat");
for (i=0;i<n;i++)
{printf("\n\n %f\t %f\t %f\t %f\t",ab3[i][0],ab3[i][1],ab3[i][2],ab3[i][3]);}
*/

    for(i=0;i<n;i++)
{ for (j=0;j<n;j++)
{sum_s=0.0;
for (k=0;k<n;k++)
{sum_s= sum_s + a[i][k]*ab3[k][j];
a4[i][j]= sum_s ;
}}}
for (i=0;i<n;i++)
{ sum4= sum4+ a4[i][i];}

p4= sum4/ 4.0 ;

```

```

for (i=0;i<n;i++)
{ for (j=0;j<n;j++)
{ m_4[i][j] = p4*I[i][j];}}

for (i=0;i<n;i++)
{ for (j=0;j<n;j++)
{ sum_s=0.0;
sum_s=sum_s+a4[i][j]-m_4[i][j];
ab4[i][j]=sum_s;}}

    for(i=0;i<n;i++)
{ for (j=0;j<n;j++)
{sum_s=0.0;
for (k=0;k<n;k++)
{sum_s= sum_s + a[i][k]*ab4[k][j];
a5[i][j]= sum_s ;
}}}

for (i=0;i<n;i++)
{ sum5= sum5+ a5[i][i];}
p5= sum5/ 5.0 ;

for (i=0;i<n;i++)
{ for (j=0;j<n;j++)
{m5[i][j]=p5*I[i][j];}}

for (i=0;i<n;i++)
{ for (j=0;j<n;j++)
{ sum_s=0.0;
sum_s=sum_s+a5[i][j]-m5[i][j];
ab5[i][j]=sum_s;}}

    for(i=0;i<n;i++)
{ for (j=0;j<n;j++)
{sum_s=0.0;
for (k=0;k<n;k++)
{sum_s= sum_s + a[i][k]*ab5[k][j];
a6[i][j]= sum_s ;
}}}

for (i=0;i<n;i++)
{ sum6= sum6+ a6[i][i];}
p6= sum6/ 6.0 ;

for (i=0;i<n;i++)
{ for (j=0;j<n;j++)
{inverse[i][j]=ab5[i][j]/p6;}}

```

```

for(i=0;i<n;i++)
{ for (j=0;j<n;j++)
  {sum_s=0.0;
  for (k=0;k<n;k++)
  {sum_s= sum_s + inverse[i][k]*q[k][j];
  test[i][j]= sum_s ;
  }}}
for(i=0;i<n;i++)
{ for (j=0;j<n;j++)
  {sum_s=0.0;
  for (k=0;k<n;k++)
  {sum_s= sum_s + inverse[i][k]*r[k];
  ya[i]= sum_s ;
  }}}

```

```

for(i=0;i<n;i++)
{ for (j=0;j<n;j++)
  {sum_s=0.0;
  for (k=0;k<n;k++)
  {sum_s= sum_s + Yb[i][k]*ya[k];
  m4[j]= sum_s ;
  }}}

```

```

y1old=yb[0];
y2old=yb[1];
y3old=yb[2];
y4old=yb[3];
y5old=yb[4];
y6old=yb[5];
for (i=0;i<n;i++)
{ for (j=0;j<n;j++)
{ sum_s=0.0;
sum_s=sum_s+m4[i]-l1[i];
yb[i]=sum_s;}}

```

```

y1new=yb[0] ;
y2new=yb[1] ;
y3new=yb[2] ;
y4new=yb[3] ;
y5new=yb[4] ;
y6new=yb[5] ;

```

```

for (b=0;b<n;b++)
{yt[b]=ya[b];}

```

```

convgnc1= fabs(y1new-y1old) ;
convgnc2= fabs(y2new-y2old) ;
convgnc3= fabs(y3new-y3old) ;

```

```

convgnc4= fabs(y4new-y4old) ;
convgnc5= fabs(y5new-y5old) ;
convgnc6= fabs(y6new-y6old) ;

} /* for acc loop */ /* while loop end */

for (b=0;b<n;b++)
  {yt[b]=ya[b];}
/* yt[0]=0.05;
yt[1]=0.06;
yt[2]=0.07;
yt[3]=0.08;
*/
t=0.0;

for (k=0;k<n;k++)
  { for (b=0;b<n;b++)
    {k1[k][b]=k2[k][b];}}
for (k=0;k<n;k++)
  { for (b=0;b<n;b++)
    {K[k][b]=k2[k][b];}}
for (k=0;k<n;k++)
  { for (b=0;b<n;b++)
    {L[k][b]=k2[k][b];}}
for (k=0;k<n;k++)
  { for (b=0;b<n;b++)
    {M[k][b]=k2[k][b];}}
for (k=0;k<n;k++)
  { for (b=0;b<n;b++)
    {N[k][b]=k2[k][b];}}
for (k=0;k<n;k++)
  { for (b=0;b<n;b++)
    {Y[k][b]=Y1[k][b];}}

fputs (" t(s)\t k_1/fl *y1(x1)(m)\t y2(m/s)\t k_1/fl *y3(x2)(m)\t y4(m/s)\t k_1 *y5/fl (x3)(m)\t y6(m/s) \n",fp);
fprintf(fp," %ft %ft %ft %ft %ft %ft %f\n ",t,k_1/fl *yt[0],yt[1],k_1/fl *yt[2],yt[3],k_1/fl *yt[4],yt[5]);
for (i=0;i<(n1-1);i++)
  {
k1[0][0]=h * yt[1];
k1[0][1]=h * fn1(t, yt[0],yt[1], yt[2], yt[3], yt[4], yt[5], w);
k1[0][2]=h * yt[3];
k1[0][3]=h * fn2(t, yt[0],yt[1], yt[2], yt[3], yt[4], yt[5], w);
k1[0][4]=h * yt[5];
k1[0][5]=h * fn3(t, yt[0],yt[1], yt[2], yt[3], yt[4], yt[5], w);

k1[1][0]=h * (yt[1]+k1[0][1]/2.0);
k1[1][1]=h * fn1(t+h/2.0, yt[0]+k1[0][0]/2.0, yt[1]+k1[0][1]/2.0, yt[2]+k1[0][2]/2.0, yt[3]+k1[0][3]/2.0,
yt[4]+k1[0][4]/2.0, yt[5]+k1[0][5]/2.0, w) ;

```

```

k1[1][2]=h * (yt[3]+k1[0][3]/2.0);
k1[1][3]=h * fn2(t+h/2.0, yt[0]+k1[0][0]/2.0, yt[1]+k1[0][1]/2.0, yt[2]+k1[0][2]/2.0, yt[3]+k1[0][3]/2.0,
yt[4]+k1[0][4]/2.0, yt[5]+k1[0][5]/2.0, w) ;
k1[1][4]=h * (yt[5]+k1[0][5]/2.0);
k1[1][5]=h * fn3(t+h/2.0, yt[0]+k1[0][0]/2.0, yt[1]+k1[0][1]/2.0, yt[2]+k1[0][2]/2.0, yt[3]+k1[0][3]/2.0,
yt[4]+k1[0][4]/2.0, yt[5]+k1[0][5]/2.0, w) ;

k1[2][0]=h * (yt[1]+k1[1][1]/2.0);
k1[2][1]=h * fn1(t+h/2.0, yt[0]+k1[1][0]/2.0, yt[1]+k1[1][1]/2.0, yt[2]+k1[1][2]/2.0, yt[3]+k1[1][3]/2.0,
yt[4]+k1[1][4]/2.0, yt[5]+k1[1][5]/2.0, w) ;
k1[2][2]=h * (yt[3]+k1[1][3]/2.0);
k1[2][3]=h * fn2(t+h/2.0, yt[0]+k1[1][0]/2.0, yt[1]+k1[1][1]/2.0, yt[2]+k1[1][2]/2.0, yt[3]+k1[1][3]/2.0,
yt[4]+k1[1][4]/2.0, yt[5]+k1[1][5]/2.0, w) ;
k1[2][4]=h * (yt[5]+k1[1][5]/2.0);
k1[2][5]=h * fn3(t+h/2.0, yt[0]+k1[1][0]/2.0, yt[1]+k1[1][1]/2.0, yt[2]+k1[1][2]/2.0, yt[3]+k1[1][3]/2.0,
yt[4]+k1[1][4]/2.0, yt[5]+k1[1][5]/2.0, w) ;

k1[3][0]=h * (yt[1]+k1[2][1]/2.0);
k1[3][1]=h * fn1(t+h/2.0, yt[0]+k1[2][0]/2.0, yt[1]+k1[2][1]/2.0, yt[2]+k1[2][2]/2.0, yt[3]+k1[2][3]/2.0,
yt[4]+k1[2][4]/2.0, yt[5]+k1[2][5]/2.0, w) ;
k1[3][2]=h * (yt[3]+k1[2][3]);
k1[3][3]=h * fn2(t+h/2.0, yt[0]+k1[2][0]/2.0, yt[1]+k1[2][1]/2.0, yt[2]+k1[2][2]/2.0, yt[3]+k1[2][3]/2.0,
yt[4]+k1[2][4]/2.0, yt[5]+k1[2][5]/2.0, w) ;
k1[3][4]=h * (yt[5]+k1[2][5]/2.0);
k1[3][5]=h * fn3(t+h/2.0, yt[0]+k1[2][0]/2.0, yt[1]+k1[2][1]/2.0, yt[2]+k1[2][2]/2.0, yt[3]+k1[2][3]/2.0,
yt[4]+k1[2][4]/2.0, yt[5]+k1[2][5]/2.0, w) ;

k1[4][0]=h * (yt[1]+k1[3][1]/2.0);
k1[4][1]=h * fn1(t+h/2.0, yt[0]+k1[3][0]/2.0, yt[1]+k1[3][1]/2.0, yt[2]+k1[3][2]/2.0, yt[3]+k1[3][3]/2.0,
yt[4]+k1[3][4]/2.0, yt[5]+k1[3][5]/2.0, w) ;
k1[4][2]=h * (yt[3]+k1[3][3]);
k1[4][3]=h * fn2(t+h/2.0, yt[0]+k1[3][0]/2.0, yt[1]+k1[3][1]/2.0, yt[2]+k1[3][2]/2.0, yt[3]+k1[3][3]/2.0,
yt[4]+k1[3][4]/2.0, yt[5]+k1[3][5]/2.0, w) ;
k1[4][4]=h * (yt[5]+k1[3][5]/2.0);
k1[4][5]=h * fn3(t+h/2.0, yt[0]+k1[3][0]/2.0, yt[1]+k1[3][1]/2.0, yt[2]+k1[3][2]/2.0, yt[3]+k1[3][3]/2.0,
yt[4]+k1[3][4]/2.0, yt[5]+k1[3][5]/2.0, w) ;

k1[5][0]=h * (yt[1]+k1[4][1]);
k1[5][1]=h * fn1(t+h, yt[0]+k1[4][0], yt[1]+k1[4][1], yt[2]+k1[4][2], yt[3]+k1[4][3], yt[4]+k1[4][4], yt[5]+k1[4][5],
w) ;
k1[5][2]=h * (yt[3]+k1[4][3]);
k1[5][3]=h * fn2(t+h, yt[0]+k1[4][0], yt[1]+k1[4][1], yt[2]+k1[4][2], yt[3]+k1[4][3], yt[4]+k1[4][4], yt[5]+k1[4][5],
w) ;
k1[5][4]=h * (yt[5]+k1[4][5]);
k1[5][5]=h * fn3(t+h, yt[0]+k1[4][0], yt[1]+k1[4][1], yt[2]+k1[4][2], yt[3]+k1[4][3], yt[4]+k1[4][4], yt[5]+k1[4][5],
w) ;

for(z=0;z<n;z++)

```





```

K[5][0]=h * (Y[1][z]+K[4][1]) ;
K[5][1]=h * FN1(t+h, Y[0][z]+K[4][0], Y[1][z]+K[4][1], Y[2][z]+K[4][2], Y[3][z]+K[4][3], Y[4][z]+K[4][4],
Y[5][z]+K[4][5], yt[0], yt[1], yt[2], yt[3], yt[4], yt[5]) ;
K[5][2]=h * (Y[3][z]+K[3][3]/2.0) ;
K[5][3]=h * FN2(t+h, Y[0][z]+K[4][0], Y[1][z]+K[4][1], Y[2][z]+K[4][2], Y[3][z]+K[4][3], Y[4][z]+K[4][4],
Y[5][z]+K[4][5], yt[0], yt[1], yt[2], yt[3], yt[4], yt[5]) ;
K[5][4]=h * (Y[5][z]+K[4][5]/2.0) ;
K[5][5]=h * FN3(t+h, Y[0][z]+K[4][0], Y[1][z]+K[4][1], Y[2][z]+K[4][2], Y[3][z]+K[4][3], Y[4][z]+K[4][4],
Y[5][z]+K[4][5], yt[0], yt[1], yt[2], yt[3], yt[4], yt[5]) ;

for(l=0;l<n;l++)
{Y[l][z]=Y[l][z]+ (K[0][l]+ 2.0*K[1][l]+ 2.0*K[2][l]+ 2.0*K[3][l]+ 2.0*K[4][l]+ K[5][l])/6.0 ;}

}

t=t+h;

for (j=0;j<n;j++)
{ yt[j]= yt[j] + (k1[0][j]+ 2.0*k1[1][j]+ 2.0*k1[2][j]+ 2.0*k1[3][j]+ 2.0*k1[4][j]+ k1[5][j])/6.0 ; }
printf("\n at t=%f y1=%ft y3=%ft y5=%ft \n",t, yt[0], yt[2], yt[4]);
fprintf(fp, " %ft %ft %ft %ft %ft %ft %f\n ",t,k_1/fl*yt[0],yt[1],k_1/fl*yt[2],yt[3], k_1/fl*yt[4], yt[5]);

}
fprintf(fp, " %ft %ft %ft %ft %ft %ft %f\n ",t+h,k_1/fl*yb[0],yb[1],k_1/fl*yb[2],yb[3], k_1/fl*yb[4],
yb[5]);

fclose(fp);

getch();

}

```

### C. 4 Code for Nondimensional Displacement with Forcing Frequency for 3 DOFS

```
# include <stdio.h>
# include <conio.h>
# include <math.h>
# include <stdlib.h>
# define n 6
# define ma1 20.0
# define ma2 20.0
# define ma3 20.0
# define k_1 50
# define k_1p 0.05
# define k_2 50.0
# define k_2p 0.05
# define k_3 50.0
# define k_3p 0.05
# define c1 0.0
# define c1p 0.0
# define c2 0.00
# define c2p 0.0
# define c3 0.0
# define c3p 0.0
# define f1 20.0
# define f2 0.0
# define f3 0.0
# define tb 20.0
# define h 0.02
# define zero 0.0
# define wf 10.0
# define wi 0.0
# define dw 0.1

/* mass m in kg : spring constant k in N/m : damping constant c in Ns/m : force amplitude f in N: angular frequency
w in rad/s : tb or t in second(s) */

double fn1(double t ,double y1,double y2, double y3, double y4, double y5, double y6, double w1)
{
double z=0.0;
z=(-k_1*y1-k_1p*pow(y1,3)-k_2*y1+k_2*y3-k_2p*pow(y1-y3,3)-c1*y2-c1p*y2*pow(y1,2)-c2*y2+c2*y4-
c2p*(y2-y4)*pow(y1-y3,2)-f1*sin(w1*t))/ma1 ;
return z;
}

double fn2(double t, double y1,double y2, double y3, double y4, double y5, double y6, double w1)
{
double z=0.0;
```

```

z= (-k_3*(y3-y5)+k_3p*pow(y3-y5,2)+c3*(y4-y6)+c3p*(y4-y6)*pow(y2-y5,2)-k_2*(y1-y3)-k_2p*pow(y1-y3,3)-
c2*(y2-y4)-c2p*(y2-y4)*pow(y1-y3,2)-f2*sin(w1*t)) /ma2 ;
return z;
}

```

```

double fn3(double t, double y1, double y2, double y3, double y4, double y5, double y6, double w1)
{
double z=0.0;
z= (-k_3*(y3-y5)-k_3p*pow(y3-y5,3)-c3*(y4-y6)-c3p*(y4-y6)*pow(y3-y5,2)-f3*sin(w1*t)) /ma3 ;
return z;
}

```

```

float FN1(float t, float Y0, float Y1, float Y2, float Y3, float Y4, float Y5, float y1, float y2, float y3, float y4, float
y5, float y6)
{
float z=0.0;
z= (-k_1*Y0-k_1p*3*pow(y1,2)*Y0-k_2*(Y0-Y2)-k_2p*(Y0-Y2)*3*pow(y1-y3,2)-c1*Y1-c1p*Y1*pow
(y1,2)-c1p*2*y1*y2*Y0-c2*(Y1-Y3)-c2p*pow(y1-y3,2)*(Y1-Y3)-c2p*2*(y2-y4)*(y1-y3)*(Y0-Y2)) /ma1;
return z;
}

```

```

float FN2(float t, float Y0, float Y1, float Y2, float Y3, float Y4, float Y5, float y1, float y2, float y3, float y4, float
y5, float y6)
{
float z=0.0;
z= (-k_3*(Y2-Y4)+k_3p*2*(y3-y5)*(Y2-Y4)+c3*(Y3-Y5)+2*c3p*(y2-y5)*(y4-y6)*(Y1-Y4)+c3p*pow(y2-
y5,2)*(Y3-Y5)-k_2*(Y0-Y2)-k_2p*3*pow(y1-y3,2)*(Y0-Y2)-c2*(Y1-Y3)-c2p*pow(y1-y3,2)*(Y1-Y3)-
c2p*2*(y2-y4)*(y1-y3)*(Y0-Y2)) /ma2;
return z;
}

```

```

float FN3(float t, float Y0, float Y1, float Y2, float Y3, float Y4, float Y5, float y1, float y2, float y3, float y4, float
y5, float y6)
{
float z=0.0;
z= (-k_3*(Y2-Y4)-k_3p*3*pow(y3-y5,2)*(Y2-Y4)-c3*(Y3-Y5)-c3p*pow(y3-y5,2)*(Y3-Y5)-c3p*(y4-
y6)*2*(y3-y5)*(Y2-Y4)) /ma3;
return z;
}

```

```

void main()
{
int i,j,n1,l,b,k,u,acc,freq,z;
double yt[n];
float Y1[n][n] = {{1.0},
{0.0, 1.0}, {0.0, 0.0, 1.0}, {0.0, 0.0, 0.0, 1.0}, {0.0, 0.0, 0.0, 0.0, 1.0}, {0.0, 0.0, 0.0, 0.0, 0.0, 1.0}};

```

```

float n2,nf,t,w,y1new=0.0,y1old=1.0, convgnc1,
y2new=0.0,y2old=1.0,convgnc2,y3new=0.0,y3old=1.0,convgnc3,y4new=0.0,y4old=1.0,convgnc4,y5new=0.0,
y5old=1.0, convgnc5, y6new=0.0, y6old=1.0, convgnc6;
double k1[n][n],K[n][n],L[n][n],M[n][n],N[n][n],Y[n][n];
double k2[n][n]={0.0},{0.0},{0.0},{0.0},{0.0},{0.0};
float m_1[n],m_2[n][n],m_3[n],m_4[n][n];
float l1[n],q[n][n],r[n],ya[n],yb[n], Yb[n][n];
float
a[n][n],a1[n][n],a2[n][n],a3[n][n],a4[n][n],a5[n][n],a6[n][n],m1[n][n],m2[n][n],m3[n][n],m4[n],m5[n][n],ab1[n][n],a
b2[n][n],ab3[n][n],ab4[n][n],ab5[n][n],inverse[n][n],test[n][n];
float p1,p2,p3,p4,p5,p6,sum,sum_s,freq1;
float yta[n], ytb[n];
float ytini[n]={0.0,0.0,0.0,0.0,0.0,0.0};
float sum6,sum5,sum4,sum3,sum2,sum1;
FILE *fp;

float I[n][n]= {{1.0,0.0,0.0,0.0,0.0,0.0},
                {0.0,1.0,0.0,0.0,0.0,0.0},
                {0.0,0.0,1.0,0.0,0.0,0.0},
                {0.0,0.0,0.0,1.0,0.0,0.0},
                {0.0,0.0,0.0,0.0,1.0,0.0},
                {0.0,0.0,0.0,0.0,0.0,1.0}};

float A[n][n]={01.0,0.0,0.0,0.0,0.0,0.0},
              {0.0,0.0,0.0,0.0,0.0,0.0},
              {0.0,0.0,01.0,0.0,0.0,0.0},
              {0.0,0.0,0.0,0.0,0.0,0.0},
              {0.0,0.0,0.0,0.0,01.0,0.0},
              {0.0,0.0,0.0,0.0,0.0,0.0}};

float B[n][n]={0.0,0.0,0.0,0.0,0.0,0.0},
              {0.0,01.0,0.0,0.0,0.0,0.0},
              {0.0,0.0,0.0,0.0,0.0,0.0},
              {0.0,0.0,0.0,01.0,0.0,0.0},
              {0.0,0.0,0.0,0.0,0.0,0.0},
              {0.0,0.0,0.0,0.0,0.0,01.0}};

float C[n]={0.05,0.06,-0.07,-0.06,0.0,0.0};

t= zero;
n2=((tb-t)/h);
n1=n2;
w=wi;
freq1=(wf-w)/dw;
freq=freq1;
nf=pow((k_1/ma1),0.5);
fp=fopen("g:\\result\\3dofw.xls","w");

```

```

clrscr();

fputs ("t(s)\t w/w1\t k1/fl*y1\t k1/fl*y3\t k1/fl*y5 \n",fp);

for(u=0;u<=freq;u++)
{ y1new=0.0;
y2new=0.0;
y3new=0.0;
y4new=0.0;
y5new=0.0;
y6new=0.0;
y1old=1.0;
y2old=1.0;
y3old=1.0;
y4old=1.0;
y5old=1.0;
y6old=1.0;

convgnc1=fabs(y1new-y1old);
convgnc2=fabs(y2new-y2old);
convgnc3=fabs(y3new-y3old);
convgnc4=fabs(y4new-y4old);
convgnc5=fabs(y5new-y5old);
convgnc6=fabs(y6new-y6old);

for (b=0;b<n;b++)
{yt[b]=ytini[b];}
for (b=0;b<n;b++)
{yta[b]=ytini[b];}
for (b=0;b<n;b++)
{ytb[b]=ytini[b];}
for (b=0;b<n;b++)
{r[b]=ytini[b];}
for (b=0;b<n;b++)
{l1[b]=ytini[b];}

for (k=0;k<n;k++)
{ for (b=0;b<n;b++)
{Yb[k][b]=k2[k][b];}}
for (k=0;k<n;k++)
{ for (b=0;b<n;b++)
{m2[k][b]=k2[k][b];}}
for (k=0;k<n;k++)
{ for (b=0;b<n;b++)
{q[k][b]=k2[k][b];}}

```

```

while (0.0001<convgnc1 && 0.0001<convgnc2 && 0.0001<convgnc3 && 0.0001<convgnc4 &&
0.0001<convgnc5 && 0.0001<convgnc6)
{
    yt[0]=0.05;
/* yt[1]=-0.724; */
    yt[2]=0.07;
/* yt[3]=0.0183; */
/* printf("\n\n convergence1=%f ",convgnc1);
printf("\n\n convergence2=%f ",convgnc2);
printf("\n\n convergence3=%f ",convgnc3);
printf("\n\n convergence4=%f ",convgnc4);
*/
    for (b=0;b<n;b++)
    {yta[b]=ytini[b];}
    for (b=0;b<n;b++)
    {ytb[b]=ytini[b];}
    for (b=0;b<n;b++)
    {r[b]=ytini[b];}
    for (b=0;b<n;b++)
    {l1[b]=ytini[b];}

    for (k=0;k<n;k++)
    { for (b=0;b<n;b++)
    {Yb[k][b]=k2[k][b];}}
    for (k=0;k<n;k++)
    { for (b=0;b<n;b++)
    {m2[k][b]=k2[k][b];}}
    for (k=0;k<n;k++)
    { for (b=0;b<n;b++)
    {q[k][b]=k2[k][b];}}

    for (b=0;b<n;b++)
    {yta[b]=yt[b];}
/* printf("\n\n at acc=%d yta= %f\t%f\t%f\t%f",acc,yta[0][0],yta[1][0],yta[2][0],yta[3][0]);

printf("\n\n value of yt matrix");
    for (l=0;l<n;l++)
    {printf(" \n %f ",yt[l]);}
*/
sum1=0.0;
sum2=0.0;
sum3=0.0;
sum4=0.0;
sum=0.0;
sum_s=0.0;
p1=0.0;
p2=0.0;

```

```

p3=0.0;
p4=0.0;
p5=0.0;
p6=0.0;
t=0.0;
for (k=0;k<n;k++)
{ for (b=0;b<n;b++)
{k1[k][b]=k2[k][b];}}
for (k=0;k<n;k++)
{ for (b=0;b<n;b++)
{K[k][b]=k2[k][b];}}
for (k=0;k<n;k++)
{ for (b=0;b<n;b++)
{L[k][b]=k2[k][b];}}
for (k=0;k<n;k++)
{ for (b=0;b<n;b++)
{M[k][b]=k2[k][b];}}
for (k=0;k<n;k++)
{ for (b=0;b<n;b++)
{N[k][b]=k2[k][b];}}
for (k=0;k<n;k++)
{ for (b=0;b<n;b++)
{Y[k][b]=Y1[k][b];}}

for (i=0;i<n1;i++)
{
k1[0][0]=h * yt[1];
k1[0][1]=h * fn1(t, yt[0],yt[1], yt[2], yt[3], yt[4], yt[5], w);
k1[0][2]=h * yt[3];
k1[0][3]=h * fn2(t, yt[0],yt[1], yt[2], yt[3], yt[4], yt[5], w);
k1[0][4]=h * yt[5];
k1[0][5]=h * fn3(t, yt[0],yt[1], yt[2], yt[3], yt[4], yt[5], w);

k1[1][0]=h * (yt[1]+k1[0][1]/2.0);
k1[1][1]=h * fn1(t+h/2.0, yt[0]+k1[0][0]/2.0, yt[1]+k1[0][1]/2.0, yt[2]+k1[0][2]/2.0, yt[3]+k1[0][3]/2.0,
yt[4]+k1[0][4]/2.0, yt[5]+k1[0][5]/2.0, w) ;
k1[1][2]=h * (yt[3]+k1[0][3]/2.0);
k1[1][3]=h * fn2(t+h/2.0, yt[0]+k1[0][0]/2.0, yt[1]+k1[0][1]/2.0, yt[2]+k1[0][2]/2.0, yt[3]+k1[0][3]/2.0,
yt[4]+k1[0][4]/2.0, yt[5]+k1[0][5]/2.0, w) ;
k1[1][4]=h * (yt[5]+k1[0][5]/2.0);
k1[1][5]=h * fn3(t+h/2.0, yt[0]+k1[0][0]/2.0, yt[1]+k1[0][1]/2.0, yt[2]+k1[0][2]/2.0, yt[3]+k1[0][3]/2.0,
yt[4]+k1[0][4]/2.0, yt[5]+k1[0][5]/2.0, w) ;

k1[2][0]=h * (yt[1]+k1[1][1]/2.0);
k1[2][1]=h * fn1(t+h/2.0, yt[0]+k1[1][0]/2.0, yt[1]+k1[1][1]/2.0, yt[2]+k1[1][2]/2.0, yt[3]+k1[1][3]/2.0,
yt[4]+k1[1][4]/2.0, yt[5]+k1[1][5]/2.0, w) ;
k1[2][2]=h * (yt[3]+k1[1][3]/2.0);

```



```

k1[2][3]=h * fn2(t+h/2.0, yt[0]+k1[1][0]/2.0, yt[1]+k1[1][1]/2.0, yt[2]+k1[1][2]/2.0, yt[3]+k1[1][3]/2.0,
yt[4]+k1[1][4]/2.0, yt[5]+k1[1][5]/2.0, w) ;
k1[2][4]=h * (yt[5]+k1[1][5]/2.0);
k1[2][5]=h * fn3(t+h/2.0, yt[0]+k1[1][0]/2.0, yt[1]+k1[1][1]/2.0, yt[2]+k1[1][2]/2.0, yt[3]+k1[1][3]/2.0,
yt[4]+k1[1][4]/2.0, yt[5]+k1[1][5]/2.0, w) ;

k1[3][0]=h * (yt[1]+k1[2][1]/2.0);
k1[3][1]=h * fn1(t+h/2.0, yt[0]+k1[2][0]/2.0, yt[1]+k1[2][1]/2.0, yt[2]+k1[2][2]/2.0, yt[3]+k1[2][3]/2.0,
yt[4]+k1[2][4]/2.0, yt[5]+k1[2][5]/2.0, w) ;
k1[3][2]=h * (yt[3]+k1[2][3]);
k1[3][3]=h * fn2(t+h/2.0, yt[0]+k1[2][0]/2.0, yt[1]+k1[2][1]/2.0, yt[2]+k1[2][2]/2.0, yt[3]+k1[2][3]/2.0,
yt[4]+k1[2][4]/2.0, yt[5]+k1[2][5]/2.0, w) ;
k1[3][4]=h * (yt[5]+k1[2][5]/2.0);
k1[3][5]=h * fn3(t+h/2.0, yt[0]+k1[2][0]/2.0, yt[1]+k1[2][1]/2.0, yt[2]+k1[2][2]/2.0, yt[3]+k1[2][3]/2.0,
yt[4]+k1[2][4]/2.0, yt[5]+k1[2][5]/2.0, w) ;

k1[4][0]=h * (yt[1]+k1[3][1]/2.0);
k1[4][1]=h * fn1(t+h/2.0, yt[0]+k1[3][0]/2.0, yt[1]+k1[3][1]/2.0, yt[2]+k1[3][2]/2.0, yt[3]+k1[3][3]/2.0,
yt[4]+k1[3][4]/2.0, yt[5]+k1[3][5]/2.0, w) ;
k1[4][2]=h * (yt[3]+k1[3][3]);
k1[4][3]=h * fn2(t+h/2.0, yt[0]+k1[3][0]/2.0, yt[1]+k1[3][1]/2.0, yt[2]+k1[3][2]/2.0, yt[3]+k1[3][3]/2.0,
yt[4]+k1[3][4]/2.0, yt[5]+k1[3][5]/2.0, w) ;
k1[4][4]=h * (yt[5]+k1[3][5]/2.0);
k1[4][5]=h * fn3(t+h/2.0, yt[0]+k1[3][0]/2.0, yt[1]+k1[3][1]/2.0, yt[2]+k1[3][2]/2.0, yt[3]+k1[3][3]/2.0,
yt[4]+k1[3][4]/2.0, yt[5]+k1[3][5]/2.0, w) ;

k1[5][0]=h * (yt[1]+k1[4][1]);
k1[5][1]=h * fn1(t+h, yt[0]+k1[4][0], yt[1]+k1[4][1], yt[2]+k1[4][2], yt[3]+k1[4][3], yt[4]+k1[4][4], yt[5]+k1[4][5],
w) ;
k1[5][2]=h * (yt[3]+k1[4][3]);
k1[5][3]=h * fn2(t+h, yt[0]+k1[4][0], yt[1]+k1[4][1], yt[2]+k1[4][2], yt[3]+k1[4][3], yt[4]+k1[4][4], yt[5]+k1[4][5],
w) ;
k1[5][4]=h * (yt[5]+k1[4][5]);
k1[5][5]=h * fn3(t+h, yt[0]+k1[4][0], yt[1]+k1[4][1], yt[2]+k1[4][2], yt[3]+k1[4][3], yt[4]+k1[4][4], yt[5]+k1[4][5],
w) ;

for(z=0;z<n;z++)
{
K[0][0]=h * Y[1][z] ;
K[0][1]=h * FN1(t, Y[0][z], Y[1][z], Y[2][z], Y[3][z], Y[4][z], Y[5][z], yt[0], yt[1], yt[2], yt[3], yt[4], yt[5] ) ;
K[0][2]=h * Y[3][z] ;
K[0][3]=h * FN2(t, Y[0][z], Y[1][z], Y[2][z], Y[3][z], Y[4][z], Y[5][z], yt[0], yt[1], yt[2], yt[3], yt[4], yt[5] ) ;
K[0][4]=h * Y[5][z] ;
K[0][5]=h * FN3(t, Y[0][z], Y[1][z], Y[2][z], Y[3][z], Y[4][z], Y[5][z], yt[0], yt[1], yt[2], yt[3], yt[4], yt[5] ) ;

K[1][0]=h * (Y[1][z]+K[0][1]/2.0) ;
K[1][1]=h * FN1(t+h/2.0, Y[0][z]+K[0][0]/2.0, Y[1][z]+K[0][1]/2.0, Y[2][z]+K[0][2]/2.0, Y[3][z]+K[0][3]/2.0,
Y[4][z]+K[0][4]/2.0, Y[5][z]+K[0][5]/2.0, yt[0], yt[1], yt[2], yt[3], yt[4], yt[5] ) ;

```

$K[1][2]=h * (Y[3][z]+K[0][3]/2.0);$   
 $K[1][3]=h * FN2(t+h/2.0, Y[0][z]+K[0][0]/2.0, Y[1][z]+K[0][1]/2.0, Y[2][z]+K[0][2]/2.0, Y[3][z]+K[0][3]/2.0, Y[4][z]+K[0][4]/2.0, Y[5][z]+K[0][5]/2.0, yt[0], yt[1], yt[2], yt[3], yt[4], yt[5]);$   
 $K[1][4]=h * (Y[5][z]+K[0][5]/2.0);$   
 $K[1][5]=h * FN3(t+h/2.0, Y[0][z]+K[0][0]/2.0, Y[1][z]+K[0][1]/2.0, Y[2][z]+K[0][2]/2.0, Y[3][z]+K[0][3]/2.0, Y[4][z]+K[0][4]/2.0, Y[5][z]+K[0][5]/2.0, yt[0], yt[1], yt[2], yt[3], yt[4], yt[5]);$

$K[2][0]=h * (Y[1][z]+K[1][1]/2.0);$   
 $K[2][1]=h * FN1(t+h/2.0, Y[0][z]+K[1][0]/2.0, Y[1][z]+K[1][1]/2.0, Y[2][z]+K[1][2]/2.0, Y[3][z]+K[1][3]/2.0, Y[4][z]+K[1][4]/2.0, Y[5][z]+K[1][5]/2.0, yt[0], yt[1], yt[2], yt[3], yt[4], yt[5]);$   
 $K[2][2]=h * (Y[3][z]+K[1][3]/2.0);$   
 $K[2][3]=h * FN2(t+h/2.0, Y[0][z]+K[1][0]/2.0, Y[1][z]+K[1][1]/2.0, Y[2][z]+K[1][2]/2.0, Y[3][z]+K[1][3]/2.0, Y[4][z]+K[1][4]/2.0, Y[5][z]+K[1][5]/2.0, yt[0], yt[1], yt[2], yt[3], yt[4], yt[5]);$   
 $K[2][4]=h * (Y[5][z]+K[1][5]/2.0);$   
 $K[2][5]=h * FN3(t+h/2.0, Y[0][z]+K[1][0]/2.0, Y[1][z]+K[1][1]/2.0, Y[2][z]+K[1][2]/2.0, Y[3][z]+K[1][3]/2.0, Y[4][z]+K[1][4]/2.0, Y[5][z]+K[1][5]/2.0, yt[0], yt[1], yt[2], yt[3], yt[4], yt[5]);$

$K[3][0]=h * (Y[1][z]+K[2][1]/2.0);$   
 $K[3][1]=h * FN1(t+h/2.0, Y[0][z]+K[2][0]/2.0, Y[1][z]+K[2][1]/2.0, Y[2][z]+K[2][2]/2.0, Y[3][z]+K[2][3]/2.0, Y[4][z]+K[2][4]/2.0, Y[5][z]+K[2][5]/2.0, yt[0], yt[1], yt[2], yt[3], yt[4], yt[5]);$   
 $K[3][2]=h * (Y[3][z]+K[2][3]/2.0);$   
 $K[3][3]=h * FN2(t+h/2.0, Y[0][z]+K[2][0]/2.0, Y[1][z]+K[2][1]/2.0, Y[2][z]+K[2][2]/2.0, Y[3][z]+K[2][3]/2.0, Y[4][z]+K[2][4]/2.0, Y[5][z]+K[2][5]/2.0, yt[0], yt[1], yt[2], yt[3], yt[4], yt[5]);$   
 $K[3][4]=h * (Y[5][z]+K[2][5]/2.0);$   
 $K[3][5]=h * FN3(t+h/2.0, Y[0][z]+K[2][0]/2.0, Y[1][z]+K[2][1]/2.0, Y[2][z]+K[2][2]/2.0, Y[3][z]+K[2][3]/2.0, Y[4][z]+K[2][4]/2.0, Y[5][z]+K[2][5]/2.0, yt[0], yt[1], yt[2], yt[3], yt[4], yt[5]);$

$K[4][0]=h * (Y[1][z]+K[3][1]/2.0);$   
 $K[4][1]=h * FN1(t+h/2.0, Y[0][z]+K[3][0]/2.0, Y[1][z]+K[3][1]/2.0, Y[2][z]+K[3][2]/2.0, Y[3][z]+K[3][3]/2.0, Y[4][z]+K[3][4]/2.0, Y[5][z]+K[3][5]/2.0, yt[0], yt[1], yt[2], yt[3], yt[4], yt[5]);$   
 $K[4][2]=h * (Y[3][z]+K[3][3]/2.0);$   
 $K[4][3]=h * FN2(t+h/2.0, Y[0][z]+K[3][0]/2.0, Y[1][z]+K[3][1]/2.0, Y[2][z]+K[3][2]/2.0, Y[3][z]+K[3][3]/2.0, Y[4][z]+K[3][4]/2.0, Y[5][z]+K[3][5]/2.0, yt[0], yt[1], yt[2], yt[3], yt[4], yt[5]);$   
 $K[4][4]=h * (Y[5][z]+K[3][5]/2.0);$   
 $K[4][5]=h * FN3(t+h/2.0, Y[0][z]+K[3][0]/2.0, Y[1][z]+K[3][1]/2.0, Y[2][z]+K[3][2]/2.0, Y[3][z]+K[3][3]/2.0, Y[4][z]+K[3][4]/2.0, Y[5][z]+K[3][5]/2.0, yt[0], yt[1], yt[2], yt[3], yt[4], yt[5]);$

$K[5][0]=h * (Y[1][z]+K[4][1]);$   
 $K[5][1]=h * FN1(t+h, Y[0][z]+K[4][0], Y[1][z]+K[4][1], Y[2][z]+K[4][2], Y[3][z]+K[4][3], Y[4][z]+K[4][4], Y[5][z]+K[4][5], yt[0], yt[1], yt[2], yt[3], yt[4], yt[5]);$   
 $K[5][2]=h * (Y[3][z]+K[3][3]/2.0);$   
 $K[5][3]=h * FN2(t+h, Y[0][z]+K[4][0], Y[1][z]+K[4][1], Y[2][z]+K[4][2], Y[3][z]+K[4][3], Y[4][z]+K[4][4], Y[5][z]+K[4][5], yt[0], yt[1], yt[2], yt[3], yt[4], yt[5]);$   
 $K[5][4]=h * (Y[5][z]+K[4][5]/2.0);$   
 $K[5][5]=h * FN3(t+h, Y[0][z]+K[4][0], Y[1][z]+K[4][1], Y[2][z]+K[4][2], Y[3][z]+K[4][3], Y[4][z]+K[4][4], Y[5][z]+K[4][5], yt[0], yt[1], yt[2], yt[3], yt[4], yt[5]);$

for(l=0;l<n;l++)

```

{Y[i][z]=Y[i][z]+ (K[0][i]+ 2.0*K[1][i]+ 2.0*K[2][i]+ 2.0*K[3][i]+ 2.0*K[4][i]+ K[5][i])/6.0 ;}

}

t=t+h;

for (j=0;j<n;j++)
{ yt[j]= yt[j] + (k1[0][j]+ 2.0*k1[1][j]+ 2.0*k1[2][j]+ 2.0* k1[3][j]+ 2.0*k1[4][j]+ k1[5][j])/6.0 ; }
/* printf("\n at t=%f and h=%f\n y1=%ft y2=%ft y3=%ft y4=%f\n",t,h,yt[0],yt[1],yt[2],yt[3]);
*/
}

for (i=0;i<n;i++)
{ytb[i]=yt[i];}
/* printf("\n\n value of ytb matrix");
for (l=0;l<n;l++)
{printf (" \n %ft %ft %ft %f ",ytb[l][0],ytb[l][1],ytb[l][2],ytb[l][3]);}
*/
for (i=0;i<n;i++)
{for (j=0;j<n;j++)
{Yb[i][j]=Y[i][j];}}

/* printf("\n\n value of yta matrix");
for (l=0;l<n;l++)
{printf (" \n %ft %ft %ft %f ",yta[l][0],yta[l][1],yta[l][2],yta[l][3]);}

printf("\n\n value of ytb matrix");
for (l=0;l<n;l++)
{printf (" \n %ft %ft %ft %f ",ytb[l][0],ytb[l][1],ytb[l][2],ytb[l][3]);}

printf("\n\nvalue of Y matrix ");
for (l=0;l<n;l++)
{printf (" \n %ft %ft %ft %f ",Y[l][0],Y[l][1],Y[l][2],Y[l][3]);}
*/

for(i=0;i<n;i++)
{ for (j=0;j<n;j++)
{sum=0.0;
for (k=0;k<n;k++)
{sum= sum + Yb[i][k]*yta[k];
m_1[i]= sum ;
}}
}

for(i=0;i<n;i++)
{ for (j=0;j<n;j++)
{sum=0.0;

```

```

sum= sum + m_1[i]-ytb[i];
l1[i]= sum ;
}}

/* printf("\n\n l1 matrix is");
for (i=0;i<n;i++)
{printf("\n\n %ft %ft %ft %ft",l1[i][0],l1[i][1],l1[i][2],l1[i][3]);}*/

for(i=0;i<n;i++)
{ for (j=0;j<n;j++)
{sum=0.0;
for (k=0;k<n;k++)
{sum= sum + B[i][k]*Yb[k][j];
m_2[i][j]= sum ;
}}}

for(i=0;i<n;i++)
{ for (j=0;j<n;j++)
{sum=0.0;
for (k=0;k<n;k++)
{sum= sum + B[i][k]*l1[k];
m_3[i]= sum ;
}}}

for(i=0;i<n;i++)
{ for (j=0;j<n;j++)
{sum=0.0;
sum= sum + A[i][j]+m_2[i][j];
q[i][j]= sum ;
}}

for(i=0;i<n;i++)
{ for (j=0;j<n;j++)
{sum=0.0;
sum= sum + C[i]+m_3[i];
r[i]= sum ;
}}
/* printf("\n\n q matrix is\n");
for(i=0;i<n;i++)
{
printf("\n\n %ft %ft %ft %ft ",q[i][0],q[i][1],q[i][2],q[i][3]);
}
printf("\n\n r matrix is\n");
for(i=0;i<n;i++)
{
printf("\n\n %ft %ft %ft %ft ",r[i][0],r[i][1],r[i][2],r[i][3]);
} */

```

```

for(i=0;i<n;i++)
{for(j=0;j<n;j++)
{a[i][j]=q[i][j];
}}
/* printf("\nvalue of a mat");
for (i=0;i<n;i++)
{printf("\n\n %f\t %f\t %f\t %f\t",a[i][0],a[i][1],a[i][2],a[i][3]); }
*/
for(i=0;i<n;i++)
{for(j=0;j<n;j++)
{a1[i][j]=a[i][j];
}}

for (i=0;i<n;i++)
{sum1=sum1+a1[i][i];
}
p1=sum1/1;

for (i=0;i<n;i++)
{ for (j=0;j<n;j++)
{m1[i][j]=p1*i[i][j];}}

for (i=0;i<n;i++)
{ for (j=0;j<n;j++)
{ sum_s=0.0;
sum_s=sum_s+a1[i][j]-m1[i][j];
ab1[i][j]=sum_s;}}
/* printf("\nvalue of ab1 mat");
for (i=0;i<n;i++)
{printf("\n\n %f\t %f\t %f\t %f\t",ab1[i][0],ab1[i][1],ab1[i][2],ab1[i][3]); }
*/

for(i=0;i<n;i++)
{ for (j=0;j<n;j++)
{sum_s=0.0;
for (k=0;k<n;k++)
{sum_s= sum_s + a[i][k]*ab1[k][j];
a2[i][j]= sum_s ;
}}}
/* printf("\nvalue of a2 mat");
for (i=0;i<n;i++)
{printf("\n\n %f\t %f\t %f\t %f\t",a2[i][0],a2[i][1],a2[i][2],a2[i][3]);}
*/
for (i=0;i<n;i++)
{sum2=sum2+a2[i][i];
}
p2=sum2/2;

```

```

for (i=0;i<n;i++)
{ for (j=0;j<n;j++)
{m2[i][j]=p2*I[i][j];}}

```

```

for (i=0;i<n;i++)
{ for (j=0;j<n;j++)
{ sum_s=0.0;
sum_s=sum_s+a2[i][j]-m2[i][j];
ab2[i][j]=sum_s;}}

```

```

for(i=0;i<n;i++)
{ for (j=0;j<n;j++)
{sum_s=0.0;
for (k=0;k<n;k++)
{sum_s= sum_s + a[i][k]*ab2[k][j];
a3[i][j]= sum_s ;
}}}
/* printf("\nvalue of a3 mat");
for (i=0;i<n;i++)
{printf("\n\n %ft %ft %ft %ft",a3[i][0],a3[i][1],a3[i][2],a3[i][3]);
*/
for (i=0;i<n;i++)
{sum3=sum3+a3[i][i];
}
p3=sum3/3;

```

```

for (j=0;i<n;i++)
{ for (j=0;j<n;j++)
{m3[i][j]=p3*I[i][j];}}

```

```

for (i=0;i<n;i++)
{ for (j=0;j<n;j++)
{ sum_s=0.0;
sum_s=sum_s+a3[i][j]-m3[i][j];
ab3[i][j]=sum_s;}}
/* printf("\nvalue of ab3 mat");
for (i=0;i<n;i++)
{printf("\n\n %ft %ft %ft %ft",ab3[i][0],ab3[i][1],ab3[i][2],ab3[i][3]);}
*/

```

```

for(i=0;i<n;i++)
{ for (j=0;j<n;j++)
{sum_s=0.0;
for (k=0;k<n;k++)
{sum_s= sum_s + a[i][k]*ab3[k][j];
a4[i][j]= sum_s ;
}
}

```

```

}}}
/* printf("\nvalue of a4 mat");
for (i=0;i<n;i++)
    {printf("\n\n %ft %ft %ft %ft",a4[i][0],a4[i][1],a4[i][2],a4[i][3]);}
*/
for (i=0;i<n;i++)
    { sum4= sum4+ a4[i][i];}
/* printf ("\n\n value of sum4= %f",sum4); */

p4= sum4/ 4.0 ;
/* printf ("\n\n value of p4= %f",p4); */

for (i=0;i<n;i++)
    { for (j=0;j<n;j++)
        { m_4[i][j] = p4*I[i][j];}}

for (i=0;i<n;i++)
    { for (j=0;j<n;j++)
        { sum_s=0.0;
sum_s=sum_s+a4[i][j]-m_4[i][j];
ab4[i][j]=sum_s;}}

    for(i=0;i<n;i++)
    { for (j=0;j<n;j++)
        {sum_s=0.0;
for (k=0;k<n;k++)
        {sum_s= sum_s + a[i][k]*ab4[k][j];
a5[i][j]= sum_s ;
}}}

for (i=0;i<n;i++)
    { sum5= sum5+ a5[i][i];}
p5= sum5/ 5.0 ;

for (i=0;i<n;i++)
    { for (j=0;j<n;j++)
        {m5[i][j]=p5*I[i][j];}}

for (i=0;i<n;i++)
    { for (j=0;j<n;j++)
        { sum_s=0.0;
sum_s=sum_s+a5[i][j]-m5[i][j];
ab5[i][j]=sum_s;}}

    for(i=0;i<n;i++)
    { for (j=0;j<n;j++)
        {sum_s=0.0;
for (k=0;k<n;k++)

```

```

{sum_s= sum_s + a[i][k]*ab5[k][j];
a6[i][j]= sum_s ;
}}}

for (i=0;i<n;i++)
{ sum6= sum6+ a6[i][i];}
p6= sum6/ 6.0 ;

for (i=0;i<n;i++)
{ for (j=0;j<n;j++)
{inverse[i][j]=ab5[i][j]/p6;}}

/* fputs ("\n\ninverse of q matrix is",fp);
for (i=0;i<n;i++)
{fprintf(fp, "\n %ft %ft %ft %ft",inverse[i][0],inverse[i][1],inverse[i][2],inverse[i][3]);}
*/
/* printf ("\n\ninverse of q matrix is");
for (i=0;i<n;i++)
{printf("\n %ft %ft %ft %ft",inverse[i][0],inverse[i][1],inverse[i][2],inverse[i][3]);}
*/

for(i=0;i<n;i++)
{ for (j=0;j<n;j++)
{sum_s=0.0;
for (k=0;k<n;k++)
{sum_s= sum_s + inverse[i][k]*q[k][j];
test[i][j]= sum_s ;
}}}

/* fputs ("\n\n test for inverse of q matrix is",fp);
for (i=0;i<n;i++)
{fprintf(fp, "\n %ft %ft %ft %ft",test[i][0],test[i][1],test[i][2],test[i][3]);}
*/
/* printf ("\n\n test for inverse of q matrix is");
for (i=0;i<n;i++)
{printf("\n %ft %ft %ft %ft",test[i][0],test[i][1],test[i][2],test[i][3]);}
*/

for(i=0;i<n;i++)
{ for (j=0;j<n;j++)
{sum_s=0.0;
for (k=0;k<n;k++)
{sum_s= sum_s + inverse[i][k]*r[k];
ya[i]= sum_s ;
}}}
/* fputs ("\n\n ya matrix is",fp);
for (i=0;i<n;i++)
{fprintf(fp, "\n %ft %ft %ft %ft",ya[i][0],ya[i][1],ya[i][2],ya[i][3]);}

```



```

printf("\n\n ya matrix is");
for (i=0;i<n;i++)
{printf("\n %ft ",ya[i]);}
*/

for(i=0;i<n;i++)
{ for (j=0;j<n;j++)
{sum_s=0.0;
for (k=0;k<n;k++)
{sum_s= sum_s + Yb[i][k]*ya[k];
m4[j]= sum_s ;
}}}
y1old=yb[0];
y2old=yb[1];
y3old=yb[2];
y4old=yb[3];
y5old=yb[4];
y6old=yb[5];
for (i=0;i<n;i++)
{ for (j=0;j<n;j++)
{ sum_s=0.0;
sum_s=sum_s+m4[i]-11[i];
yb[j]=sum_s;}}

y1new=yb[0] ;
y2new=yb[1] ;
y3new=yb[2] ;
y4new=yb[3] ;
y5new=yb[4] ;
y6new=yb[5] ;
/* fputs ("\n\n yb matrix is",fp);
for (i=0;i<n;i++)
{fprintf(fp, "\n %ft %ft %ft %ft",yb[i][0],yb[i][1],yb[i][2],yb[i][3]);}

printf("\n\n yb matrix is");
for (i=0;i<n;i++)
{printf("\n %ft ",yb[i]);}
*/

for (b=0;b<n;b++)
{yt[b]=ya[b];}

convgnc1= fabs(y1new-y1old) ;
convgnc2= fabs(y2new-y2old) ;
convgnc3= fabs(y3new-y3old) ;
convgnc4= fabs(y4new-y4old) ;
convgnc5= fabs(y5new-y5old) ;

```

```
convgn6= fabs(y6new-y6old) ;
```

```
 } /* for acc loop */ /* while loop end */
```

```
 printf("\n at t=%f w=%f y1=%f y3=%f y5=%f\n",t,w,yb[0],yb[2],yb[4]);
```

```
 fprintf(fp," %ft %ft %ft %ft %f\n ",t,w/nf,k_1/fl*yb[0],k_1/fl*yb[2],k_1/fl*yb[4]);
```

```
 w=w+dw;
```

```
 }
```

```
 fclose(fp);
```

```
 getch();
```

```
 }
```

