Palestine Polytechnic University

Collage of Engineering & Technology

Mechanical Engineering Department

Graduation Project

Theoretical and experimental study of structure dynamic

Project Team:

Anwar Atteyia

Lu’ai Al-Shrouf

Raid Abu-Malash

Project Supervisor:

Eng. Jalal Al-salaiymeh

Hebron-Palestine

June, 2004

Palestinian Polytechnic University
Hebron-Palestine
Department of Mechanical Engineering

Theoretical and experimental study of structure dynamic

Project team

Anwar Atteyia Lu'ai Shrouf
Raid Abu-Malash

According to the direction of the project supervisor and by agreement the committee member’s entire .This project was submitted to department of mechanical Engineering and Technology to partially fulfill to the bachelor requirement for the department

Supervisor Signature

……………………

Committee member Signature

1…………………….. 2……………………. 3…………………………

Department head Signature

………………………………

July, 2004

Abstract

When the building response to earthquakes, each floor can be idealized of a total mass at a floor level, and the walls can be modeled as a massless transverse springs.

A three story buildings model is designed and a scotch-yoke mechanism is used to simulate an earthquake to give the base a sinusoidal motion.

The continuous system is analyzed as a discrete system, and dynamic behavior is described by a set of three order differential equation, solved by an elementary method procedure.

The particular and homogenous solution is found and simulated theoretically, and the comparison to practical results is done

To simulate this results a matlab simulation program is used to give a good user friendly environments.

Dedication

To our parents for there supports

To our teachers for there advices

To our friends

Acknowledgement

To begin with our prophet saying" that" who ever does not thank people, he or she would not thank God. So we would like to extend our gratitude to the administration of Palestine Polytechnic University in general, and the administration of the Mechanical Engineering Department in particular. At the same time express our special thanks to the teachers of the university with special reference to technician. Raid Abu markhia and Twfiq Amro and Mr. Amjad A. Kamal.
But all the while it was Eng. Younis Al-Fasfous who has indulged great contribution of enlightening our vision with his ideas in mechanical design.
Last but least, special acknowledgment of appreciation and respect would be rendered to our project Supervisor Eng. Jalal Al-salaiymeh. Little words remain short to express our regard and thank to our Supervisor for his sound ideas and suggestions that he used to give without grudges or complaints.
It's our pleasure to thank whoever participated in backing up this project to be a real and successful one.

Lu'ai Shrouf
Anwar Atteyia
Raid Abu-Malash

Table of Contents

List of Figures

List of symbols

Symbol	Definition	SI Units
$[a]$	flexibility matrix	m/N
b	width of column	m
c	Greatest distance between the centriod and the outer section.	m
CK	clock signal for flip flop chip	
d_1	Diameter of the motor shaft	
d_2	Diameter of the disk	
$[D]$	dynamical matrix	
E	modulus of elasticity	Pa
Eg	induced voltage	Volt
$F(t)$	external force applied to the plate	N
$F_{(Ti)}$	amplitude of transmitted force	N
F_0	amplitude of force $F(t)$	N
h	length of each column	m
I	moment of inertia	$kg.m^2$
I_a	armature current	A
$[I]$	identity matrix	
k	spring constant	N/m
Ka	motor constant	
k_c	shear stiffness for each columns	N/m
[k]	stiffness matrix	N/m

m	mass of each floor	kg
m_c	mass of each column	kg
[m]	mass matrix	kg
$M_{allowable}$	Maximum allowable moment.	$N.m$
P	total power	watt
q(t)	time dependant generalized displacement coordinates	
$q_i(0)$	initial generalized displacements	m
$\dot{q}_i(0)$	initial generalized velocities	m/s
Q(t)	generalized force	
Q	flip flop chip output	
r_a	armature Resistance	Ohm
t	thickness of each column	m
Ts	tensile strength	Pa
v	velocity	m/s
V	volume of the column	m^3
n_1	Nominal motor speed	
n_2	Desired or disk speed	
V_a	terminal Voltage	Volt
$Vi(s)$	input voltage of PI controller	Volt
$V_o(s)$	output voltage of PI controller	Volt
W	total work	Joule
x_j	displacement of jth mass	m
$\vec{x}_h(t)$	homogenous displacement of system	m
$\vec{x}_p(t)$	particular displacement of system	m
$\{x_j\}$	n-dimensional State vector for n-th order system	
$\dot{x}_j$	differential equation of system	

$X_i^{(j)}$	ith component of jth mode	
$\ddot{x}(t)$	acceleration vector	m/s^2
$x(t)$	displacement vector	m
$\ddot{Z}(t)$	acceleration acting on the system	m/s^2
¤	flux per pole	
ª	eigenvalue$1/\mu^2$	s^2
±	shear stress	N/m^2
²	period of oscillation	S
P_{cr}	Maximum critical load	N
Se'	Endurance limit	MPa
Ka	Surface condition modification factor	
Kb	Size modification factor	
Kc	Load modification factor	
Kd	Temperature modification factor	
Ke	Miscellaneous-effects modification factor	
Se'	Rotary-beam endurance limit	MPa
f	The fraction of Su_t	
S_f	The fatigue strength	MPa

Chapter One

Overview

Chapter One
Overview

1.1 Introduction

This chapter contains the project objectives and importance in addition to the Literature review.

1.2 Objectives of the project

The mains object of this project is to build a three block building model to examine the response behavior of building to earthquakes.

Also this project has the following educational objectives:

1- To examine how modeling the continuous systems as a discrete systems influences the normal mode of the systems.

2-To understand special physical properties of the dynamics characteristics

3- To relate experimental measurements to computer simulations of a system's dynamic behavior.

1.3 Importance of project

The importance of this project is because it deals with a vibration influences and there effects on a structure. So its effect on continues three degree mass system is to be studied to made a simple experiment to give a good indication about this effect .
In this project there is an opportunity to change the mass, stiffness and the force so that this project can be used as experimental laborites that can be used in labs

Also, this project can be used to test the effect of vibration on materials, by simulating the natural vibration as any type of force to affect the shaker.

1.4 Literature Review

The subject of vibration effects has many studying and research, most of these studies are focused on earthquakes and its influences on buildings, so that the researcher on there practical experiments made a vibrating system to work as a natural vibration like a shaker plates, They Also simulate the structure of building as a simple spring-mass system to give an indication a bout the vibration influences.

A simple three mass spring damped system was presented as a graduation project in PPU University. They affect the system by a linear reciprocating force and compute the displacement of each individual mass and by using the computer software mat lab they plot the displacement versus time for each mass [1].

A Three story building was presented by Henri Gavin using computer software to compute the response of a three-story building to simulated earthquakes. You may specify the stiffness and damping of the structure, and observe how modifications in stiffness and damping affect the building's response. The simulated earthquake is actually sinusoidal with frequencies and amplitudes changing over time. You may specify the initial frequency, the initial amplitude, the final frequency and the final amplitude of the simulated earthquake. When the data is plotted, you may zoom-in on any portion of the data using this software [7].

For other type of input force Cory Fisher used a variable speed by an balance masses disk to give input vibration to the system, he made a practical experiment and connect it to computer using a mat lab software to determine the acceleration encountered on sinusoidal vibration that simulate an earthquake [8].

Also , there are a different use of shaker by a researchers to simplify a study or to made an Experimental works , L.Braile made an experiment in shaker table to examine the damage effects of vibration on building structure , he used a different type of shakers using a table with a rollers and simulate the effects using computer software [9] .

To reduce the damage effects of vibration Tsukuba-shi, Fudo Kenken developed a system that can cause a rocking vibration under appropriate control during earthquakes .One of this system has weak base plates at the bottom of each steel column of the first story. When the weak base plates yield during a strong earthquake, the building causes rocking vibration. This paper examines the effect of rocking vibration with base plates yielding on earthquakes response of building. It is conclude that the rocking systems with weak base plates can reduce earthquake response of building by casing rocking vibration, based on nonlinear time history analysis [6].

The simulation of vibration using the mat lab software has been presented in many research working , Park, Jeong Gyu printed a book that give a good helping in using the mat lab software to simulate the vibration , he give a several examples in vibration and programming technique to solve it [10].

Chapter Two

Mathematical modeling

Chapter Two
Mathematical modeling

2.1 Introduction

This chapter contains a system modeling in its dynamic differential equations, and its solution by the elementary method to find the system natural frequencies; also it includes the harmonic motion analysis.

2.2 Mathematical modeling of the system

The system shown in figure (2.1) represent a multi-story building consist of three floors , each floor is consider as a rigid mass and each columns as a flexible spring .

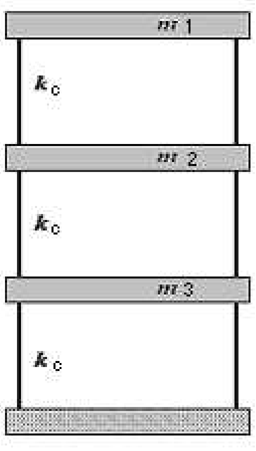


Figure (2.1): Idealization of system frame

For a base movement each floor is subjected to a lateral vibration and the transformation of the system is as shown in the figure (2.2) below.

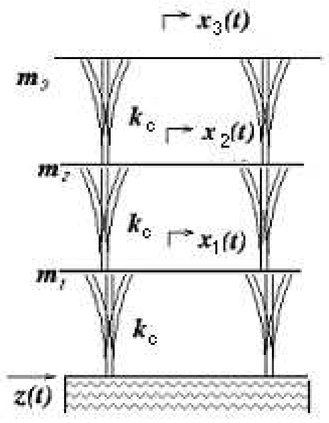


Figure (2. 2): Lateral vibration effects

The motion of the system is described by the coordinates $x_1(t)$, $x_2(t)$, $x_3(t)$ which define the position of the mass m_1 ,m_2 ,m_3 respectively . The external force F(t) acts on the system is due to base motion as in figure(2.3) .

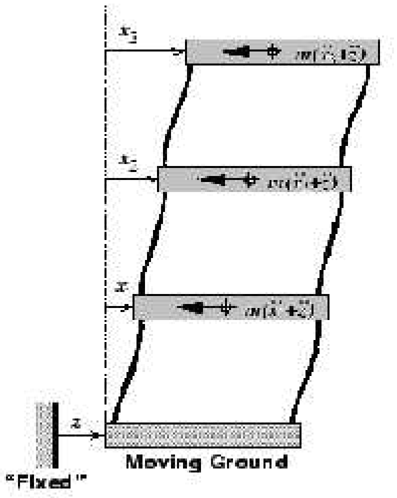


Figure (2.3): Mass movements

The equivalent spring-mass system is shown in figure (2.4) and a free body diagrams for each masses in figures (2.5) ,(2.6) ,(2.7)

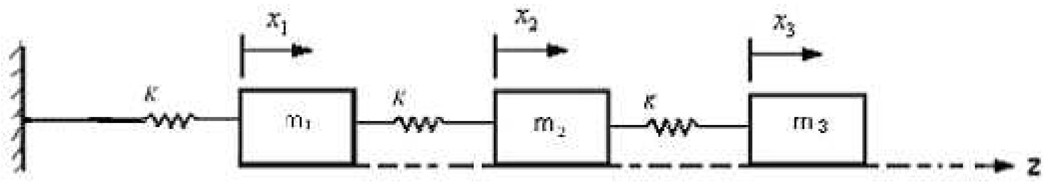


Figure (2. 4): Equivalent spring mass system

The columns are assumed to be of identical stiffness k_i and masses with equivalent mass m . Then the application of Newton second law for each mass according to the free body diagrams shown in figure (2.5), (2.6), (2.7) then the equation will be

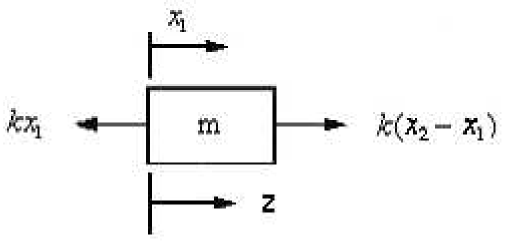


Figure (2. 5): Free body diagram of m_1

$$m(\ddot{x}_1 + \ddot{z}) = -kx_1 + k(x_2 - x_1)$$

$$m\ddot{x}_1 + 2kx_1 - kx_2 = -m\ddot{z} \tag{2.1}$$

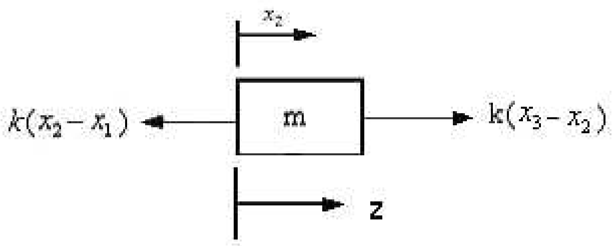


Figure (2.6): Free body diagram of m$_2$

$m(\ddot{x}_2 + \ddot{z}) = -k(x_2 - x_1) + k(x_3 - x_2)$

$m\ddot{x}_2 - kx_1 + 2kx_2 - kx_3 = -m\ddot{z}$ (2.2)

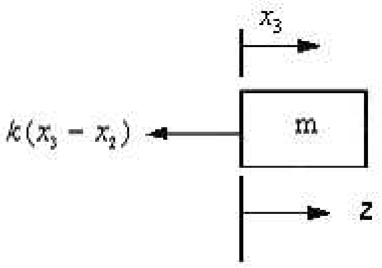


Figure (2. 7): Free body diagram of m$_3$

$m(\ddot{x}_3 + \ddot{z}) = -k(x_3 - x_2)$

$m\ddot{x}_3 - kx_2 + kx_3 = -m\ddot{z}$ (2.3)

From the three coupled differential equation the system can be written in a matrix form as in Eq (2.4)

$$\begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & m \end{bmatrix}\begin{bmatrix} \ddot{x}_1 \\ \ddot{x}_2 \\ \ddot{x}_3 \end{bmatrix} + \begin{bmatrix} 2k & -k & 0 \\ -k & 2k & -k \\ 0 & -k & k \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_2 \end{bmatrix} = -\begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & m \end{bmatrix}\ddot{z} \qquad (2.4)$$

$[m]\ddot{x}(t) + [k]x(t) = -[m]\ddot{z}(t)$ (2.5)

Where

[m] : mass matrix .

[k] : stiffness matrix .

$\ddot{z}(t)$: The force acting on the system.

$\ddot{x}(t)$: The acceleration vector.

$x(t)$: The displacement vector.

2.3 Natural frequencies for the system

To find the natural frequency of the system the eigenvalue problem is solved using an elementary method. The dynamical matrix is given by [1]:

$$[D] = [k]^{-1}[m] \equiv [a]\cdot[m] \tag{2.6}$$

Where

$$[a] = \frac{1}{k}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 2 \\ 1 & 2 & 3 \end{bmatrix} \tag{2.7}$$

And

$$[m] = m\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.8}$$

Thus,

$$[D] = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 2 \\ 1 & 2 & 3 \end{bmatrix}\frac{m}{k} \tag{2.9}$$

By setting the characteristic determinant equal to zero, we obtain the frequency equation:

$$=0 \qquad (2.10)\left\| \begin{bmatrix} a & 0 & 0 \\ 0 & a & 0 \\ 0 & 0 & a \end{bmatrix} - \frac{m}{k}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 2 \\ 1 & 2 & 3 \end{bmatrix} \right\| = \left| a[I] - [D] \right| \Delta=$$

Where

$$a = \frac{1}{\mu^{2}} \qquad (2.11)$$

By dividing throughout by λ Eq (2.10) gives

$$\begin{vmatrix} 1-\sim & -\sim & -\sim \\ -\sim & 1-2\sim & -2\sim \\ -\sim & -2\sim & 1-3\sim \end{vmatrix} =^{3}\text{-}5\sim \ \sim^{2} + 6 \sim -1 = 0 \qquad (2.12)$$

Where

$$\sim = \qquad (2.13)\ \frac{m}{ka} = \frac{m\mu^{2}}{k}$$

The roots of the cubic Eq(2.12) are given by

$$\sim_{1} = \frac{m\mu_{1}^{2}}{k} = 0.19806 \qquad (2.14)$$

$$\sim_{2} = \frac{m\mu_{2}^{2}}{k} = 1.5553 \qquad (2.15)$$

$$\sim_{3} = \frac{m\mu_{3}^{2}}{k} = 3.2490 \qquad (2.16)$$

So that

$$(2.17)\ \mu_{1} = 0.44504\sqrt{\frac{k}{m}}$$

$$(2.18)\ \mu_{2} = 1.2471\sqrt{\frac{k}{m}}$$

$$(2.19)\ \mu_{3} = 1.8025\sqrt{\frac{k}{m}}$$

2.4 Mode shapes

Once the natural frequencies are known, the mode shapes or eigenvectors can be calculated using Eq(2.20) [1]

$$\left[\mathrm{a}_i[I]-[D]\right]\vec{X}^{(i)}=\vec{0} \qquad , \mathrm{i} = 1, 2, 3 \tag{2.20}$$

Where

$$\vec{X}^{i}=\begin{Bmatrix} X_1^{(i)} \\ X_2^{(i)} \\ X_3^{(i)} \end{Bmatrix}$$

Denotes the i'th mode shape.

First mode: By substituting the value of μ_1 (i.e., $\mathrm{a}_1 = 5.0489\frac{m}{k}$) in Eq (2.20), we obtain

$$\left[5.0489\frac{m}{k}\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}-\frac{m}{k}\begin{bmatrix}1&1&1\\1&2&2\\1&2&3\end{bmatrix}\right]\begin{Bmatrix}X_1^{(1)}\\X_2^{(1)}\\X_3^{(1)}\end{Bmatrix}=\begin{Bmatrix}0\\0\\0\end{Bmatrix}$$

That is,

$$\begin{bmatrix}4.0489&-1.0&-1.0\\-1.0&3.0489&-2.0\\-1.0&-2.0&2.0489\end{bmatrix}\begin{Bmatrix}X_1^{(1)}\\X_2^{(1)}\\X_3^{(1)}\end{Bmatrix}=\begin{Bmatrix}0\\0\\0\end{Bmatrix} \tag{2.21}$$

Eq(2.21) denotes a system of three homogeneous linear equations in the three un-known $X_1^{(1)}$, $X_2^{(1)}$ and $X_3^{(1)}$. Any two of these unknowns can be expressed in terms of the remaining one. If we choose, arbitrarily, to express $X_2^{(1)}$ and $X_3^{(1)}$ in terms of $X_1^{(1)}$, we obtain from the first two rows of Eq(2.21)

$$\begin{aligned} X_2^{(1)} + X_3^{(1)} &= 4.0489 X_1^{(1)} \\ 3.0489 X_2^{(1)} - 2.0 X_3^{(1)} &= X_1^{(1)} \end{aligned} \qquad (2.22)$$

Once Eqs (2.22) are satisfied, the third row of Eq (2.21) is satisfied automatically.
The solution of Eqs (2.22) can be obtained:

$$X_2^{(1)} = 1.8019 X_1^{(1)} \quad \text{And} \quad X_3^{(1)} = 2.2470 X_1^{(1)} \qquad (2.23)$$

Thus the first mode shape is given by

$$\vec{X}^{(1)} = X_1^{(1)} \begin{Bmatrix} 1.0 \\ 1.8019 \\ 2.2470 \end{Bmatrix} \qquad (2.24)$$

Where the value of $X_1^{(1)}$ can be chosen arbitrarily.

Second mode: By substituting the value of $\mu_2 \left(i.e., \alpha_2 = 0.6430 \frac{m}{k} \right)$ in Eq (2.20) leads to

$$\left[0.6430 \frac{m}{k} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \frac{m}{k} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 2 \\ 1 & 2 & 3 \end{bmatrix} \right] \begin{Bmatrix} X_1^{(2)} \\ X_2^{(2)} \\ X_3^{(2)} \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 0 \end{Bmatrix}$$

That is,

$$\begin{bmatrix} -0.3570 & -1.0 & -1.0 \\ 1.0 & -1.3570 & -2.0 \\ -1.0 & -2.0 & -2.0489 \end{bmatrix} \begin{Bmatrix} X_1^{(2)} \\ X_2^{(2)} \\ X_3^{(2)} \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 0 \end{Bmatrix} \tag{2.25}$$

As before, the first two rows of Eq (2.25) can be used to obtain

$$\begin{aligned} -X_2^{(2)} - X_3^{(2)} &= 0.3570 X_1^{(2)} \\ -X_2^{(2)} - 2.0 X_3^{(2)} &= X_1^{(2)} \end{aligned} \tag{2.26}$$

The solution of Eqs (2.26) leads to

$$X_2^{(2)} = 0.4450 X_1^{(2)} \quad \text{And} \quad X_3^{(2)} = -0.8020 X_1^{(2)} \tag{2.27}$$

Thus the second mode shape can be expressed as

$$\vec{X}^{(2)} = X_1^{(2)} \begin{Bmatrix} 1.0 \\ 0.4450 \\ -0.8020 \end{Bmatrix} \tag{2.28}$$

Where the value of $X_1^{(2)}$ can be chosen arbitrarily.

Third mode: To find the third mode, we substitute the value of $\mu_3 \left(i.e., \text{a}_3 = 0.3078 \frac{m}{k} \right)$

in Eq (2.20) and obtain

$$\left[0.3078 \frac{m}{k} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \frac{m}{k} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 2 \\ 1 & 2 & 3 \end{bmatrix} \right] \begin{Bmatrix} X_1^{(3)} \\ X_2^{(3)} \\ X_3^{(3)} \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 0 \end{Bmatrix}$$

That is,

$$\begin{bmatrix} -0.6922 & -1.0 & -1.0 \\ 1.0 & -1.6922 & -2.0 \\ -1.0 & -2.0 & -2.6922 \end{bmatrix} \begin{Bmatrix} X_1^{(3)} \\ X_2^{(3)} \\ X_3^{(3)} \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 0 \end{Bmatrix} \tag{2.29}$$

The first two rows of Eq (2.29) can be written as

$$\begin{aligned} &-X_2^{(3)} - X_3^{(3)} = 0.6922 X_1^{(3)} \\ &-1.6922 X_2^{(3)} - 2.0 X_3^{(3)} = X_1^{(3)} \end{aligned} \tag{2.30}$$

Eq(2.30) give

$$X_2^{(3)} = -1.2468 X_1^{(3)} \quad \text{And} \quad X_3^{(3)} = 0.5544 X_1^{(3)} \tag{2.31}$$

Hence the third mode shape can be written as

$$\vec{X}^{(3)} = X_1^{(3)} \begin{Bmatrix} 1.0 \\ -1.2468 \\ 0.5544 \end{Bmatrix} \tag{2.32}$$

Where the value of $X_1^{(3)}$ is arbitrary. The values of $X_1^{(1)}$, $X_1^{(3)}$ and $X_1^{(3)}$ are usually taken as 1, and the mode shapes are shown in Fig (2.8)

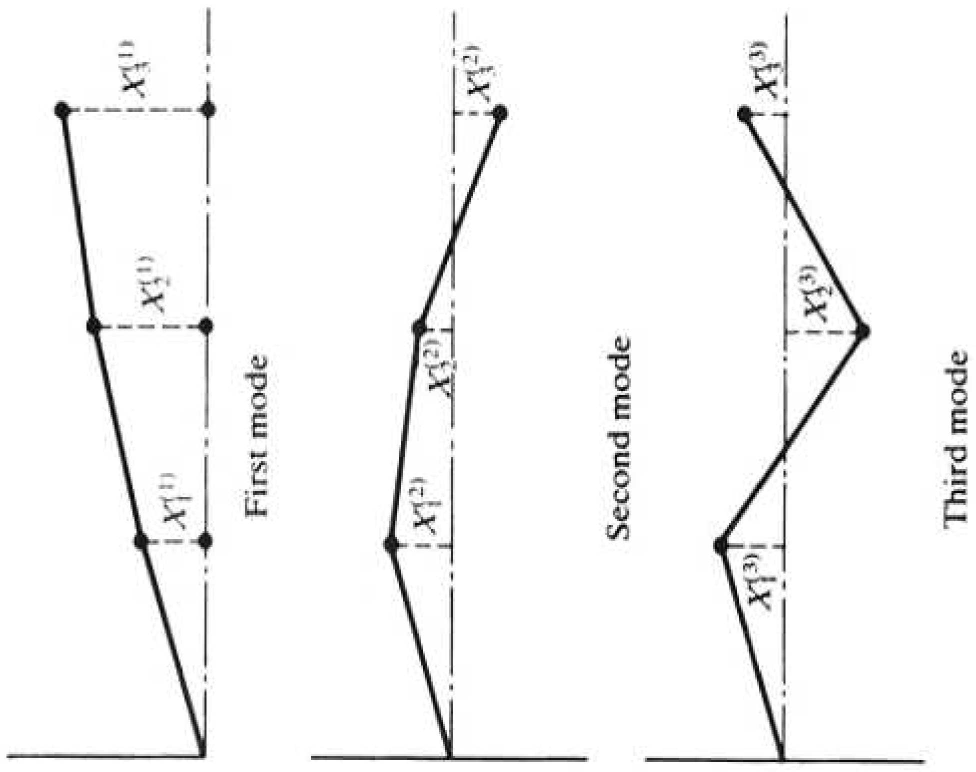


Figure (2.8): mode shape

2.5 Final system response

The total response or physical displacement which consist two parts; the first part homogenous displacement, and the second part the particular displacement.

From Eqs (2.24) and (2.28) and (2.32) the value of X_1 $^{(1)}$, X_1 $^{(2)}$, X1 $^{(3)}$ can be found using an orthonormalized of the Eigenvectors.
The mass matrix is given by

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

(2.33)

The eigenvector $\vec{X}^{(i)}$ is said to be [m] – orthonormal if the following condition is satisfied:

$$\vec{X}^{(i)T}[m]\vec{X}^{(i)} = 1 \qquad (2.34)$$

Thus for $i = 1$, Eq (2.34) leads to

$$m\left(X_1^{(1)}\right)^2\left(1.0^2 + 1.0819^2 + 2.2470^2\right) = 1$$

Or

$$X_1^{(1)} = \frac{1}{\sqrt{m(9.2959)}} = \frac{0.3280}{\sqrt{m}}$$

Similarly, for i = 2 and i = 3, Eq (2.34) gives

$$m\left(X_1^{(2)}\right)^2\left(1.0^2 + 0.4450^2 + (-0.8020)^2\right) = 1 \quad \text{Or} \quad X_1^{\,2} = \frac{0.7370}{\sqrt{m}}$$

And,

$$m\left(X_1^{(3)}\right)^2\left(1.0^2 + (-1.2468)^2 + 0.5544^2\right) = 1 \quad \text{Or} \quad X_1^{\,3} = \frac{0.5911}{\sqrt{m}}$$

Thus the modal vector can be expressed as

$$[X] = [\vec{X}^{(1)}\ \vec{X}^{(2)}\ \vec{X}^{(3)}] = \frac{1}{\sqrt{m}}\begin{bmatrix}0.328 & 0.737 & 0.5911\\ 0.5911 & 0.328 & -0.737\\ 0.737 & -0.5911 & 0.328\end{bmatrix} \qquad (2.35)$$

The generalized force $\vec{Q}(t)$ vector can be expressed as, [1]

$$\vec{Q}(t) = [X]^T \vec{F}(t) = \frac{1}{\sqrt{m}}\begin{bmatrix}0.328 & 0.737 & 0.5911\\ 0.5911 & 0.328 & -0.737\\ 0.737 & -0.5911 & 0.328\end{bmatrix}\begin{bmatrix}m & 0 & 0\\ 0 & m & 0\\ 0 & 0 & m\end{bmatrix}\ddot{z}$$

$$= \begin{Bmatrix} Q_{10} \\ Q_{20} \\ Q_{30} \end{Bmatrix} \cos \mu t \qquad (2.36)$$

Where

$[m]\ddot{z}$: Equal to the force applied, and

$\ddot{z} = -\mu^2 Z \cos \mu t$

So that we obtain,

$$\vec{Q}(t) = \frac{1}{\sqrt{m}} \begin{bmatrix} 0.328 & 0.737 & 0.5911 \\ 0.5911 & 0.328 & -0.737 \\ 0.737 & -0.5911 & 0.328 \end{bmatrix} \begin{bmatrix} -mZ\mu^2 & 0 & 0 \\ 0 & -mZ\mu^2 & 0 \\ 0 & 0 & -mZ\mu^2 \end{bmatrix} \begin{Bmatrix} \\ \cos \mu t \\ \\ \end{Bmatrix}$$

$$\vec{Q}(t) = \frac{1}{\sqrt{m}} \begin{bmatrix} 0.328 & 0.737 & 0.5911 \\ 0.5911 & 0.328 & -0.737 \\ 0.737 & -0.5911 & 0.328 \end{bmatrix} \begin{Bmatrix} \\ F_0 \cos \mu t \\ \\ \end{Bmatrix}$$

$$= \begin{Bmatrix} Q_{10} \\ Q_{20} \\ Q_{30} \end{Bmatrix} \cos \mu t \qquad (2.37)$$

Where

$$F_0 = \left| -mZ\mu^2 \right| \qquad (2.38)$$

Thus from Eq (2.37) we obtain

$$Q_{10} = 1.6561 \frac{1}{\sqrt{m}} F_0$$

$$Q_{20} = 0.473 \frac{1}{\sqrt{m}} F_0$$

$$Q_{30} = 0.1812 \frac{1}{\sqrt{m}} F_0$$

The final system response can be expressed as [1]

$$\vec{x}(t) = [X]\vec{q}(t) \qquad (2.39)$$

Where

$$\vec{x}(t)=\begin{Bmatrix}x_1(t)\\x_2(t)\\x_3(t)\end{Bmatrix}.$$

$$\vec{q}(t)=\begin{Bmatrix}q_1(t)\\q_2(t)\\q_3(t)\end{Bmatrix}.$$

$\vec{q}(t)$: Time dependant generalized displacement coordinates.

By substituting the variables in Eq (2.39)

$$\begin{bmatrix}x_1(t)\\x_2(t)\\x_3(t)\end{bmatrix}=\frac{1}{\sqrt{m}}\begin{bmatrix}0.328 & 0.737 & 0.5911\\0.5911 & 0.328 & -0.737\\0.737 & -0.5911 & 0.328\end{bmatrix}\begin{bmatrix}q_1(t)\\q_2(t)\\q_3(t)\end{bmatrix}$$

$$\begin{bmatrix}x_1(t)\\x_2(t)\\x_3(t)\end{bmatrix}=\frac{1}{\sqrt{m}}\begin{bmatrix}0.328q_1(t)+0.737q_2(t)+0.5911q_3(t)\\0.5911q_1(t)+0.328q_2(t)-0.737q_3(t)\\0.737q_1(t)-0.5911q_2(t)+0.328q_3(t)\end{bmatrix}\quad(2.40)$$

$$q_i(t)=q_i(0)\cos\mu_i t+(\frac{\dot{q}_i(0)}{\mu_i})\sin\mu_i t+\frac{1}{\mu_i}\int_0^t Q_i({}^2)\sin\mu_i(t-{}^2)d^2\quad(2.41)$$

The above equation represents the total generalized displacements of system (Homogenous and particular generalized displacements).

The final or physical displacement $\vec{x}(t)$ can be found using Eq(2.40), but this equation represent the total response or physical displacement which consist two parts; the particular and homogenous displacement .

$$\vec{x}(t)=\vec{x}_h(t)+\vec{x}_p(t)\quad(2.42)$$

Where

$\vec{x}_h(t)$: Homogenous displacement.

$\vec{x}_p(t)$: Particular displacement.

2.5.1 Particular displacement

When the system operates under excited force all initial condition $q_i(0), \dot{q}_i(0)$, are assumed to be equal zero, So that Eq (2.41) leads to

$$q_i(t) \;=\; \frac{1}{\mu_i}\int_0^t Q_i({}^2)\sin\mu_i(t-{}^2)d^2$$

$$q_i(t) \;=\; \frac{1}{\mu_i}Q_i\int_0^t \cos\mu^2 \cdot \sin\mu_i(t-{}^2)d^2$$

$$q_i(t) = \frac{1}{2\mu_i}Q_i\int_0^t [\sin(\mu^2+\mu_i(t-{}^2))+\sin(\mu^2-\mu_i(t-{}^2))]d^2$$

$$q_i(t) = \frac{1}{2\mu_i}Q_i[-\frac{1}{\mu-\mu_i}\cos(\mu_i t+{}^2(\mu-\mu_i))-\frac{1}{\mu+\mu_i}\cos({}^2(\mu+\mu_i)-\mu_i t)]_0^t$$

$$q_i(t) = \frac{1}{2\mu_i}Q_i\left[\left(\frac{-1}{\mu-\mu_i}-\frac{1}{\mu+\mu_i}\right)\cos(\mu t)+\left(\frac{1}{\mu-\mu_i}+\frac{1}{\mu+\mu_i}\right)\cos(\mu_i t)\right] \qquad (2.43)$$

Finally Eq(2.43) represent the particular generalized displacement of system.

So that,

$$q_1(t) = 0.82805\frac{\sqrt{m}Z\mu^2}{\mu_1}\left[\left(\frac{-1}{\mu-\mu_1}-\frac{1}{\mu+\mu_1}\right)\cos(\mu t)+\left(\frac{1}{\mu-\mu_1}+\frac{1}{\mu+\mu_1}\right)\cos(\mu_1 t)\right]$$

(2.44)

$$q_2(t) = 0.2365\frac{\sqrt{m}Z\mu^2}{\mu_2}\left[\left(\frac{-1}{\mu-\mu_2}-\frac{1}{\mu+\mu_2}\right)\cos(\mu t)+\left(\frac{1}{\mu-\mu_2}+\frac{1}{\mu+\mu_2}\right)\cos(\mu_2 t)\right]$$

(2.45)

$$q_3(t) = 0.0906\frac{\sqrt{m}Z\mu^2}{\mu_3}\left[\left(\frac{-1}{\mu-\mu_3}-\frac{1}{\mu+\mu_3}\right)\cos(\mu t)+\left(\frac{1}{\mu-\mu_3}+\frac{1}{\mu+\mu_3}\right)\cos(\mu_3 t)\right]$$

(2.46)

Final particular physical displacement $\vec{x}_p(t)$ can be expressed as follows:

$$\vec{x}_p(t) = \begin{bmatrix} x_{p1}(t) \\ x_{p2}(t) \\ x_{p3}(t) \end{bmatrix} = \frac{1}{\sqrt{m}} \begin{bmatrix} 0.328q_1(t) + 0.737q_2(t) + 0.5911q_3(t) \\ 0.5911q_1(t) + 0.328q_2(t) - 0.737q_3(t) \\ 0.737q_1(t) - 0.5911q_2(t) + 0.328q_3(t) \end{bmatrix} \tag{2.47}$$

2.5.2 Homogenous displacement

When the system operates without any excited force (free vibration) the generalized force $Q_i(^2) = 0$ and the homogenous generalized displacement becomes:

$$q_i(t) = q_i(0)\cos\mu_i t + (\frac{\dot{q}_i(0)}{\mu_i})\sin\mu_i t \tag{2.48}$$

The generalized displacement can be expressed as [1]

$$q_i(0) = [X]^T [m]\vec{x}(0) \tag{2.49}$$

By substituting the values of the variables in Eq (2.49)

$$q_i(0) = \frac{1}{\sqrt{m}} \begin{bmatrix} 0.328 & 0.737 & 0.5911 \\ 0.5911 & 0.328 & -0.737 \\ 0.737 & -0.5911 & 0.328 \end{bmatrix} \begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & m \end{bmatrix} \vec{x}(0)$$

$$q_i(0) = \sqrt{m} \begin{bmatrix} 0.328 & 0.737 & 0.5911 \\ 0.5911 & 0.328 & -0.737 \\ 0.737 & -0.5911 & 0.328 \end{bmatrix} \vec{x}(0) \tag{2.50}$$

The generalized velocities is expressed as [1]

$$\dot{q}_i(0) = [X]^T [m]\dot{\vec{x}}(0) \tag{2.51}$$

By substituting the variables in Eq (2.51):

$$\dot{q}_i(0) = \sqrt{m}\begin{bmatrix} 0.328 & 0.737 & 0.5911 \\ 0.5911 & 0.328 & -0.737 \\ 0.737 & -0.5911 & 0.328 \end{bmatrix}\dot{\vec{x}}(0) \tag{2.52}$$

$$q_i(t) = \sqrt{m}\begin{bmatrix} 0.328 & 0.737 & 0.5911 \\ 0.5911 & 0.328 & -0.737 \\ 0.737 & -0.5911 & 0.328 \end{bmatrix}\vec{x}(0)\cos\mu_i t$$
$$+\frac{\sqrt{m}}{\mu_i}\begin{bmatrix} 0.328 & 0.737 & 0.5911 \\ 0.5911 & 0.328 & -0.737 \\ 0.737 & -0.5911 & 0.328 \end{bmatrix}\dot{\vec{x}}(0)\sin\mu_i t \tag{2.53}$$

The final homogenous physical displacement $\vec{x}_h(t)$ can be expressed as follows:

$$\vec{x}_h(t) = \begin{bmatrix} x_{h1}(t) \\ x_{h2}(t) \\ x_{h3}(t) \end{bmatrix} = \frac{1}{\sqrt{m}}\begin{bmatrix} 0.328q_1(t) + 0.737q_2(t) + 0.5911q_3(t) \\ 0.5911q_1(t) + 0.328q_2(t) - 0.737q_3(t) \\ 0.737q_1(t) - 0.5911q_2(t) + 0.328q_3(t) \end{bmatrix} \tag{2.54}$$

2.6 Harmonic motion

The earthquake is to be represented by harmonic motion generated by scotch yoke mechanism as designed in chapter three.

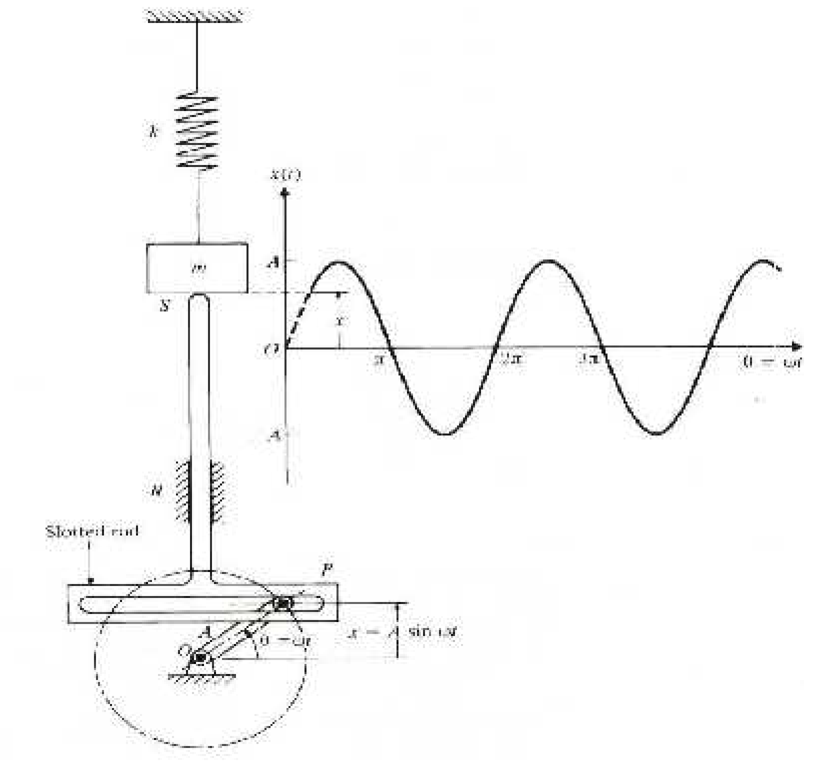


Figure (2.9) scotch Yoke mechanism

Chapter Three

Mechanical Design

Chapter Three

Mechanical Design

3.1 Introduction

In this chapter the necessary mechanical analysis is done, the required dimensions are calculated under certain considerations and the needed input power is calculated.

3.2 Mechanical system components and considerations

The small-scale model of the three-degree of freedom structure is shown in Figure(3.1)

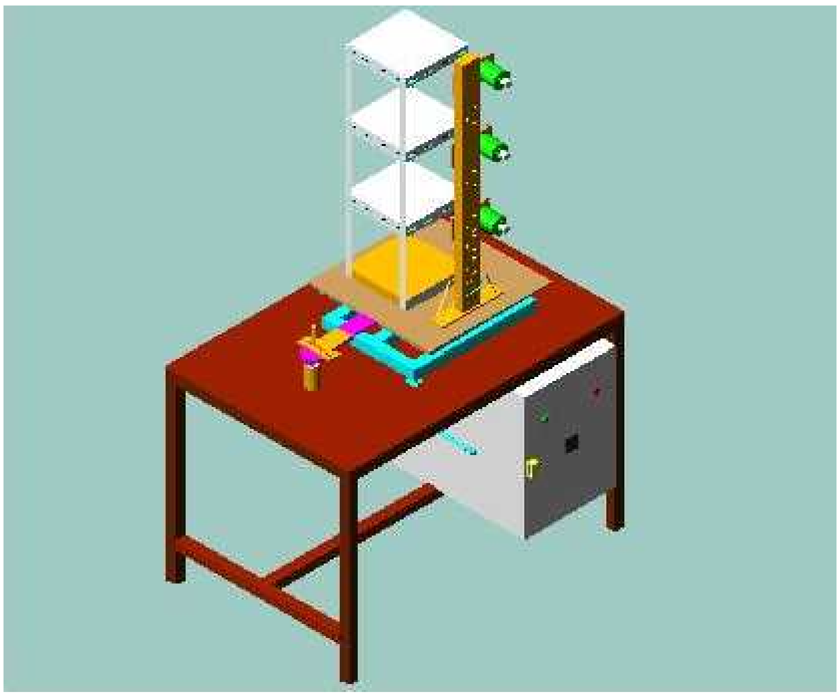


Figure (3.1): Structure general form

The structure specification:

1-Columns made from different materials like aluminum and steel.

2-Each of the four columns assumed massless with rectangular cross section of $b \times t$

3-Each layer has a floor-to-ceiling clear height h, and the structural columns are tightly clamped at each floor.

4-The floors are rigid plates of steel, and the weight can change by adding a masses to each layer .

5-The building model is placed on a table called shaking plate which can slide in one horizontal direction with almost no friction.

6-shaking plate is made to oscillate sinusoidal by scotch yoke mechanism.

3.3 Mechanical analysis design

The mechanical design is done for the cast iron columns by specifying the columns dimensions, determining the stiffness, the maximum deflection, maximum force to each column and the input power to the system
The aluminum analysis and other material is the same, so it's not presented here

3.3.1 Columns Design

A column which available in the local market is made up of cast iron (density ° =7200kg/m^3, modulus of elasticity $E = 100GPa$, tensile strength $Ts = 300MPa$), and with rectangular cross section of (b=20mm, t=2mm).

To find the shear stiffness (k_c) for each columns it's assumed to be a cantilever then,[3]

$$k_c = \frac{3EI}{l^3} \tag{3.1}$$

With $l = h = 18mm$ (assumed)

The moment of inertia I equal to

$$I = \frac{bt^3}{12} = \frac{20\times10^{-3}\times(2\times10^{-3})^3}{12} \tag{3.2}$$

$$= 13.33\times10^{-12}\, kg.m^2$$

From Eq(3.1) k_c is formed to be

$$k_c = \frac{3\times100\times10^9\times13.33\times10^{-12}}{0.18^3}$$

$$= 685.7 N/m$$

And the mass of each column can be calculated to be

$$m = ° \times V$$

Where

V : Volume of the column

m : Mass of each column

$$= ° \times(b\times t\times h) \tag{3.4}$$

$$= 7200\times0.02\times2\times10^{-3}\times0.18 = 0.052kg$$

Which can be neglected according to the mass of floor (rigid plate $m = 3kg$).

3.3.2 Equivalent shear stiffness

Each floor is assumed to be put over a four identical column which is can be consider as a four springs in the parallel each with $k_c = 685.7 N/m$,then k for each floor is:

$$k = k_{c1} + k_{c2} + k_{c3} + k_{c4} \tag{3.5}$$

$$= 4k_c = 4 \times 685.7$$

$$= 2742.3 N/m$$

3.3.3 Maximum Amplitude consideration

The maximum deflection of beams should be taken in consideration in order to be away from the failure or the plastic region of deformation which will give different properties of materials.

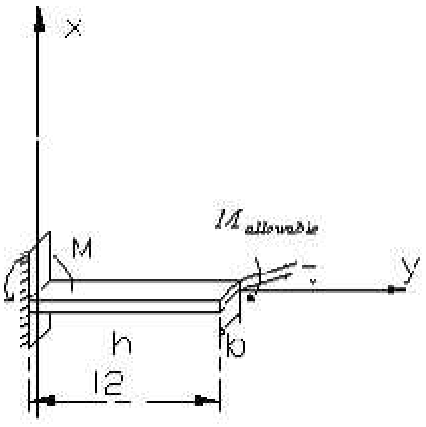


Figure (3.2): Column analysis

The maximum deflection in cantilever beam, [3]

$$x_{\max} = \frac{M_{allowable} \times h^2}{2 \times E \times I} \qquad (3.6)$$

And the shear stress given by, [3]

$$\pm = \frac{Mc}{I} \qquad (3.7)$$

To find the allowable moment the shear stress and tensile strength related to the relation $(\pm = 0.5 \times Ts)$ [2]

So Eq (3.7) leads to

$$M_{allowable} = \frac{Ts \times I}{2 \times c}$$

Where

$$c = \frac{t}{2}$$

$$c = \frac{2 \times 10^{-3}}{2} = 1 \times 10^{-3}\, m$$

The substation of Eq (3.7) into Eq (3.6) will give

$$x_{\max} = \frac{Ts \times I \times h^2}{2 \times c \times E \times I}$$

$$x_{\max} = \frac{Ts \times h^2}{2 \times c \times E} \qquad (3.8)$$

Where

c: The greatest distance between the centriod and the outer section.

$M_{allowable}$: Maximum allowable moment.

By substituting the magnitude of each variable in the above equation we find

$$x_{\max} = \frac{300 \times 10^{6} \times 0.18^{2}}{4 \times 1 \times 10^{-3} \times 100 \times 10^{9}}$$

$$= 0.0243m = 2.43mm$$

3.3.4 Maximum load (masses) consideration

The masses that added as a variable parameter in the system have limited considerations to be away from the buckling happing in columns

The columns is fixed from the two end, the relation is for maximum loading is given by [3]

$$P_{cr} = \frac{C \times \circledR \times E \times I}{l^{2}} \tag{3.9}$$

Where

C: The end condition constant and have a recommended value of 1.2, [3]

By substituting the variable in the above equation we obtains

$$P_{cr} = \frac{1.2 \times \circledR^{2} \times 100 \times 10^{9} \times 13.33 \times 10^{-12}}{0.18^{2}}$$

$$P_{cr} = 487.2N$$

The above load represent the critical load that place the column in the unstable equilibrium

For the four identical columns the critical load equal to

$$P_{cr} = 4 \times 487.2$$

$$P_{cr} = 1948.8N$$

3.3.5 Fatigue analysis

The fatigue due to the strength on column is determine with the number of cycle of the fatigue strength

The ultimate tensile strength for steel $Su_t = 341MPa$,[3]

The endurance limit Se' is given by, [3]

$$Se' = 0.506 \times Su_t \tag{3.10}$$

By substituting the ultimate strength in the above equation

$$Se' = 0.506 \times 341 \times 10^6$$

$$Se' = 172.5MPa$$

The Marin equation for the endurance limit is given by ,[3]

$$Se = Ka \cdot Kb \cdot Kc \cdot Kd \cdot Ke \cdot Se' \tag{3.11}$$

Where

Ka : Surface condition modification factor

Kb : Size modification factor

Kc : Load modification factor

Kd : Temperature modification factor

Ke : Miscellaneous-effects modification factor

Se' : Rotary-beam endurance limit

The surface modification factor Ka equal to, [3]

$$Ka = a \times Su_t^{\,b} \tag{3.12}$$

Where a and b are constant for hot rolled steel given by, [3]

$a = 56.1MPa$

$b = -0.719$

So that eq (3.12) become

$$Ka = 56.1 \times 341^{-0.719}$$

$$Ka = 0.847$$

The load modification factor Kc equal to, [3]

$$Kc = 1.23 \times Su_t^{-0.0778} \tag{3.13}$$

By substituting the ultimate strength Su_t in eq (3.13) we obtain

$$Kc = 1.23 \times 341^{-0.0778}$$

$$Kc = 0.781$$

The other factor is consider to be one, So that the endurance limit in eq (3.11) become equal to

$$Se = 0.847 \times 1 \times 0.781 \times 1 \times 1 \times 172.5 \times 10^6$$

$$Se = 114 MPa$$

The fatigue strength coefficient $\pm_f{}'$ equal to, [3]

$$\pm_f{}' = Su_t + 345 MPa \tag{3.14}$$

So that

$$\pm_f{}' = 341 \times 10^6 + 345 \times 10^6$$

$$= 680 MPa$$

The fraction of $Su_t \Rightarrow f$ is given by, [3]

$$f = \frac{\pm_f{}'}{Su_t}(2 \times 10^6)^b \tag{3.15}$$

Where the constant b equal to ,[3]

$$b = -\frac{\log(\frac{\pm_F{}'}{Se})}{\log(2N_e)} \tag{3.16}$$

So that,

$$b = \frac{\log(\frac{680 \times 10^6}{114 \times 10^6})}{\log(2 \times 10^6)}$$

$$b = -0.123$$

Eq (3.15) become

$$f = \frac{680\times10^6}{341\times10^6}(2\times10^3)^{-0.123}$$

$$f = 0.782$$

The fatigue strength is S_f equal to, [3]

$$S_f = aN^b \qquad (3.17)$$

Where a and b are constant with the relation,[3]

$$a = \frac{f^2 \times Su_t^{\,2}}{Se} \qquad (3.18)$$

$$a = \frac{0.782^2 \times (341\times10^6)^2}{114\times10^6}$$

$$a = 623.7MPa$$

$$b = -\tfrac{1}{3}\log(\frac{f \times Su_t}{Se}) \qquad (3.19)$$

$$b = -\frac{1}{3}\log(\frac{0.782\times341\times10^6}{114\times10^6})$$

$$b = -0.123$$

So that the fatigue strength for 10^4 cycle equal to

$$(S_f) \to 10^4 = 630\times10^6 \times (10^4)^{-0.106}$$

$$S_f = 186MPa$$

The expected life cycle to failure N can be expressed as, [3]

$$N = (\frac{\pm_a}{a})^{\frac{1}{b}} \qquad (3.20)$$

Where

$\pm_a$: The reserved stress

The area on which the force of $30N$ is concentrated is $20cm$,So that

$$\pm_a = \frac{30}{0.2} = 150N/m$$

The number of cycle to failure finally equal to, [3]

$$N = (\frac{150}{623.7\times10^6})^{\frac{-1}{0.123}}$$

$$N = 6.48\times10^{53}\,cycle$$

3.3.6 Scotch-Yoke Design

The base is to be excited sinusoidal by a scotch-Yoke mechanism, since $z = Z\cos \mu t$ where Z represent the displacement amplitude. The change in the amplitude is done by changing the hole as shown in figure (3.3)

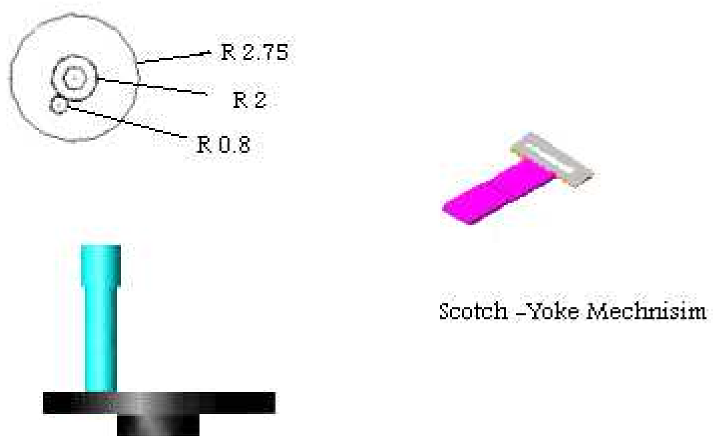


Figure (3.3): scotch –yoke mechanism

3.3.7 Shaking plate

The shaking consist of two parts, the lower part which is fixed figure(3.4.b), and the upper part which is the movement part; the building is fixed on the upper part as shown in figure(3.1).The upper part have guide to cancel any lateral movements. In other hand the lower part have a rollers to minimize the friction as possible figure (3.4 a)

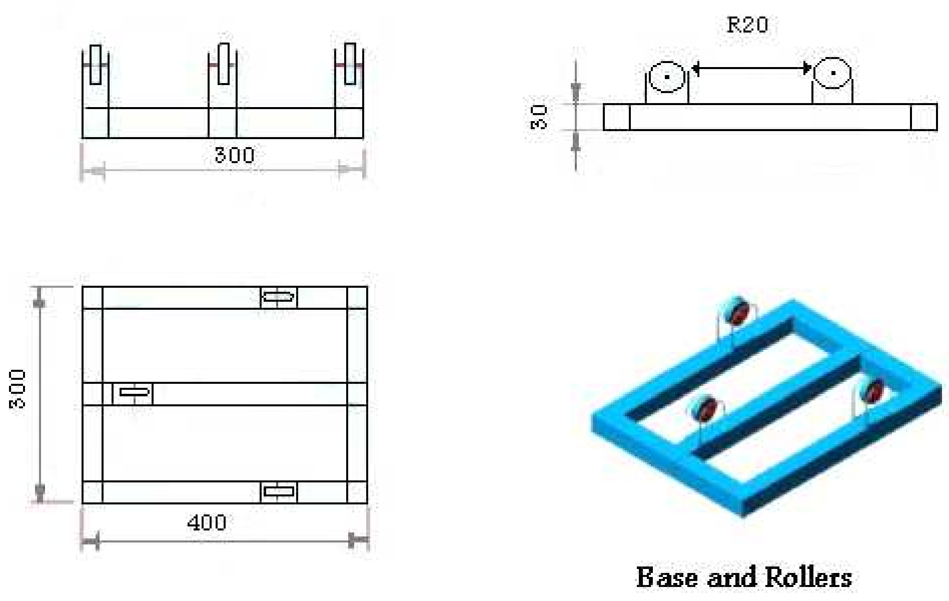


Figure (3.4 a): Lower part of shaking plate

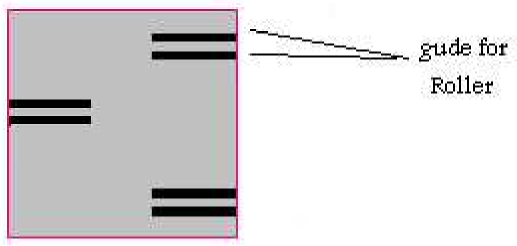


Figure (3.4 b): Higher part of shaking plate

3.3.8 Clamps

The selection of clamps to fixed the system because the clamps insure a uniform deflection along the beams, also it give a chance to change the length of columns to obtain a different stiffness of calculation

3.3.9 Optical encoder fixing

The rectangular beam figure (3.5) in which the optical encoder are fixed is selected to be very rigid to insure that there are no deflection or movements occur, So that the results of displacements measured by optical encoders have no error.

The rectangular beam also has a different point to fixed the optical encoder to match other tested structure dimension

Figure (3.5): Rectangular beam specification

The optical encoder fixing figure (3.6) is obtain by the rectangular plates with a three screws to cancel any movement of encoder that cause errors in results

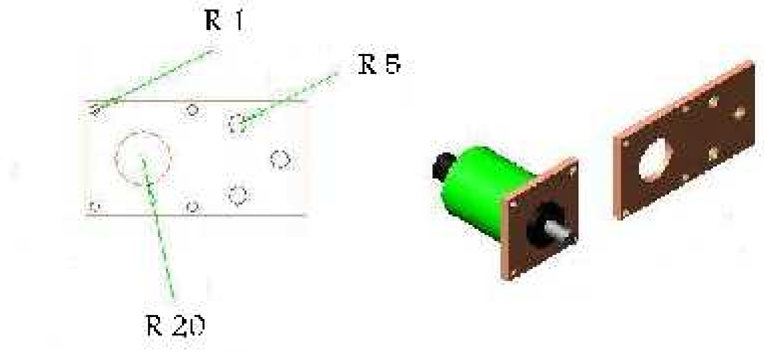


Figure (3.6): Optical Encoder fixing

3.3.10 Reducing motor speed and increasing its torque

A mechanism is design to increase the torque developed by the motor, and decreasing its speed .this mechanism consist of a disk made from polymer called echelon meshed with the motor shaft like pinion gear. The diameter of motor shaft (12mm) and the disk is design with a diameter of (125mm) as shown in figure (3.7)

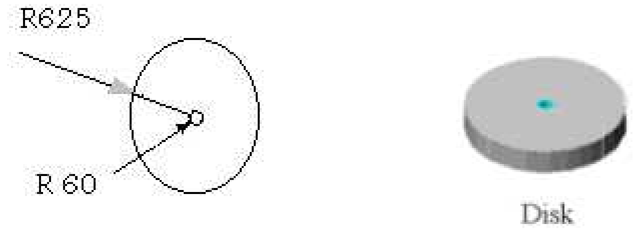


Figure (3.7): Ackelon disk

From the data sheet we see that the nominal motor speed 4950 rpm which is very high and the torque is 20 N.cm; by using this mechanism we can get the desired speed and torque using the following rule, [4]

$$\frac{n_2}{n_1} = \frac{T_1}{T_2} = \frac{d_1}{d_2} \tag{3.21}$$

By substituting the above information we obtain

$$\frac{n_2}{4950} = \frac{20}{T_2} = \frac{12}{125}$$

So that the final speed and torque is

$n_2 = 475.5 rpm$

$T_2 = 208.33 N.cm$

Where

n_1: Nominal motor speed

n_2: Desired or disk speed

d_1: Diameter of the motor shaft

d_2: Diameter of the disk

The motor shaft is covered with a rubber, and the disk has a course surface to make strong meshing without any sliding

3.3.11 Plate or layer selection

Plates are design with a square section (200 mm), figure (3.8) and have four holes to fix columns on it.

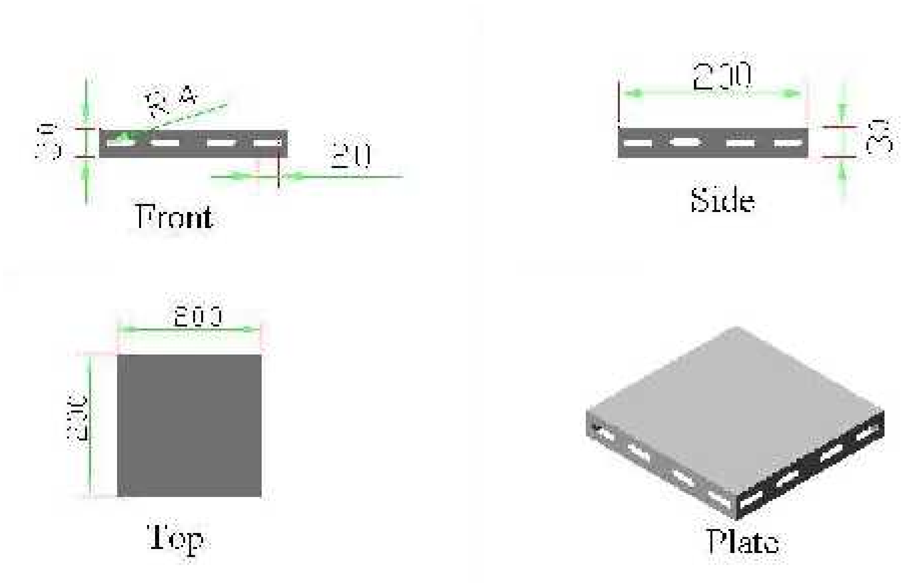


Figure (3.8): Square Section Plates

3.3.12 Other System Design Specification:

The shaft that handles the scotch -Yoke mechanism is fixed by cylinder guide to insure a pure rotation and to cancel unbalance rotation in the system; the guide contains of two bearing at the end to reduce friction as shown in figure (3.9).

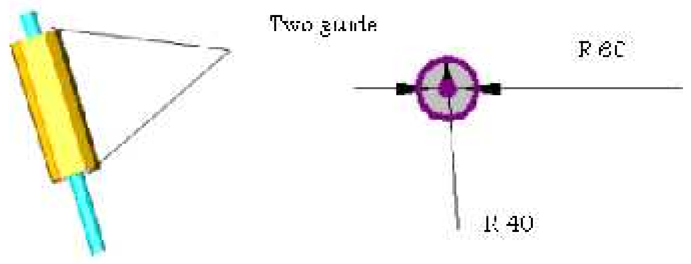


Figure (3.9): Cylinder guides for shaft

The whole system is constructed on a table, the length of 90cm,width of 60cm and the high equal 50cm.in which the guide and the system is in the top, the motor and gear are in bottom side, which give better shape.

The motor is fixed on a pin to mesh and release the disk to work as a clutch if any error is happening Figure (3.10)

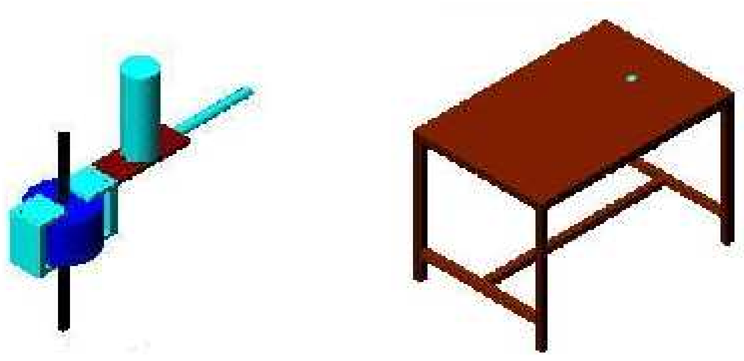


Figure (3.10) Table and motor clutch

3.4 Power Determination

In order to determine the maximum input power, the maximum angular velocity for the motor should be determine

From Eq(2.19)the maximum natural frequency is ω_3

$$\mu_3 = 1.8025\sqrt{\frac{k}{m}}$$

$$\mu_3 = 1.8025\sqrt{\frac{2742.3}{3}}$$

$$= 54.42 \text{ rad/s}$$

The study required frequency greater and less than the natural frequency of the system, So that the selected frequency is about 50 rad /s

The work done on the system equal to

$$dW = F \cdot dz \qquad (3.22)$$

For harmonic motion with an amplitude Z , the resultant work equal to

$$W = 4\int_0^Z m\ddot{z} \cdot dz$$

But $\ddot{z} = -Z\mu^2$, So that

$$W = 4\int_0^Z m\mu^2 z \cdot dz$$

$$W = 2m\mu^2 Z^2 \text{ in J unit} \qquad (3.23)$$

In watt unit

$$P = W \times \frac{\mu}{2\circledR} = 2m\mu^2 Z^2 \times \frac{\mu}{2\circledR}$$

$$P = \frac{m\mu^3 Z^2}{\circledR}(W) \qquad (3.24)$$

The system is assumed to move as a rigid body with a total mass *m* ,which equal to the movement part of the shaking plate mass, the rectangular beam, three optical encoder and three floor masses.

The total mass without the three floor equal 16.72kg, and each floor as determine before to be equal to 3 kg, then the whole mass will be 20.02 kg

The desired input power finally equal

$$P = \frac{25.72 \times 50^{3} \times (1 \times 10^{-2})^{2}}{®} = 131.1W$$

Chapter Four

Electrical Design

Chapter Four
Electrical Design

4.1 Introduction

This chapter contains the electrical design including the speed control of dc motor and the components used, also it include the type of sensor to measure the displacements The block diagram in figure (4.1) specifies the electrical procedure for controlling the system to create a control system for measuring a vibration effects.

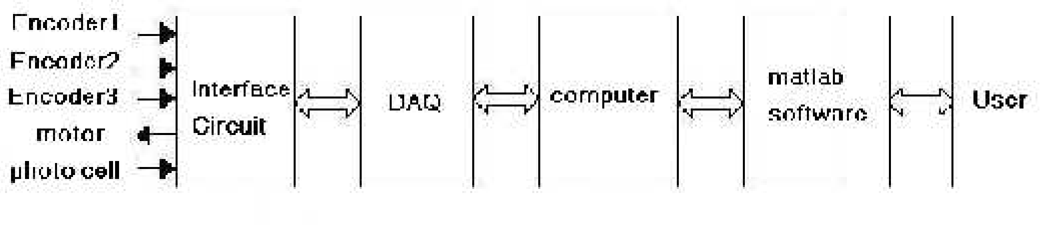


Figure (4.1): The block diagram of system

4.2 DC motor

The devices used to convert electrical energy into mechanical energy, by electromagnetic means.

The permanent magnetic dc motor is used to obtain a mechanical action, which will convert to a sinusoidal motion using a scotch-Yoke mechanism.

4.2.1 Speed control

Speed control can be done in several methods, one of these methods is the armature voltage control, which is done by increasing the voltage which will increase the speed and vise versa; but increasing the voltage is up to the nominal value, and above this method is not allowed.

The characteristic is hard and parallel, and controlled rectifier or chopper can do this. Figure (4.2)

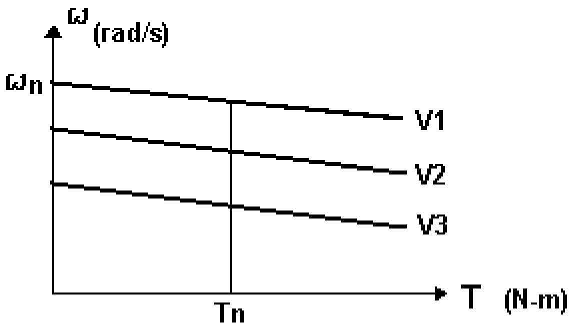


Figure (4.2) Voltage control of DC motor

The armature voltage control is used because it’s the most popular way used for large and small dc machine, and the speed is easily controlled from zero to maximum safe speed in either forward or reverse direction.

The electric circuit model for dc-machine shown in figure (4.3) below

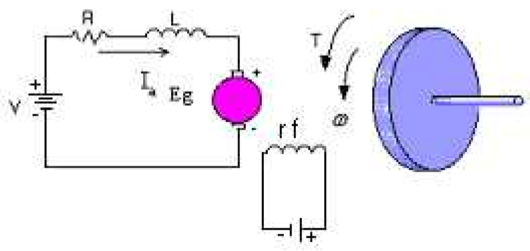


Figure (4.3) Electric circuit for Dc motor

The kirchhoffs law voltage equation for the armature gives (where the inductance is very small and neglected)

$Eg = V_a - I_a r_a$ (4.1)

Where

Eg : Induced voltage

V_a : Terminal Voltage

I_a : Armature Current

r_a :Armature Resistance

The induced voltage Eg is [2]

$Eg = K_a ¤ µ$ (4.2)

So Eq (4.1) becomes

$V_a = K_a ¤ µ + I_a r_a$ (4.3)

Produce µ Solving for

$$µ = \frac{V_a - I_a r_a}{K_a ¤}$$

So that motor speed

$$(4.4)\ \mu = \frac{V_a}{K_a ¤} - \frac{I_a r_a}{K_a ¤}$$

4.3 Closed loop system for Dc motor speed

The use of a closed loop system for controlling the speed of DC motor is to get a constant speed which will give a constant frequency which is the goal of this type of control on this project.

For a closed loop figure (4.4) below refer to the control system components

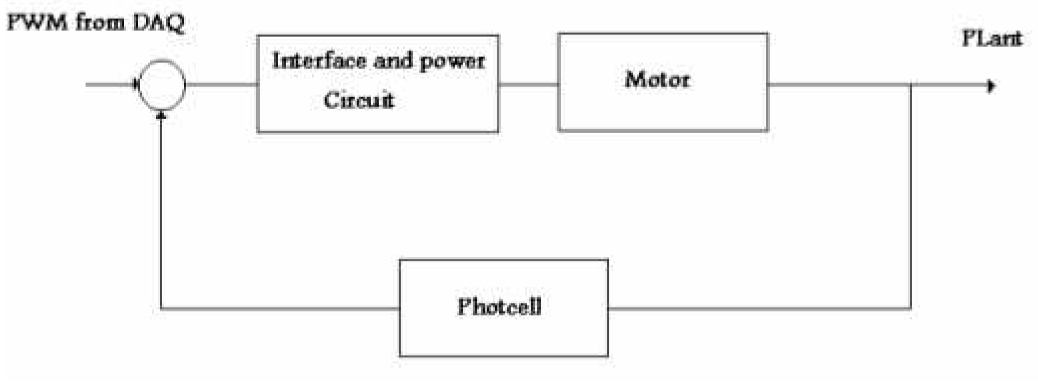


Figure (4.4) Block Diagram of speed control

From the figure (4.4) a different components are used on this closed loop system

4.3.1 Controller

The controller used is PI -controller to reduce steady-state error reducing. The implementation of this controller is internally software in the main program, and not by it physical components in the closed loop system.

4.3.2 Photocell

The photocell consists of photodiode and emitting PNP-transistor which used an emitting light from the sender diode to work as a switch.

The circuit work as an simple optical encoder in which a disk with one hole is placed on the shaft as shown in figure (4.5), pulse is generated when the hole is in line with the photodiode and emitting transistor, a pulses that is produced is counted internally by a computer software to obtain the real speed .

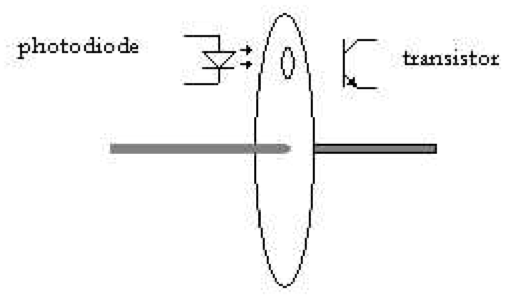


Figure (4.5): Photocell Principle

4.3.3 Power circuit

The drive circuit shown in figure (4.6) show the power E-MOSFET N-channel with a product number (IRFZ44), the gate of the MOSFET is controlled from a train of pulse generated from the DAQ, The motor is placed above the drain, with an input voltage of 25 V DC, the diode is placed to protect the transistor from the reversing voltage generated by the motor.

The gate of the MOSFET is connected to an isolation circuit (optocuplar) to protect the DAQ from any reversing current.

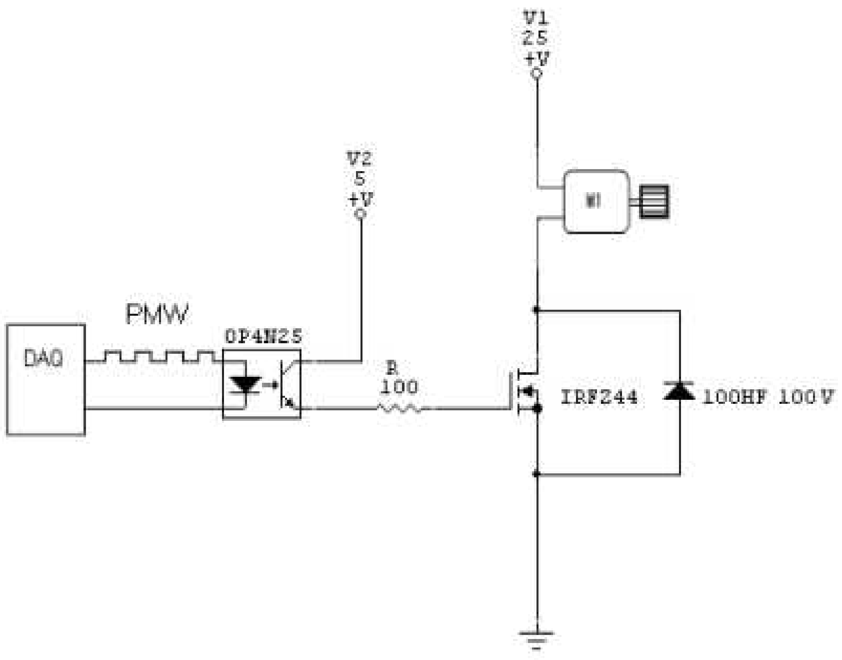


Figure (4.6): Driving circuit of Dc-motor

4.3.4 Pulse width modulation

Pulse width modulation (PWM) is a powerful technique for controlling analog circuits with a processor's digital outputs. PWM is employed in a wide variety of applications, ranging from measurement and communications to power control and conversion.

Figure (4.7) shows a PWM output at a 10% duty cycle. That is, the signal is on for 10% of the period and off the other 90%.

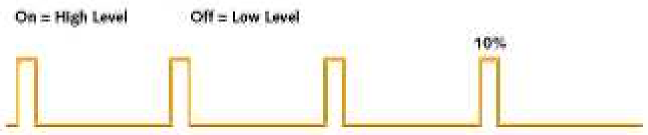


Figure (4.7): PWM signal duty cycles

The pulse width that used to control the E-MOSFET transistor (IRFZ44) is obtain from the DAQ-interface system by programming internally software program that generate a pulse with level voltage equal 5 V

4.4 Sensors

A sensor is a device that converts a physical phenomenon into a signal that can be fed into a processing unit like computer.

There are two main types of sensors based on the output they produce; the digital sensors and analog sensors.

The sensor type used in this project is an optical encoder .The use of this type because of it's measuring of rotational motion that can be used to measure the linear motion by the use of wire as rack and pinion mechanisms.

4.4.1 Optical encoder

An encoder is a device that provides a digital output as a result of a linear or angular displacement.

An optical encoder is a disk with a number of small rectangular slits placed around it as shown in figure (4.8). The light from the source is received at the sensor through a slit only when the slit is in line with the source and the sensor.

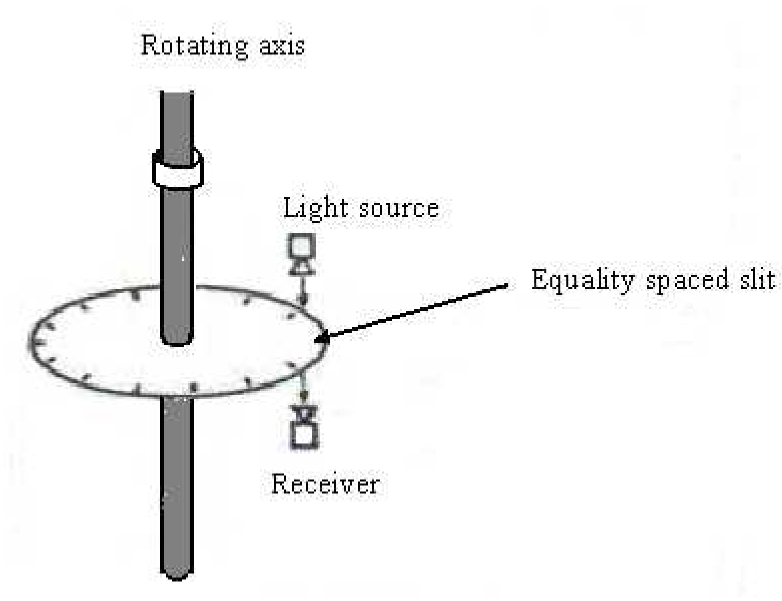


Figure (4.8): Optical encoder

The output pulses can be counted to detect the speed and velocity of rotary and linear motions of the system.

However, the pulse counts do not convey any information related to the direction of the rotation. In order to detect the direction of the motion, the encoder provides two output lines (A and B) that are 90° out of phase as shown in figure (4.9). Some encoders have a Z output to establish a "zero index". One pulse appears at the Z output per revolution.

The encoder frequency can be calculated as:

$$Encoder\_frequency = \frac{Encoder(rpm) \times Encoder(PPR)}{60}$$

Where

rpm: Revolution Per Minute.

PPR: Pulse per Revolution.

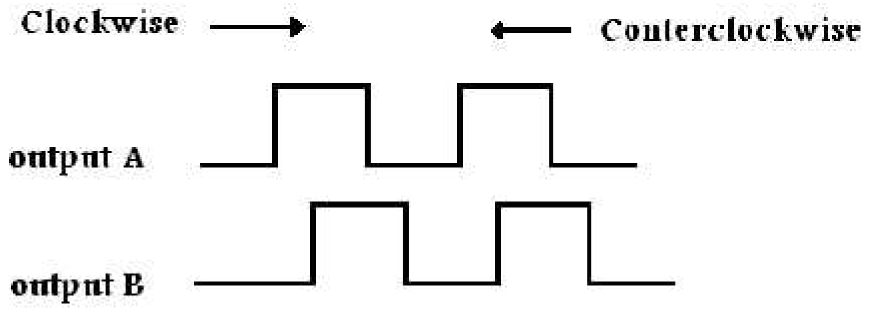


Figure (4.9): Rotary encoder output phases

4.4.1.1 Encoder Direction Detection Circuit

There are two output phases for the rotary encoder. The purpose of the two phases figure (4.9) is to determine the rotating direction, the edge-triggered D flip-flop is used in determining the rotating direction.

With an edge-triggered D flip-flop figure (4.10), the output Q is equal to the input D when the CK (clock) signal transit from low-to-high, so by connecting the two output phases A and B to the pins of the D flip-flop chip as shown in figure(4.10), the direction can be determined as in the following:

When the encoder rotates clockwise, from left to right in fig (4.9), and when B signal that is connected to the CK pin figure(4.10) transits from low-to-high, the A signal is always high as shown in figure(4.9). Thus, the output Q (flip flop chip output) is the same as signal A. Or in the other words, output Q equals 1 when the encoder rotates clockwise

On the other hand, when the encoder rotates counterclockwise, from right to left in figure (4.9) and when signal B, that is connected to the CK pin figure (4.10) transits from low to high, the A signal is always low as shown in figure(4.9). Thus, the output Q (flip flop chip output) is the same as signal A. Or in the other words, output Q equal zero when the encoder rotates counterclockwise.

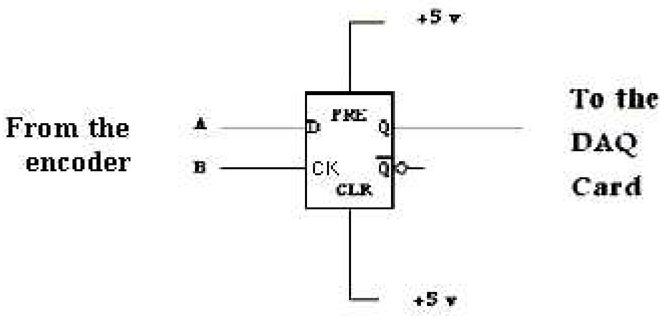


Figure (4.10): The edge-triggered D-flip flop

For the edge-triggered D flip-flop there are two other inputs called preset (PRE) and clear (CLR). A low on preset set the output Q to 1 while a low on clear clears the output ,setting Q to 0 .we connect the (PRE) and (CLR) to high because there is no need to set or clear the output.

Figure (4.11) explain the interfacing circuit with the DAQ system, it shows the D flip-flop and isolation components (optocouplars) that used for protection to DAQ.

4.4.2 Photocell

It’s a type of sensors which used to measure speed of the motor. Work principle of Photocell was explained in section (4.3.2)

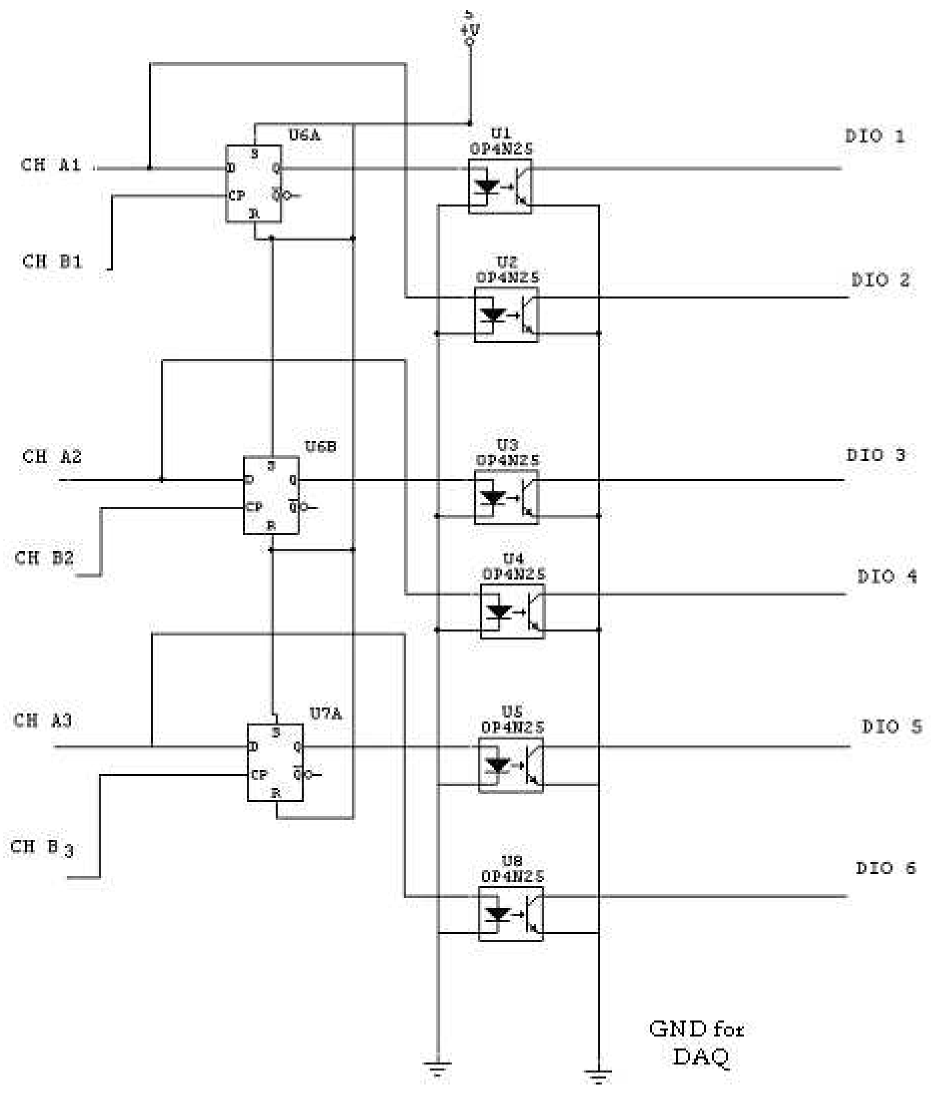


Figure (4.11): Interface circuit for optical encoder

4.5 Data Acquisition card

In order to be able to connect any physical system with a computer, as well as being able to control that system using a specific software, it is necessary to have some sort of hardware connection between these two worlds. This is the main aim behind using a DAQ or a data acquisition card.

A data acquisitions card consists mainly from:

• Analog input subsystem (A/D converter): Converts real analog signals (From a sensor) into bits that can be manipulated by a computer.

• Analog output subsystem (D/A converter): Converts digital data stored on the computer to real-world analog signal.

• Digital input/output subsystem (DIO): Designed to input and output digital value (logic levels) to and from hardware components.

• Counter/ Timer subsystem: Used for event counting, frequency and period measurement and pulse train generation.

The DAQ used in the case under consideration has a product number of (NI PCI-6601), and the main aim behind choosing such a cart is it internally counter and it ability of dealing with several programming languages (C++, Visual Basic, Matlab).

Chapter Five

Project discussion and implementation

Chapter Five
Project Implementation and Discussion

5.1 Introduction

This chapter contain the the implementation software and hardware that use to al forceto a sinsodi response simulate the system

5.2 Theoretical simulation

The response of the system to a sinusoidal force is derived in Eq (2.47) and Eq(2.54). This equations is implemented in the mat lab software using a GUI (graphical user interface), with a different variable (frequency, mass, stiffness, amplitude, initial displacement and velocities, time)

The new GUI is used to create a user-friendly operation environment .It is a user interface built with graphical objects, such as buttons, slider, text field, and menus.

The new GUI can be implemented from File→ New → GUI, then a new window will open as in figure (5.1), which help us in design the desired program properties

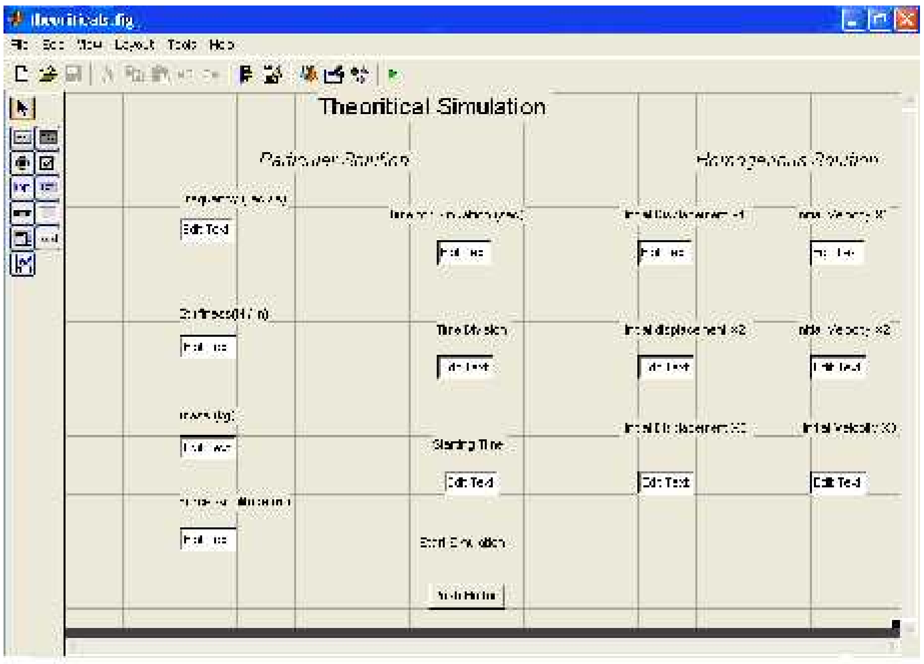


Figure (5.1) GUI-Window in Mat lab program

After the GUI is design with the desired input of mass, force …, the program is executed by typing the program name in the command window
When the program is executed it appear in a different form as in the figure (5.2) below

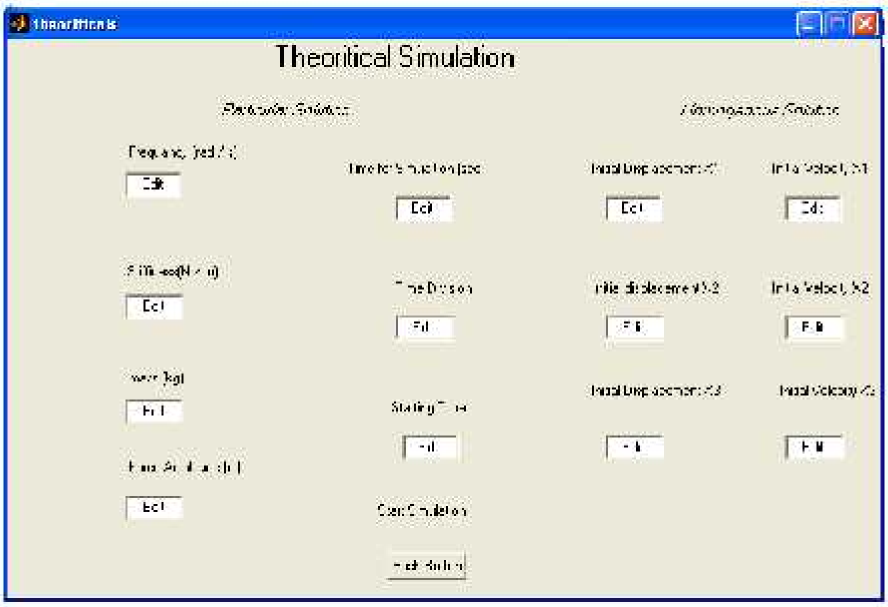


Figure (5.2) Executed theoretical GUI- Form

As an example for the implementation of GUI-program, let us consider the iron columns describe in chapter three .The desired columns stiffens is (2742.3N/m) and the floor weight is (1.96 kg).So the natural frequencies are

$$\mu_1 = 0.44504\sqrt{\frac{k}{m}} = 0.44504\sqrt{\frac{2742.3}{1.96}} = 16.6467 rad/s$$

$$\mu_2 = 1.2471\sqrt{\frac{k}{m}} = 1.2471\sqrt{\frac{2742.3}{1.96}} = 46.65 rad/s$$

$$\mu_3 = 1.8025\sqrt{\frac{k}{m}} = 1.8025\sqrt{\frac{2742.3}{1.96}} = 67.42 rad/s$$

If we assume the force amplitude to be (1 cm) with a speed of (16.6467 rad/s) which represent the first natural frequency.

The initial condition (x_1 = 1cm) and (x_2=2cm) and (x_3=4cm) with other values equal to zero.

Then after plugging the above values in the GUI-window, the desired system response is shown in the figure (5.3a and 5.3 b) below

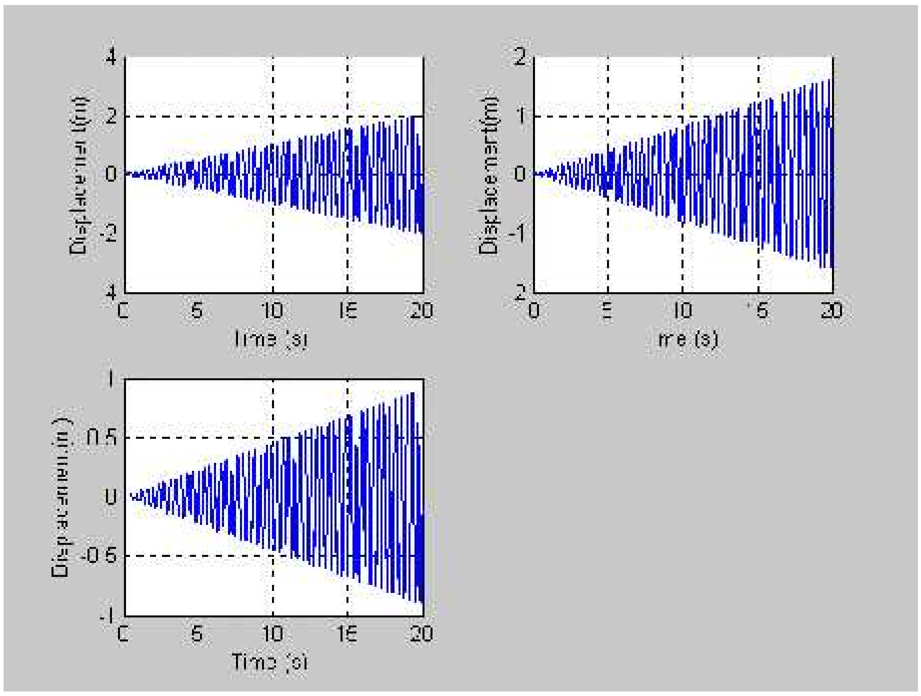


Figure (3.5 a) Particular Displacement Simulation

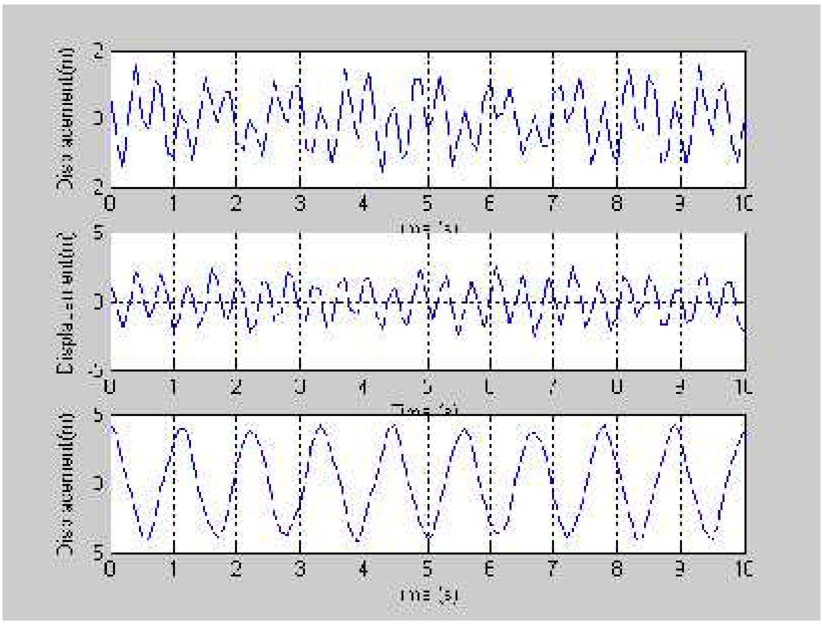


Figure (3.5 b) Homogenous Displacement Simulation

If we change the speed to 18 rad/s beating phenomena occur because the speed near to the natural frequency as shown in figure(5.c)

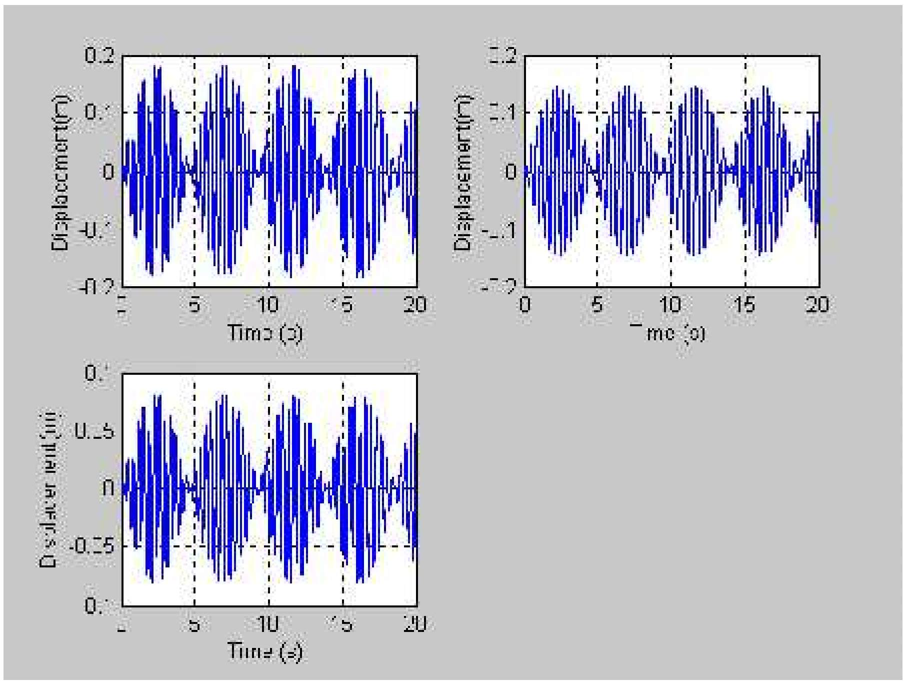


Figure (3.5 c) Particular Displacement Simulation

5.3 Practical Simulation

As we do in the previous GUI , we construct a GUI for the practical Simulation as shown in figure (5.4), the GUI require to input the time for Applying the force, and the motor speed
When the Push Button is pressed, the system start, and the analytical simulation from optical encoders is plotted on the mat lab simulation

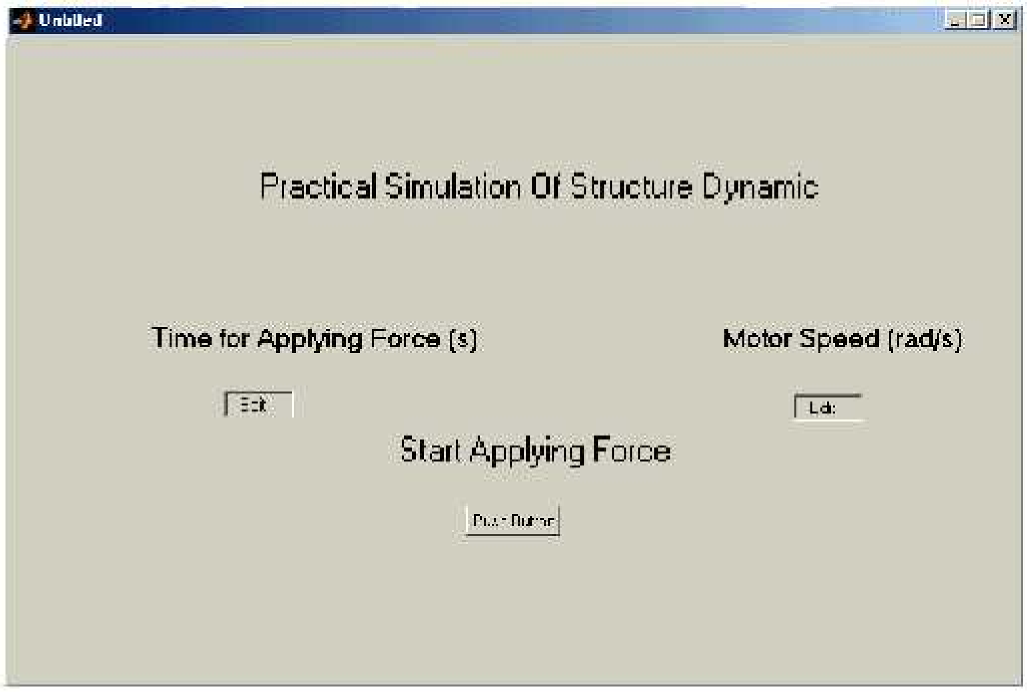

Untitled
Practical Simulation Of Structure Dynamic
Time for Applying Force (s)
Motor Speed (rad/s)
Edit
Edit
Start Applying Force
Push Button

Figure (5.4) Executed Practical GUI-Form

For the previous example, we adjust the weight for each floor, and the stiffness for each column, then as determine the desired frequency is input with the desired time. If the above data is calculated and enter, and the motor speed is enter with (17 rad /s) and the simulation is taken for (5 sec) then we obtain the desired plot as shown in the figure (5.5) below

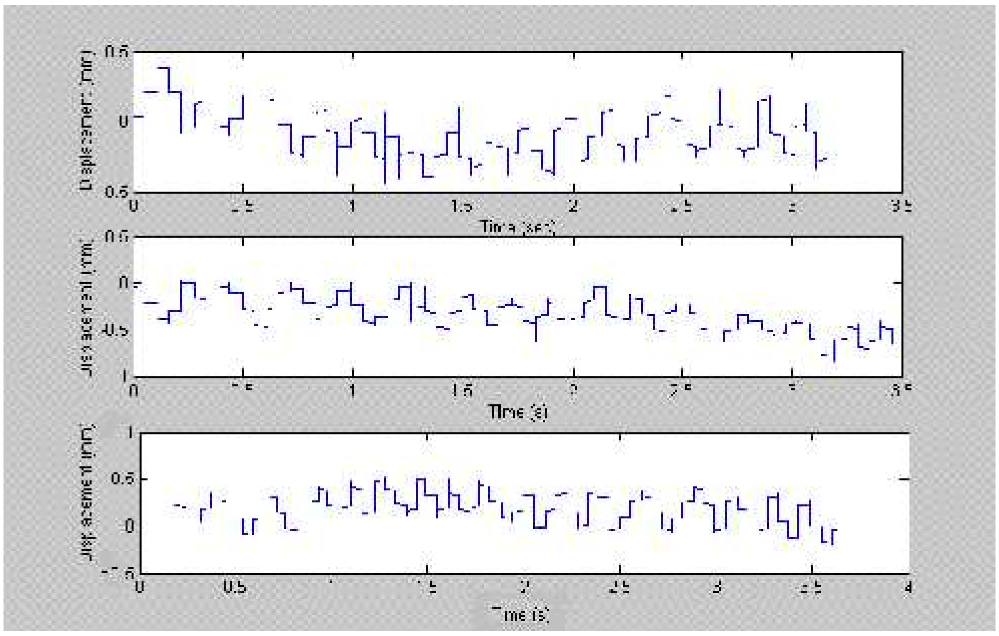


Figure (5.5) Practical Mat lab Simulation

5.4 Visual Basic Simulation

As we see from the previous plotted simulation in mat lab, the discrete signal coming from speed of plotting in mat lab regardless to slow rate of taken data from the DAQ , So we construct a Visual Basic program that take data directly to internal counter , then the data is taken a gain to mat alb program and plotted

The program user friendly interface window as shown in the figure (5.6) below

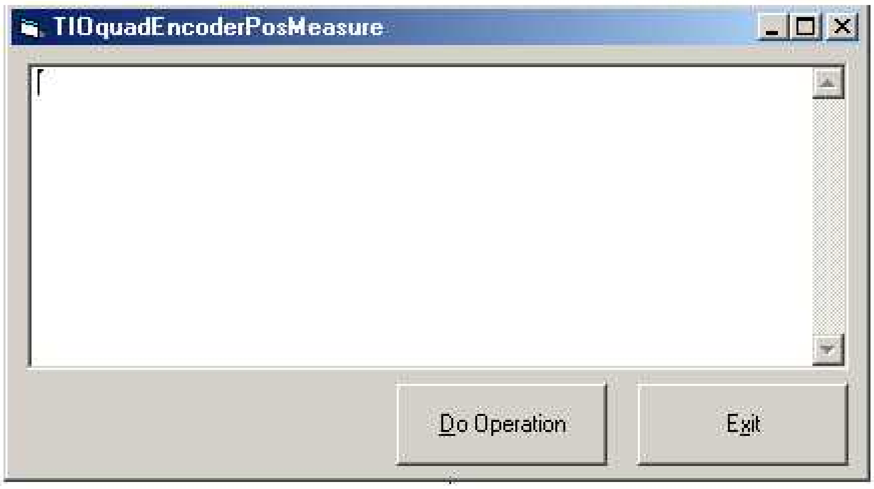


Figure (5.6) Visual Basic Operation window

As an example we take an aluminum column and made the calculation to the stiffness and frequency.

From Eq (3.2), with $l = h = 12cm, b = 20\times10^{-3}, t = 2\times10^{-3}$ the moment of inertia I equal to

$$I = \frac{20\times10^{-3}\times(2\times10^{-3})^3}{12}$$

$$= 13.33\times10^{-12}\,kg.m^2$$

From Eq(3.1) k_c is formed to be

$$k_c = \frac{3\times69\times10^9\times13.33\times10^{-12}}{0.12^3} = 1597N/m$$

$$k = 4k_c = 4\times1597 = 6388N/m$$

And the natural frequencies is calculated as

$$\mu_1 = 0.44504\sqrt{\frac{k}{m}} = 0.44504\sqrt{\frac{6388}{2.76}} = 21.4105rad/s$$

$$\mu_2 = 1.2471\sqrt{\frac{k}{m}} = 1.2471\sqrt{\frac{6388}{2.76}} = 59.997 rad/s$$

$$\mu_3 = 1.8025\sqrt{\frac{k}{m}} = 1.8025\sqrt{\frac{6388}{2.76}} = 86.7168 rad/s$$

The practical simulation to the above calculation in the visual basic program is appear in figure (5.7a)

If the above value in the pervious example is held constant, but the force amplitude is change to (2 cm), the desired simulation will be as shown in the figure (5.7b)

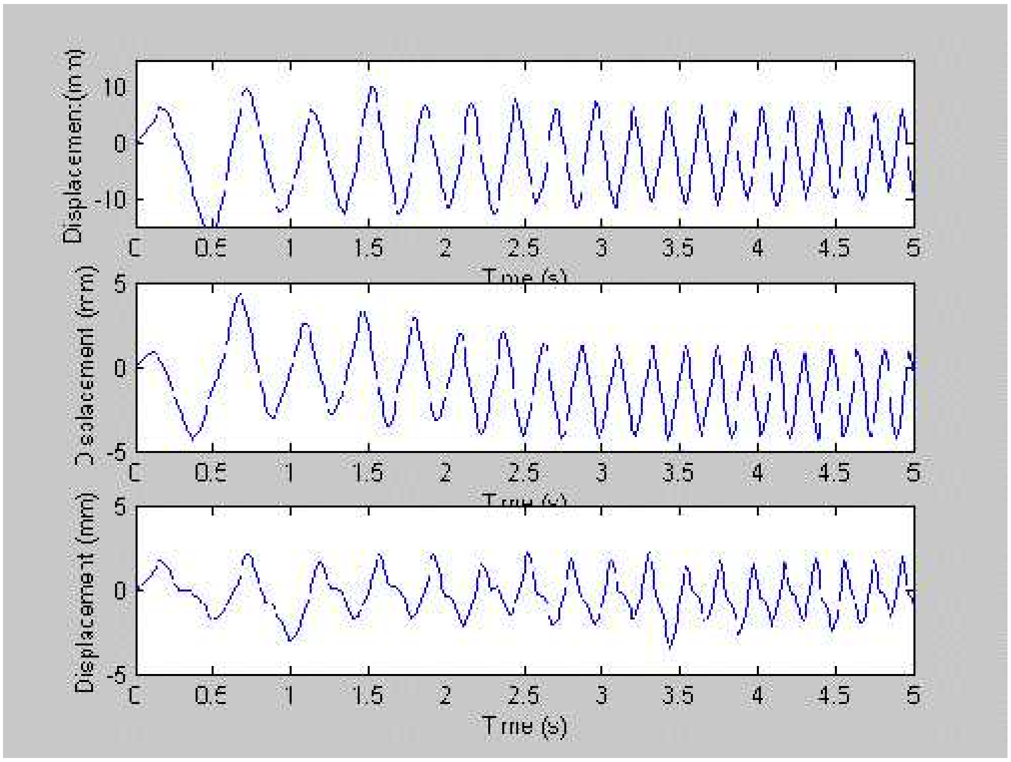


Figure (5.7a) Plotted Data from Visual Basic with mat lab

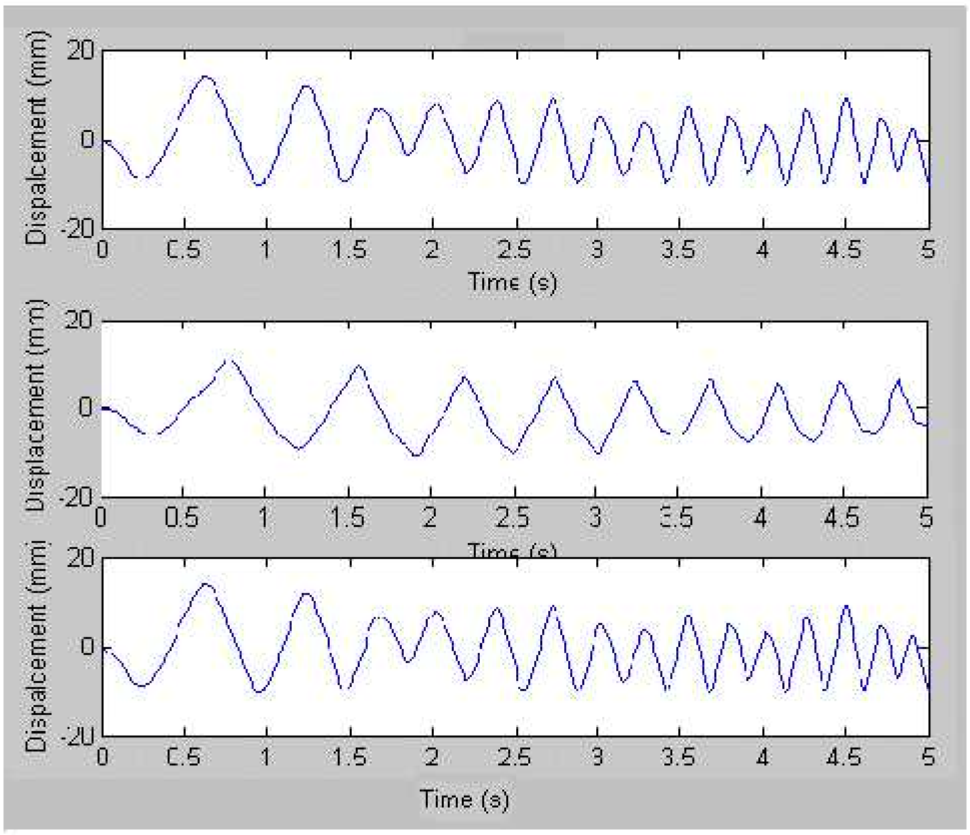


Figure (5.7b) Plotted Data from Visual Basic with mat lab

When we compare the above two figure, we see that deflection for each layer increase by increasing the force amplitude

The theoretical simulation for the particular solution in the mat lab appear in the figure (5.8a) below

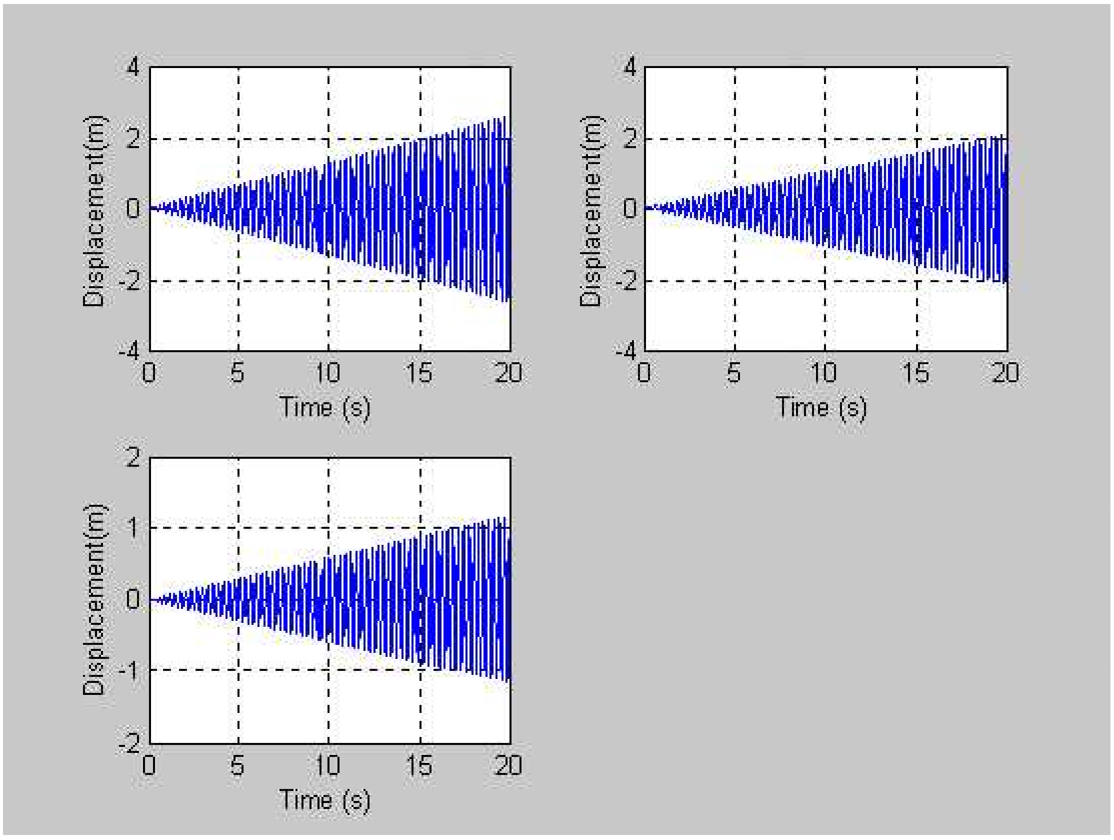


Figure (5.8a) particular displacement Simulation with force amplitude (1 cm)

When $\frac{\mu}{\mu_1} = 1$

When forcing frequency is close to but not exactly equal to, the natural frequency of the system, a phenomenon known as beating may occur. In this kind of vibration the amplitude builds up and then diminishes in a regular pattern [1]

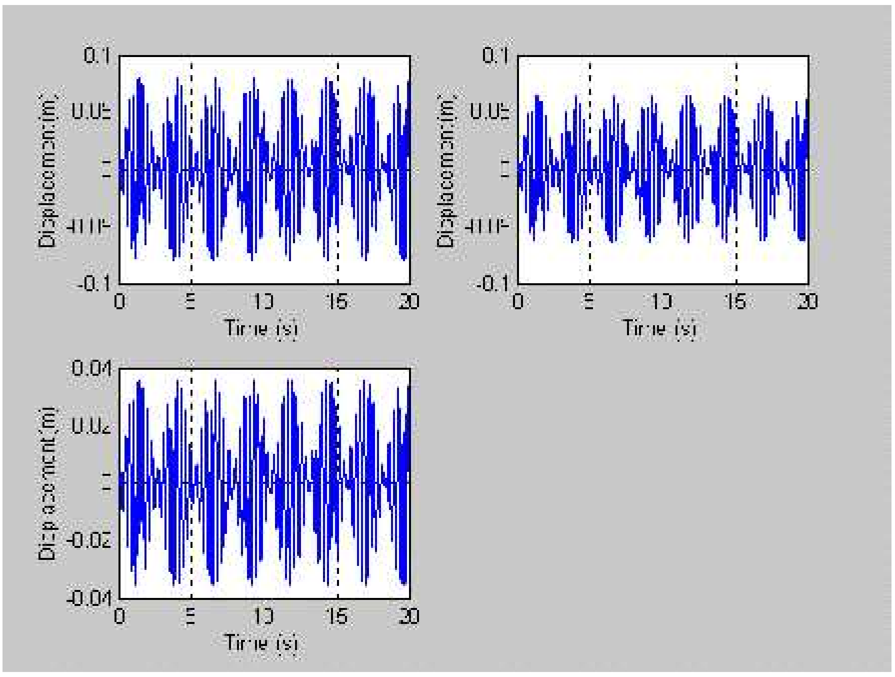


Figure (5.8b) particular displacement Simulation when $\frac{\mu}{\mu_1} < 1, \mu = 19$

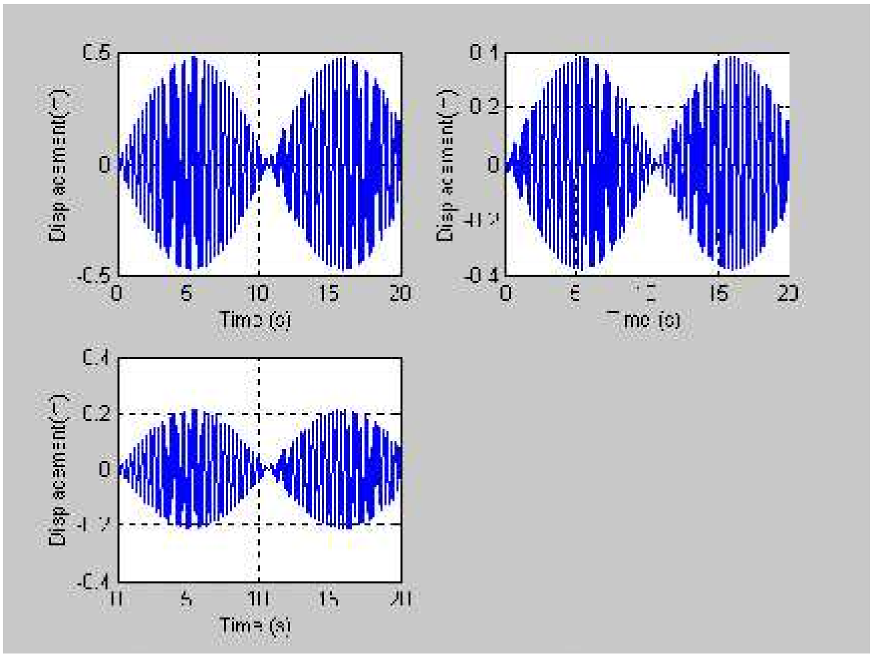


Figure (5.8c) particular displacement Simulation when $\frac{\mu}{\mu_1} > 1, \mu = 22$

The homogenous simulation shown in figure (5.8d)

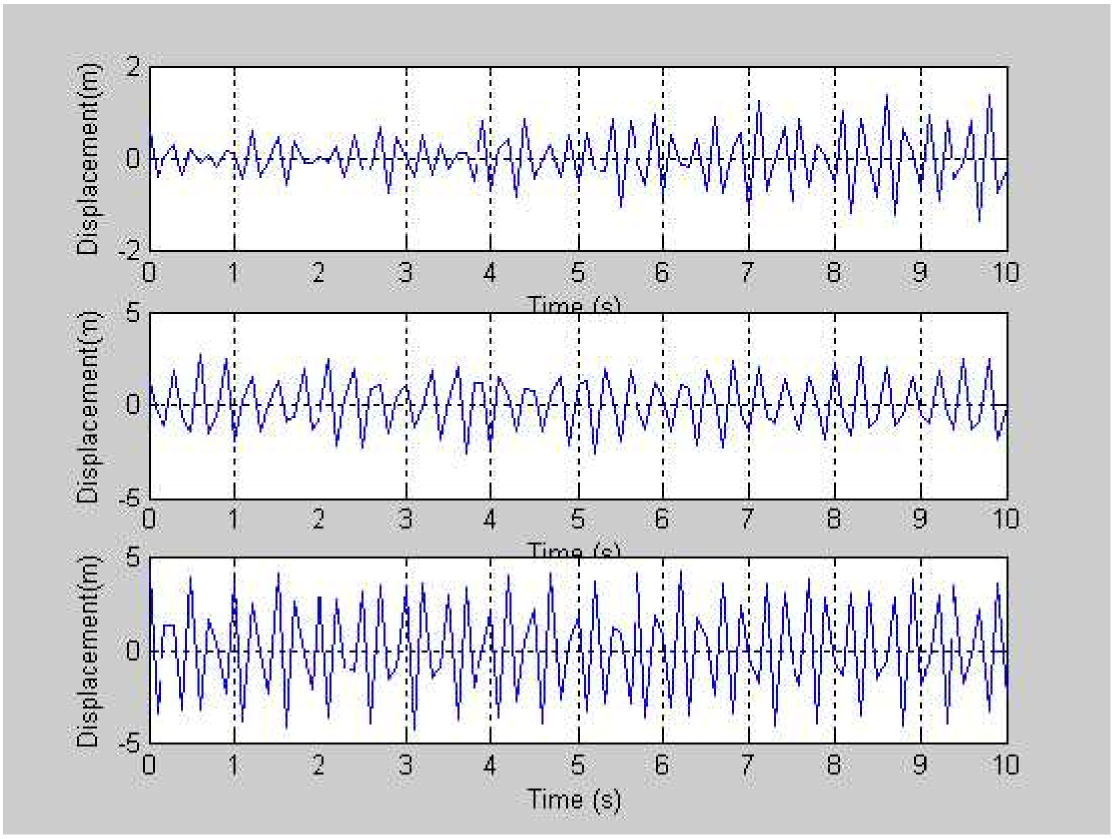


Figure (5.8d) homogenous displacement Simulation

Chapter Six

Conclusions and Recommendations

Chapter Six
Conclusion and Recommendations

6.1 Conclusion

1-The un agreement in the results refer to:

a-The damping factor is considered zero through the study, but there is a small damping in the system structure.

b- The system is studied as a discrete system, but the actual structure is a continuous system.

c- The fiction and losses found in the system structure.

d- The damping produced from using the tide, and the damping from encoder itself .

2- The system does not damage at the calculated natural frequency, because we not use the exact solution of the system.

3- The deflection for each layer increase by increasing the force amplitude.

4-The high natural frequencies haven't large effect on the building.

5- The discrete simulation result in the mat lab drawing coming from the plotting speed of mat lab comparing to data reading from the DAQ.

6.2 Recommendations

1- Using this device to make experiment s in lab to test the sinusoidal force effects

2- Build structure from concrete and make experiments on it

3- Study the three story building as continuous system and compare the result with practical solution.

4- Use vibration sensor instead of optical encoder.

References

[1]- Ayman Alzaro and Mohammad Ismaiel, "computerized vibration system", ppu Hebron, Palestine, 2001

[2]-George McPherson and Robert D. Lara more, An introduction to electrical machines and transformers, John wiley and son, Inc, University of Missouri and Purdue university, Canada, 1990

[3]-Joseph E .Shigley and Charles R. Mischke , Mechanical Engineering Design , Mc-Grow Hill university Of Michigan and Iowa university ,Canada , 2001

[4]-Joseph Edward, john Joseph and Uicker .JR, Theory of machine and mechanism, McGraw-Hill, University of Michigan, Singapore, 1980

[5]- Singiresus .RAO, Mechanical Vibration, Addison-Wesley publishing company, Purdue university, 1995

[6]- Tsukuba-shi, Fudo Kenken , "Damage Reduction Effect due to Basement Uplift of Buildings" AIJ, No.485, 53-62, Jul., Japanese, 1996

http://www.shodor.org/~reneeg/weave/module1/general.html [7]-

www.ceet.n iu.edu/faculty/kim/mee424/earthquate%[8]- http://

[9]-http:// www.eas.puwww.rdue.edu/~braile

www.pag.edu/tim.gyn/rerw/10 [10]-http:\\

Appendix A

The mat lab Software for the Theoretical Implementation of Structure dynamic is written below

```
function varargout = theoriticals(varargin)
% THEORITICALS M-file for theoriticals.fig
%      THEORITICALS, by itself, creates a new THEORITICALS or raises the
existing
%      singleton*.
%
%      H = THEORITICALS returns the handle to a new THEORITICALS or the
handle to
%      the existing singleton*.
%
%      THEORITICALS('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in THEORITICALS.M with the given input
arguments.
%
%      THEORITICALS('Property','Value',...) creates a new THEORITICALS or
raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before theoriticals_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to theoriticals_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help theoriticals

% Last Modified by GUIDE v2.5 22-Jun-2004 16:16:30

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...

                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @theoriticals_OpeningFcn, ...
                   'gui_OutputFcn',  @theoriticals_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
```

```
if nargin & isstr(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before theoriticals is made visible.
function theoriticals_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to theoriticals (see VARARGIN)

% Choose default command line output for theoriticals

handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes theoriticals wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = theoriticals_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
```

```
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a double


% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a double


% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3 as a double


% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
```

```
function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit4 as text
%        str2double(get(hObject,'String')) returns contents of edit4 as a double


% --- Executes during object creation, after setting all properties.
function edit7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function edit7_Callback(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit7 as text
%        str2double(get(hObject,'String')) returns contents of edit7 as a double


% --- Executes during object creation, after setting all properties.
function edit8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
```

```
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function edit8_Callback(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit8 as text
%        str2double(get(hObject,'String')) returns contents of edit8 as a double


% --- Executes during object creation, after setting all properties.
function edit9_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function edit9_Callback(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit9 as text
%        str2double(get(hObject,'String')) returns contents of edit9 as a double


% --- Executes during object creation, after setting all properties.
function edit10_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function edit10_Callback(hObject, eventdata, handles)
% hObject    handle to edit10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit10 as text
%        str2double(get(hObject,'String')) returns contents of edit10 as a double


% --- Executes during object creation, after setting all properties.
function edit11_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function edit11_Callback(hObject, eventdata, handles)
% hObject    handle to edit11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit11 as text
%       str2double(get(hObject,'String')) returns contents of edit11 as a double


% --- Executes during object creation, after setting all properties.
function edit12_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function edit12_Callback(hObject, eventdata, handles)
% hObject    handle to edit12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit12 as text
%       str2double(get(hObject,'String')) returns contents of edit12 as a double


% --- Executes during object creation, after setting all properties.
function edit13_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit13 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
```

```
function edit13_Callback(hObject, eventdata, handles)
% hObject    handle to edit13 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit13 as text
%        str2double(get(hObject,'String')) returns contents of edit13 as a double


% --- Executes during object creation, after setting all properties.
function edit14_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit14 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function edit14_Callback(hObject, eventdata, handles)
% hObject    handle to edit14 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit14 as text
%        str2double(get(hObject,'String')) returns contents of edit14 as a double


% --- Executes during object creation, after setting all properties.
function edit15_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit15 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
```

```
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function edit15_Callback(hObject, eventdata, handles)
% hObject    handle to edit15 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit15 as text
%        str2double(get(hObject,'String')) returns contents of edit15 as a double



% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)



fre=get(handles.edit1,'string');
frequancy=str2double(fre)

kk=get(handles.edit2,'string');
sttifness=str2double(kk);


mm=get(handles.edit3,'string');
mass=str2double(mm);

zz=get(handles.edit4,'string');
Amplitude=str2double(zz);

x11=get(handles.edit7,'string');
x10=str2double(x11);

x22=get(handles.edit8,'string');
x20=str2double(x22);

x33=get(handles.edit9,'string');
```

```
x30=str2double(x33);

x1=get(handles.edit10,'string');
x1dot=str2double(x1);

x2=get(handles.edit11,'string');
x2dot=str2double(x2);

x3=get(handles.edit12,'string');
x3dot=str2double(x3);


T=get(handles.edit13,'string');
Time=str2double(T)



dd=get(handles.edit14,'string');
d=str2double(dd)

hh=get(handles.edit15,'string');
h=str2double(hh)






w=frequancy;
k=sttifness;
m=mass;
z=Amplitude;
A=((-w^2)*z);
t=h:d:Time;


w1=0.44504*(k/m).^0.5
w2=1.2471*(k/m).^0.5
w3=1.8025*(k/m).^0.5

% Particular solution
qp1=(0.82805*m.^(0.5)*z*w.^2/w1)*(((-w+w1).^(-1)-(w+w1).^(-
1))*cos(w*t)+((1/(w-w1)+1/(w+w1))*cos(w1*t)));
qp2=(0.2365*m.^(0.5)*z*w.^2/w2)*(((-w+w2).^(-1)-(w+w2).^(-
1))*cos(w*t)+((1/(w-w2)+1/(w+w2))*cos(w2*t)));
```

```
qp3=(0.09065*m.^(0.5)*z*w.^2/w3)*(((-w+w3).^(-1)-(w+w3).^(-
1))*cos(w*t)+((1/(w-w3)+1/(w+w3))*cos(w3*t)));

x1=(1/m.^.5).*(.328*qp1+.737*qp2+.5911*qp3);
x2=(1/m.^.5).*(.5911*qp1+.328*qp2-.737*qp3);
x3=(1/m.^.5).*(.737*qp1-.5911*qp2+.328*qp3);
figure(2)

subplot(2,2,3);plot(t,x1);ylabel('Displacement(m)');xlabel('Time
(s)');grid
subplot(2,2,2);plot(t,x2);ylabel('Displacement(m)');xlabel('Time
(s)');grid
subplot(2,2,1);plot(t,x3);ylabel('Displacement(m)');xlabel('Time
(s)');grid




 figure(3)

 % Homogenous Solution

w1=0.44504*(k/m).^0.5
w2=1.2471*(k/m).^0.5
w3=1.8025*(k/m).^0.5
q1=(m.^0.5)*(0.328*x10*cos(w1*t)+0.737*x20*cos(w2*t)+0.5911*x30*cos(w
3*t))+(m.^0.5)*((0.328*x1dot)*sin(w1*t)/w1+(0.737*x2dot)*sin(w2*t)/w2
+(0.5911*x3dot)*sin(w3*t)/w3);
q2=(m.^0.5)*(0.5911*(x10)*cos(w1*t)+0.328*x20*cos(w2*t)-
0.737*x30*cos(w3*t))+(m.^0.5)*((0.5911*x1dot)*sin(w1*t)/w1+(0.328*x2d
ot)*sin(w2*t)/w2-(0.737*x3dot)*sin(w3*t)/w3);
q3=(m.^0.5)*(0.737*(x10)*cos(w1*t)-
0.5911*x20*cos(w2*t)+0.328*x30*cos(w3*t))+(m.^0.5)*((0.737*x1dot)*sin
(w1*t)/w1-(0.5911*x2dot)*sin(w2*t)/w2+(0.328*x3dot)*sin(w3*t)/w3);

xh1=(1/m.^.5).*(0.328*q1+0.737*q2+0.5911*q3);

xh2=(1/m.^.5).*(0.5911*q1+0.328*q2-0.737*q3);
xh3=(1/m.^.5).*(0.737*q1-0.5911*q2+0.328*q3);

subplot(3,1,1),plot(t,xh1) ;ylabel('Displacement(m)');xlabel('Time
(s)');grid
subplot(3,1,2),plot(t,xh2);ylabel('Displacement(m)');xlabel('Time
(s)');grid
subplot(3,1,3),plot(t,xh3);ylabel('Displacement(m)');xlabel('Time
(s)');grid
```

The mat lab software for the practical software is written below

```
function varargout = experimantal(varargin)
% EXPERIMANTAL M-file for experimantal.fig
%      EXPERIMANTAL, by itself, creates a new EXPERIMANTAL or raises the
existing
%      singleton*.
%
%      H = EXPERIMANTAL returns the handle to a new EXPERIMANTAL or the
handle to
%      the existing singleton*.
%
%      EXPERIMANTAL('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in EXPERIMANTAL.M with the given input
arguments.
%
%      EXPERIMANTAL('Property','Value',...) creates a new EXPERIMANTAL or
raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before experimantal_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to experimantal_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help experimantal

% Last Modified by GUIDE v2.5 01-Jan-1998 01:15:17

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...

                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @experimantal_OpeningFcn, ...
                   'gui_OutputFcn',  @experimantal_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin & isstr(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
```

```
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before experimantal is made visible.
function experimantal_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to experimantal (see VARARGIN)

% Choose default command line output for experimantal

handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes experimantal wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = experimantal_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a double


% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a double
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


tim=get(handles.edit1,'string');
time=str2double(tim);

spe=get(handles.edit1,'string');
speed=str2double(spe);



% DAQ initilization

dio1=digitalio('nidaq');
dio2=digitalio('nidaq');
dio3=digitalio('nidaq');
dio4=digitalio('nidaq');
dio5=digitalio('nidaq');
dio6=digitalio('nidaq');
dio7=digitalio('nidaq');
dio8=digitalio('parallel');


addline(dio1,0,'in');
addline(dio2,1,'in');
addline(dio3,2,'in');
addline(dio4,3,'in');
addline(dio5,4,'in');
addline(dio6,5,'in');
addline(dio7,6,'in');
addline(dio8,0,'out');


Ton=0;
Toff=0;


% counter resetting
```

```
counter1=0;
counter2=0;
counter3=0;


a1=0;
oldvalo(1)=0;

a2=0;
oldvalo(2)=0;

a3=0;
oldvalo(3)=0;

ci1=0;
ci2=0;
ci3=0;

% Time resetting

t1=0;
t2=0;
t3=0;


t11=0;
t22=0;
t33=0;

% conversion pulses to distance
D=3;
inc=D*pi/600;

vout = (speed*5)/400;

kk=cputime;

a11=getvalue(dio1);
a22=getvalue(dio3);
a33=getvalue(dio5);

Vout=5;
```

```
while (cputime-kk<time)

    t=cputime;


%*** PWM generation


Vin=5;
T=0.00004;

Ton=(Vout/Vin)*T


Toff=T-Ton

putvalue(dio8,[1]);
pause(Ton);
putvalue(dio8,[0]);
pause(Toff);

% *** End of PMW

% Speed reding and closed loop

   s=getvalue(d2);

      if (s==1)
         b=b+1;
         while(s==1&cputime-kk<time)
            s=getvalue(d2)
         end

      end

   g=((b*60)/time;

   volt = (g*400)/5;
   vnew=5-volt;
   Vout=vnew+Vout;

% optical encoder resding program

a1=getvalue(dio1);
```

```
dire1=getvalue(dio2);
a2=getvalue(dio3);
dire2=getvalue(dio4);
a3=getvalue(dio5);
dire3=getvalue(dio6);


if (a11~=a1)
    a11=a1;
 if(dire1==1)
counter1=(counter1)+inc;
else
    counter1=counter1-inc;
end
else
    counter1=counter1;
end
 vector1=[ci1 counter1];
ci1=vector1;

e=cputime-t;
t1=t1+e;
timvector1=[t11 t1];
t11=timvector1;

if (a22~=a2)
    a22=a2;
 if(dire2==1)
counter2=(counter2)+inc;
else
    counter2=counter2-inc;
end
else
    counter2=counter2;
end

 vector2=[ci2 counter2];
ci2=vector2;

e=cputime-t;
t2=t2+e;
timvector2=[t22 t2];
t22=timvector2;
```

```
if (a33~=a3)
    a33=a3;

 if(dire3==1)
counter3=(counter3)+inc;
else
    counter3=counter3-inc;
end
else

    counter3=counter3;
end
  vector3=[ci3 counter3];
ci3=vector3;

e=cputime-t;
t3=t3+e;
timvector3=[t33 t3];
t33=timvector3;

end

% Result ploting

subplot(3,1,1)
 plot(timvector1,vector1);ylabel('Displacement(mm)');xlabel('Time (s)');title(' X1
Response')
 subplot(3,1,2)
 plot(timvector2,vector2);ylabel('Displacement(mm)');xlabel('Time (s)');title(' X2
Response');
 subplot(3,1,3)
 plot(timvector3,vector3);ylabel('Displacement(mm)');xlabel('Time (s)');title(' X2
Response');

delete(dio1)
clear dio1
delete(dio2)
clear dio2
delete(dio3)
clear dio3
delete(dio4)
clear dio4
delete(dio5)
```

```
clear dio5
delete(dio6)
clear dio6
delete(dio7)
clear dio7
delete(dio8)
clear dio8
```

The Visual basic program is written below

```
' ********************************************************************
'
'  Example Program:
'    TIOquadEncoderPosMeasure.FRM
'
'  Description:
'    Counts the number of quadrature encoded digital pulses using a
'     general purpose counter 0 in a loop. Also accepts a Z-index pulse
'     so that the counter initializes automatically (for NI-TIO based
'     devices)
'
'  Example Category:
'    CTR
'
'  Example Task Types:
'    EVENTCNT, 1PT
'
'  List of key parameters:
'    ulGpctrNum, ulZIndexCount, ulInitCount
'
'    [Since variables are hardcoded, there is no guarantee that this
'     program will work for your setup.  This example is simply
'     presented as a code snippet of how you can use NI-DAQ functions
'     to perform a task.]
'
'  List of NI-DAQ Functions used in this example:
'    GPCTR_Control, NIDAQErrorHandler, GPCTR_Set_Application,
'     GPCTR_Change_Parameter, Line_Change_Attribute, GPCTR_Watch,
'     NIDAQYield
'
'    [NOTE: For further details on each NI-DAQ function, please refer
'     to the NI-DAQ On-Line Help (NIDAQPC.HLP).]
'
'  Pin Connection Information:
'    Connect your encoder channel A signal to the default source pin
'     (PFI 39), the channel B signal to the default auxiliary line (PFI
'     37), and the Z-index pulse signal to the default gate pin (PFI
'     38).  Also, connect the ground reference to the DIG GND pin.
'
'    [For further I/O connection details, please refer to your hardware
'     User Manual.]
```

```
'
'    [For further details on how to run this example, please refer to
'     the NI-DAQ Examples On-Line Help (NIDAQEx.HLP).]
'
' ************************************************************************
Option Explicit
Option Base 0
'
' Constant for PrintText
'
Const LEN_PRINTTEXT = 4096


'
*********************************************************************
***
' SUBROUTINE:  PrintText
' DESCRIPTION: PrintText to desired TextBox (upto 4096 characters)
' INPUTS:      txtBox - TextBox to print on
'              strText - Text to print
'
*********************************************************************
***
Sub PrintText(txtBox As TextBox, strText As String)

    txtBox.Text = Right$(txtBox.Text + strText$ + Chr$(13) + Chr$(10),
LEN_PRINTTEXT)

    txtBox.SelStart = Len(CStr(txtBox.Text))

    DoEvents

End Sub


'
*********************************************************************
***
' SUBROUTINE:  cmdExit_Click
' DESCRIPTION: Clean up and exit
'
*********************************************************************
***
Sub cmdExit_Click()
```

```
    End

End Sub

'
*********************************************************************
***
' SUBROUTINE:  Form_Load
' DESCRIPTION: Gets automatically called at startup
'
*********************************************************************
***
Sub Form_Load()


End Sub


'
*********************************************************************
***
' SUBROUTINE:  cmdDoOperation_Click
' DESCRIPTION: The main NI-DAQ operations are here
'
*********************************************************************
***
Sub cmdDoOperation_Click()

    '
    '  Local Variable Declarations:


    Dim iStatus As Integer
    Dim iRetVal As Integer
    Dim iDevice As Integer
    Dim ulGpctrNum As Long
    Dim ulCount As Long
    Dim ulZIndexCount As Long
    Dim ulInitCount As Long
    Dim ulTCReached As Long
    Dim iLoopCount As Long
    Dim iIgnoreWarning As Integer
    Dim iYieldON As Integer
    iDevice% = 1
    ulGpctrNum& = ND_COUNTER_0
    ulTCReached& = ND_NO
    iLoopCount& = 500
    iYieldON% = 1

    ' Temporarily disable buttons for protection from 'DoEvents'
    cmdDoOperation.Enabled = False
    cmdExit.Enabled = False

    iStatus% = GPCTR_Control(iDevice%, ulGpctrNum&, ND_RESET)

    iRetVal% = NIDAQErrorHandler(iStatus%, "GPCTR_Control/RESET",
iIgnoreWarning%)

'     Setup for a position measurement application.

'     NOTE: If you want to measure speed at the same time, you must
'      timestamp each reading at the exact moment you take a quadrature
'      encoded position measurement. By determining the position change
'      from the previous measurement and the time difference since the
'      previous measurement, you can calculate the speed.

    iStatus% = GPCTR_Set_Application(iDevice%, ulGpctrNum&,
ND_POSITION_MSR)

    iRetVal% = NIDAQErrorHandler(iStatus%, "GPCTR_Set_Application",
iIgnoreWarning%)

'     Setup the encoder type for Quadrature Encoder (X1) measurement.
'      You can change this to X2 or X4 if you wish.

    Call GPCTR_Change_Parameter(iDevice%, ulGpctrNum&,
ND_ENCODER_TYPE, ND_QUADRATURE_ENCODER_X1)

    iRetVal% = NIDAQErrorHandler(iStatus%,
"GPCTR_Change_Parameter/QUADRATURE_ENCODER_X1", iIgnoreWarning%)

'     Activate a Z-index pulse to reset the counter to an initial value,
'      specified by 'ulCount' later.

    Call GPCTR_Change_Parameter(iDevice%, ulGpctrNum&,
ND_Z_INDEX_ACTIVE, ND_YES)
```

```
   iRetVal% = NIDAQErrorHandler(iStatus%,
"GPCTR_Change_Parameter/Z_INDEX_PULSE", iIgnoreWarning%)

'    Specify the value that gets loaded when a Z-index pulse arrives.

   iStatus% = GPCTR_Change_Parameter(iDevice%, ulGpctrNum&,
ND_Z_INDEX_VALUE, ulZIndexCount&)

   iRetVal% = NIDAQErrorHandler(iStatus%, "GPCTR_Change_Parameter/Z-
INDEX_COUNT", iIgnoreWarning%)

'    Load initial count.

   iStatus% = GPCTR_Change_Parameter(iDevice%, ulGpctrNum&,
ND_INITIAL_COUNT, ulInitCount&)

   iRetVal% = NIDAQErrorHandler(iStatus%,
"GPCTR_Change_Parameter/INITCOUNT", iIgnoreWarning%)

'    Signals from quadrature encoders often have noise and glitches
'     that result in measurement errors. Setup 5 usec filtering on each
'     input from the quadrature encoder.

'    Setup filter for Channel A.

   iStatus% = Line_Change_Attribute(iDevice%, ND_PFI_39, ND_LINE_FILTER,
ND_5_MICROSECONDS)

   iRetVal% = NIDAQErrorHandler(iStatus%,
"Line_Change_Attribute/ND_PFI_39", iIgnoreWarning%)

'    Setup filter for Channel B.

   iStatus% = Line_Change_Attribute(iDevice%, ND_PFI_37, ND_LINE_FILTER,
ND_5_MICROSECONDS)

   iRetVal% = NIDAQErrorHandler(iStatus%,
"Line_Change_Attribute/ND_PFI_37", iIgnoreWarning%)

'    Setup filter for Z-index Channel.

   iStatus% = Line_Change_Attribute(iDevice%, ND_PFI_38, ND_LINE_FILTER,
ND_5_MICROSECONDS)
```

```
   iRetVal% = NIDAQErrorHandler(iStatus%,
"Line_Change_Attribute/ND_PFI_38", iIgnoreWarning%)

   'Call PrintText(txtStatusBox, "Connect the encoder Channel A signal to PFI 39. ")

   'Call PrintText(txtStatusBox, "Connect the encoder Channel B signal to PFI 37. ")

   'Call PrintText(txtStatusBox, "Connect the Z-index signal to PFI 38. ")

   iStatus% = GPCTR_Control(iDevice%, ulGpctrNum&, ND_PROGRAM)

   iRetVal% = NIDAQErrorHandler(iStatus%, "GPCTR_Control/PROGRAM",
iIgnoreWarning%)

'    Loop 100 times.

   Do

      iStatus% = GPCTR_Watch(iDevice%, ulGpctrNum&, ND_COUNT, ulCount&)

      iRetVal% = NIDAQErrorHandler(iStatus%, "GPCTR_Watch/COUNT",
iIgnoreWarning%)

      If (iStatus% = 0) Then

         Call PrintText(txtStatusBox, " " + Trim$(Str$(ulCount&)))

      End If

      iLoopCount = iLoopCount - 1

      DoEvents

   Loop While ((iLoopCount& > 0) And (iStatus% = 0))

   iRetVal% = NIDAQErrorHandler(iStatus%, "GPCTR_Watch", iIgnoreWarning%)

'    CLEANUP - Don't check for errors on purpose.

'    Clear filter settings.

   iStatus% = Line_Change_Attribute(iDevice%, ND_PFI_39, ND_LINE_FILTER,
ND_NONE)
```

```
    iStatus% = Line_Change_Attribute(iDevice%, ND_PFI_37, ND_LINE_FILTER,
ND_NONE)

    iStatus% = Line_Change_Attribute(iDevice%, ND_PFI_38, ND_LINE_FILTER,
ND_NONE)

'     Reset GPCTR.

    iStatus% = GPCTR_Control(iDevice%, ulGpctrNum&, ND_RESET)

    Call PrintText(txtStatusBox, "Done with quadrature encoded position
measurement! ")

    ' Re-enable buttons
    cmdDoOperation.Enabled = True
    cmdExit.Enabled = True

End Sub
```

Appendix B

1-The Data Acquisition Card

Counter/Timer Specifications

Counter/Timer Specifications Data Acquisition and Signal Conditioning

Specifications

Level Minimum Maximum

Input low voltage -0.3 V 0.8 V

Input high voltage 2.0 V 5.25 V

Output low voltage (Iout = 4 mA) – 0.4 V

Output high voltage (Iout = 4 mA) 2.4 V –

Family without Prescaling With Prescaling

NI 6601 20 MHz 60 MHz

NI 6602 80 MHz 125 MHz

NI 6608 80 MHz 125 MHz

Family Frequency to Measure Min/Max Frequency to Generate

NI 6601 20 MHz 10 MHz

NI 6602 80 MHz 40 MHz

NI 6608 80 MHz 40 MHz

Level Minimum Maximum

Input low voltage -0.3 V 0.8 V

Input high voltage 2.0 V 5.25 V

Output low voltage (Iout = 4 mA) – 0.4 V

Output high voltage (Iout = 4 mA) 2.4 V –

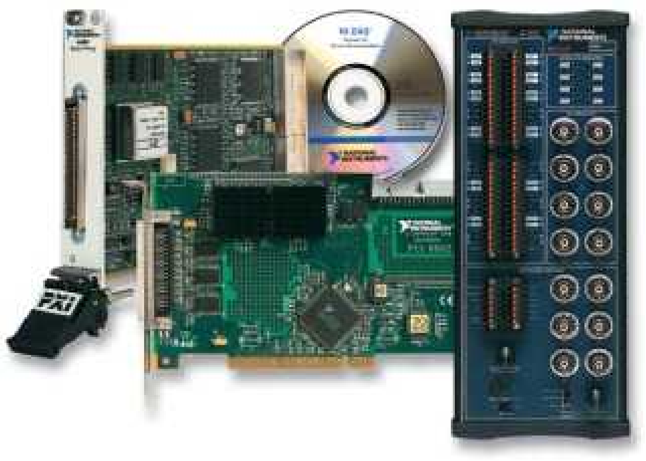
Device +5 VDC (±5%)* Power Available at I/O Connector

NI 6601 0.4 to 0.75 A +4.65 to +5.25 VDC, 1 A

NI 6602 0.5 to 1.5 A +4.65 to +5.25 VDC, 1 A

NI 6608 1 to 2.5 A +4.65 to +5.25 VDC, 1 A

*Excludes power consumed through I/O connector

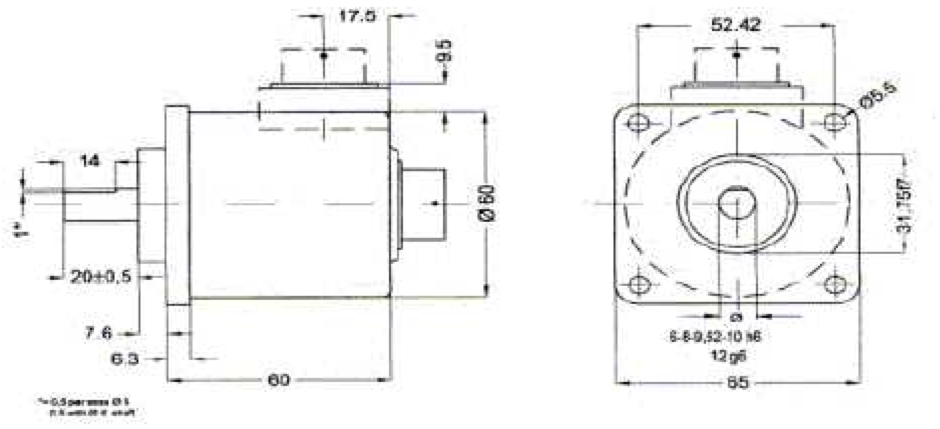


2- The Digital Encoder

Digital Encoder picture

Mechanical Specifications	
Dimensions	See the next figure
Shaft Loading (axial and radial)	100N max
Shaft Rotational speed	6000 rpm max
Weight	Almost 0.4 Kg
Number of pulse per revolution	600rpm

Electrical Specifications	
Power Supply	+5V/+30V
Output Current	40mA max
Output Frequency	60kHz max
Power Consumption	1.2W



Engineering drawing of Digital Encoder

3- Dc Motor

Electric DC motor without gear assembly **(**DPD · 24 V 104 W**)**

Motor picture

Part number 0 130 111 130

Electrical and Mechanical Specifications	
Nominal voltage $\boldsymbol{U}_N$	24 V
Nominal voltage $\boldsymbol{P}_N$	104 W
Nominal current $\boldsymbol{I}_N$	7 A
Nominal speed $\boldsymbol{n}_N$	4950 min^{-1}
Continuous torque $\boldsymbol{M}_D$	20 Ncm
Breakaway torque $\boldsymbol{M}_A$	170 Ncm
Direction of rotation	R
Type of duty	S 1
Degree of protection	IP 10
Weight	1,1 kg
Part number	**0 130 111 130**

Connection Diagram:

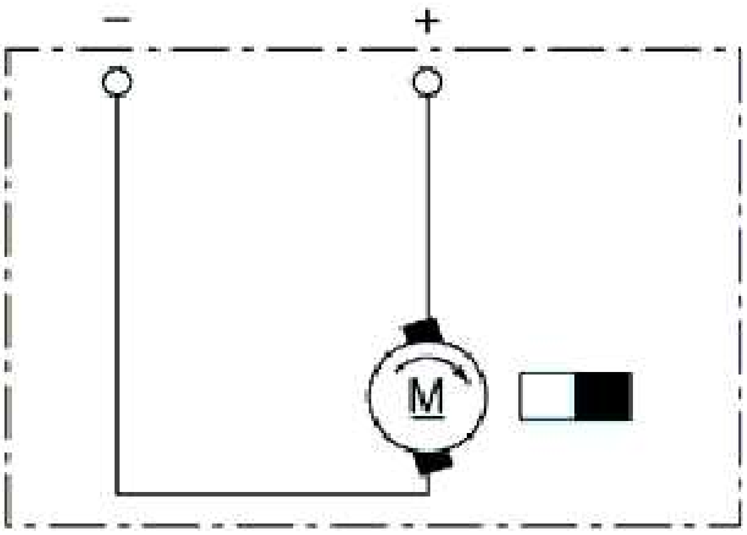
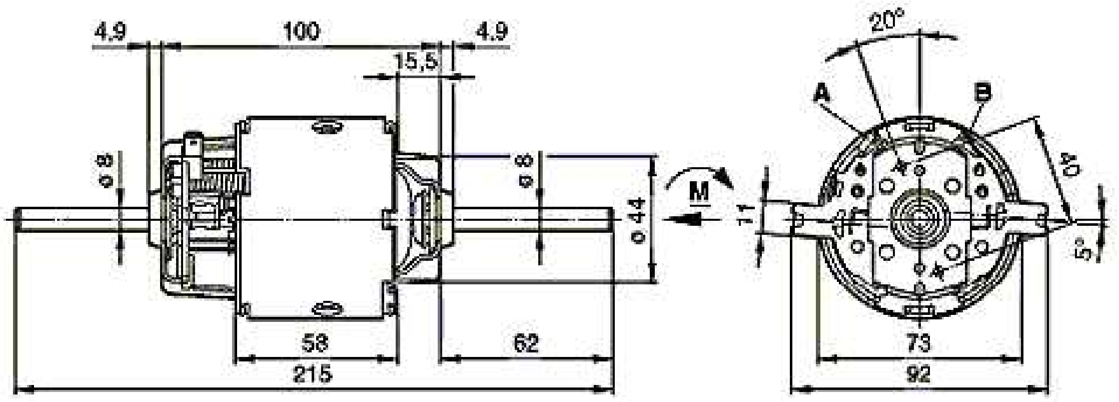


Figure ().: Connection diagram Part number 0 130 111 130

Engineering Drawing:

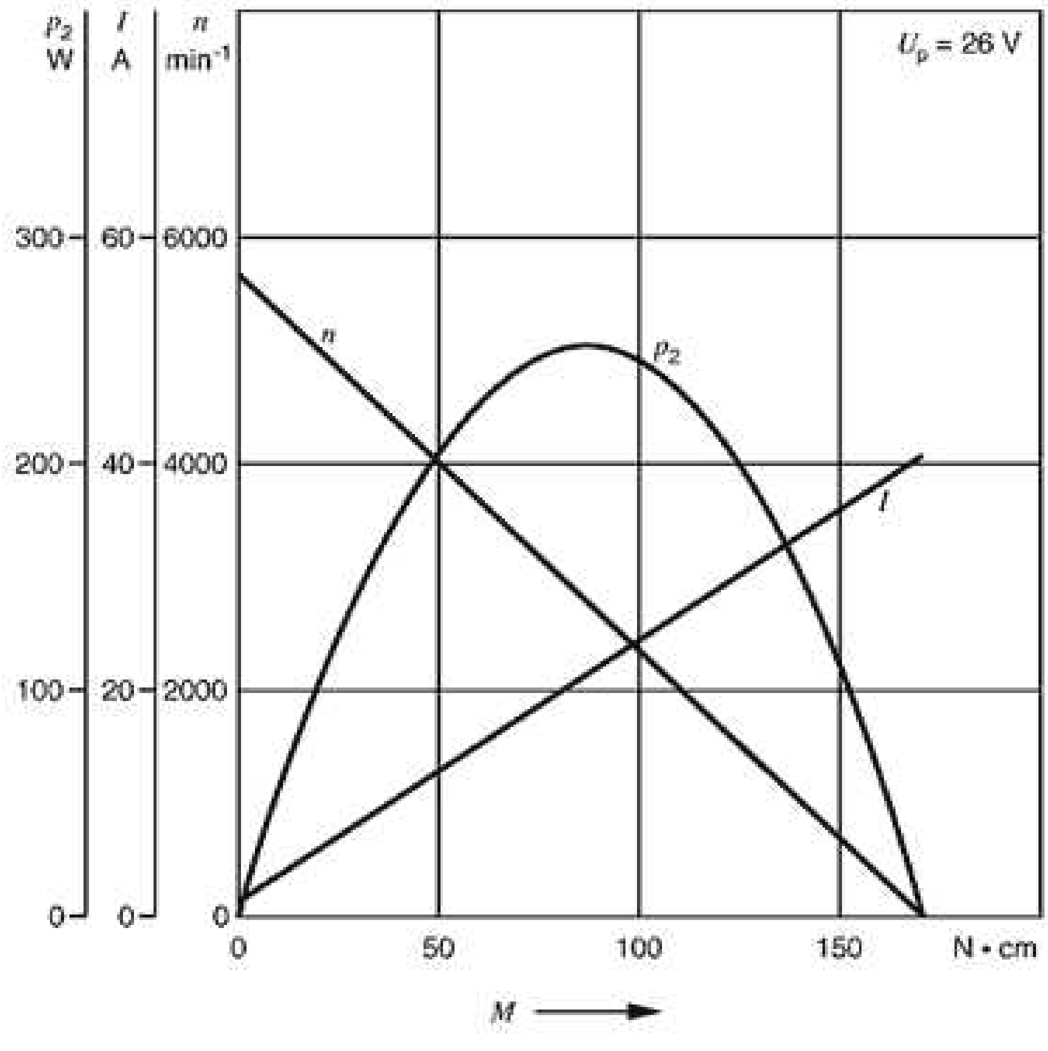


Engineering drawing for Part number 0 130 111 130

A:Blade receptacel for Blade terminal 6,3 x 0,8

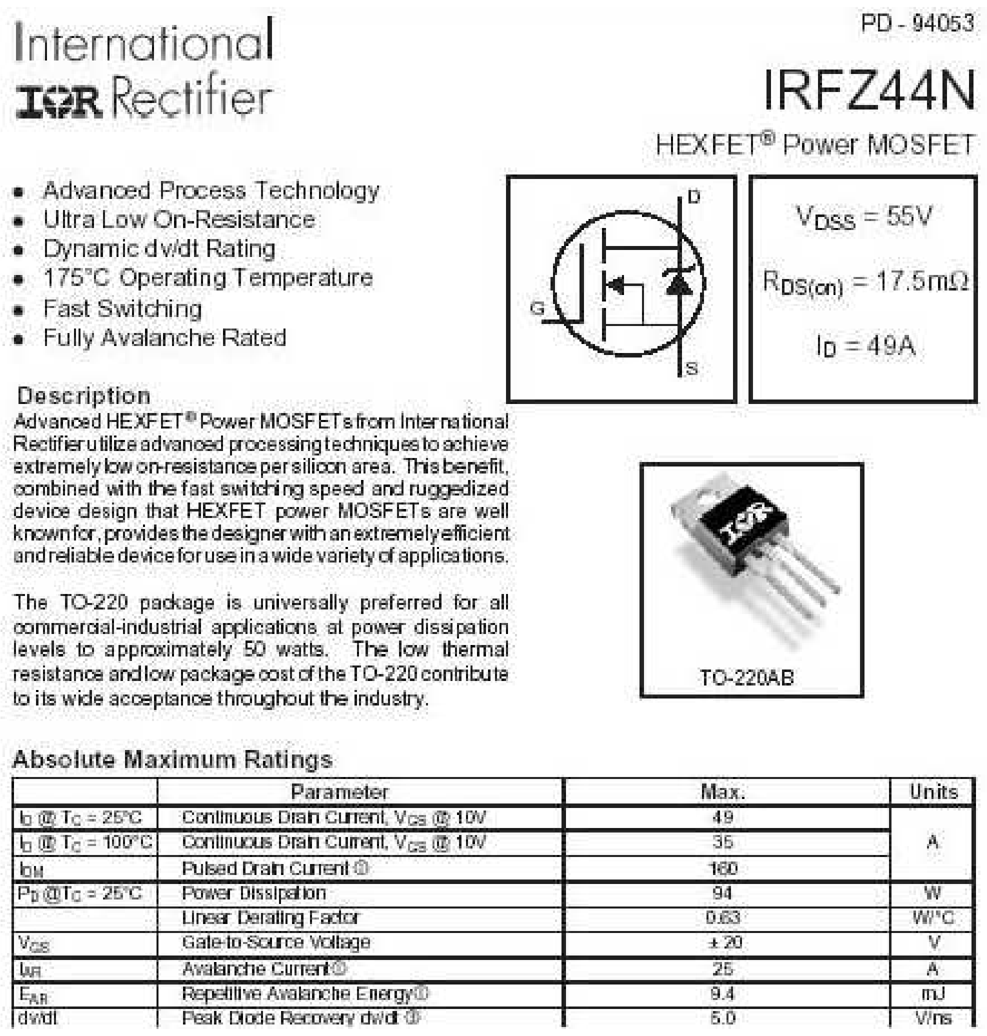
B:Blade terminal 6,3 x 0,8

Characteristic curve:



Characteristic curve Part number 0 130 111 130

4- Power Transistor (IRFZ44)



International
IOR Rectifier

PD - 94053

IRFZ44N

HEXFET® Power MOSFET

- Advanced Process Technology
- Ultra Low On-Resistance
- Dynamic dv/dt Rating
- 175°C Operating Temperature
- Fast Switching
- Fully Avalanche Rated

$V_{DSS} = 55V$

$R_{DS(on)} = 17.5m\Omega$

$I_D = 49A$

Description

Advanced HEXFET® Power MOSFETs from International Rectifier utilize advanced processing techniques to achieve extremely low on-resistance per silicon area. This benefit, combined with the fast switching speed and ruggedized device design that HEXFET power MOSFETs are well known for, provides the designer with an extremely efficient and reliable device for use in a wide variety of applications.

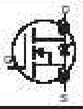
The TO-220 package is universally preferred for all commercial-industrial applications at power dissipation levels to approximately 50 watts. The low thermal resistance and low package cost of the TO-220 contribute to its wide acceptance throughout the industry.

TO-220AB

Absolute Maximum Ratings

	Parameter	Max.	Units
I_D @ T_C = 25°C	Continuous Drain Current, V_{GS} @ 10V	49	A
I_D @ T_C = 100°C	Continuous Drain Current, V_{GS} @ 10V	35	
I_{DM}	Pulsed Drain Current ①	160	
P_D @T_C = 25°C	Power Dissipation	94	W
	Linear Derating Factor	0.63	W/°C
V_{GS}	Gate-to-Source Voltage	± 20	V
I_{AR}	Avalanche Current①	25	A
E_{AR}	Repetitive Avalanche Energy①	9.4	mJ
dv/dt	Peak Diode Recovery dv/dt ③	5.0	V/ns

Electrical Characteristics @ T_J = 25°C (unless otherwise specified)

	Parameter	Min.	Typ.	Max.	Units	Conditions
$V_{(BR)DSS}$	Drain-to-Source Breakdown Voltage	55	–––	–––	V	V_{GS} = 0V, I_D = 250μA
$\Delta V_{(BR)DSS}/\Delta T_J$	Breakdown Voltage Temp. Coefficient	–––	0.058	–––	V/°C	Reference to 25°C, I_D = 1mA
$R_{DS(on)}$	Static Drain-to-Source On-Resistance	–––	–––	17.5	mΩ	V_{GS} = 10V, I_D = 25A [4]
$V_{GS(th)}$	Gate Threshold Voltage	2.0	–––	4.0	V	V_{DS} = V_{GS}, I_D = 250μA
g_{fs}	Forward Transconductance	19	–––	–––	S	V_{DS} = 25V, I_D = 25A[4]
I_{DSS}	Drain-to-Source Leakage Current	–––	–––	25	μA	V_{DS} = 55V, V_{GS} = 0V
		–––	–––	250		V_{DS} = 44V, V_{GS} = 0V, T_J = 150°C
I_{GSS}	Gate-to-Source Forward Leakage	–––	–––	100	nA	V_{GS} = 20V
	Gate-to-Source Reverse Leakage	–––	–––	-100		V_{GS} = -20V
Q_g	Total Gate Charge	–––	–––	63	nC	I_D = 25A
Q_{gs}	Gate-to-Source Charge	–––	–––	14		V_{DS} = 44V
Q_{gd}	Gate-to-Drain ("Miller") Charge	–––	–––	23		V_{GS} = 10V, *See Fig. 6 and 13*
$t_{d(on)}$	Turn-On Delay Time	–––	12	–––	ns	V_{DD} = 28V
t_r	Rise Time	–––	60	–––		I_D = 25A
$t_{d(off)}$	Turn-Off Delay Time	–––	44	–––		R_G = 12Ω
t_f	Fall Time	–––	45	–––		V_{GS} = 10V, *See Fig. 10* [4]
L_D	Internal Drain Inductance	–––	4.5	–––	nH	Between lead, 6mm (0.25in.) from package and center of die contact 
L_S	Internal Source Inductance	–––	7.5	–––		
C_{iss}	Input Capacitance	–––	1470	–––	pF	V_{GS} = 0V
C_{oss}	Output Capacitance	–––	360	–––		V_{DS} = 25V
C_{rss}	Reverse Transfer Capacitance	–––	88	–––		*ƒ* = 1.0MHz, *See Fig. 5*
E_{AS}	Single Pulse Avalanche Energy[2]	–––	530[5]	150[6]	mJ	I_{AS} = 25A, L = 0.47mH