UNIVERSITY OF CINCINNATI

Date: May 5, 2006

I, Amit Sharma,

hereby submit this work as part of the requirements for the degree of:

Master of Science

in:

Mechanical Engineering

It is entitled:

Dynamic Characteristics of a Shaft-Universal Joint System

This work and its defense approved by:

Chair: Dr. Teik C. Lim

Dr. David Thompson

Dr. Ronal Huston

Dynamic Characteristics of a Shaft-Universal Joint System

A Thesis submitted to the

Division of Research and Advanced Studies

of the University of Cincinnati

in partial fulfillment of the

requirements for the degree of

MASTER OF SCIENCE

in the Department of Mechanical, Industrial and Nuclear Engineering

of the College of Engineering

2006

By

AMIT SHARMA

Bachelor of Engineering (B.E.)

Regional Engineering College, Bhopal, India, 2001

Committee Chair: Dr. Teik C. Lim

ABSTRACT

An analytical model for the dynamical analysis of a non-rotating shaft-universal joint model using Multibody Dynamics approach [14] along with Finite Element Techniques is developed. The shaft-universal joint model is presented as a multibody system of elastic bodies discretized into three dimensional elastic beam elements. The equations of motions of the beam elements are derived using the Lagrange's equation and the principal of virtual work and for the complete system are developed by applying constraint equations in the multi-body model. These equations are then solved to calculate the dynamic stiffness matrix of the universal joint system that is then employed to the finite element models of the shafts. Modal analysis is then performed to calculate the natural frequencies and the mode shapes. The effect of the geometrical parameters on the modal parameters of the system is presented. Results of some of the particular examples cases are compared and validated using general purpose finite element software.

ACKNOWLEDGEMENT

I would like to thank Dr. Teik C. Lim, my advisor, for his time discussing this topic as well as my academic interests. I would also like to thank the committee members Dr. David Thompson and Dr. Ronald Huston and for their time and valuable suggestions.

I would like to thank my family for providing me the means and instilling the desire to pursue an education.

Lastly, I thank my friends for their support and encouragement in the completion of this work.

TABLE OF CONTENTS

Abstract i

Acknowledgements iii

1 INTRODUCTION 7

1.1 Literature review……..……...7

1.2 Structure of the thesis.……....9

2 KINEMATICS OF UNIVERSAL JOINT 11

2.1 Joint description……..……..11

2.2 Non-parallel shaft system.……12

3 DYNAMICS OF UNIVERSAL JOINT 18

3.1 Formulation of the problem……..18

3.2 Point displacement of flexible body……..20

3.3 Equation of motion……..……..25

3.3.1 Calculation for left part of equation of motion.....26

3.3.2 Calculation for right part of equation of motion.....28

3.4 Constraint equation……..…….30

3.5 Assembly of the system.....33

4 NUMERICAL SOLUTION AND VALIDATION 34

4.1 Solution of Equation.....……..34

4.2 Numerical Solution.....……..……..35

4.3 Graphical user interface.....35

4.4 Example……..……..37

4.5 Validation……..……43

5 CONCLUSIONS AND SCOPE OF FUTURE WORK 46

5.1 Conclusion……..……...46

REFERENCES 48

APPENDIX A 51

APPENDIX B 54

B.1 MATLAB script for non-parallel shafts.……...……...54

B.2 MATLAB script for dynamic analysis ……....57

B.2.1 Main……...57

B.2.2 Mass beam……..……...82

B.2.3 Stiffness beam……..…….…85

B.2.4 Mass matrix cross pin...…….…87

B.2.5 Stiffness matrix cross pin……...…90

B.2.6 Mass matrix input shaft...……..92

B.2.7 Stiffness matrix input shaft……....95

B.2.8 Mass matrix output shaft...…….97

B.2.9 Stiffness matrix output shaft……..100

B.2.10 Transformation_1 ...…….……102

B.2.11 Transformation_2……...…….102

B.2.10 Transformation_3 ...…….……102

B.2.11 Transformation_4……..……..102

B.2.11 Graphical User Interface……....103

LIST OF FIGURES

Figure 2.1 Geometry of the shaft-universal joint system inclined in XZ plane.....13

Figure 2.2 Cross pin's position vector and coordinate systems for non-parallel shaft-universal joint system ……..…….....……13

Figure 2.3 Driven shaft rotation for 180 degrees rotation of driving shaft for angle of inclination 0 (___), 20 (----), 27 (-.-.-.), 33 (___), 40 (.....) degrees.....16

Figure 2.4 Velocity variation of the driven shaft for 360 degree rotation of the driving shaft for the angle of inclination of 0 (___), 20 (----), 27 (-.-.-.), 33 (___), 40 (.....) degrees.....……...16

Figure 3.1 Shaft-Universal joint system model with degrees of freedom $u = \{u_x, u_y, u_z, \phi_x, \phi_y, \phi_z\}$ ……19

Figure 3.2 Primary bodies of the shaft-universal joint system XYZ – Global coordinate system, x_i, y_i, z_i - Body coordinate system for i^{th} body..…19

Figure 3.3 Position vector of an arbitrary point in a beam with XYZ – Global coordinate system, x_i, y_i, z_i - Body coordinate system, x_{ij}, y_{ij}, z_{ij} - Intermediate coordinate system, $x_{ije}, y_{ije}, z_{ije}$ - Element coordinate system and i, j are the body and element numbers respectively.....21

Figure 3.4 Position vector of an arbitrary point in deformed and un-deformed state for i^{th} body …….....……...21

Figure 3.5 Non-parallel shaft arrangement .....……..……30

Figure 3.6 Driving shaft and yoke position vectors.....31

Figure 3.7 Cross pin rotational degree of freedom.....32

Figure 4.1 Graphical user interface…….……..……..…..….36

Figure 4.2 Variation of natural frequencies – Case I vs. Case II……...…….....39

Figure 4.3 Variation of natural frequencies – Case I vs. Case III...…...……....40

Figure 4.4 Frequency response function (X-direction), case IV……....41

Figure 4.5 Frequency response function (Y-direction), case IV……....42

Figure 4.6 Frequency response function (Z-direction), case IV…….....42

Figure 4.7 Frequency response function (X-direction), case II..……....44

Figure 4.8 Frequency response function (Y-direction), case II..……....44

Figure 4.9 Frequency response function (Y-direction), case II..……....45

LIST OF TABLES

Table 4.1 Material geometrical parameters.....37

Table 4.2 Different cases of geometrical variation.....38

Table 4.3 Natural frequencies for various cases.....38

Table 4.4 Comparisons of the natural frequencies calculated from the analytical method and ANSYS for the particular case of universal joint system….43

NOMENCLATURE

ω :	Angular velocity
XYZ :	Global coordinate system
xyz	Body coordinate system
$X_i^{ij} Y_i^{ij} Z_i^{ij}$	Element coordinate system
$X^{ij} Y^{ij} Z^{ij}$	Intermediate coordinate system
i, j, k :	Unit vectors along the coordinate system
ψ , θ :	Angle of rotation of the driving shaft and driven shaft respectively
α, ρ :	Angle of Inclination the shafts
$P(\psi)$, $Q(\theta)$:	Position vector of the cross-pin branches
[A]	Transformation matrix
[S]	Shape function matrix
{e}	Nodal coordinate vector in element/intermediate coordinate system
{q}	Nodal coordinate vector in body/global coordinate system
{w}	Displacement field vector (element/intermediate coordinate system)
$\bar{\mathbf{c}}^{ij}$	Transformation matrix (Body to intermediate coordinate system)
$\mathbf{c}^{ij}$	Transformation matrix (Intermediate to body coordinate system)
T	Kinetic energy
E	Modulus of elasticity

CHAPTER 1

INTRODUCTION

1.1 LITERATURE REVIEW

The universal joint has a wide application in rotary machines and automobile industries. The dynamical properties of universal joint such as Cardan joint or Hooke's joint, that transmits rotational motion and torque load between two non-parallel shafts are still relatively unknown. This lack of understanding makes the design of the shaft-universal joint system for the purpose of tuning vibration response very difficult. Due to the need for higher rotational speed and greater precision in newer drives, it becomes more important to be able to tune the design of the shaft-universal joint system to avoid resonances and other forced vibration issues. Much of the existing work focus on computing the mean reaction forces torques in the universal joint. A fairly thorough treatment of universal joint kinematics and design for the mean forces and torques is given in the design handbook by Wagner [1]. Yang [2] presented a universal joint force and torque analysis for a four bar spherical mechanism. The analysis by Rosenberg [3] in 1958 investigated the bending stability of a shaft disk system connected to a pair of joints at both ends which is perhaps one of the earliest studies of shaft-universal joint dynamics. That work showed the influence of joint angle on the critical speed of the driven shaft. Freudenstein's team also conducted a series universal joint system. The work specific to the internal force and torque calculation for the case of a stationary, rigid, frictionless universal joint was accomplished by Fischer and Freudenstein [4]. That study also investigated effect of manufacturing tolerances in the universal joint system. In an

extension of this work, Freudenstein and Chen [5] analyzed the dynamic forces and moments in the universal joint. Finally Freudenstein and Macey [6] proposed a model for inertia torque calculation of a rigid Hooke joint model.

Studies on the lateral and torsional behaviors of the rotating shaft driven by a universal joint were performed by Ota and Kato [7-9]. Those analyses assumed rigid driving shaft and universal joint but flexible driven shaft. The later of their three papers included viscous and coulomb's friction parameters. A paper dealing with the effects of the joint angle and friction on the intermediate shaft design between two universal joints was presented by Sheu, Chieng and Lee [10]. They reported that the intermediate shaft critical speeds are affected by the axial torques that in turn depends on the viscous friction and joint angle.

More recently Mazzei, Argento and Scott [11] did a dynamic stability of universal joint coupled shafts. The effects of distributed shaft flexibility and inertia on the parametric resonance along with non-zero joint angle and system nonlinearities were presented. Furthermore Biancolini et. al. [12-13] investigated the dynamics, mechanical efficiency and fatigue analysis of cardan joints.

No significant paper on the modal behavior of a shaft-universal joint system that is critical to the understanding and design of this class of mechanical structure was seen in the open literature. The purpose of this paper is to address part of this gap.

1.2 STRUCTURE OF THE THESIS

Chapter one gives an introduction of the thesis and the background of the research work done in the field of the universal joint dynamics.

Chapter two describes the kinematics of the universal joint using a rigid body model. A model similar to the one developed by Sheu, Chieng and Lee [10] is presented in this section for the kinematics study of the joint. This section presents a rigid non-parallel shaft-universal joint model. The effect of joint angle on the angle of rotation and angular velocity of the driving and driven shaft is explained in this chapter. Comparison study of effect of joint angle for various cases is presented with the graphs.

Chapter three describes the dynamics of the universal joint using finite element approach. This section explains the application of flexible multibody dynamics approach developed by Shabana [14] in shaft-universal joint model. The dynamical equation of motion of the flexible shaft-universal joint system are derived and solved to yield the dynamic stiffness matrix. The result is then incorporated into a finite element model of the complete shaft-universal joint drive to predict overall vibration response of the system.

Chapter four represents numerical solution and an example of particular case of universal joint modeling and solution using MATLAB and validation of results using ANSYS. The effect of variation of the geometrical parameters and material properties on the behavior of the joint system is demonstrated. The various frequency response

functions for the particular case are shown in this section along with the natural frequency comparison table.

Chapter five gives a conclusion of the thesis. The shape function, mass matrix and stiffness matrix for the beam element are presented in the appendix A. Appendix B shows the MATLAB scripts for the complete formulation.

CHAPTER 2
KINEMATICS OF THE UNIVERSAL JOINT

2.1 JOINT DESCRIPTION

This section describes the kinematical relationship between the angle of rotation and angular velocity of the driving and driven shafts for a general universal joint structure consists of cross pin and yoke. Because of the complicated geometry of the universal joint the angle of rotation and the angular velocity of the output shaft are different from that of the input shaft although the angular velocity of the input shaft is constant. Normally the universal joint is used for small misalignment angles, but in the case of inclination in one plane only, it can be used up to 40 degrees without adverse effect. The relationship of the angle of rotation and angular velocity of non-parallel driving and driven shaft is presented below.

The rotation angle and angular velocity of the driven shaft depend on the geometrical angle between the driving and driven shafts. Normally the universal joint is used for small misalignment angles, but in the case of inclination in one plane only, it can be used up to 40 degrees without adverse effect. The relationship of the angle of rotation and angular velocity of non-parallel driving and driven shaft is presented below. A model similar to the one developed by Sheu, Chieng and Lee [10] is presented in this section for the kinematics study of the joint.

Expressions showing the relationships for the two cases are derived along with the MATLAB code. The angles of inclinations can be changed in the script to compare the effect of changing the angle between the driving and the driven shaft. A comparison study is represented in this section.

2.2 **NON-PARALLEL SHAFT SYSTEM**

Figure 2.1 shows the inclination of the output shaft with respect to the input shaft. Both shaft are connected by universal joint and inclined in X-Z plane at an angle α, where, XYZ is the coordinate system that defines the orientation of the driven shaft.

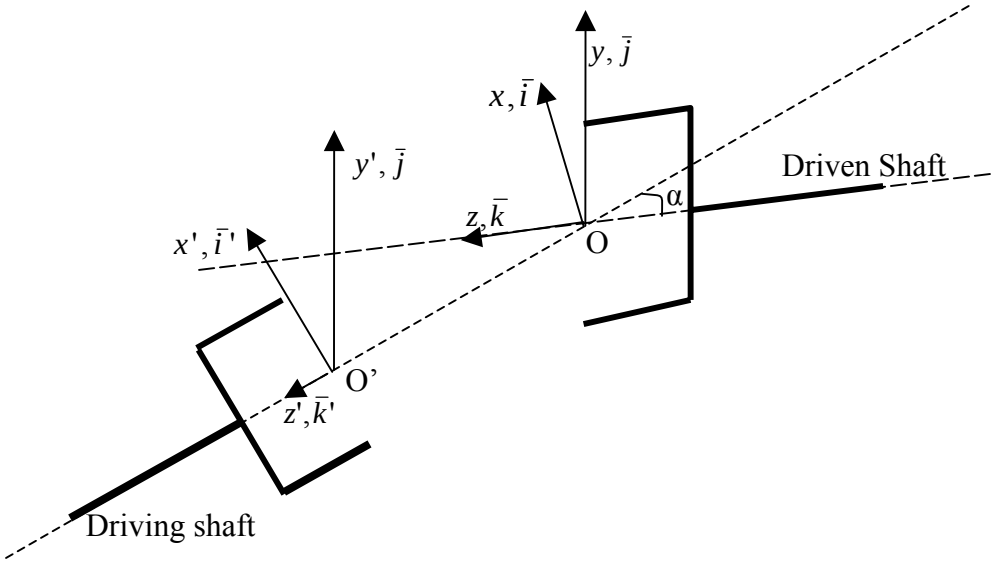

Figure 2.1. Geometry of the shaft-universal joint system inclined in XZ plane.

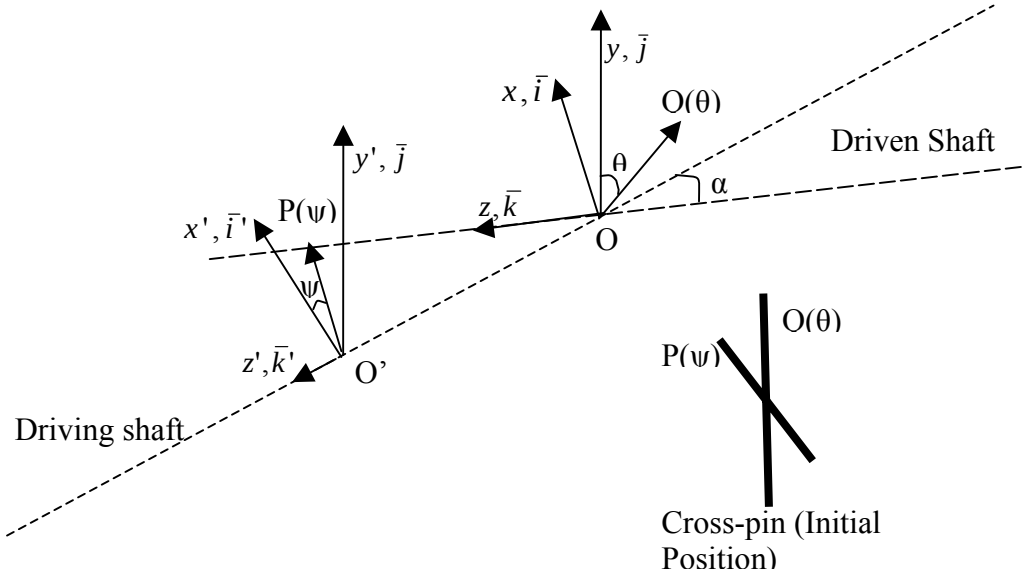

Figure 2.2. Cross pin's position vector and coordinate systems for non-parallel shaft-universal joint system

As per the model developed in [10], the coordinate system X'Y'Z' defines the position of the driving shaft. The driving shaft is assumed to rotate at a constant angular velocity ω relative to $X'Y'Z'$. The triads of unit vectors i', j', k' and i, j, k are associated with $X'Y'Z'$ and XYZ coordinate systems respectively. The position unit vector of the input shaft cross pin branch $P(\psi)$ at time $t = 0$ is assumed to be the unit vector of the X' axis, i.e. $P(\psi) = i'$. The angle between unit vectors i and i' is α in the XZ plane. Initially the origins at O' and O are co-incident and the unit vectors j and j' along Y and Y' axes respectively are also co-incident. The driven shaft cross pin branch is orientated along the Y axis at time $t = 0$, i.e. the position unit vector of the pin is $Q(\theta) = j$. The orientation of the position unit vector $P(\psi)$ and $Q(\theta)$ are shown in Figure 2.2. The system is rigid and so there is no relative shaft motion in translational direction which implies that the cross-pin branches will remain perpendicular to each other. At a given time t>0, the driving shaft has rotated by an angle $\psi = \omega t$. The corresponding position vectors of the cross pin branches can be written as [10],

$$P(\psi) = (\cos\psi \cos\alpha)i + (\sin\psi)\, j - (\cos\psi \sin\alpha)k \tag{1a}$$

$$Q(\theta) = -(\sin\theta)i + (\cos\theta)\, j \tag{1b}$$

Since two branches of the cross pins are always perpendicular, the dot product of $P(\psi)$ and $Q(\theta)$ is zero i.e. $P(\psi).Q(\theta) = 0$. Substitution of the position vectors into the orthogonality relation yields,

$$\cos\alpha \tan\theta = \tan\psi \tag{2}$$

which defines the relationship between the rotations of the two shafts. To obtain the relationship between the angular velocities, differentiate Equation (2) with respect to time once, i.e.

$$\frac{d}{dt}(P(\psi))Q(\theta) + P(\psi)\frac{d}{dt}(Q(\theta)) = 0\,. \tag{3a}$$

$$\frac{d}{dt}(\theta) = [\{\sin\theta\sin\psi\cos\alpha + \cos\psi\cos\theta\}/\{\cos\theta\cos\psi\cos\alpha + \sin\psi\sin\theta\}]\frac{d}{dt}\psi \tag{3b}$$

The variations of the angle of rotations and angular velocities for five different angles of inclinations are shown in Figures 2.3 and 2.4, respectively.

Figure 2.3. Driven shaft rotation for 180 degrees rotation of driving shaft for angle of inclination 0 (___), 20 (----), 27 (-.-.-.), 33 (___), 40 (.....) degrees.

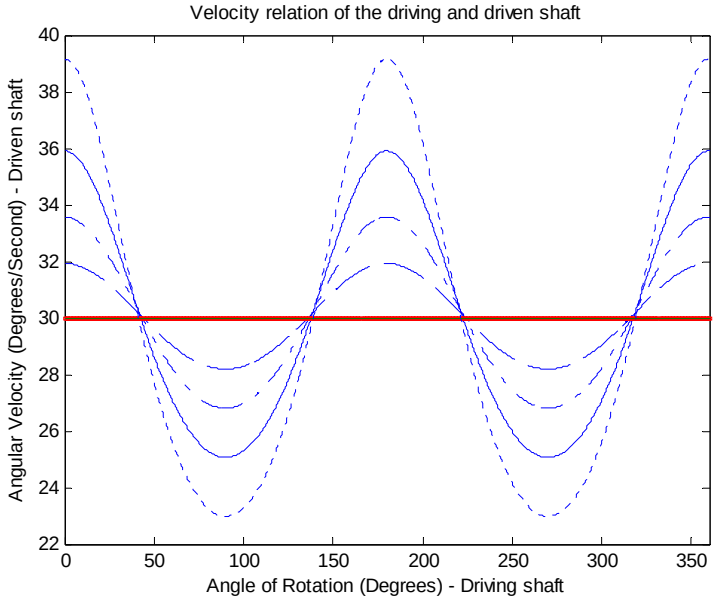

Figure 2.4. Velocity variation of the driven shaft for 360 degree rotation of the driving shaft for the angle of inclination of 0 (___), 20 (----), 27 (-.-.-.), 33 (___), 40 (.....) degrees.

In Figure 2.3, it can be seen that the variation of the rotation angle of the driven and driving shafts increases with the angle of inclination, similarly in Figure 2.4 the variation of the angular velocity of the driven shaft for a constant angular velocity input of the driving shaft increases as well with the angle of inclination. For each cycle of 90 degree rotation, the output shaft accelerates or decelerates, but the average velocity remains constant. The expressions and figures presented in this section show the effect of the angle of inclination on the angle of rotation and angular velocity of the driven shaft with respect to the driving shaft.

It has been shown in Figure 2.4 that for zero angle of inclination the driven shaft rotates with the same angular velocity as that of the driven shaft which can be used for the dynamic analysis of the rotating shaft-universal joint arrangement.

CHAPTER 3

DYNAMICS OF THE UNIVERSAL JOINT

3.1 FORMULATION OF PROBLEM

In this section, the dynamic equations of motion of the shaft-universal joint system shown in Figure 3.1 are formulated. The problem of the general non-parallel shafts coupled by a universal joint is considered. The analysis assumes the system as a discrete-continuous multibody dynamic model, where the discrete universal joint the shaft bodies are formulated as 2 noded beam finite element. Each finite element node consists of three translational and three rotational degrees of freedom. The cross-pin is connected to the input and the output yokes by four frictionless pin joints. Using the approach for multibody dynamics given by Shabana [14], the system can be considered as a combination of various flexible and rigid bodies. Each body is tracked by a set of coordinate systems as shown in Figure 3.2, which will be decided later. The input end of the driving shaft will have a specified motion that is similar to the effect of a fixed point for the purpose of deriving the dynamic stiffness of the system.

At first, the equations for the position vector and velocity of an arbitrary point in a beam element is presented, which is then used to calculate the kinetic energy expressions [14]. The equation of motion of the element is then formulated using Lagrange's equation. Combining the universal joint kinematical relations and the finite element representation, the system equations of motion are formulated by applying the constraint equations.

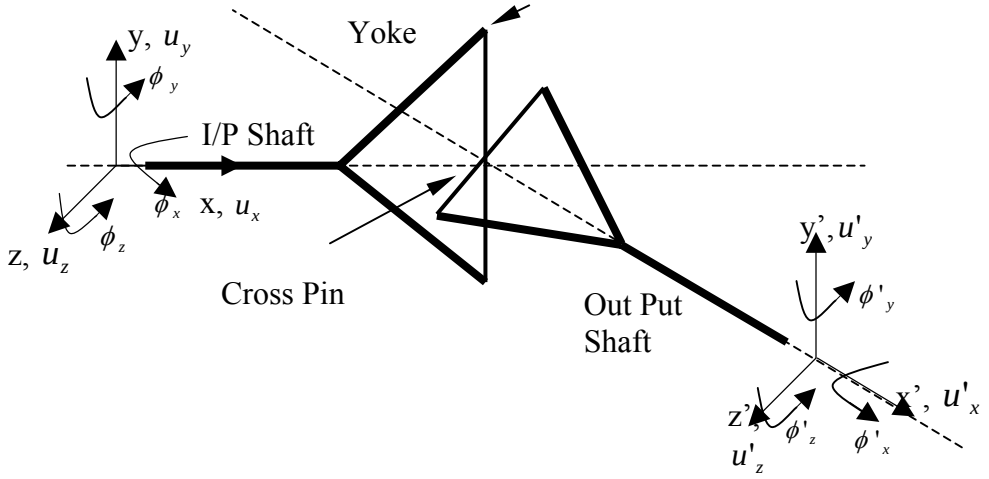

Figure 3.1. Shaft-Universal joint system model with degrees of freedom
$u = \{u_x, u_y, u_z, \phi_x, \phi_y, \phi_z\}$

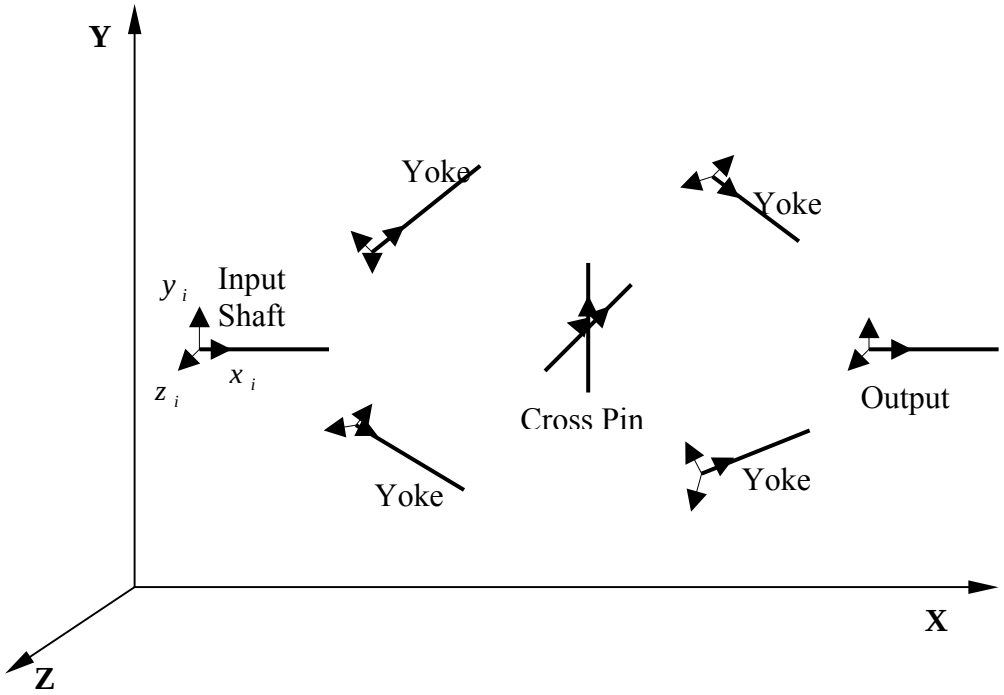

Figure 3.2. Primary bodies of the shaft-universal joint system

XYZ – Global coordinate system, x_i, y_i, z_i - Body coordinate system for i^{th} body.

3.2 POINT DISPLACMENT OF A FLEXIBLE BODY

This section employs a set of Cartesian coordinates to describe the orientation and location of the individual bodies in the system [14]. The connectivity between the adjacent bodies in a system is achieved by a set of constraint equations. The beam element [14] shown in Figure 3.3 is applied to the shaft-universal joint system as noted earlier.

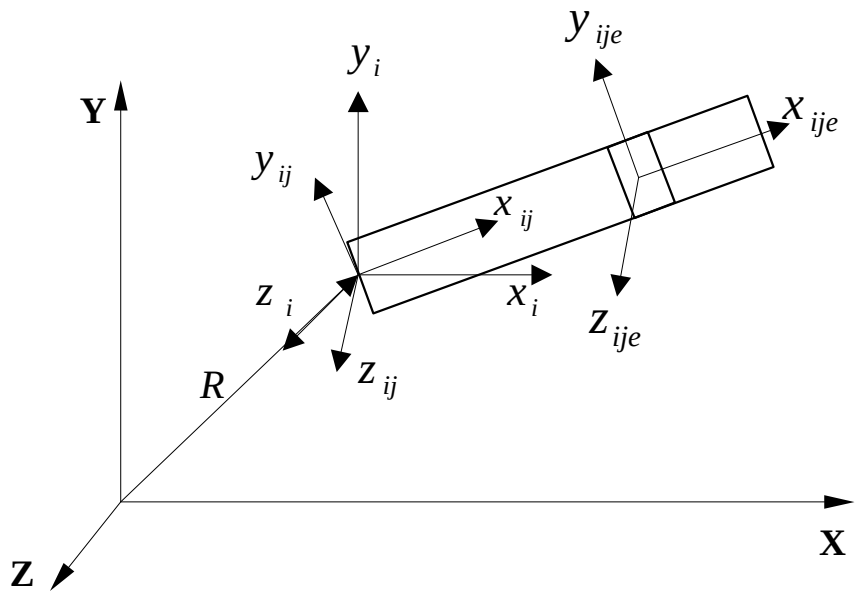

Figure 3.3. Position vector of an arbitrary point in a beam with XYZ – Global coordinate system, x_i, y_i, z_i - Body coordinate system, x_{ij}, y_{ij}, z_{ij} - Intermediate coordinate system, $x_{ije}, y_{ije}, z_{ije}$ - Element coordinate system and i, j are the body and element numbers respectively.

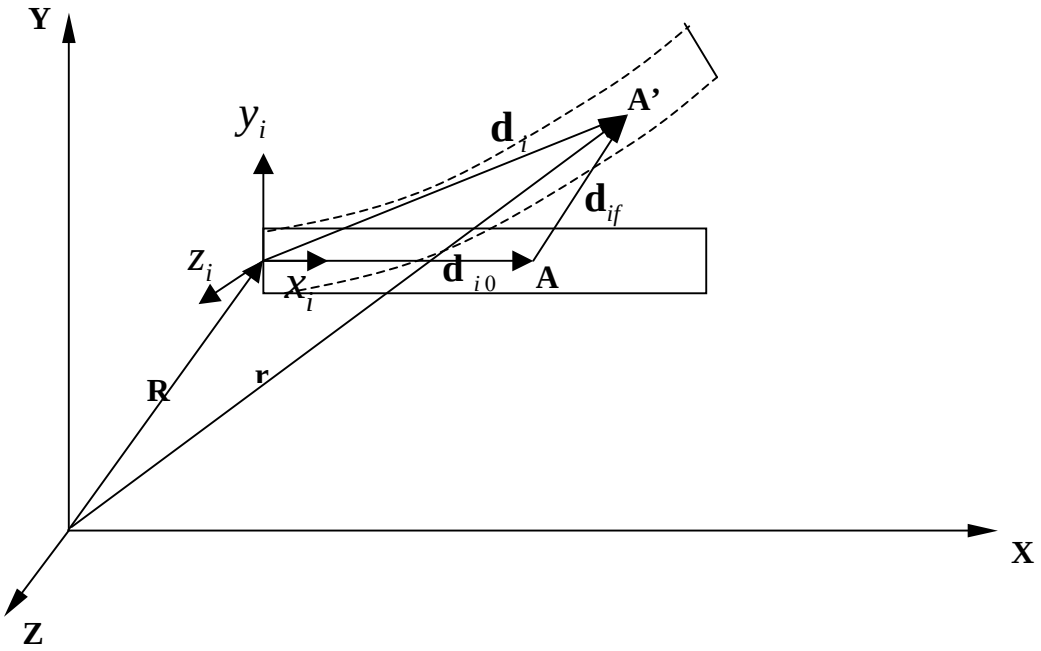

Figure 3.4. Position vector of an arbitrary point in deformed and un-deformed state for i^{th} body

In Figure 3.3, XYZ represents the global coordinate system, $x_i y_i z_i$ is the body coordinate system for any body i. Since we are dealing with multibody finite element system all the bodies are descritized into different number of elements. For a body i and element j, $X_{ije} Y_{ije} Z_{ije}$ represents element coordinate system, $X_{ij}\ Y_{ij}\ Z_{ij}$ represents intermediate coordinate system [14]. These subscripts are used in further discussion. Since the body coordinate system is rigidly attached to the body, the position vector of the body coordinate system origin gives the rigid body motion relative to the global coordinate system. Each element will possess an element coordinate system. The intermediate coordinate system is used to relate the body to the element coordinate systems.

The position vector of an arbitrary point in any elastic body can be explained using global and body coordinate system. Consider an elastic beam element placed in a global coordinate system XYZ as shown in the Figure 3.4. After undergoing deformation, point A moves to A'. The new position vector of the point A', with respect to the body coordinate system, can now be defined as

$$\mathbf{d} = \mathbf{d}_0 + \mathbf{d}_f \tag{4}$$

where, $\mathbf{d}$ is the position vector of A' with respect to the body coordinate system, $\mathbf{d}_0$ is the constant position vector in the un-deformed state and $\mathbf{d}_f$ defines the elastic deformation of point A. Hence, the new position vector of the point A' with respect to the global coordinate system can be defined as [14]

$$\mathbf{r} = \mathbf{R} + [\mathbf{A}]\ \mathbf{d}, \tag{5}$$

as illustrated in figure 3.4. Here, the vector **r** is the position vector with respect to the global coordinate system and **R** is the position vector of the origin of the body coordinate system with respect to the global coordinate system. The motion of the body coordinate system can be related to the global coordinate system by using a transformation matrix **[A]**. In case of a non-rotating universal joint analysis the transformation matrix **[A]** takes the form of the identity matrix.

Next, consider a beam element with two nodes and six degrees of freedom per node (three translational and three rotational) for which the nodal coordinate are expressed as **u** in Figure 3.1, subscript i and j will be used further for body and element number respectively. The displacements of these nodes are unknown and a piecewise fit is used to uniquely describe the shape of each element. The corresponding displacement functions of the beam element can be written using Finite element techniques as [14],

$$\mathbf{w}_{ije} = \mathbf{s}\,\mathbf{u}_{ije} \ , \tag{6}$$

where, $\mathbf{w}_{ije}$ is the displacement field vector with respect to element coordinate system, **S** is the shape function matrix and $\mathbf{u}_{ije}$ is the nodal coordinate vector with respect to the element coordinate system. Since we are dealing with a multibody model, indices i and j refer to body numbers and element numbers respectively.

From Equation 3.2, the nodal coordinates can be written relative to an intermediate coordinate system that is initially parallel to the element coordinate system

as, $\mathbf{u}_{ij} = \mathbf{u}_{ij0} + \mathbf{u}_{ijf}$, where $\mathbf{u}_{ij0}$ is the nodal coordinate vector for the un-deformed coordinates and $\mathbf{u}_{ijf}$ is the nodal coordinate vector of the deformed coordinates.

Since the intermediate coordinate system was initially parallel to the element coordinate system, the displacement field can be written in the intermediate coordinate system as,

$$\mathbf{w}_{ij} = \mathbf{s}\ \mathbf{u}_{ij}, \tag{7}$$

where, $\mathbf{w}_{ij}$ is assumed displacement field in intermediate coordinate system, and $\mathbf{u}_{ij}$ is nodal coordinate system in intermediate coordinate system.

As defined earlier the intermediate coordinate system maintains a fixed orientation with respect to the body coordinate system, and hence the vector of nodal coordinates in intermediate system can be defined in terms of the vector of the nodal coordinate in body coordinate system as,

$$\mathbf{u}_{ij} = \mathbf{c}_i \mathbf{u}_i \tag{8}$$

where, $\mathbf{u}_i$ is vector of nodal coordinates of the arbitrary point in body coordinate system, and $\mathbf{c}_i$ is transformation matrix (from body to intermediate coordinate systems). Also, the assumed displacement field in body coordinate system can be defined as,

$$\mathbf{w}_i = \mathbf{c}_{ij}\ \mathbf{w}_{ij} \tag{9}$$

where, $\mathbf{c}_{ij}$ is transformation matrix from intermediate coordinate system to body coordinate system. Substituting the value of $\mathbf{w}_{ij}$ from Equation 7 into Equation 9 yields,

$\mathbf{w}_i = \mathbf{c}_{ij}\ \mathbf{s}\ \mathbf{u}_{ij}$, then substituting the value of $\mathbf{u}_{ij}$ from Equation 8 into this equation yields [14],

$$\mathbf{w}_i = \mathbf{c}_{ij}\mathbf{s}\ \mathbf{c}_i\mathbf{u}_i \tag{10}$$

The above equation defines the deformation coordinates of an arbitrary point on the finite element with respect to the body coordinate system. Combining Equation (10) and Equation (5), will give rise to [14],

$$\mathbf{r} = \mathbf{R} + [\mathbf{A}]\ \mathbf{c}_{ij}\mathbf{s}\ \mathbf{c}_i\mathbf{u}_i\,, \tag{11 a}$$

$$\mathbf{r} = \mathbf{R} + [\mathbf{A}]\mathbf{N}\mathbf{u}_i\,, \tag{11 b}$$

$$\mathbf{N} = \mathbf{c}_{ij}\mathbf{s}\ \mathbf{c}_i\ . \tag{12}$$

3.3 EQUATION OF MOTION

The equations of motion are derived using the Lagrange technique. For a multibody system the Lagrange formulation takes the form of

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}}\right)^T - \left(\frac{\partial T}{\partial q}\right)^T + \left(\frac{\partial V}{\partial q}\right)^T = Q \tag{13}$$

where, T and V is the kinetic and potential energies respectively, q is generalized degrees of freedom and Q_e the vector of external forcing function.

3.3.1 *Kinetic energy calculation:*

The kinetic energy can be obtained using the displacement vector from the equation [5],

$$T = \frac{1}{2}\int_V \rho \ \ \dot{r}^T . \dot{r} \ . dv \tag{14}$$

Differentiating equation (11 b) with respect to time once, substituting $\mathbf{u} = \mathbf{u}_0 + \mathbf{u}_f$ as shown above and recognizing the $\mathbf{u}_0$ is time invariant yields,

$$\frac{d}{dt}(\mathbf{r}) = \frac{d}{dt}(\mathbf{R}) + \frac{d}{dt}([\mathbf{A}])\ \mathbf{u} + [\mathbf{A}]\frac{d}{dt}(\mathbf{u}_f)$$

which can be written as [14],

$$\dot{\mathbf{r}} = \dot{\mathbf{R}} + [\dot{\mathbf{A}}]\ \mathbf{u} + [\mathbf{A}]\ \dot{\mathbf{u}}_f \tag{15}$$

In the above equation **R** and [**A]** can be time-dependant or constant depending on whether or not the body coordinate system is translating and rotating along with the body. For the case when the relative translation and rotation between body coordinate system and the global coordinate system is allowed, the above equation can be written as [14],

$$\dot{\mathbf{r}} = \dot{\mathbf{R}} - [\overline{\mathbf{A}}]\ \mathbf{u}\ \dot{\theta} + [\mathbf{A}]\mathbf{N}\dot{\mathbf{u}}_f\,, \tag{16}$$

where, $\dot{\mathbf{A}} = -\overline{\mathbf{A}}\dot{\theta}$ and $\dot{\mathbf{u}}_f = \mathbf{N}\ \dot{\mathbf{u}}_f$,

The above equation can further be written in the matrix form as [14],

$$\dot{\mathbf{r}} = \begin{bmatrix}\mathbf{I} & -[\overline{\mathbf{A}}].\mathbf{u} & [\mathbf{A}]\mathbf{N}\end{bmatrix}\begin{bmatrix}\dot{\mathbf{R}} \\ \dot{\theta} \\ \dot{\mathbf{u}}_f\end{bmatrix}, \tag{17}$$

where, **I** is an identity matrix. This expression for the velocity vector is incorporated into the kinetic energy equation to formulate the mass matrix [M] as,

$$T = \frac{1}{2}\int_V \rho^i \dot{r}^T . \dot{r} \; .dv = \frac{1}{2}.\mathbf{q}^T .[M].\mathbf{q}$$

where, $\mathbf{q} = \left[\mathbf{R}^{\mathbf{T}} \quad \theta^{\mathbf{T}} \quad \mathbf{u}_{\mathbf{f}}^{\mathbf{T}}\right]^{\mathbf{T}}$

$$M = \int_{\mathbf{v}} \rho . \begin{bmatrix} \mathbf{I} & -\overline{\mathbf{A}}\mathbf{u} & \mathbf{A}N \\ & \mathbf{u}^T . \overline{A^T} . \overline{A}\mathbf{u}^T & \mathbf{u}^T . \overline{A^T} . A.N \\ Sym & & N^T . N \end{bmatrix} .dv \tag{18}$$

The mass matrix beam element presented above is the generalized case when the relative translation and rotation between body coordinate system and the global coordinate system is allowed. These equations are the generalized equations and can be used for the dynamic analysis of rotating universal joint and shaft system. But in case of a non-rotating universal joint, when the reference motion is not allowed, the mass matrix from equation (18) takes the form shown below:

$$M = \int_{\mathbf{v}} \rho . \mathbf{N}^T \mathbf{N} . dv$$

The kinetic energy expression mentioned above gives the energy expression for one element. The kinetic energy for the complete body can be directly calculated by adding up the kinetic energy for each element of the body. If there are n elements in a body the total kinetic energy of the body can be written as,

$$T_i = \sum_{j=1}^{n} T_{ije}$$

where, i and j are body number and element number respectively.

$$T_i = \frac{1}{2}.\sum_{j=1}^{n} \dot{q}_{ije}{}^T \; M_{ije} \; \dot{q}_{ije}$$

$$T_i = \frac{1}{2}\dot{q}_i{}^T \; [\sum_{j=1}^{n} M_{ije} \;]\dot{q}_i$$

$$M_i = [\sum_{j=1}^{n} M_{ij} \;] \tag{19}$$

This is the complete mass matrix for one component with n number of elements.

Detailed expressions of the individual elements of [M] are presented in the Appendix A.

3.3.2 *Potential energy calculation:*

The potential energy expression can be written as,

$$V = -\frac{1}{2}\int_v \sigma^T \delta\varepsilon \; dv \tag{20}$$

Where, σ is the stress vector and ε is the strain vector. Substitution of the strain displacement and stress-strain relationships in the potential energy expression yields,

$$V = -\frac{1}{2}\int_v [q_f{}^T (D \; N \;)^T E \; (D \; N \;)dv \;]\delta q_f$$

where, D is the differential operator, u_f is the deformation vector and E is the constitutive matrix. The above expression can be written as,

$$V = -\frac{1}{2} q_f{}^T [k] q_f \tag{21}$$

Where, $[k]$ is the stiffness matrix of the beam element. The stiffness matrix for a component of the universal joint can be written as the sum of the stiffness matrices of the element of that body.

$$K_i = [\sum_{j=1}^{n} k_{ije}]$$

Detailed expressions for individual elements of the stiffness matrix for the beam element are represented in the Appendix A.

Then substituting the kinetic and potential energy expressions into the Lagrange's equation yields,

$$M_i . \ddot{q}_i + K_i . q = Q_e - \dot{M}_i . \dot{q}_i + \left[\frac{\partial}{\partial q_i} \left(\frac{1}{2} \dot{q}_i{}^T M_i . \dot{q}_i \right) \right]^T \quad (22)$$

where, the term $(-\dot{M}_i . \dot{q}_i + \left[\frac{\partial}{\partial q_i} \left(\frac{1}{2} \dot{q}_i{}^T M_i \dot{q}_i \right) \right]^T)$ represents the quadratic velocity vector [14] resulting from the differentiation of the kinetic energy with respect to time and with respect to the body coordinates. Note that the term contains the gyroscopic and Coriolis force components.

The equation (22) above gives an expression for the equation of motion for a beam element. After finding out the mass matrix and the stiffness matrix for one component, the same formulation can be applied to all other components of the universal joint structure by changing the material properties and the geometry. Now since the mass matrix for a non-rotating case is time invariant the Equation of motion shown in equation (22) takes the form,

$$M_i . \ddot{q}_i + K_i . q_i = Q_e \quad (23)$$

3.4 CONSTRAINT EQUATION

The equation of motion and calculations for the mass and the stiffness matrices for a beam element are presented in the previous section. For the finite element application and for assembling the various components to form the complete shaft- universal joint system dynamic stiffness matrix, constraint equations are required. For the non-rotating universal joint arrangement, as shown in Figure 3.5 with non-parallel shaft arrangement the constraint equations that need to be applied are presented below.

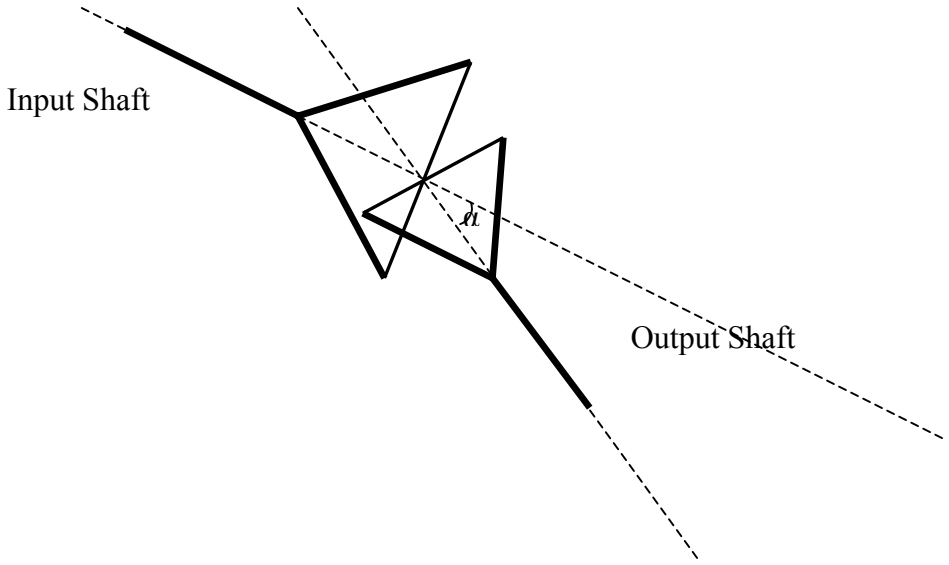

Figure 3.5. Non-parallel shaft arrangement

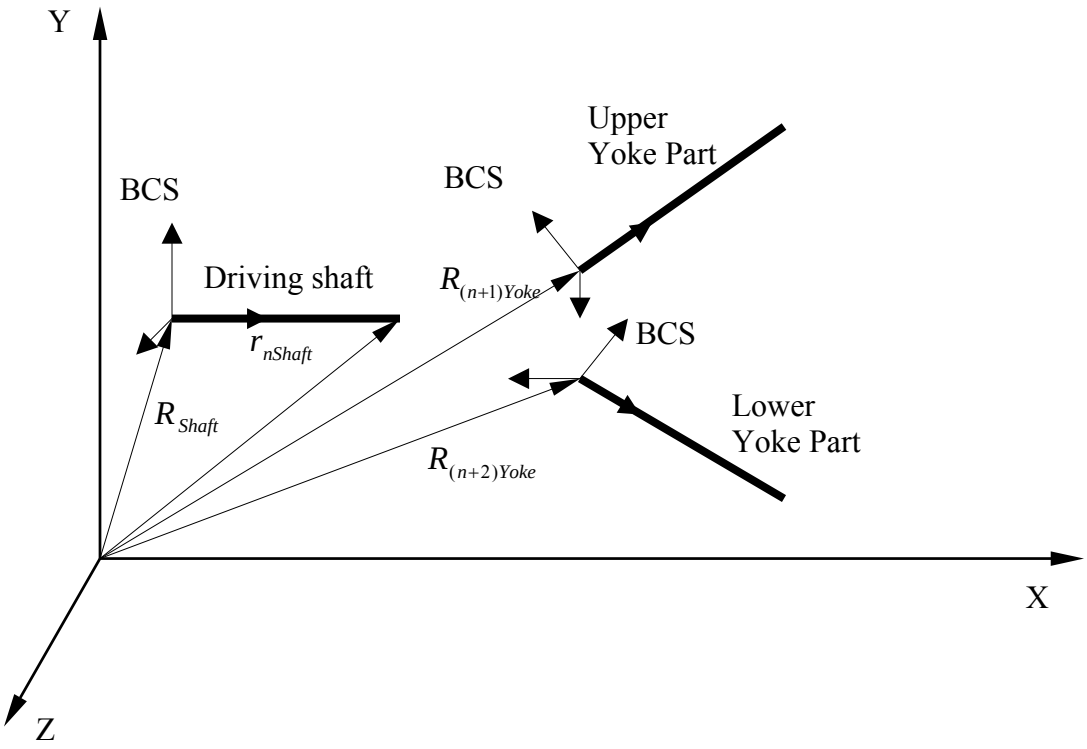

Figure 3.6. Driving shaft and yoke position vectors

For the Input shaft and the input yoke with XYZ as global coordinate system, from Figure 3.6,

$$r_{nShaft} = R_{(n+1)Yoke} ,$$

and, $$r_{nShaft} = R_{(n+2)Yoke} , \quad (24)$$

where, r_{nShaft} is the position vector of the last node of the input shaft. Since the input shaft has been modeled to be connected to the input yoke branches at the end, so there are two equations above. $R_{(n+1)Yoke}$ is the position vector of the first node of the upper yoke branch. Also, $R_{(n+2)Yoke}$ is the position vector of the first node of the lower yoke branch. All the degrees of freedom of the last node of the shaft are connected to that of the yoke

branches. Similar constraint equations are applicable for the output shaft and the output yoke.

In the model the two pins are modeled a separate pieces and are connected to each other at the center of each pin. It can be represented in the form shown below:

$$r_{cp1} = r_{cp2} \tag{25}$$

All the degrees of freedom of the two pins are connected and form a single cross pin piece. Also, in the model the cross pin and the yokes are two different parts and are connected to each other forming a pin joint.

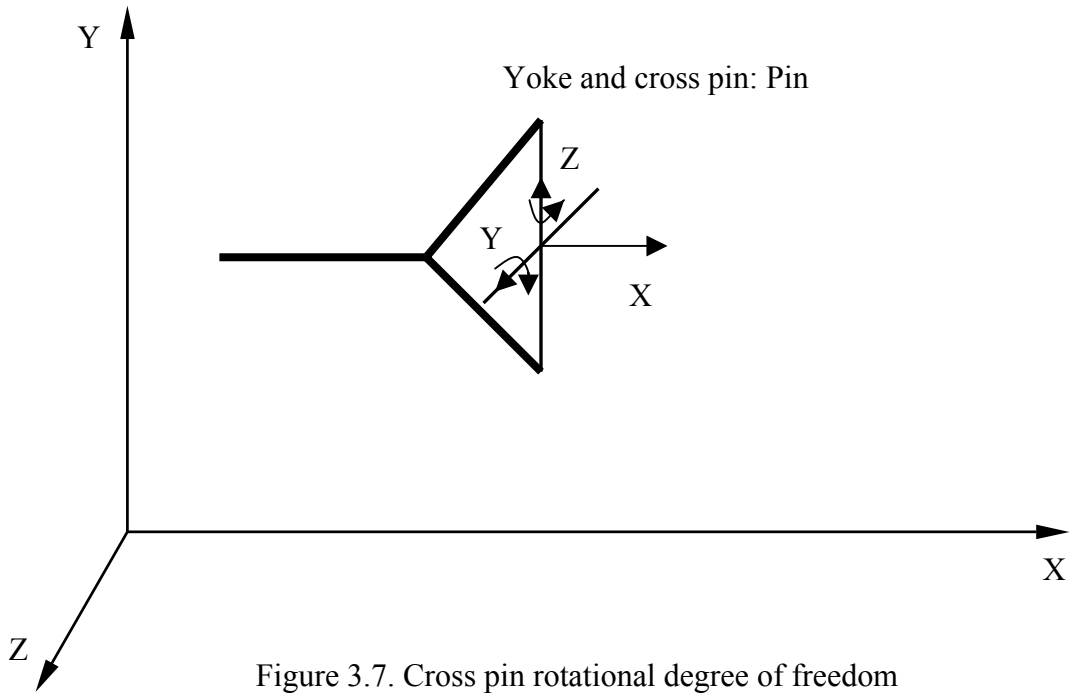

Figure 3.7. Cross pin rotational degree of freedom

The cross pin is connected to the yoke through roller bearing hence the pin can rotate in the longitudinal axis while the other two rotational and all of the translational

degrees of freedom are connected to the yoke. One of the degrees of freedom is free hence the pins are allowed to rotate along the longitudinal axis. As shown in the Figure 3.7 the cross pin branch attached to the input yoke is allowed to rotate along the longitudinal Z axis. No friction has been considered in the pin joint. The input shaft in this study has been considered to be constrained as a fixed condition.

3.5 ASSEMBLY OF THE SYSTEM

To assemble to mass and the stiffness matrices of the complete system the individual mass and stiffness matrices of the components are calculated in the local coordinate system and then transformed to the global coordinate system and then assembled according to the constraint equations mentioned in the previous section.

CHAPTER 4

NUMERICAL SOLUTION AND VALIDATION

4.1 SOLUTION OF EQUATION

The final equation of motion of the system would be used to get the modal frequency and the mode shapes of the complete system. In this study there is no external force acting on the joint, and the input shaft is fixed at one end. After calculation of the mass matrix and the final equation will be solved using the state –space transformation. This transformation reformulates the original equation of the system into equivalent sets of 2N first order differential equation known as Hamilton canonical equation.

It can be represented as,

$$[M]\{\ddot{x}\}+[c]\{\dot{x}\}+[k]\{x\}=\{f\}$$

And the system of equation can be written as,

$$[M]\{\dot{x}\}-[M]\{\dot{x}\}=\{0\}$$

Above two equations can be written as,

$$[A]\{\dot{y}\}+[B]\{y\}=\{f\}$$

where, $[A]=\begin{bmatrix}[0] & [M]\\ [M] & [c]\end{bmatrix}$ and $[B]=\begin{bmatrix}-[M] & [0]\\ [0] & [K]\end{bmatrix}$

$$\{\dot{y}\}=\begin{matrix}\{\ddot{x}\}\\ \{\dot{x}\}\end{matrix},\qquad \{y\}=\begin{matrix}\{\dot{x}\}\\ \{x\}\end{matrix},\qquad \{f\}=\begin{matrix}\{0\}\\ \{f\}\end{matrix}$$

The solution of above equation yields the complex valued natural frequencies (eigenvalues) and complex valued modal vectors (eigenvectors).The formulation is solved using the above transformation to get natural frequencies and mode shapes.

4.2 NUMERICAL SOLUTION

For the solution of the complete equation of the universal joint system, a MATLAB code is developed. The code is based on the analytical procedure discussed in the previous sections. The code is developed for the study of the various cases of the universal joints with different material and geometric properties. In case of analysis of large assemblies with universal joints, the system Dynamic Stiffness Matrix calculated from the analytical method can directly be incorporated in the computational analysis tool for further calculation. The geometrical information like length, width and heights and the material properties like the young's modulus and poisons ratio are required for the solution. The code uses a graphical user interface for inputting the values.

4.3 GRAPHICAL USER INTERFACE

The GUI has been introduced easily input the geometry, material properties and number of elements to be descritized. It allows inputting different geometry and different material properties for different components. Material properties like young's modulus and poisons ratio are required to complete the solution.

After inputting the material properties and geometrical properties the MATLAB code uses the calculations procedure defined in the previous sections. The natural frequency and the mode shapes are extracted using the above mentioned formulation.

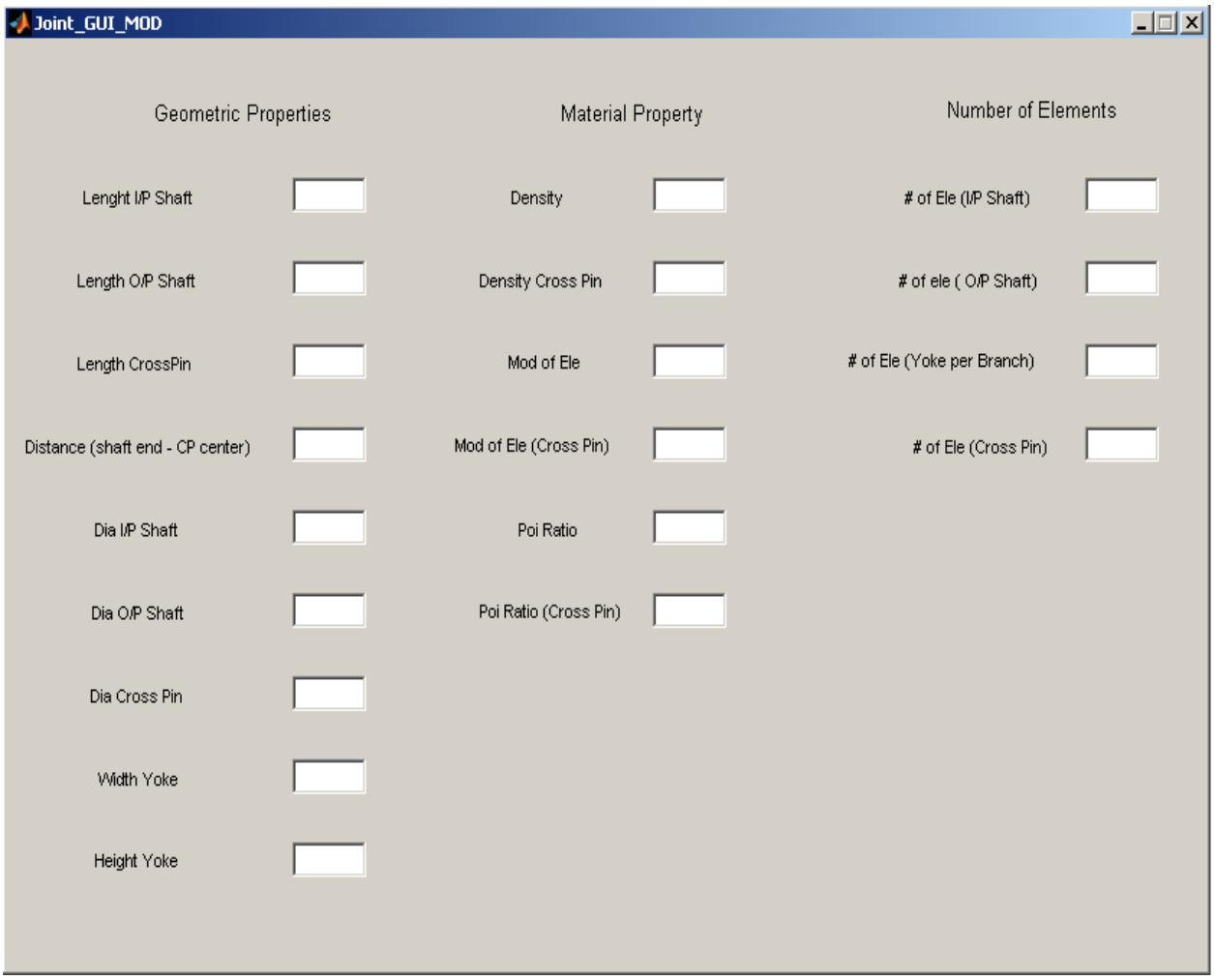

Figure 4.1: Graphical User Interface

4.4 EXAMPLES

The above formulation is a generalized presentation of system with two non-parallel shafts coupled by universal joint. The particular example demonstrated in this sections are for two non-rotating parallel shaft arrangement with an input shaft, output shaft, yokes and cross pin. The angle of inclination of the input and the output shafts are considered to be negligible. The expression for this arrangement can be derived by considering the inclination angle to be zero in the above formulation.

Although different geometry and material properties for different components of the system can be used but in the examples the material properties i.e. the young's modulus, density and the poisons ratio of all of the components are considered to be the same. The geometry is symmetrical and the angle of inclination between the input and the output shafts is zero i.e. the axes of the two shafts are parallel. The various parameters of the universal joint used in this formulation are given below.

Material properties		**Geometrical parameters**			
Young's Modulus (N/m^2)	2.00E+11	Length I/P Shaft (m)	1	Diameter I/P Shaft (m)	0.01
Density (Kg/m^2)	7850	Length O//P Shaft (m)	1	Diameter O/P Shaft (m)	0.01
Poisson's Ratio	0.3				

Table 4.1. Material and geometrical parameters

Geometrical Parameter	Case I	Case II	Case III	Case IV
Width of Yoke (mm)	10	10	20	20
Height of Yoke (mm)	10	10	20	20
Diameter of Cross Pin (mm)	5	10	5	10

Table 4.2. Typical design parameters for automotive applications

Four different cases are mentioned in Table 4.2 to demonstrate the variation of the natural frequencies with the variation in geometrical parameters. The comparison of the natural frequencies, for the typical operating range of 0 to 100 Hz, for the cases mentioned in Table 4.2 is shown in the Table 4.3

Natural Frequencies (Hz)			
Case I	Case II	Case III	Case IV
0	0	2.10E-05	0
0.21902	0.36233	0.1341	0.24693
0.6747	1.1858	0.39762	0.97166
0.82322	1.7836	0.5147	1.4827
1.661	3.9335	0.91372	2.8892
2.6123	4.9211	1.4852	4.3757
5.1451	7.0427	3.4086	6.0087
5.8276	7.3096	5.5996	7.3027
7.1358	7.567	6.0413	7.5623
7.6087	10.653	7.6255	9.472
8.4287	11.898	8.8056	11.679
13.668	16.395	18.166	27.276

Table 4.3. Natural frequencies for different cases

By changing the geometry of the universal joint the critical resonance zones and undesirable mode shapes can be avoided. It can be seen from the frequency response

functions that by changing the geometry of the universal joint system the natural frequencies changes. It is observed from the Figure 4.2, that by changing the diameter of the cross pin the total stiffness of the system increases and the natural frequencies shifts towards the higher end.

Figure 4.2 below presents the comparison of the variation of the natural frequency for case I and case II.

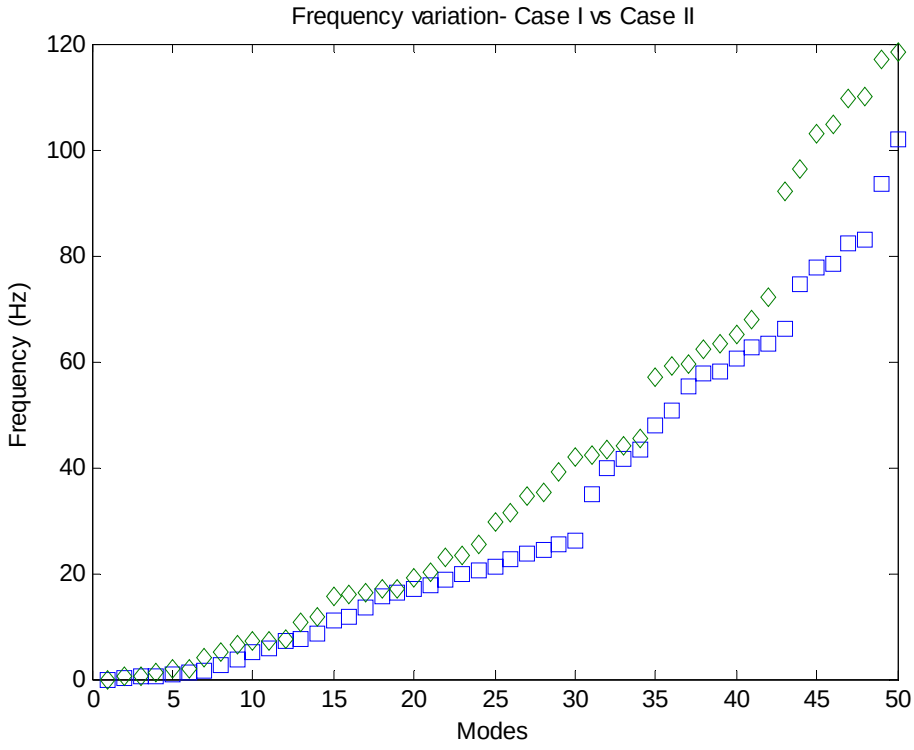

Figure 4.2: Variation of natural frequencies- Case I vs. Case II

From Figure 4.3, comparison of case I and Case III, it can be seen that by increasing the cross section area of the input and the output yoke makes the system stiffer and shifts the higher natural frequencies to the higher side. But, increase in the cross

sectional area of the yoke also increases the mass of the system and it can be seen that it shifts the lower frequencies of the system to the lower end.

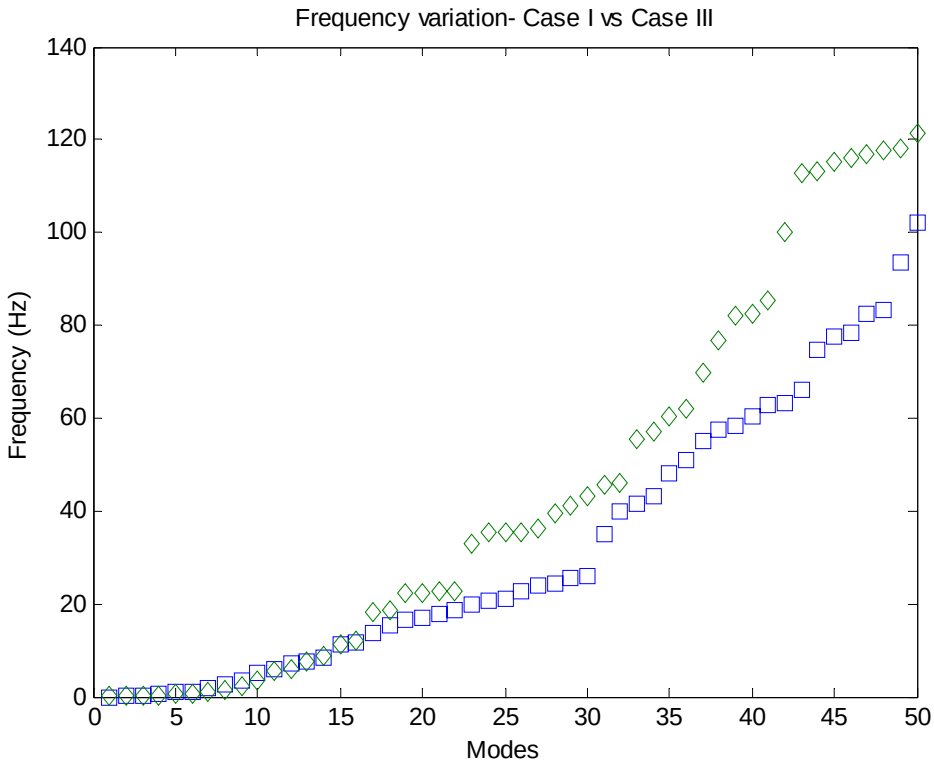

Figure 4.3: Variation of natural frequencies- Case I vs. Case III

The various components of the model can be descritized into any desirable number of finite elements but due to computational limitations lesser number of elements are considered in these examples. The input shaft, output shafts and the yokes for this solution are descritized into five elements and the cross pin is descritized into four elements.

Figures 4.4 - 4.6 below shows some typical frequency response functions for the universal joint system (case I) for the normal operating frequency range of 0 to 200 Hz. 1% damping is considered in the system.

These frequency response function plots show the plot of the log magnitude versus frequency for second node on the driving shaft.

From the figures 4.4 - 4.6, it is observed that the universal joint has fewer modes of vibration in the lateral direction (X-direction) than in transverse direction (Y and Z). The system has more modes of vibration in transverse directions and also these modes show up at lower frequencies. From figure 4.5 and 4.6 it can be seen that the behavior of the joint is almost similar in transverse directions because of the symmetry of the structure.

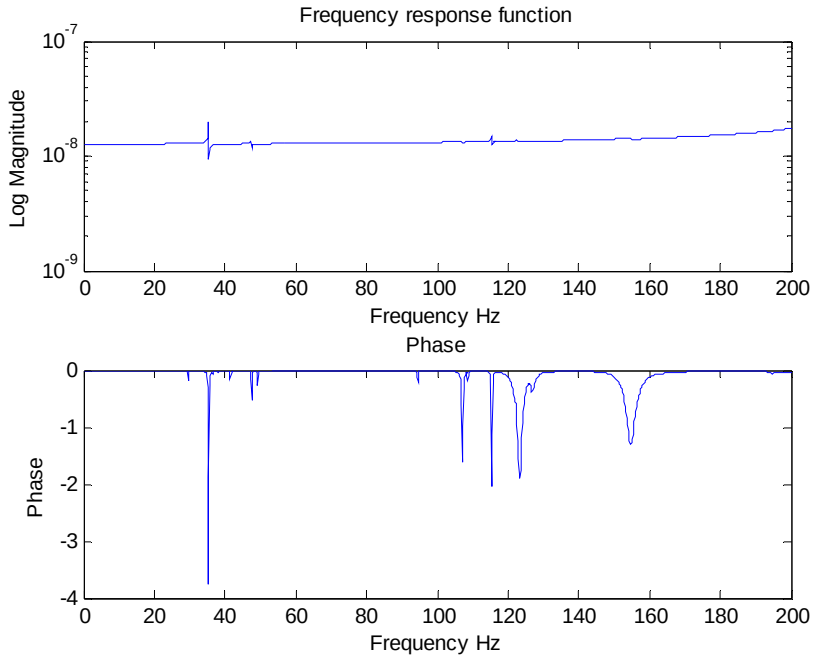

Figure 4.4. FRF X-direction (case IV)

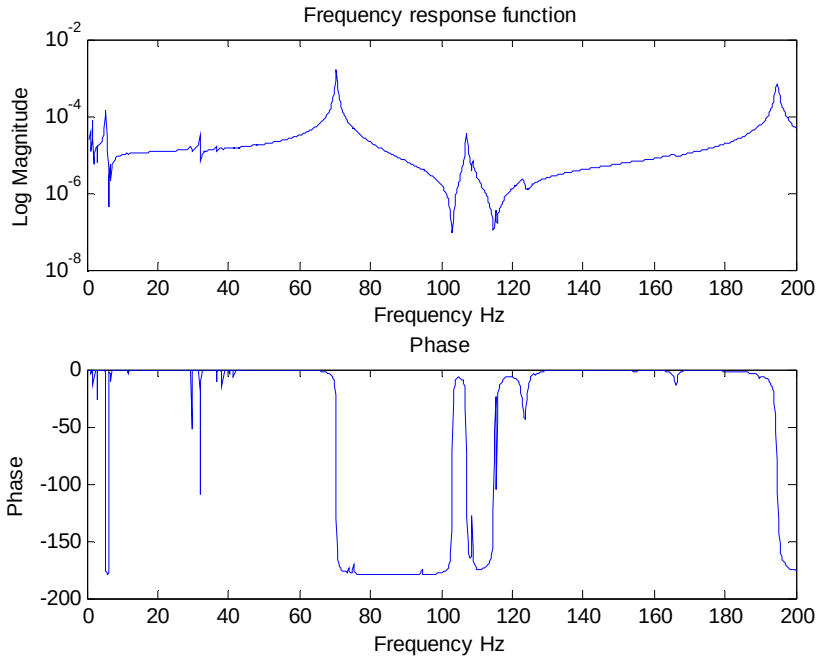

Figure 4.5. FRF Y-direction (case IV)

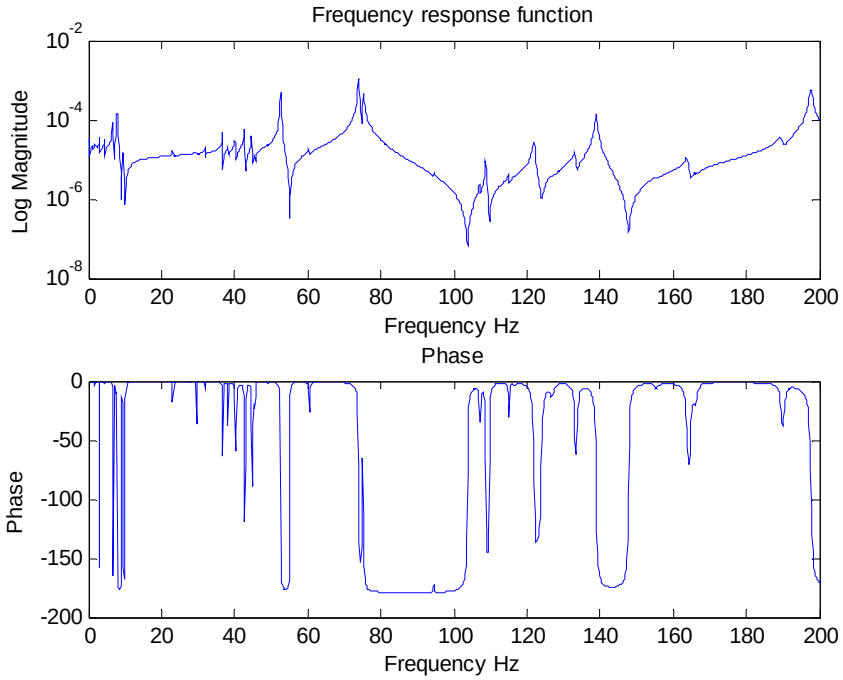

Figure 4.6. FRF Z-direction (case IV)

4.5 VALIDATION

For the validation purposes a finite element model is produced using general purpose finite element simulation software. The formulation is done using the line beam element and each line beam element has two nodes and each node has six degrees of freedom, Three translational and three rotational. The model has been solved to validate the results of the analytical model.

The table below shows a comparison of natural frequencies from the analytical and finite element models. All the material properties are same as mentioned in Table 4.1.

Frequency (Hz)			Frequency (Hz)		
Sr. #	**ANSYS**	**MATLAB**	**Sr. #**	**ANSYS**	**MATLAB**
1	0	0	11	7.9487	8.4287
2	0	0.21902	12	13.968	13.668
3	0.68531	0.6747	13	16.481	16.467
4	0.7689	0.82322	14	17.864	17.76
5	1.7384	1.661	15	18.755	18.605
6	2.3024	2.6123	16	21.21	20.533
7	5.0885	5.1451	17	22.877	21.152
8	5.5982	5.8276	18	22.916	22.676
9	7.1461	7.1358	19	23.088	23.807
10	7.2399	7.6087	20	23.263	24.233

Table 4.4. Comparison of natural frequencies from analytical and FE model

The natural frequency results from the finite element model are matching with the natural frequency results of the analytical model.

Figures 4.7 - 4.9 below shows some typical frequency response functions for the universal joint system (case I) for the normal operating frequency range of 0 to 200 Hz.

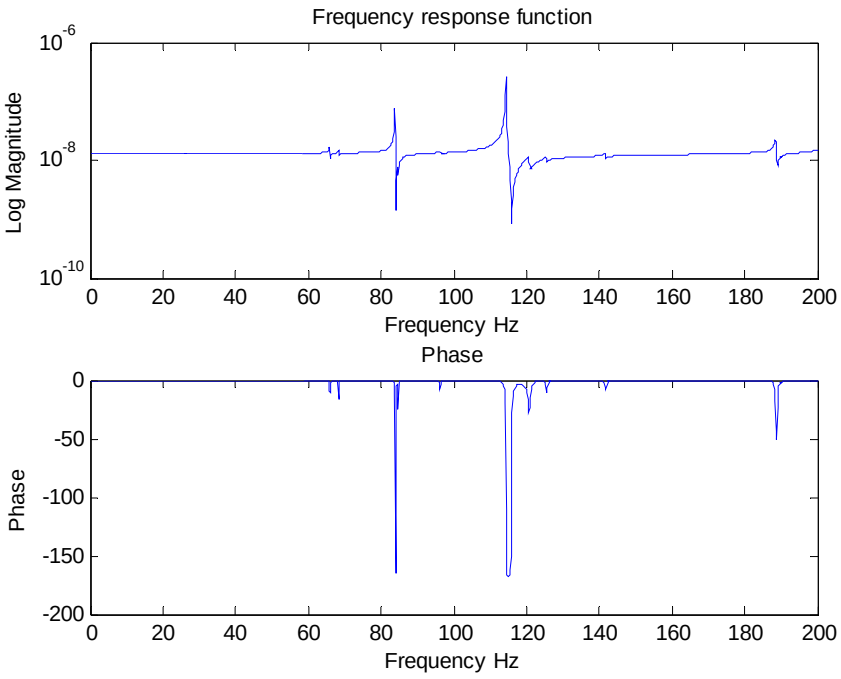

Figure 4.7: Frequency response function X- direction (case I)

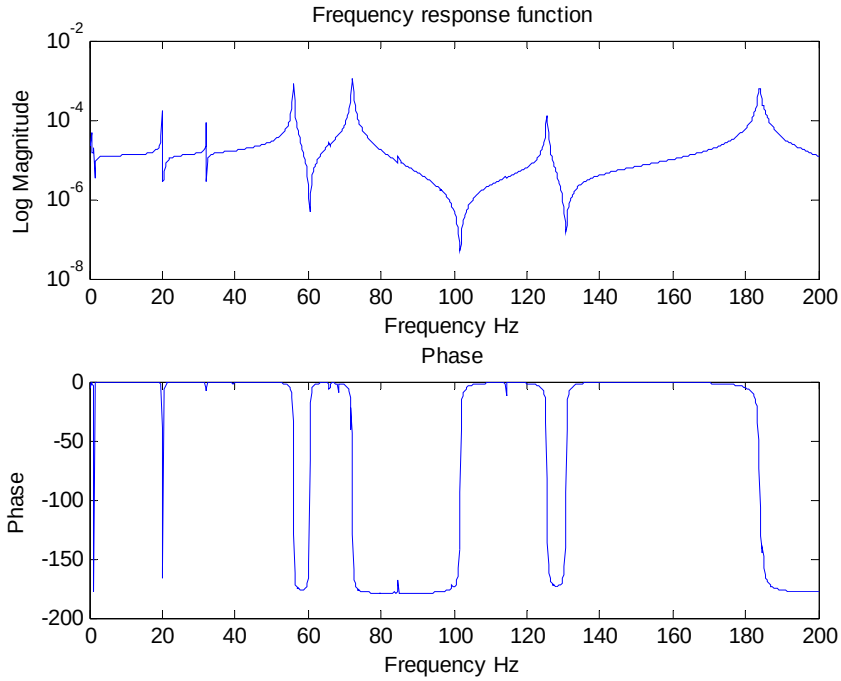

Figure 4.8: Frequency response function Y- direction (case I)

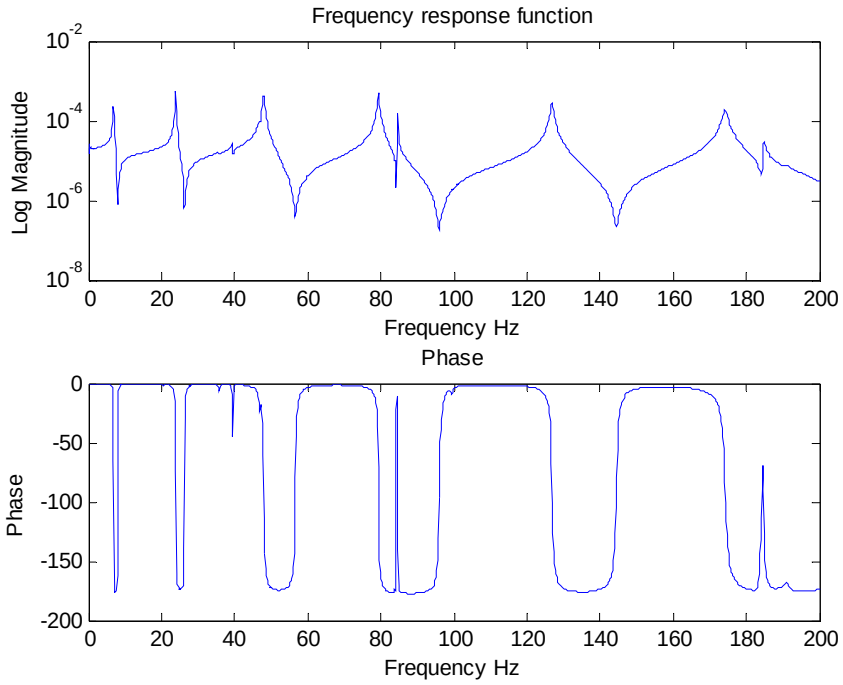

Figure 4.9: Frequency response function Z- direction (case I)

CHAPTER 5

CONCLUSION

The kinematics and the dynamics of a shaft-universal joint system are investigated in the paper. The kinematics study gives a relationship between the angle of rotation and angular velocity of the driving and the driven shaft. It is found that for a non-parallel shaft arrangement the variation in the angle of rotation and the angular velocity of the driven shaft increases with the inclination angle. Also there is no difference in the angle of rotation and angular velocity of the driving and the driven shaft for zero inclination angle.

The dynamical analysis of the shaft-universal joint system is proposed in the paper. The equation of motion and dynamic stiffness matrix for the complete model is presented. The dynamic stiffness matrix for the universal joint is calculated and then incorporated to the finite element shaft model. Modal solution of the complete system is performed to calculate the natural frequencies and the mode shape of the system. It is found that the universal joint system has fewer longitudinal modes of vibration as compared with transverse modes. The variation in the natural frequencies of the system with the variation in the geometrical parameters is observed. A comparison study of the effect of the geometry of the bodies of the model on the natural frequency is presented. It is shown that this generalized procedure proposed can be used for the design of the universal joint in terms of avoiding critical resonance frequencies and undesirable mode shapes. The dynamic stiffness matrix developed in this paper can be employed to any

shaft model or large assemblies. The results of the particular example solved in the paper are validated from the general purpose finite element software. The general equation of motion, formulated in this paper, can be used for the vibration analysis of rotating set of two shafts coupled by universal joint as a part of future work.

REFERENCES

1. Wagner, E.R. and Cooney, C.E. (1979) *Universal Joint and Driveshaft Design Manual*, Society of Automotive Engineers, Warren dale, PA,

2. Yang, A.T., (1965) "Static force and torque analysis of spherical four-bar mechanisms," *ASME Journal of Engineering for Industry*, Vol.87, pp.221-227.

3. Rosenberg, R.M., (1958) "On the dynamic behavior of rotating shafts driven by universal (Hooke) couplings", *ASME Journal of Applied Mechanics*, pp.47-51.

4. Fischer, I. and Freudenstein, F. (1984) "Internal force and moment transmission in a cardan joint with manufacturing tolerances," *ASME Journal of Mechanisms, Transmission and Automation in Design*, Vol. 106, pp.301-311.

5. Chen, C.K., and Freudenstein, F. (1986), "Dynamic analysis of a universal joint with manufacturing tolerances," *ASME Journal of Mechanisms, Transmission and Automation in Design*, Vol. 108, pp.524-532.

6. Freudenstein, F. and Macey, J.P. (1990), 'The inertia torque of the Hooke Joint', Proc. of the 21st Biennial ASME Mechanism Conference,' September 16-19, Chicago, DE-Vol. 24, pp.407-413.

7. Ota, H. and Kato, M. (1984) "Lateral vibration of a rotating shaft driven by a universal joint – 1st report", Bulletin of JSME 27, No. 231, September, pp.2002-2007.

8. Ota, H. Kato, M. and Sugita, H. (1985), "Lateral vibration of a rotating shaft driven by a universal joint – 2nd report," Bulletin of JSME 28, No. 241, August, pp.1749-1755.

9. Ota, H. and Kato, M. (1990), "Lateral excitation of a rotating shaft driven by a universal joint with friction," *Journal of Vibration and Acoustics*, Vol. 112, pp.298-303.

10. Sheu, P.-P., Chieng, W.-H., Lee, A.-C. (1996) "Modeling and analysis of intermediate shaft between two universal joints," *Journal of Vibration and Acoustics* Vol. 118, pp.88-99.

11. Mazzei Jr., A.J., Argento, A., Scott, R.A. (1999), "Dynamic stability of a rotating shaft driven through a universal joint," *Journal of Sound and Vibration*, Vol. 222, pp.19-47.

12. Brutti, C., Pennestri, E., Biancolini, M.E. (1999), 'On the dynamics of the transmission with a double cardan joint,' Tenth world conference on the theory of machines and mechanisms, Oulu, Finland, June 20-24.

13. Brutti, C., Pennestri, E., Biancolini, M.E., Valentini, P.P. (2003), 'Dynamic, mechanical efficiency, and fatigue analysis of the double cardan homokinetic joint', International Journal of Vehicle Design, Vol. 32, Nos.3/4, pp.231-249.

14. Shabana, A.A., Dynamics of Multibody Systems, Second Edition, Cambridge University Press

15. Przemieniecki, S., Theory of Matrix structural analysis McGraw-Hill Book Company.

16. Cook, R.D., Malkus, D.S., Plesha, M.E., Concept and applications of finite element analysis, Third Edition, John Wiley & Sons (ASIA) Pte Ltd.

APPENDIX A

From the equation (41), the mass matrix [14],

$$\mathbf{M} = \int_{\mathbf{v}} \rho . \begin{bmatrix} \mathbf{I} & -\overline{\mathbf{A}}\overline{U} & \mathbf{A}N \\ & \overline{U^T}.\overline{A^T}.\overline{A}\overline{U} & \overline{U^T}.\overline{A^T}.A.N \\ Sym & & N^T.N \end{bmatrix} .dv$$

If the beam element is fixed at one end, then there will be no relative motion between the global and the body coordinate system. And for the non-rotating case the body coordinate system maintains a fixed orientation w.r.t. global coordinate system, hence for this case the mass matrix of the beam element reduce down to $[M] = \int_{v} N^T.N.dv$

Where, N is the shape function matrix. The shape function of a beam element with two nodes and six degrees of freedom per node has been calculated using the Hermitian interpolating functions. It is known that the shape function of a beam element in the longitudinal direction are same as the bar element. The longitudinal shape functions of a bar element are calculated using the liner interpolation functions. Hence the shape functions for a beam element of length 'l' in the longitudinal direction can be written as,

$$N_x = [1 - \frac{x}{l}, \frac{x}{l}]$$

For the transverse direction shape function that will be same in both y and z direction can be found out using the cubic interpolation functions.

The shape functions for a beam element in the transverse direction can be given as,

$$N_1 = 1 - 3\left(\frac{x}{l}\right)^2 + 2\left(\frac{x}{l}\right)^3 \qquad N_2 = x - 2\left(\frac{x^2}{l}\right) + \left(\frac{x^3}{l^2}\right)$$

$$N_3 = 3\left(\frac{x}{l}\right)^2 - 2\left(\frac{x}{l}\right)^3 \qquad N_4 = -\left(\frac{x^2}{l}\right) + \left(\frac{x^3}{l^2}\right)$$

Now the shape function for a beam element can be expanded to three dimensional shape functions by combining the shape functions in longitudinal and transverse direction [14].

$$[N] = \begin{bmatrix} 1-\xi & 0 & 0 \\ 6.(\xi-\xi^2).\eta & 1-3.\xi^2+2.\xi^3 & 0 \\ 6.(\xi-\xi^2).\zeta & 0 & 1-3.\xi^2+2.\xi^3 \\ 0 & -(1-\xi).l.\zeta & -(1-\xi).l.\eta \\ (1-4.\xi+3.\xi^2).l.\zeta & 0 & (-\xi+2.\xi^2-\xi^3).l \\ (-1+4.\xi-3.\xi^2).l.\eta & (\xi-2.\xi^2+\xi^3).l & 0 \\ \xi & 0 & 0 \\ 6.(-\xi+\xi^2).\eta & 3.\xi^2-2.\xi^3 & 0 \\ 6.(-\xi+\xi^2).\zeta & 0 & 3.\xi^2-2.\xi^3 \\ 0 & -l.\xi.\zeta & -l.\xi.\eta \\ (-2\xi+3\xi^2).l.\zeta & 0 & (\xi^2-\xi^3).l \\ (2\xi-3\xi^2).l.\eta & (-\xi^2+\xi^3).l & 0 \end{bmatrix}$$

where, $\xi = x/l$, $\eta = y/l$ and $\zeta = z/l$.

x, y and z are the spatial coordinate along the element axes.

Now these shape functions can be used to find out the mass matrix for the beam element.

The Mass matrix [15] can be represented as,

$$\rho Al\begin{bmatrix}
\frac{1}{3} & & & & & & & & & & & \\
0 & \frac{13}{35}+\frac{6I_z}{5Al^2} & & & & & & & & & & \\
0 & 0 & \frac{13}{35}+\frac{6I_y}{5Al^2} & & & & & & \textit{Symmetric} & & & \\
0 & 0 & 0 & \frac{j_x}{3A} & & & & & & & & \\
0 & 0 & -\frac{11l}{210}-\frac{I_y}{10Al} & 0 & \frac{l^2}{105}+\frac{2I_y}{15A} & & & & & & & \\
0 & \frac{11l}{210}+\frac{I_z}{10Al} & 0 & 0 & 0 & \frac{l^2}{105}+\frac{2I_z}{15A} & & & & & & \\
0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{3} & & & & & \\
0 & \frac{9}{70}-\frac{6I_z}{5Al^2} & 0 & 0 & 0 & \frac{13l}{420}-\frac{I_z}{10Al} & 0 & \frac{13}{35}+\frac{6I_z}{5Al^2} & & & & \\
0 & 0 & \frac{9}{70}-\frac{6I_y}{5Al^2} & 0 & -\frac{13}{420}+\frac{I_y}{10Al} & 0 & 0 & 0 & \frac{13}{35}+\frac{6I_y}{5Al^2} & & & \\
0 & 0 & 0 & \frac{j_x}{6A} & 0 & 0 & 0 & 0 & 0 & \frac{j_x}{3A} & & \\
0 & 0 & \frac{13}{420}-\frac{I_y}{10Al} & 0 & -\frac{l^2}{140}-\frac{I_y}{30A} & 0 & 0 & 0 & \frac{11l}{210}+\frac{I_y}{10Al} & 0 & \frac{l^2}{105}+\frac{2I_y}{15A} & \\
0 & -\frac{13}{420}+\frac{I_z}{10Al} & 0 & 0 & 0 & -\frac{l^2}{140}-\frac{I_z}{30A} & 0 & -\frac{11l}{210}-\frac{I_z}{10Al} & 0 & 0 & 0 & \frac{l^2}{105}+\frac{2I_z}{15A}
\end{bmatrix}$$

The stiffness matrix [15] can be represented as,

$$\begin{bmatrix}
\frac{EA}{l} & & & & & & & & & & & \\
0 & \frac{12EI_z}{l^3(1+\phi_y)} & & & & & & & & & & \\
0 & 0 & \frac{12EI_y}{l^3(1+\phi_z)} & & & & & & & & & \\
0 & 0 & 0 & \frac{Gj}{l} & & & & & & & & \\
0 & 0 & \frac{-6EI_y}{l^2(1+\phi_z)} & 0 & \frac{(4+\phi_z)EI_y}{l(1+\phi_z)} & & & & & & & \\
0 & \frac{6EI_z}{l^2(1+\phi_y)} & 0 & 0 & 0 & \frac{(4+\phi_y)EI_z}{l(1+\phi_y)} & & & & & & \\
-\frac{EA}{l} & 0 & 0 & 0 & 0 & 0 & \frac{AE}{l} & & & & & \\
0 & -\frac{12EI_z}{l^3(1+\phi_y)} & 0 & 0 & 0 & \frac{-6EI_z}{l^2(1+\phi_y)} & 0 & \frac{12EI_z}{l^3(1+\phi_y)} & & & & \\
0 & 0 & \frac{-12EI_y}{l^3(1+\phi_z)} & 0 & \frac{6EI_y}{l^2(1+\phi_z)} & 0 & 0 & 0 & \frac{12EI_y}{l^3(1+\phi_z)} & & & \\
0 & 0 & 0 & \frac{-Gj}{l} & 0 & 0 & 0 & 0 & 0 & \frac{Gj}{l} & & \\
0 & 0 & \frac{-6EI_y}{l^2(1+\phi_z)} & 0 & \frac{(2-\phi_z)EI_y}{l(1+\phi_z)} & 0 & 0 & 0 & \frac{6EI_y}{l^2(1+\phi_z)} & 0 & \frac{(4+\phi_z)EI_y}{l(1+\phi_z)} & \\
0 & \frac{6EI_z}{l^2(1+\phi_y)} & 0 & 0 & 0 & \frac{(2-\phi_y)EI_z}{l(1+\phi_y)} & 0 & \frac{-6EI_z}{l^2(1+\phi_y)} & 0 & 0 & 0 & \frac{(4+\phi_y)EI_z}{l(1+\phi_y)}
\end{bmatrix}$$

APPENDIX B

B.1 MATLAB Scripts for angle of rotation and angular velocity relationship for non-parallel shaft case

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%
% Kinematics of a universal joint with non-parallel shafts
% code for non-parallel shafts
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%
clc;
clear all;
phi=linspace(0,360,1000)

ang=linspace(45,60,4)
alph=ang*pi/180;

thet2=zeros(4,1000);

for ii=1:4
 for pp=1:1000
thet1(pp)=atan(tan(pi*phi(pp)/180)/cos(alph(ii)));
end


thet=abs(thet1)*180/pi;

for kk=1:1000
    if(kk<=249)
        thet2(ii,kk)=thet(kk);
    else
```

```
        if(250<=kk & kk<=499)
            thet2(ii,kk)=180-thet(kk);
        else
            if(500<=kk & kk<=749)
                thet2(ii,kk)=180+thet(kk);
            else
                thet2(ii,kk)=360-thet(kk);
            end
        end
    end
end
end
%plot(thet2(4,:))
figure
plot(phi,thet2(1,:),'--')
hold on
plot(phi,thet2(2,:),'-.')
hold on
plot(phi,thet2(3,:),'-')
hold on
plot(phi,thet2(4,:),':')
hold on
plot(phi,phi,'--rs','linewidth',.1,'markersize',1.5)
axis([0,180,0,200])
title('Angle of rotation relatioship - driving and the driven shaft')
xlabel('Angle of Rotation of driving shaft')
ylabel('Angle of Rotation of driven shaft')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
```

```
num=zeros(1,1000);
den=zeros(1,1000);
vel_phi=30;

for ii=1:4
for jj=1:1000
   num(jj)=(sin(thet2(ii,jj)*pi/180)*sin(phi(jj)*pi/180)*cos(alph(ii)))...
      +(cos(phi(jj)*pi/180)*cos(thet2(ii,jj)*pi/180));
   den(jj)=(cos(thet2(ii,jj)*pi/180)*cos(phi(jj)*pi/180)*cos(alph(ii)))...
      +(sin(phi(jj)*pi/180)*sin(thet2(ii,jj)*pi/180));
   vel_thet(ii,jj)=(num(jj)/den(jj))*vel_phi;
end
end
figure
vel_phi_1=30*ones(1,1000);
plot(phi,vel_thet(1,:),'--',phi,vel_phi_1)
hold on
plot(phi,vel_thet(2,:),'-.',phi,vel_phi_1)
hold on
plot(phi,vel_thet(3,:),'-',phi,vel_phi_1,'--rs','linewidth',.1,'markersize',2)
hold on
plot(phi,vel_thet(4,:),':',phi,vel_phi_1)
axis([0,360,22,40])

title('Velocity relation of the driving and driven shaft')
xlabel('Angle of Rotation (Degrees) - Driving shaft')
ylabel('Angular Velocity (Degrees/Second) - Driven shaft')
```

B.2.1.Main

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CODE FOR THE VIBRATIONAL ANALYSIS OF UNIVERSAL JOINT
CONSIDERING IT TO BE
% A 3-D BEAM STRUCTURE.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Amit Sharma
% Graduate Student
% Vibro-Acoustics Lab
% MINE Department
% University of Cincinnati, OH
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clc;
clear all;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Loading the ANSYS Mass and Stiffness matrices for the validation of
% MATLAB results using ANSYS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
load('mt1.mtx')
load('kt1.mtx')
% Square matrix formulation of ANSYS output
mass_mat_an=zeros(220,220);
stiff_mat_an=zeros(220,220);
for ii=1:1405
   mass_mat_an(mt1(ii,1),mt1(ii,2))=mt1(ii,3);
   stiff_mat_an(kt1(ii,1),kt1(ii,2))=kt1(ii,3);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Loading the GUI values
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
load yoke_beam_height;load len_is;load len_os;load len_cp;load len_y_cp;
load dia_is;load dia_os;load dia_cp;load yoke_beam_width;
load den;load den_cp;load mod_E;load mod_E_cp;load poi;
load poi_cp;load num_ele_is;load num_ele_os;load num_ele_y;load num_ele_cp;
ndof=12;            % Number of degrees of freedom per element
den=den;            % Density of the material of the beam
den_cp=den_cp;      % density of the material of cross pin

L_is=len_is;        % Length of the input shaft
```

```
L_os=len_os;          % Length of the output shaft
L_cp=len_cp;          % Length of the cross-pin
L_y_cp=len_y_cp;      % Distance between crosspin center and I/P shaft end
L_y= abs(sqrt(L_cp^2+(L_y_cp)^2)); % Lenght of the yoke beam

dia_is=dia_is;        % Diameter of the input shafts
dia_os=dia_os;        % Diameter of the output shafts
dia_cp=dia_cp;        % Diameter of the crosspin

dia_is=0.5;        % Diameter of the input shafts
dia_os=0.5;        % Diameter of the output shafts
dia_cp=0.2;        % Diameter of the crosspin



a=yoke_beam_width;    % width yoke beam
b=yoke_beam_height;   % Height yoke beam
%A=a*b;               % Cross sectional area of the yoke beam

a=0.25;    % width yoke beam
b=0.25;   % Height yoke beam

n_is=num_ele_is;      % Number of elemtns for FE descritization of i/p shaft
n_os=num_ele_os;      % Number of elemtns for FE descritization of o/p shaft
n_y=num_ele_y;        % Number of elemtns for FE descritization of yoke beam
n_cp=num_ele_cp;      % Number of elemtns for FE descritization of cross pin

l_is=L_is/n_is;       % Lenght of one element of i/p shaft
l_os=L_os/n_os;       % Lenght of one element of the o/p shaft
l_y=L_y/n_y;          % Lenght of one element of the yoke
l_cp=2*L_cp/n_cp;     % Lenght of one element of the cross pin
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Calculating other parameters
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
angle=(180/pi)*atan(L_cp/(L_y_cp));     % Angle of the yoke inclination
                                % from the center line

E=mod_E;                        % Modulus of Elasticity
E_cp=mod_E_cp;                  % Modulus of Elasticity of Cross pin
v=poi;                          % Poissions ratio
v_cp=poi_cp;                    % Poissions ratio fo Cross pin

G=E/(2*(1+v));                  % Shear Modulus
G_cp=E_cp/(2*(1+v_cp));         % Shear Modulus
```

```
% Defining the total no of nodes of the structure
nodes=n_is+n_os+4*n_y+2*n_cp+3;
% Defining the total no of dof of the system
dof=6*nodes-20;
% Clearing the unwanted parameters
clear edit1 edit2 edit3 edit4 edit5
clear edit6 edit7 edit8 edit9 edit10
clear edit11 edit12 edit13 edit14 edit15
clear edit22 edit23 edit24 edit25
clear eventdata handles ans hObject

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% FORMULATION AND ASSEMBLY OF INPUT SHAFT AND YOKE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Assembling the mass and element matrices for the input shaft
m_ele_is=mass_ip_shaft(n_is,dia_is,den,l_is,ndof,E,G);
k_ele_is=stiff_ip_shaft(n_is,dia_is,den,l_is,ndof,E,G);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Formulating the mass marix for 45 degrees inclined beam element

m_ele2=mass_beam(n_y,a,b,den,ndof,E,G,l_y);
k_ele2=stiff_beam(n_y,a,b,den,ndof,E,G,l_y);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% One part of the input yoke has been considered to be inclined at an
% angles + Angle and other part to be inclined at an angle - Angle degrees

% Defining the trasformation angle for the input yoke positive part

theta=angle;
trans=trans_45(theta);
k_ele_p=trans'*k_ele2*trans;

% Defining the trasformation angle for the input yoke positive part

theta=-angle;
trans=trans_45(theta);
k_ele_n=trans'*k_ele2*trans;
```

```
% Predefining the shaft, input and output yoke final stiffness matrices

K_is=zeros(dof,dof);
K_p=zeros(dof,dof);
K_n=zeros(dof,dof);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Assembling the stiffness matrices for input shaft only

for kk=1:n_is
   intK_s=zeros(dof,dof);
   n_c=linspace(1,n_is+1,n_is+1);
   % n_c=[1,2,3,4,5,6]
   r1=n_c(kk); r2=n_c(kk+1);
   A1=[6*(r1-1)+1,6*(r1-1)+2,6*(r1-1)+3,6*(r1-1)+4,6*(r1-1)+5,6*(r1-1)+6];
   B1=[6*(r2-1)+1,6*(r2-1)+2,6*(r2-1)+3,6*(r2-1)+4,6*(r2-1)+5,6*(r2-1)+6];
   for ii=1:6
      for jj=1:6
         intK_s(A1(ii),A1(jj))=k_ele_is(ii,jj);
         intK_s(B1(ii),B1(jj))=k_ele_is(6+ii,6+jj);
         intK_s(A1(ii),B1(jj))=k_ele_is(ii,6+jj);
         intK_s(B1(ii),A1(jj))=k_ele_is(6+ii,jj);
      end
   end
   K_is=K_is+intK_s;
end
clear r1 r2 n_c A1 B1;

% Assembling the stiffness matrices for positive part(upper) of I/P yoke

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% NOTE: In ANSYS start numbering from negative yoke branch (after last node
% of the input shaft)else formulation may give erroneous results
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for kk=1:n_y
   intK_p=zeros(dof,dof);
   n_c=(n_is+1):2:(n_is+(2*n_y)+1);
   %n_c=[6,8,10,12,14,16]
   r1=n_c(kk); r2=n_c(kk+1);
   A1=[6*(r1-1)+1,6*(r1-1)+2,6*(r1-1)+3,6*(r1-1)+4,6*(r1-1)+5,6*(r1-1)+6];
   B1=[6*(r2-1)+1,6*(r2-1)+2,6*(r2-1)+3,6*(r2-1)+4,6*(r2-1)+5,6*(r2-1)+6];
   for ii=1:6
```

```
        for jj=1:6
            intK_p(A1(ii),A1(jj))=k_ele_p(ii,jj);
            intK_p(B1(ii),B1(jj))=k_ele_p(6+ii,6+jj);
            intK_p(A1(ii),B1(jj))=k_ele_p(ii,6+jj);
            intK_p(B1(ii),A1(jj))=k_ele_p(6+ii,jj);
        end
    end
    K_p=K_p+intK_p;
end
clear r1 r2 n_c A1 B1;

% Assembling the stiffness matrices for negative part (lower) of I/P yoke

for kk=1:n_y
    intK_n=zeros(dof,dof);
    n_c_int1=(n_is+2):2:(n_is+(2*n_y));
    n_c_int2=n_is+1;
    n_c(1,1)=n_c_int2;
    n_c(1,2:n_y+1)=n_c_int1;
    %n_c=[6,7,9,11,13,15]
    r1=n_c(kk); r2=n_c(kk+1);
    A1=[6*(r1-1)+1,6*(r1-1)+2,6*(r1-1)+3,6*(r1-1)+4,6*(r1-1)+5,6*(r1-1)+6];
    B1=[6*(r2-1)+1,6*(r2-1)+2,6*(r2-1)+3,6*(r2-1)+4,6*(r2-1)+5,6*(r2-1)+6];
    for ii=1:6
        for jj=1:6
            intK_n(A1(ii),A1(jj))=k_ele_n(ii,jj);
            intK_n(B1(ii),B1(jj))=k_ele_n(6+ii,6+jj);
            intK_n(A1(ii),B1(jj))=k_ele_n(ii,6+jj);
            intK_n(B1(ii),A1(jj))=k_ele_n(6+ii,jj);
        end
    end
    K_n=K_n+intK_n;
end
clear r1 r2 n_c A1 B1;

% Final assembled matrix for the input shaft and yoke

K_fin=K_is+K_p+K_n;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Defining the transformation matrix for the mass matrix for the Input part

% Mass matrix for the positive part of the beam

theta=angle;
```

```
trans=trans_45(theta);
m_ele_p=trans'*m_ele2*trans;

% Mass matrix for the negative part of the beam

theta=-angle;
trans=trans_45(theta);
m_ele_n=trans'*m_ele2*trans;

% Predefining the mass matrics for the shaft, positive and negative yoke

M_is=zeros(dof,dof);
M_p=zeros(dof,dof);
M_n=zeros(dof,dof);

% Assembling the mass matrix for the input shaft

for kk=1:n_is
   intM_s=zeros(dof,dof);
   n_c=linspace(1,n_is+1,n_is+1);
   %n_c=[1,2,3,4,5,6]
   r1=n_c(kk); r2=n_c(kk+1);
   A1=[6*(r1-1)+1,6*(r1-1)+2,6*(r1-1)+3,6*(r1-1)+4,6*(r1-1)+5,6*(r1-1)+6];
   B1=[6*(r2-1)+1,6*(r2-1)+2,6*(r2-1)+3,6*(r2-1)+4,6*(r2-1)+5,6*(r2-1)+6];
   for ii=1:6
      for jj=1:6
         intM_s(A1(ii),A1(jj))=m_ele_is(ii,jj);
         intM_s(B1(ii),B1(jj))=m_ele_is(6+ii,6+jj);
         intM_s(A1(ii),B1(jj))=m_ele_is(ii,6+jj);
         intM_s(B1(ii),A1(jj))=m_ele_is(6+ii,jj);
      end
   end
   M_is=M_is+intM_s;
end
clear r1 r2 n_c A1 B1;

% Assembling the mass matrix for the positive (upper) part of the input yoke

for kk=1:n_y
   intM_p=zeros(dof,dof);
   n_c=(n_is+1):2:(n_is+(2*n_y)+1);
   %n_c=[6,8,10,12,14,16]
   r1=n_c(kk); r2=n_c(kk+1);
   A1=[6*(r1-1)+1,6*(r1-1)+2,6*(r1-1)+3,6*(r1-1)+4,6*(r1-1)+5,6*(r1-1)+6];
   B1=[6*(r2-1)+1,6*(r2-1)+2,6*(r2-1)+3,6*(r2-1)+4,6*(r2-1)+5,6*(r2-1)+6];
   for ii=1:6
```

```
        for jj=1:6
            intM_p(A1(ii),A1(jj))=m_ele_p(ii,jj);
            intM_p(B1(ii),B1(jj))=m_ele_p(6+ii,6+jj);
            intM_p(A1(ii),B1(jj))=m_ele_p(ii,6+jj);
            intM_p(B1(ii),A1(jj))=m_ele_p(6+ii,jj);
        end
    end
    M_p=M_p+intM_p;
end
clear r1 r2 n_c A1 B1;

% Assembling the mass matrix for the negative (lower) part of the input yoke

for kk=1:n_y
    intM_n=zeros(dof,dof);
    n_c_int1=(n_is+2):2:(n_is+(2*n_y));
    n_c_int2=n_is+1;
    n_c(1,1)=n_c_int2;
    n_c(1,2:n_y+1)=n_c_int1;
    %n_c=[6,7,9,11,13,15]
    r1=n_c(kk); r2=n_c(kk+1);
    A1=[6*(r1-1)+1,6*(r1-1)+2,6*(r1-1)+3,6*(r1-1)+4,6*(r1-1)+5,6*(r1-1)+6];
    B1=[6*(r2-1)+1,6*(r2-1)+2,6*(r2-1)+3,6*(r2-1)+4,6*(r2-1)+5,6*(r2-1)+6];
    for ii=1:6
        for jj=1:6
            intM_n(A1(ii),A1(jj))=m_ele_n(ii,jj);
            intM_n(B1(ii),B1(jj))=m_ele_n(6+ii,6+jj);
            intM_n(A1(ii),B1(jj))=m_ele_n(ii,6+jj);
            intM_n(B1(ii),A1(jj))=m_ele_n(6+ii,jj);
        end
    end
    M_n=M_n+intM_n;
end
clear r1 r2 n_c A1 B1;

% Final assembled matrix for the input shaft and the yoke

M_fin=M_is+M_p+M_n;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% clearing any temporary variables used

clear intK_n intK_s intK_p;
clear intM_n intM_s intM_p;
clear M_is M_p M_n;
```

```
clear K_is K_p K_n;
clear n_c n_c_int1 n_c_int2 ii jj kk r1 r2;
clear theta trans;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CROSS-PIN FORMULATION %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Formulating the mass and stiffness matrices for the cross pin

m_ele_cp=mass_cp(n_cp,dia_cp,den_cp,ndof,E_cp,G_cp,l_cp);
k_ele_cp=stiff_cp(n_cp,dia_cp,den_cp,ndof,E_cp,G_cp,l_cp);

% Defining the tranformation matrix for one of the cross pin

theta=-90;
trans=trans_45_cpv(theta);

% Defining the mass and stiffness matrix for the verticle cross pin part

k_ele_cpvp=trans'*k_ele_cp*trans;
m_ele_cpvp=trans'*m_ele_cp*trans;

% Assembling the cross pin with the global input shaft and yoke matrix

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The crosspin will be assembled with the end elements first then the
% elements inbetween the ends and the center part of the cross pin
% End nodes have only one DOF open ie for the Y-axis rotation every other
% dof has been coupled to the end node of the positive and negative yokes.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Assembling the last element of the lower part of the cross pin (Element 17-18 as in
old_mod)

% Mass Matrix
```

```
mib_int1=zeros(dof,dof);
p=n_is+2*n_y+(n_cp/2);          % 17 as in old mod
A1=[6*(p-1)+1,6*(p-1)+2,6*(p-1)+3,6*(p-1)+4,6*(p-1)+5,6*(p-1)+6];
                        % Node 17: A1=[97,98,99,100,101,102]
q=n_is+2*n_y;                   % 15 as in old mod
B1=[6*(q-1)+1,6*(q-1)+2,6*(q-1)+3,6*(q-1)+4,6*p+1,6*(q-1)+6];
                        % Node 18: B1=[85,86,87,88,103,90]
for ii=1:6
   for jj=1:6
      mib_int1(A1(ii),A1(jj))=m_ele_cpvp(ii,jj);
      mib_int1(B1(ii),B1(jj))=m_ele_cpvp(6+ii,6+jj);
      mib_int1(A1(ii),B1(jj))=m_ele_cpvp(ii,6+jj);
      mib_int1(B1(ii),A1(jj))=m_ele_cpvp(6+ii,jj);
   end
end
clear A1 B1;

% Stiffness Matrix

kib_int1=zeros(dof,dof);
p=n_is+2*n_y+(n_cp/2);          % 17 as in old mod
A1=[6*(p-1)+1,6*(p-1)+2,6*(p-1)+3,6*(p-1)+4,6*(p-1)+5,6*(p-1)+6];
                        % Node 17: A1=[97,98,99,100,101,102]
q=n_is+2*n_y;                   % 15 as in old mod
B1=[6*(q-1)+1,6*(q-1)+2,6*(q-1)+3,6*(q-1)+4,6*p+1,6*(q-1)+6];
                        % Node 18: B1=[85,86,87,88,103,90]
for ii=1:6
   for jj=1:6
      kib_int1(A1(ii),A1(jj))=k_ele_cpvp(ii,jj);
      kib_int1(B1(ii),B1(jj))=k_ele_cpvp(6+ii,6+jj);
      kib_int1(A1(ii),B1(jj))=k_ele_cpvp(ii,6+jj);
      kib_int1(B1(ii),A1(jj))=k_ele_cpvp(6+ii,jj);
   end
end
clear A1 B1;

% Defining the final matrix till cross pin lower verticle end elements part

m_f1=M_fin+mib_int1;
k_f1=K_fin+kib_int1;

clear mib_int1 kib_int1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Assembling the upper part of the cross pin (Element 24-25 as in old mod)

% Mass Matrix

mib_int2=zeros(dof,dof);
p1=n_is+2*n_y+2*n_cp+2;          % 25 as in old mod (Calc for node 24)
p2=n_is+2*n_y+1;                 % 16 as in old mod
A1=[6*(p1-2)+1-5,6*(p1-2)+2-5,6*(p1-2)+3-5,6*(p1-2)+4-5,6*(p1-2)+5-5,6*(p1-2)+6-
5];
                              % A1=[134,135,136,137,138,139]
B1=[6*p2-5,6*p2-4,6*p2-3,6*p2-2,6*(p1-2)+7-5,6*p2];
                              % B1=[91,92,93,94,140,96]
for ii=1:6
   for jj=1:6
      mib_int2(A1(ii),A1(jj))=m_ele_cpvp(ii,jj);
      mib_int2(B1(ii),B1(jj))=m_ele_cpvp(6+ii,6+jj);
      mib_int2(A1(ii),B1(jj))=m_ele_cpvp(ii,6+jj);
      mib_int2(B1(ii),A1(jj))=m_ele_cpvp(6+ii,jj);
   end
end

clear A1 B1;

% Stiffness matrix

kib_int2=zeros(dof,dof);
p1=n_is+2*n_y+2*n_cp+2;          % 25 as in old mod (Calc for node 24)
p2=n_is+2*n_y+1;                 % 16 as in old mod
A1=[6*(p1-2)+1-5,6*(p1-2)+2-5,6*(p1-2)+3-5,6*(p1-2)+4-5,6*(p1-2)+5-5,6*(p1-2)+6-
5];
                              % A1=[134,135,136,137,138,139];
B1=[6*p2-5,6*p2-4,6*p2-3,6*p2-2,6*(p1-2)+7-5,6*p2];
                              % B1=[91,92,93,94,140,96];
for ii=1:6
   for jj=1:6
      kib_int2(A1(ii),A1(jj))=k_ele_cpvp(ii,jj);
      kib_int2(B1(ii),B1(jj))=k_ele_cpvp(6+ii,6+jj);
      kib_int2(A1(ii),B1(jj))=k_ele_cpvp(ii,6+jj);
      kib_int2(B1(ii),A1(jj))=k_ele_cpvp(6+ii,jj);
   end
end
clear A1 B1;

% Defining the final matrix till cross pin verticle end elements part
```

```
m_f2=m_f1+mib_int2;
k_f2=k_f1+kib_int2;
clear kib_int2 mib_int2;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Assembling the reamining elements to the globle matrix
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This part assembles the elements in between the end and the mid element of
% the upper and lower part of the cross pin
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Assembling the lower part of the verticle cross pin

m_i1=zeros(dof,dof);m_i2=zeros(dof,dof);
k_i1=zeros(dof,dof);k_i2=zeros(dof,dof);
val=(n_cp/2)-1;
p3=p1-val;

val_ln1=6*p2+1;            % 17 as in old mod when n_cp=4 for lower
val_ln2=6*(p3-1)+1-5;      % 24 as in old mod when n_cp=4 for upper

% NOTE: when loop gets executed val_ln2 will change accordingly

% Assembling the lower part ie 17-18-19 as in n_cp=6

% Loop will only be be executed if number of crosspin elements is more than 4

for ii=1:(n_cp-4)/2
   m_i1=zeros(dof,dof);
   k_i1=zeros(dof,dof);
   m_i1(6*(p2)+1+(6*(ii-1)):6*(p2)+12+(6*(ii-1)),6*(p2)+1+(6*(ii-
1)):6*(p2)+12+(6*(ii-1)))=m_ele_cpvp(1:12,1:12);
   k_i1(6*(p2)+1+(6*(ii-1)):6*(p2)+12+(6*(ii-1)),6*(p2)+1+(6*(ii-1)):6*(p2)+12+(6*(ii-
1)))=k_ele_cpvp(1:12,1:12);
   m_i2=m_i2+m_i1;
   k_i2=k_i2+k_i1;
end

% Assembling the upper part of the verticle cross pin
```

```
m_i1=zeros(dof,dof);m_i3=zeros(dof,dof);
k_i1=zeros(dof,dof);k_i3=zeros(dof,dof);

% Assembling the upper part ie 27-28-29 for n_cp=6

for ii=1:(n_cp-4)/2
   m_i1=zeros(dof,dof);
   k_i1=zeros(dof,dof);
   val=(n_cp/2)-1;
   p3=p1-val;
   m_i1(6*(p3-1)+1+(6*(ii-1))-5:6*(p3-1)+12+(6*(ii-1))-5,6*(p3-1)+1+(6*(ii-1))-
5:6*(p3-1)+12+(6*(ii-1))-5)=m_ele_cpvp(1:12,1:12);
   k_i1(6*(p3-1)+1+(6*(ii-1))-5:6*(p3-1)+12+(6*(ii-1))-5,6*(p3-1)+1+(6*(ii-1))-
5:6*(p3-1)+12+(6*(ii-1))-5)=k_ele_cpvp(1:12,1:12);
   val_ln2=6*(p2+3*(n_cp/2)+1)+1-5;       % 27 as in old mod for n_cp=6
   m_i3=m_i3+m_i1;
   k_i3=k_i3+k_i1;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This part assembles the center two elemets of the cross pin
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

m_iv1=zeros(dof,dof);m_iv2=zeros(dof,dof);
k_iv1=zeros(dof,dof);k_iv2=zeros(dof,dof);

% Assembling the upper part 20-27 as in old mod n_cp=6

mn=n_is+2*n_y+(n_cp/2)+2;   % Mid-node 19 as in old mod
mid_node=6*(mn-1)+1-5;

m_iv1=zeros(dof,dof);m_iv2=zeros(dof,dof);
k_iv1=zeros(dof,dof);k_iv2=zeros(dof,dof);
m_iv1(mid_node:mid_node+5,mid_node:mid_node+5)=m_ele_cpvp(1:6,1:6);
m_iv1(val_ln2:val_ln2+5,val_ln2:val_ln2+5)=m_ele_cpvp(7:12,7:12);
m_iv1(mid_node:mid_node+5,val_ln2:val_ln2+5)=m_ele_cpvp(1:6,7:12);
m_iv1(val_ln2:val_ln2+5,mid_node:mid_node+5)=m_ele_cpvp(7:12,1:6);

k_iv1(mid_node:mid_node+5,mid_node:mid_node+5)=k_ele_cpvp(1:6,1:6);
k_iv1(val_ln2:val_ln2+5,val_ln2:val_ln2+5)=k_ele_cpvp(7:12,7:12);
k_iv1(mid_node:mid_node+5,val_ln2:val_ln2+5)=k_ele_cpvp(1:6,7:12);
k_iv1(val_ln2:val_ln2+5,mid_node:mid_node+5)=k_ele_cpvp(7:12,1:6);
```

```
% Assembling the lower part 17-20 as in old n_cp=6

m_iv2(mid_node:mid_node+5,mid_node:mid_node+5)=m_ele_cpvp(1:6,1:6);
m_iv2(val_ln1:val_ln1+5,val_ln1:val_ln1+5)=m_ele_cpvp(7:12,7:12);
m_iv2(val_ln1:val_ln1+5,mid_node:mid_node+5)=m_ele_cpvp(7:12,1:6);
m_iv2(mid_node:mid_node+5,val_ln1:val_ln1+5)=m_ele_cpvp(1:6,7:12);

k_iv2(mid_node:mid_node+5,mid_node:mid_node+5)=k_ele_cpvp(1:6,1:6);
k_iv2(val_ln1:val_ln1+5,val_ln1:val_ln1+5)=k_ele_cpvp(7:12,7:12);
k_iv2(val_ln1:val_ln1+5,mid_node:mid_node+5)=k_ele_cpvp(7:12,1:6);
k_iv2(mid_node:mid_node+5,val_ln1:val_ln1+5)=k_ele_cpvp(1:6,7:12);

m_iv=m_iv1+m_iv2;
k_iv=k_iv1+k_iv2;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Assebling the final matrix till cross pin verticle part

m_ultv=m_f2+m_i2+m_i3+m_iv;
k_ultv=k_f2+k_i2+k_i3+k_iv;

clear mib_int1 mib_int2 kib_int1 kib_int2 theta trans;
clear m_i1 m_i2 m_f1 m_f2 k_i1 k_i2 m_i3 k_i3 k_f1 k_f2 m_iv1 k_iv1 m_iv2 k_iv2 ;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% DRIVEN SHAFT AND YOKE PART %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Assembling the mass and element matrices for the output shaft
m_ele_os=mass_op_shaft(n_os,dia_os,den,l_os,ndof,E,G)
k_ele_os=stiff_op_shaft(n_os,dia_os,den,l_os,ndof,E,G)

% Stiffness matrix
K_os=zeros(dof,dof);
for kk=1:n_os
   intK_s=zeros(dof,dof);
```

```
    sn=n_is+2*n_y+1+2*n_cp+1+2*n_y;
    n_c=linspace(sn+1,sn+n_os+1,n_os+1);
                              % n_c=[36,37,38,39,40,41];
    r1=n_c(kk); r2=n_c(kk+1);
    A1=[6*(r1)-25,6*(r1)-24,6*(r1)-23,6*(r1)-22,6*(r1)-21,6*(r1)-20];
    B1=[6*(r2)-25,6*(r2)-24,6*(r2)-23,6*(r2)-22,6*(r2)-21,6*(r2)-20];
    for ii=1:6
        for jj=1:6
            intK_s(A1(ii),A1(jj))=k_ele_os(ii,jj);
            intK_s(B1(ii),B1(jj))=k_ele_os(6+ii,6+jj);
            intK_s(A1(ii),B1(jj))=k_ele_os(ii,6+jj);
            intK_s(B1(ii),A1(jj))=k_ele_os(6+ii,jj);
        end
    end

    K_os=K_os+intK_s;
end

% Mass matrix
M_os=zeros(dof,dof);
for kk=1:n_os
    intM_s=zeros(dof,dof);
    sn=n_is+2*n_y+1+2*n_cp+1+2*n_y;
    n_c=linspace(sn+1,sn+n_os+1,n_os+1);
    % n_c=[36,37,38,39,40,41];
    r1=n_c(kk); r2=n_c(kk+1);
    A1=[6*(r1)-25,6*(r1)-24,6*(r1)-23,6*(r1)-22,6*(r1)-21,6*(r1)-20];
    B1=[6*(r2)-25,6*(r2)-24,6*(r2)-23,6*(r2)-22,6*(r2)-21,6*(r2)-20];
    for ii=1:6
        for jj=1:6
            intM_s(A1(ii),A1(jj))=m_ele_os(ii,jj);
            intM_s(B1(ii),B1(jj))=m_ele_os(6+ii,6+jj);
            intM_s(A1(ii),B1(jj))=m_ele_os(ii,6+jj);
            intM_s(B1(ii),A1(jj))=m_ele_os(6+ii,jj);
        end
    end

    M_os=M_os+intM_s;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Assebling the shaft and yoke part of output side
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
trans=zeros(12,12);
theta=angle;
```

```
trans=trans_45_hz(theta)
k_ele_hp=trans'*k_ele2*trans;

trans=zeros(12,12);
theta=-angle;
trans=trans_45_hz(theta)
k_ele_hn=trans'*k_ele2*trans;

trans=zeros(12,12);
theta=angle;
trans=trans_45_hz(theta)
m_ele_hp=trans'*m_ele2*trans;

trans=zeros(12,12);
theta=-angle;
trans=trans_45_hz(theta)
m_ele_hn=trans'*m_ele2*trans;


K_p=zeros(dof,dof);
K_n=zeros(dof,dof);
M_p=zeros(dof,dof);
M_n=zeros(dof,dof);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Because of the complecated node numbring pattern the elements have been
% assembled in various seperate parts

% Assembling the first element of the positive part of the yoke
% Element 26-27 as in old mod

% Stiffness matrix

intK_p1=zeros(dof,dof);
num_n=n_is+2*n_y+1+2*n_cp+1;          % 25 as in old mod
num1=n_is+2*n_y+1+(n_cp/2)+1;         % 19 as in old mod
num2=num_n+1;                         % 26 as in old mod
A1=[6*num1+1-5,6*num1+2-5,6*num1+3-5,6*num1+4-5,6*num1+5-5,6*num_n+1-
10];
B1=[6*num2+1-15,6*num2+2-15,6*num2+3-15,6*num2+4-15,6*num2+5-
15,6*num2+6-15];
for ii=1:6
   for jj=1:6
      intK_p1(A1(ii),A1(jj))=k_ele_hp(ii,jj);
      intK_p1(B1(ii),B1(jj))=k_ele_hp(6+ii,6+jj);
```

```
      intK_p1(A1(ii),B1(jj))=k_ele_hp(ii,6+jj);
      intK_p1(B1(ii),A1(jj))=k_ele_hp(6+ii,jj);
   end
end


% Mass matrix

intM_p1=zeros(dof,dof);
for ii=1:6
   for jj=1:6
      intM_p1(A1(ii),A1(jj))=m_ele_hp(ii,jj);
      intM_p1(B1(ii),B1(jj))=m_ele_hp(6+ii,6+jj);
      intM_p1(A1(ii),B1(jj))=m_ele_hp(ii,6+jj);
      intM_p1(B1(ii),A1(jj))=m_ele_hp(6+ii,jj);
   end
end
clear A1 B1;

% Assembling the first element of the negative part of the yoke
% Element 28-29 as in old mod

% Stiffness matrix

intK_p2=zeros(dof,dof);
num_n=n_is+2*n_y+1+2*n_cp+1;          % 25 as in old mod
num1=n_is+2*n_y+1+3*(n_cp/2)+1;       % 23 as in old mod
num2=num_n+3;                         % 28 as in old mod
A1=[6*(num1-1)+1-5,6*(num1-1)+2-5,6*(num1-1)+3-5,6*(num1-1)+4-5,6*(num1-1)+5-
5,6*(num2-1)+1-15];
% B1=[6*num2+1-15,6*num2+2-15,6*num2+3-15,6*num2+4-15,6*num2+5-
15,6*num2+6-15];
B1=[6*num2+1-20,6*num2+2-20,6*num2+3-20,6*num2+4-20,6*num2+5-
20,6*num2+6-20];
for ii=1:6
   for jj=1:6
      intK_p2(A1(ii),A1(jj))=k_ele_hp(ii,jj);
      intK_p2(B1(ii),B1(jj))=k_ele_hp(6+ii,6+jj);
      intK_p2(A1(ii),B1(jj))=k_ele_hp(ii,6+jj);
      intK_p2(B1(ii),A1(jj))=k_ele_hp(6+ii,jj);
   end
end

% Mass matrix

intM_p2=zeros(dof,dof);
```

```
for ii=1:6
   for jj=1:6
      intM_p2(A1(ii),A1(jj))=m_ele_hp(ii,jj);
      intM_p2(B1(ii),B1(jj))=m_ele_hp(6+ii,6+jj);
      intM_p2(A1(ii),B1(jj))=m_ele_hp(ii,6+jj);
      intM_p2(B1(ii),A1(jj))=m_ele_hp(6+ii,jj);
   end
end
clear A1 B1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Assembling the stiffness matrix for the remaining elements of the yokes

% Assembling the stiffness matrices for positive part of I/P yoke
% Elements 27-36
for kk=1:n_y-1
   intK_p3=zeros(dof,dof);
   num=n_is+2*n_y+1+2*n_cp+1;   % 25 as in old mod
   n_c=zeros(1,n_y);
   n_c(1,1)=num+2;
   n_c(1,2:n_y)=linspace(num+5,num+(2*n_y+1),n_y+1-2);
                        %n_c=[27,30,32,34,36];
   r1=n_c(kk); r2=n_c(kk+1);

   if (kk==1)
   A1=[6*(r1-1)+1-15,6*(r1-1)+2-15,6*(r1-1)+3-15,6*(r1-1)+4-15,6*(r1-1)+5-15,6*(r1-
1)+6-15];
   B1=[6*(r2-1)+1-20,6*(r2-1)+2-20,6*(r2-1)+3-20,6*(r2-1)+4-20,6*(r2-1)+5-20,6*(r2-
1)+6-20];

   else
   A1=[6*(r1-1)+1-20,6*(r1-1)+2-20,6*(r1-1)+3-20,6*(r1-1)+4-20,6*(r1-1)+5-20,6*(r1-
1)+6-20];
   B1=[6*(r2-1)+1-20,6*(r2-1)+2-20,6*(r2-1)+3-20,6*(r2-1)+4-20,6*(r2-1)+5-20,6*(r2-
1)+6-20];
   end

   for ii=1:6
      for jj=1:6
         intK_p3(A1(ii),A1(jj))=k_ele_hp(ii,jj);
         intK_p3(B1(ii),B1(jj))=k_ele_hp(6+ii,6+jj);
         intK_p3(A1(ii),B1(jj))=k_ele_hp(ii,6+jj);
         intK_p3(B1(ii),A1(jj))=k_ele_hp(6+ii,jj);
      end
   end
```

```
   K_p=K_p+intK_p3;
end
clear A1 B1;


% Assembling the mass matrices for positive part of I/P yoke
% Elements 27-36
for kk=1:n_y-1
   intM_p3=zeros(dof,dof);
   num=n_is+2*n_y+1+2*n_cp+1;
   n_c=zeros(1,n_y);
   n_c(1,1)=num+2;
   n_c(1,2:n_y)=linspace(num+5,num+(2*n_y+1),n_y+1-2);
   %n_c=[27,30,32,34,36];
   r1=n_c(kk); r2=n_c(kk+1);

   if (kk==1)
   A1=[6*(r1-1)+1-15,6*(r1-1)+2-15,6*(r1-1)+3-15,6*(r1-1)+4-15,6*(r1-1)+5-15,6*(r1-
1)+6-15];
   B1=[6*(r2-1)+1-20,6*(r2-1)+2-20,6*(r2-1)+3-20,6*(r2-1)+4-20,6*(r2-1)+5-20,6*(r2-
1)+6-20];

   else
   A1=[6*(r1-1)+1-20,6*(r1-1)+2-20,6*(r1-1)+3-20,6*(r1-1)+4-20,6*(r1-1)+5-20,6*(r1-
1)+6-20];
   B1=[6*(r2-1)+1-20,6*(r2-1)+2-20,6*(r2-1)+3-20,6*(r2-1)+4-20,6*(r2-1)+5-20,6*(r2-
1)+6-20];
   end

   for ii=1:6
      for jj=1:6
         intM_p3(A1(ii),A1(jj))=m_ele_hp(ii,jj);
         intM_p3(B1(ii),B1(jj))=m_ele_hp(6+ii,6+jj);
         intM_p3(A1(ii),B1(jj))=m_ele_hp(ii,6+jj);
         intM_p3(B1(ii),A1(jj))=m_ele_hp(6+ii,jj);
      end
   end
   M_p=M_p+intM_p3;
end

% Assembling the stiffness matrices for negative part of I/P yoke
% Elements 29-36
for kk=1:n_y-1
   intK_n4=zeros(dof,dof);
   n_c=zeros(1,n_y);
   n_c(1,1:n_y-1)=linspace(num+4,num+2*n_y,n_y-1);
```

```
   n_c(1,n_y)=num+2*n_y+1;
                          %n_c=[6,7,9,11,13,15];
   r1=n_c(kk); r2=n_c(kk+1);
   A1=[6*(r1-1)+1-20,6*(r1-1)+2-20,6*(r1-1)+3-20,6*(r1-1)+4-20,6*(r1-1)+5-20,6*(r1-
1)+6-20];
   B1=[6*(r2-1)+1-20,6*(r2-1)+2-20,6*(r2-1)+3-20,6*(r2-1)+4-20,6*(r2-1)+5-20,6*(r2-
1)+6-20];

   for ii=1:6
      for jj=1:6
         intK_n4(A1(ii),A1(jj))=k_ele_hn(ii,jj);
         intK_n4(B1(ii),B1(jj))=k_ele_hn(6+ii,6+jj);
         intK_n4(A1(ii),B1(jj))=k_ele_hn(ii,6+jj);
         intK_n4(B1(ii),A1(jj))=k_ele_hn(6+ii,jj);
      end
   end
   K_n=K_n+intK_n4;
end
clear A1 B1;

% Assembling the Mass matrices for negative part of I/P yoke
% Elements 29-36

for kk=1:n_y-1
   intM_n4=zeros(dof,dof);
   n_c=zeros(1,n_y);
   n_c(1,1:n_y-1)=linspace(num+4,num+2*n_y,n_y-1);
   n_c(1,n_y)=num+2*n_y+1;

   r1=n_c(kk); r2=n_c(kk+1);
   A1=[6*(r1-1)+1-20,6*(r1-1)+2-20,6*(r1-1)+3-20,6*(r1-1)+4-20,6*(r1-1)+5-20,6*(r1-
1)+6-20];
   B1=[6*(r2-1)+1-20,6*(r2-1)+2-20,6*(r2-1)+3-20,6*(r2-1)+4-20,6*(r2-1)+5-20,6*(r2-
1)+6-20];

   for ii=1:6
      for jj=1:6
         intM_n4(A1(ii),A1(jj))=m_ele_hn(ii,jj);
         intM_n4(B1(ii),B1(jj))=m_ele_hn(6+ii,6+jj);
         intM_n4(A1(ii),B1(jj))=m_ele_hn(ii,6+jj);
         intM_n4(B1(ii),A1(jj))=m_ele_hn(6+ii,jj);
      end
   end
   M_n=M_n+intM_n4;
end
```

```
% Assembling the complete mass and stiffness matrix for output part
M_finy2=M_os+M_p+M_n+intM_p1+intM_p2;
K_finy2=K_os+K_p+K_n+intK_p1+intK_p2;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Assembling the cross pin of the yoke # 2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Calculating the transformation matrices

trans=zeros(12,12);
theta=-90;
trans=trans_45_cph(theta);
k_ele_cphp=trans'*k_ele_cp*trans;

trans=zeros(12,12);
theta=90;
trans=trans_45_cph(theta);
k_ele_cphn=trans'*k_ele_cp*trans;

trans=zeros(12,12);
theta=-90;
trans=trans_45_cph(theta);
m_ele_cphp=trans'*m_ele_cp*trans;

trans=zeros(12,12);
theta=90;
trans=trans_45_cph(theta);
m_ele_cphn=trans'*m_ele_cp*trans;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Assembling the cross pin with the global input shaft and yoke matrix
% Assembling as per the new numbring system
% Assembling the left part of the cross pin ( 17-18 as per old_mod)

% Mass matrix
mib_inth1=zeros(dof,dof);
p4=n_is+2*n_y+1+(n_cp/2)+1;     % 19 as in old mod
A1=[6*(p4)+1-5,6*(p4)+2-5,6*(p4)+3-5,6*(p4)+4-5,6*(p4)+5-5,6*(p4)+6-5];
```

```
                        % Node 20: A1=[110,111,112,113,114,115];
B1=[6*(p4+1)+1-5,6*(p4+1)+2-5,6*(p4+1)+3-5,6*(p4+1)+4-5,6*(p4+1)+5-
5,6*(p4+1)+6-5];
                        % Node 21: B1=[116,117,118,119,120,121];
for ii=1:6
   for jj=1:6
      mib_inth1(A1(ii),A1(jj))=m_ele_cphn(ii,jj);
      mib_inth1(B1(ii),B1(jj))=m_ele_cphn(6+ii,6+jj);
      mib_inth1(A1(ii),B1(jj))=m_ele_cphn(ii,6+jj);
      mib_inth1(B1(ii),A1(jj))=m_ele_cphn(6+ii,jj);
   end
end

% Stiffness matrix
kib_inth1=zeros(dof,dof);
p4=n_is+2*n_y+1+(n_cp/2)+1;     % 19 as in old mod
A1=[6*(p4)+1-5,6*(p4)+2-5,6*(p4)+3-5,6*(p4)+4-5,6*(p4)+5-5,6*(p4)+6-5];
                        % Node 20: A1=[110,111,112,113,114,115];
B1=[6*(p4+1)+1-5,6*(p4+1)+2-5,6*(p4+1)+3-5,6*(p4+1)+4-5,6*(p4+1)+5-
5,6*(p4+1)+6-5];
                        % Node 21: B1=[116,117,118,119,120,121];
for ii=1:6
   for jj=1:6
      kib_inth1(A1(ii),A1(jj))=k_ele_cphn(ii,jj);
      kib_inth1(B1(ii),B1(jj))=k_ele_cphn(6+ii,6+jj);
      kib_inth1(A1(ii),B1(jj))=k_ele_cphn(ii,6+jj);
      kib_inth1(B1(ii),A1(jj))=k_ele_cphn(6+ii,jj);
   end
end

% Defining the final matrix till cross pin verticle part
m_fh1=M_finy2+mib_inth1;
k_fh1=K_finy2+kib_inth1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Right joint of the cross pin
% Mass matrix
p5=p4+n_cp;                     % 23 as in old mod
mib_inth2=zeros(dof,dof);
A1=[6*(p5-2)+1-5,6*(p5-2)+2-5,6*(p5-2)+3-5,6*(p5-2)+4-5,6*(p5-2)+5-5,6*(p5-2)+6-
5];
                        % A1=[134,135,136,137,138,139];
```

```
B1=[6*(p5-1)+1-5,6*(p5-1)+2-5,6*(p5-1)+3-5,6*(p5-1)+4-5,6*(p5-1)+5-5,6*(p5-1)+6-
5];
                        % B1=[140,141,142,143,144,145];
for ii=1:6
   for jj=1:6
      mib_inth2(A1(ii),A1(jj))=m_ele_cphn(ii,jj);
      mib_inth2(B1(ii),B1(jj))=m_ele_cphn(6+ii,6+jj);
      mib_inth2(A1(ii),B1(jj))=m_ele_cphn(ii,6+jj);
      mib_inth2(B1(ii),A1(jj))=m_ele_cphn(6+ii,jj);
   end
end

% Stiffness matrix
p5=p4+n_cp;                    % 23 as in old mod
kib_inth2=zeros(dof,dof);
A1=[6*(p5-2)+1-5,6*(p5-2)+2-5,6*(p5-2)+3-5,6*(p5-2)+4-5,6*(p5-2)+5-5,6*(p5-2)+6-
5];
                        % A1=[134,135,136,137,138,139];
B1=[6*(p5-1)+1-5,6*(p5-1)+2-5,6*(p5-1)+3-5,6*(p5-1)+4-5,6*(p5-1)+5-5,6*(p5-1)+6-
5];
                        % B1=[140,141,142,143,144,145];
for ii=1:6
   for jj=1:6
      kib_inth2(A1(ii),A1(jj))=k_ele_cphn(ii,jj);
      kib_inth2(B1(ii),B1(jj))=k_ele_cphn(6+ii,6+jj);
      kib_inth2(A1(ii),B1(jj))=k_ele_cphn(ii,6+jj);
      kib_inth2(B1(ii),A1(jj))=k_ele_cphn(6+ii,jj);
   end
end
m_fh2=m_fh1+mib_inth2;
k_fh2=k_fh1+kib_inth2;

% other three elements
% assembling the element 17-18-19-20-21 to the global system
% Assembling the reamining elements to the globle matrix
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%
%$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$
% assembling the element 17-18-19-20-21 to the global system
m_ih4=zeros(dof,dof);m_ih5=zeros(dof,dof);
k_ih4=zeros(dof,dof);k_ih5=zeros(dof,dof);

% Assembling the reamining elements to the globle matrix

% Assembling the left part
```

```
for ii=1:(n_cp-4)/2
   m_ih4=zeros(dof,dof);
   k_ih4=zeros(dof,dof);
   p6=p4+1;          % 20 as in old mod
   m_ih4(6*(p6)+1+(6*(ii-1))-5:6*(p6)+12+(6*(ii-1))-5,6*(p6)+1+(6*(ii-1))-
5:6*(p6)+12+(6*(ii-1))-5)=m_ele_cphn(1:12,1:12);
   k_ih4(6*(p6)+1+(6*(ii-1))-5:6*(p6)+12+(6*(ii-1))-5,6*(p6)+1+(6*(ii-1))-
5:6*(p6)+12+(6*(ii-1))-5)=k_ele_cphn(1:12,1:12);
   %val_ln=6*(p2)-2+(6*(ii-1))+6;
   m_ih5=m_ih5+m_ih4;
   k_ih5=k_ih5+k_ih4;
end

% Assembling the right part
p7=p4+n_cp/2;          % 21 as in old mod
m_ih4=zeros(dof,dof);m_ih6=zeros(dof,dof);
k_ih4=zeros(dof,dof);k_ih6=zeros(dof,dof);
for ii=1:(n_cp-4)/2
   m_ih4=zeros(dof,dof);
   k_ih4=zeros(dof,dof);
   m_ih4(6*(p7)+1+(6*(ii-1))-5:6*(p7)+12+(6*(ii-1))-5,6*(p7)+1+(6*(ii-1))-
5:6*(p7)+12+(6*(ii-1))-5)=m_ele_cphn(1:12,1:12);
   k_ih4(6*(p7)+1+(6*(ii-1))-5:6*(p7)+12+(6*(ii-1))-5,6*(p7)+1+(6*(ii-1))-
5:6*(p7)+12+(6*(ii-1))-5)=k_ele_cphn(1:12,1:12);
   %val_ln=6*(p3-1)+1-15;
   m_ih6=m_ih6+m_ih4;
   k_ih6=k_ih6+k_ih4;
end

% Center elements
% 21-19 as in old mod

mn=n_is+2*n_y+(n_cp/2)+2;      % 19 as in old mod( Mid node)
mid_node=6*(mn-1)+1-5;
val=p4+(n_cp/2)-1;             % 21 as in old mod
val_ln=6*val+1-5;
m_ih8=zeros(dof,dof);m_ih9=zeros(dof,dof);
k_ih8=zeros(dof,dof);k_ih9=zeros(dof,dof);
m_ih8(mid_node:mid_node+5,mid_node:mid_node+5)=m_ele_cphn(1:6,1:6);
m_ih8(val_ln:val_ln+5,val_ln:val_ln+5)=m_ele_cphn(7:12,7:12);
m_ih8(mid_node:mid_node+5,val_ln:val_ln+5)=m_ele_cphn(1:6,7:12);
m_ih8(val_ln:val_ln+5,mid_node:mid_node+5)=m_ele_cphn(7:12,1:6);

k_ih8(mid_node:mid_node+5,mid_node:mid_node+5)=k_ele_cphn(1:6,1:6);
k_ih8(val_ln:val_ln+5,val_ln:val_ln+5)=k_ele_cphn(7:12,7:12);
k_ih8(mid_node:mid_node+5,val_ln:val_ln+5)=k_ele_cphn(1:6,7:12);
```

```
k_ih8(val_ln:val_ln+5,mid_node:mid_node+5)=k_ele_cphn(7:12,1:6);


%19-24
val=p4+(n_cp/2);
val_ln=6*val+1-5;
m_ih9(mid_node:mid_node+5,mid_node:mid_node+5)=m_ele_cphn(1:6,1:6);
m_ih9(val_ln:val_ln+5,val_ln:val_ln+5)=m_ele_cphn(7:12,7:12);
m_ih9(val_ln:val_ln+5,mid_node:mid_node+5)=m_ele_cphn(7:12,1:6);
m_ih9(mid_node:mid_node+5,val_ln:val_ln+5)=m_ele_cphn(1:6,7:12);

k_ih9(mid_node:mid_node+5,mid_node:mid_node+5)=k_ele_cphn(1:6,1:6);
k_ih9(val_ln:val_ln+5,val_ln:val_ln+5)=k_ele_cphn(7:12,7:12);
k_ih9(val_ln:val_ln+5,mid_node:mid_node+5)=k_ele_cphn(7:12,1:6);
k_ih9(mid_node:mid_node+5,val_ln:val_ln+5)=k_ele_cphn(1:6,7:12);

m_ih=m_ih8+m_ih9;
k_ih=k_ih8+k_ih9;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

m_ulth=m_fh2+m_ih+m_ih5+m_ih6;
k_ulth=k_fh2+k_ih+k_ih5+k_ih6;
%**************************************************************************
****
m_ult=m_ulth+m_ultv;
k_ult=k_ulth+k_ultv;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Natural Frequency Calculations
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Predefining the mass and stiffness matrices
M_mod=zeros(dof-6,dof-6);
K_mod=zeros(dof-6,dof-6);
M_mod=m_ult(7:dof,7:dof);
K_mod=k_ult(7:dof,7:dof);

mass=M_mod;
stiff=K_mod;
```

```
damp=zeros(size(mass));              % Damping matrix
null=zeros(size(mass));

% Form 2N x 2N state space equation.
a=[null,mass;mass,damp];
b=[-mass,null;null,stiff];
[x,d]=eig(b,-a);
dd=dof-6;
% Sort Modal Frequencies
orig_lambda=diag(d);
[Y,I]=sort(imag(orig_lambda));
lambda=orig_lambda(I);
for ii=1:2*size(mass)
   lambda_diag(ii,ii)=lambda(ii);
end
xx=x(:,I);
pp=sort(orig_lambda)/(2*pi);
%     Normalize x matrix to real vectors if possible
for ii=1:2*dd
   xx(1:2*dd,ii)=xx(1:2*dd,ii)./xx(dd+1,ii);
end
%     Compute 'modal a' and 'modal b' matrix
ma=xx.'*a*xx;
mb=xx.'*b*xx;
%     Extract modal vectors from state-space formulation

for ii=1:2*dd
   psi(1:dd,ii)=xx(dd+1:2*dd,ii);
end

%Generating the FRF
f=linspace(0,200,1000);
for ii=1:1000
   B=-M_mod*(f(ii)*2*pi)*(f(ii)*2*pi)+K_mod;
   B_mod=B(7:dd,7:dd);
   H=inv(B_mod);
   k1(1,ii)=H(1,1);
   k2(1,ii)=H(2,2);
   k3(1,ii)=H(3,3);
   k4(1,ii)=H(4,4);
   k5(1,ii)=H(5,5);
   k6(1,ii)=H(6,6);
end
% figure(1)
% spy(K_mod)
```

```
% figure(2)
% spy(stiff_mat_an)
% figure(3)
% spy(M_mod)
% figure(4)
% spy(mass_mat_an)
kkkk=sort(imag(pp));
natural_frequency=kkkk;
figure(1)
semilogy(f,abs(k1))
title('Frequency response function')
xlabel('Frequency')
ylabel('Log Magnitude')

figure(2)
semilogy(f,abs(k2))
title('Frequency response function')
xlabel('Frequency')
ylabel('Log Magnitude')

figure(3)
semilogy(f,abs(k3));
title('Frequency response function')
xlabel('Frequency')
ylabel('Log Magnitude')
```

B.2.2 Mass matrix

```
function [m_ele2]=mass_beam(n_y,a,b,den,ndof,E,G,l_y)
n=n_y;
l=l_y;
% Defining the mass per element of the beam
A=a*b;
m=den*A*l;
I_z=a*b^3/12;      % Second area moment of the beam
I_y=a^3*b/12;      % Second area of moment of the beam
j1=I_z+I_y;
% Defining the mass matrix for the inclined beam ( Yoke Input)

% m_ele=zeros(ndof,ndof);
% m_ele(:,1)=[m/3;0;0;0;0;0;m/6;0;0;0;0;0];
% m_ele(1,:)=[m/3,0,0,0,0,0,m/6,0,0,0,0,0];
%
% m_ele(2:12,2)=[(6*den*I_z*l/(5*l)+(13*m/35));0;0;0;(I_z*den/10)+(11*m*l/210);0;...
%              -(6*den*I_z*l/(5*l))+(9*m/70);0;0;0;(I_z*den/10)-(13*m*l/420)];
% m_ele(2,2:12)=[(6*den*I_z*l/(5*l)+(13*m/35)),0,0,0,(I_z*den/10)+(11*m*l/210),0,...
%              -(6*den*I_z*l/(5*l))+(9*m/70),0,0,0,(I_z*den/10)-(13*m*l/420)];
%
% m_ele(3:12,3)=[(6*den*I_y*l/(5*l)+(13*m/35));0;-(I_y*den/10)-
(11*m*l/210);0;0;0;...
%              -(6*den*I_y*l/(5*l))+(9*m/70);0;-(I_y*den/10)+(13*m*l/420);0];
% m_ele(3,3:12)=[(6*den*I_y*l/(5*l)+(13*m/35)),0,(-I_y*den/10)-(11*m*l/210),0,0,0,...
%              -(6*den*I_y*l/(5*l))+(9*m/70),0,-(I_y*den/10)+(13*m*l/420),0];
%
% m_ele(4:12,4)=[((1/3)*den*l*(j1));0;0;0;0;0;((1/6)*den*l*(j1));0;0];
% m_ele(4,4:12)=[((1/3)*den*l*(j1)),0,0,0,0,0,((1/6)*den*l*(j1)),0,0];
%
% m_ele(5:12,5)=[(2*I_y*l*den/15)+(m*l*l/105);0;0;0;...
%              (I_y*den/10)-(13*m*l/420);0;(-I_y*l*den/30)-(m*l*l/140);0];
% m_ele(5,5:12)=[(2*I_y*l*den/15)+(m*l*l/105),0,0,0,...
%             (I_y*den/10)-(13*m*l/420),0,(-I_y*l*den/30)-(m*l*l/140),0];
%
% m_ele(6:12,6)=[(2*I_z*l*den/15)+(m*l*l/105);0;(-I_z*den/10)+(13*m*l/420)...
%              ;0;0;0;(-I_z*l*den/30)-(m*l*l/140)];
% m_ele(6,6:12)=[(2*I_z*l*den/15)+(m*l*l/105),0,(-I_z*den/10)+(13*m*l/420)...
%              ,0,0,0,(-I_z*l*den/30)-(m*l*l/140)];
%
% m_ele(7:12,7)=[m/3;0;0;0;0;0];
% m_ele(7,7:12)=[m/3,0,0,0,0,0];
%
% m_ele(8:12,8)=[(6*I_z*den/(5*l))+(13*m/35);0;0;0;(-I_z*den/10)-(11*m*l/210)];
% m_ele(8,8:12)=[(6*I_z*den/(5*l))+(13*m/35),0,0,0,(-I_z*den/10)-(11*m*l/210)];
```

```
%
% m_ele(9:12,9)=[(6*I_y*den/(5*l)+(13*m/35));0;(I_y*den/10)+(11*m*l/210);0];
% m_ele(9,9:12)=[(6*I_y*den/(5*l)+(13*m/35)),0,(I_y*den/10)+(11*m*l/210),0];
%
% m_ele(10:12,10)=[(l*den/3)*(j1);0;0];
% m_ele(10,10:12)=[(l*den/3)*(j1),0,0];
%
% m_ele(11:12,11)=[(2*I_y*l*den/15)+(m*l*l/105);0];
% m_ele(11,11:12)=[(2*I_y*l*den/15)+(m*l*l/105),0];
%
% m_ele(12,12)=[(2*I_z*l*den/15)+(m*l*l/105)];
% m_ele2=m_ele;

m_ele2=zeros(ndof,ndof);
m_ele2(:,1)=[m/3;0;0;0;0;0;m/6;0;0;0;0;0];
m_ele2(1,:)=[m/3,0,0,0,0,0,m/6,0,0,0,0,0];

m_ele2(2:12,2)=[(6*den*I_z*l/(5*l)+(13*m/35));0;0;0;(I_z*den/10)+(11*m*l/210);0;...
          -(6*den*I_z*l/(5*l))+(9*m/70);0;0;0;(I_z*den/10)-(13*m*l/420)];
m_ele2(2,2:12)=[(6*den*I_z*l/(5*l)+(13*m/35)),0,0,0,(I_z*den/10)+(11*m*l/210),0,...
          -(6*den*I_z*l/(5*l))+(9*m/70),0,0,0,(I_z*den/10)-(13*m*l/420)];

m_ele2(3:12,3)=[(6*den*I_y*l/(5*l)+(13*m/35));0;-(I_y*den/10)-(11*m*l/210);0;0;0;...
          -(6*den*I_y*l/(5*l))+(9*m/70);0;-(I_y*den/10)+(13*m*l/420);0];
m_ele2(3,3:12)=[(6*den*I_y*l/(5*l)+(13*m/35)),0,(-I_y*l*l/10)-(11*m*l/210),0,0,0,...
          -(6*den*I_y*l/(5*l))+(9*m/70),0,-(I_y*den/10)+(13*m*l/420),0];

m_ele2(4:12,4)=[((1/3)*den*l*(j1));0;0;0;0;0;((1/6)*den*l*(j1));0;0];
m_ele2(4,4:12)=[((1/3)*den*l*(j1)),0,0,0,0,0,((1/6)*den*l*(j1)),0,0];

m_ele2(5:12,5)=[(2*I_y*l*den/15)+(m*l*l/105);0;0;0;...
          (I_y*den/10)-(13*m*l/420);0;(-I_y*l*den/30)-(m*l*l/140);0];
m_ele2(5,5:12)=[(2*I_y*l*den/15)+(m*l*l/105),0,0,0,...
          (I_y*den/10)-(13*m*l/420),0,(-I_y*l*den/30)-(m*l*l/140),0];

m_ele2(6:12,6)=[(2*I_z*l*den/15)+(m*l*l/105);0;(-I_z*den/10)+(13*m*l/420)...
          ;0;0;0;(-I_z*l*den/30)-(m*l*l/140)];
m_ele2(6,6:12)=[(2*I_z*l*den/15)+(m*l*l/105),0,(-I_z*den/10)+(13*m*l/420)...
          ,0,0,0,(-I_z*l*den/30)-(m*l*l/140)];

m_ele2(7:12,7)=[m/3;0;0;0;0;0];
m_ele2(7,7:12)=[m/3,0,0,0,0,0];

m_ele2(8:12,8)=[(6*I_z*den/(5*l))+(13*m/35);0;0;0;(-I_z*den/10)-(11*m*l/210)];
m_ele2(8,8:12)=[(6*I_z*den/(5*l))+(13*m/35),0,0,0,(-I_z*den/10)-(11*m*l/210)];
```

```
m_ele2(9:12,9)=[(6*I_y*den/(5*l)+(13*m/35));0;(I_y*den/10)+(11*m*l/210);0];
m_ele2(9,9:12)=[(6*I_y*den/(5*l)+(13*m/35)),0,(I_y*den/10)+(11*m*l/210),0];

m_ele2(10:12,10)=[(l*den/3)*(j1);0;0];
m_ele2(10,10:12)=[(l*den/3)*(j1),0,0];

m_ele2(11:12,11)=[(2*I_y*l*den/15)+(m*l*l/105);0];
m_ele2(11,11:12)=[(2*I_y*l*den/15)+(m*l*l/105),0];

m_ele2(12,12)=[(2*I_z*l*den/15)+(m*l*l/105)];
```

B.2.3. Stiffness matrix

```
function [k_ele2]=stiff_beam(n_y,a,b,den,ndof,E,G,l_y)
n=n_y;
l=l_y;
A=a*b;
% Defining the mass per element of the beam
m=den*A*l;
I_z=a*b^3/12;      % Second area moment of the beam
I_y=a^3*b/12;      % Second area of moment of the beam
j1=I_z+I_y;
% Defining the stiffness matrix for the inclined beam element
k_ele2=zeros(ndof,ndof);

K2=zeros(6*(n+1),6*(n+1));

k_ele2(1:12,1)=(E*A/l)*[1;0;0;0;0;0;-1;0;0;0;0;0];
k_ele2(1,1:12)=(E*A/l)*[1,0,0,0,0,0,-1,0,0,0,0,0];

k_ele2(2:12,2)=((6*E*I_z)/(l^3))*[2;0;0;0;l;0;-2;0;0;0;l];
k_ele2(2,2:12)=((6*E*I_z)/(l^3))*[2,0,0,0,l,0,-2,0,0,0,l];

k_ele2(3:12,3)=(6*E*I_y/(l*l*l))*[2;0;-l;0;0;0;-2;0;-l;0];
k_ele2(3,3:12)=(6*E*I_y/(l*l*l))*[2,0,-l,0,0,0,-2,0,-l,0];

k_ele2(4:12,4)=(G*j1/l)*[1;0;0;0;0;0;-1;0;0];
k_ele2(4,4:12)=(G*j1/l)*[1,0,0,0,0,0,-1,0,0];

k_ele2(5:12,5)=(E*I_y/(l*l))*[4*l;0;0;0;6;0;2*l;0];
k_ele2(5,5:12)=(E*I_y/(l*l))*[4*l,0,0,0,6,0,2*l,0];

k_ele2(6:12,6)=(E*I_z/(l*l))*[4*l;0;-6;0;0;0;2*l];
k_ele2(6,6:12)=(E*I_z/(l*l))*[4*l,0,-6,0,0,0,2*l];

k_ele2(7:12,7)=(A*E/l)*[1;0;0;0;0;0];
k_ele2(7,7:12)=(A*E/l)*[1,0,0,0,0,0];

k_ele2(8:12,8)=(6*E*I_z/(l*l*l))*[2;0;0;0;-l];
k_ele2(8,8:12)=(6*E*I_z/(l*l*l))*[2,0,0,0,-l];

k_ele2(9:12,9)=(6*E*I_y/(l*l*l))*[2;0;l;0];
k_ele2(9,9:12)=(6*E*I_y/(l*l*l))*[2,0,l,0];

k_ele2(10:12,10)=(G*j1/l)*[1;0;0];
k_ele2(10,10:12)=(G*j1/l)*[1,0,0];
```

```
k_ele2(11:12,11)=(4*E*I_y/l)*[1;0];
k_ele_2(11,11:12)=(4*E*I_y/l)*[1,0];

k_ele2(12,12)=(4*E*I_z/l);
```

B.2.4 Mass cross pin

```
function [m_ele_cp]=mass_cp(n_cp,dia_cp,den_cp,ndof,E_cp,G_cp,l_cp)
d=dia_cp;
n=n_cp;
den=den_cp;
E=E_cp;
G=G_cp;
l=l_cp;

A=pi*d^2/4;
m=den*A*l;
I_y=pi*d^4/64;
I_z=pi*d^4/64;
j1=pi*d^4/32;
% Defining the mass matrix for the inclined beam ( Yoke Input)
% m_ele=zeros(ndof,ndof);
% m_ele(:,1)=[m/3;0;0;0;0;0;m/6;0;0;0;0;0];
% m_ele(1,:)=[m/3,0,0,0,0,0,m/6,0,0,0,0,0];
%
% m_ele(2:12,2)=[(6*den*I_z*l/(5*l)+(13*m/35));0;0;0;(I_z*den/10)+(11*m*l/210);0;...
%             -(6*den*I_z*l/(5*l))+(9*m/70);0;0;0;(I_z*den/10)-(13*m*l/420)];
% m_ele(2,2:12)=[(6*den*I_z*l/(5*l)+(13*m/35)),0,0,0,(I_z*den/10)+(11*m*l/210),0,...
%             -(6*den*I_z*l/(5*l))+(9*m/70),0,0,0,(I_z*den/10)-(13*m*l/420)];
%
% m_ele(3:12,3)=[(6*den*I_y*l/(5*l)+(13*m/35));0;-(I_y*den/10)-
(11*m*l/210);0;0;0;...
%             -(6*den*I_y*l/(5*l))+(9*m/70);0;-(I_y*den/10)+(13*m*l/420);0];
% m_ele(3,3:12)=[(6*den*I_y*l/(5*l)+(13*m/35)),0,(-I_y*den/10)-(11*m*l/210),0,0,0,...
%             -(6*den*I_y*l/(5*l))+(9*m/70),0,-(I_y*den/10)+(13*m*l/420),0];
%
% m_ele(4:12,4)=[((1/3)*den*l*(j1));0;0;0;0;0;((1/6)*den*l*(j1));0;0];
% m_ele(4,4:12)=[((1/3)*den*l*(j1)),0,0,0,0,0,((1/6)*den*l*(j1)),0,0];
%
% m_ele(5:12,5)=[(2*I_y*l*den/15)+(m*l*l/105);0;0;0;...
%             (I_y*den/10)-(13*m*l/420);0;(-I_y*l*den/30)-(m*l*l/140);0];
% m_ele(5,5:12)=[(2*I_y*l*den/15)+(m*l*l/105),0,0,0,...
%            (I_y*den/10)-(13*m*l/420),0,(-I_y*l*den/30)-(m*l*l/140),0];
%
% m_ele(6:12,6)=[(2*I_z*l*den/15)+(m*l*l/105);0;(-I_z*den/10)+(13*m*l/420)...
%            ;0;0;0;(-I_z*l*den/30)-(m*l*l/140)];
% m_ele(6,6:12)=[(2*I_z*l*den/15)+(m*l*l/105),0,(-I_z*den/10)+(13*m*l/420)...
%            ,0,0,0,(-I_z*l*den/30)-(m*l*l/140)];
%
% m_ele(7:12,7)=[m/3;0;0;0;0;0];
% m_ele(7,7:12)=[m/3,0,0,0,0,0];
```

```
%
% m_ele(8:12,8)=[(6*I_z*den/(5*l))+(13*m/35);0;0;0;(-I_z*den/10)-(11*m*l/210)];
% m_ele(8,8:12)=[(6*I_z*den/(5*l))+(13*m/35),0,0,0,(-I_z*den/10)-(11*m*l/210)];
%
% m_ele(9:12,9)=[(6*I_y*den/(5*l)+(13*m/35));0;(I_y*den/10)+(11*m*l/210);0];
% m_ele(9,9:12)=[(6*I_y*den/(5*l)+(13*m/35)),0,(I_y*den/10)+(11*m*l/210),0];
%
% m_ele(10:12,10)=[(l*den/3)*(j1);0;0];
% m_ele(10,10:12)=[(l*den/3)*(j1),0,0];
%
% m_ele(11:12,11)=[(2*I_y*l*den/15)+(m*l*l/105);0];
% m_ele(11,11:12)=[(2*I_y*l*den/15)+(m*l*l/105),0];
%
% m_ele(12,12)=[(2*I_z*l*den/15)+(m*l*l/105)];
% m_ele_cp=m_ele;


m_ele2=zeros(ndof,ndof);
m_ele2(:,1)=[m/3;0;0;0;0;0;m/6;0;0;0;0;0];
m_ele2(1,:)=[m/3,0,0,0,0,0,m/6,0,0,0,0,0];

m_ele2(2:12,2)=[(6*den*I_z*l/(5*l)+(13*m/35));0;0;0;(I_z*den/10)+(11*m*l/210);0;...
          -(6*den*I_z*l/(5*l))+(9*m/70);0;0;0;(I_z*den/10)-(13*m*l/420)];
m_ele2(2,2:12)=[(6*den*I_z*l/(5*l)+(13*m/35)),0,0,0,(I_z*den/10)+(11*m*l/210),0,...
          -(6*den*I_z*l/(5*l))+(9*m/70),0,0,0,(I_z*den/10)-(13*m*l/420)];

m_ele2(3:12,3)=[(6*den*I_y*l/(5*l)+(13*m/35));0;-(I_y*den/10)-(11*m*l/210);0;0;0;...
          -(6*den*I_y*l/(5*l))+(9*m/70);0;-(I_y*den/10)+(13*m*l/420);0];
m_ele2(3,3:12)=[(6*den*I_y*l/(5*l)+(13*m/35)),0,(-I_y*l*l/10)-(11*m*l/210),0,0,0,...
          -(6*den*I_y*l/(5*l))+(9*m/70),0,-(I_y*den/10)+(13*m*l/420),0];

m_ele2(4:12,4)=[((1/3)*den*l*(j1));0;0;0;0;0;((1/6)*den*l*(j1));0;0];
m_ele2(4,4:12)=[((1/3)*den*l*(j1)),0,0,0,0,0,((1/6)*den*l*(j1)),0,0];

m_ele2(5:12,5)=[(2*I_y*l*den/15)+(m*l*l/105);0;0;0;...
          (I_y*den/10)-(13*m*l/420);0;(-I_y*l*den/30)-(m*l*l/140);0];
m_ele2(5,5:12)=[(2*I_y*l*den/15)+(m*l*l/105),0,0,0,...
          (I_y*den/10)-(13*m*l/420),0,(-I_y*l*den/30)-(m*l*l/140),0];

m_ele2(6:12,6)=[(2*I_z*l*den/15)+(m*l*l/105);0;(-I_z*den/10)+(13*m*l/420)...
          ;0;0;0;(-I_z*l*den/30)-(m*l*l/140)];
m_ele2(6,6:12)=[(2*I_z*l*den/15)+(m*l*l/105),0,(-I_z*den/10)+(13*m*l/420)...
          ,0,0,0,(-I_z*l*den/30)-(m*l*l/140)];

m_ele2(7:12,7)=[m/3;0;0;0;0;0];
m_ele2(7,7:12)=[m/3,0,0,0,0,0];
```

```
m_ele2(8:12,8)=[(6*I_z*den/(5*l))+(13*m/35);0;0;0;(-I_z*den/10)-(11*m*l/210)];
m_ele2(8,8:12)=[(6*I_z*den/(5*l))+(13*m/35),0,0,0,(-I_z*den/10)-(11*m*l/210)];

m_ele2(9:12,9)=[(6*I_y*den/(5*l)+(13*m/35));0;(I_y*den/10)+(11*m*l/210);0];
m_ele2(9,9:12)=[(6*I_y*den/(5*l)+(13*m/35)),0,(I_y*den/10)+(11*m*l/210),0];

m_ele2(10:12,10)=[(l*den/3)*(j1);0;0];
m_ele2(10,10:12)=[(l*den/3)*(j1),0,0];

m_ele2(11:12,11)=[(2*I_y*l*den/15)+(m*l*l/105);0];
m_ele2(11,11:12)=[(2*I_y*l*den/15)+(m*l*l/105),0];

m_ele2(12,12)=[(2*I_z*l*den/15)+(m*l*l/105)];

m_ele_cp=m_ele2;
```

B.2.5 stiffness cross pin

```
function [k_ele_cp]=stiff_cp(n_cp,dia_cp,den_cp,ndof,E_cp,G_cp,l_cp)
d=dia_cp;
n=n_cp;
den=den_cp;
E=E_cp;
G=G_cp;
l=l_cp;

A=pi*d^2/4;
m=den*A*l;
I_y=pi*d^4/64;
I_z=pi*d^4/64;
j1=pi*d^4/32;
% Defining the stiffness matrix for the inclined beam element
k_ele2=zeros(ndof,ndof);

% K2=zeros(6*(n+1),6*(n+1));

k_ele2(1:12,1)=(E*A/l)*[1;0;0;0;0;0;-1;0;0;0;0;0];
k_ele2(1,1:12)=(E*A/l)*[1,0,0,0,0,0,-1,0,0,0,0,0];

k_ele2(2:12,2)=((6*E*I_z)/(l^3))*[2;0;0;0;l;0;-2;0;0;0;l];
k_ele2(2,2:12)=((6*E*I_z)/(l^3))*[2,0,0,0,l,0,-2,0,0,0,l];

k_ele2(3:12,3)=(6*E*I_y/(l*l*l))*[2;0;-l;0;0;0;-2;0;-l;0];
k_ele2(3,3:12)=(6*E*I_y/(l*l*l))*[2,0,-l,0,0,0,-2,0,-l,0];

k_ele2(4:12,4)=(G*j1/l)*[1;0;0;0;0;0;-1;0;0];
k_ele2(4,4:12)=(G*j1/l)*[1,0,0,0,0,0,-1,0,0];

k_ele2(5:12,5)=(E*I_y/(l*l))*[4*l;0;0;0;6;0;2*l;0];
k_ele2(5,5:12)=(E*I_y/(l*l))*[4*l,0,0,0,6,0,2*l,0];

k_ele2(6:12,6)=(E*I_z/(l*l))*[4*l;0;-6;0;0;0;2*l];
k_ele2(6,6:12)=(E*I_z/(l*l))*[4*l,0,-6,0,0,0,2*l];

k_ele2(7:12,7)=(A*E/l)*[1;0;0;0;0;0];
k_ele2(7,7:12)=(A*E/l)*[1,0,0,0,0,0];

k_ele2(8:12,8)=(6*E*I_z/(l*l*l))*[2;0;0;0;-l];
k_ele2(8,8:12)=(6*E*I_z/(l*l*l))*[2,0,0,0,-l];

k_ele2(9:12,9)=(6*E*I_y/(l*l*l))*[2;0;l;0];
k_ele2(9,9:12)=(6*E*I_y/(l*l*l))*[2,0,l,0];
```

```
k_ele2(10:12,10)=(G*j1/l)*[1;0;0];
k_ele2(10,10:12)=(G*j1/l)*[1,0,0];

k_ele2(11:12,11)=(4*E*I_y/l)*[1;0];
k_ele_2(11,11:12)=(4*E*I_y/l)*[1,0];

k_ele2(12,12)=(4*E*I_z/l);

k_ele_cp=k_ele2;
```

B.2.6 Mass input shaft

```
function [m_ele_is]=mass_ip_shaft(n_is,dia_is,den,l_is,ndof,E,G)
d=dia_is;
n=n_is;
l=l_is;
A=pi*d^2/4;
m=den*A*l;
%^^^^^^^^^^^^^^^^^^
% made some corrections in the moment of inertia from 32 to 64 check
I_y=pi*d^4/64;
I_z=pi*d^4/64;
j1=pi*d^4/32;
%^^^^^^^^^^^^^^^^^^
% % m_ele=zeros(ndof,ndof);
% % m_ele(:,1)=[m/3;0;0;0;0;0;m/6;0;0;0;0;0];
% % m_ele(1,:)=[m/3,0,0,0,0,0,m/6,0,0,0,0,0];
% %
% %
m_ele(2:12,2)=[(6*den*I_z*l/(5*l)+(13*m/35));0;0;0;(I_z*den/10)+(11*m*l/210);0;...
% %              -(6*den*I_z*l/(5*l))+(9*m/70);0;0;0;(I_z*den/10)-(13*m*l/420)];
% %
m_ele(2,2:12)=[(6*den*I_z*l/(5*l)+(13*m/35)),0,0,0,(I_z*den/10)+(11*m*l/210),0,...
% %              -(6*den*I_z*l/(5*l))+(9*m/70),0,0,0,(I_z*den/10)-(13*m*l/420)];
% %
% % m_ele(3:12,3)=[(6*den*I_y*l/(5*l)+(13*m/35));0;-(I_y*den/10)-
(11*m*l/210);0;0;0;...
% %              -(6*den*I_y*l/(5*l))+(9*m/70);0;-(I_y*den/10)+(13*m*l/420);0];
% % m_ele(3,3:12)=[(6*den*I_y*l/(5*l)+(13*m/35)),0,(-I_y*den/10)-
(11*m*l/210),0,0,0,...
% %              -(6*den*I_y*l/(5*l))+(9*m/70),0,-(I_y*den/10)+(13*m*l/420),0];
% %
% % m_ele(4:12,4)=[((1/3)*den*l*(j1));0;0;0;0;0;((1/6)*den*l*(j1));0;0];
% % m_ele(4,4:12)=[((1/3)*den*l*(j1)),0,0,0,0,0,((1/6)*den*l*(j1)),0,0];
% %
% % m_ele(5:12,5)=[(2*I_y*l*den/15)+(m*l*l/105);0;0;0;...
% %              (I_y*den/10)-(13*m*l/420);0;(-I_y*l*den/30)-(m*l*l/140);0];
% % m_ele(5,5:12)=[(2*I_y*l*den/15)+(m*l*l/105),0,0,0,...
% %             (I_y*den/10)-(13*m*l/420),0,(-I_y*l*den/30)-(m*l*l/140),0];
% %
% % m_ele(6:12,6)=[(2*I_z*l*den/15)+(m*l*l/105);0;(-I_z*den/10)+(13*m*l/420)...
% %             ;0;0;0;(-I_z*l*den/30)-(m*l*l/140)];
% % m_ele(6,6:12)=[(2*I_z*l*den/15)+(m*l*l/105),0,(-I_z*den/10)+(13*m*l/420)...
% %             ,0,0,0,(-I_z*l*den/30)-(m*l*l/140)];
% %
% % m_ele(7:12,7)=[m/3;0;0;0;0;0];
```

```
% % m_ele(7,7:12)=[m/3,0,0,0,0,0];
% %
% % m_ele(8:12,8)=[(6*I_z*den/(5*l))+(13*m/35);0;0;0;(-I_z*den/10)-(11*m*l/210)];
% % m_ele(8,8:12)=[(6*I_z*den/(5*l))+(13*m/35),0,0,0,(-I_z*den/10)-(11*m*l/210)];
% %
% % m_ele(9:12,9)=[(6*I_y*den/(5*l)+(13*m/35));0;(I_y*den/10)+(11*m*l/210);0];
% % m_ele(9,9:12)=[(6*I_y*den/(5*l)+(13*m/35)),0,(I_y*den/10)+(11*m*l/210),0];
% %
% % m_ele(10:12,10)=[(l*den/3)*(j1);0;0];
% % m_ele(10,10:12)=[(l*den/3)*(j1),0,0];
% %
% % m_ele(11:12,11)=[(2*I_y*l*den/15)+(m*l*l/105);0];
% % m_ele(11,11:12)=[(2*I_y*l*den/15)+(m*l*l/105),0];
% %
% % m_ele(12,12)=[(2*I_z*l*den/15)+(m*l*l/105)];


m_ele=zeros(ndof,ndof);
m_ele(:,1)=[m/3;0;0;0;0;0;m/6;0;0;0;0;0];
m_ele(1,:)=[m/3,0,0,0,0,0,m/6,0,0,0,0,0];

m_ele(2:12,2)=[(6*den*I_z*l/(5*l)+(13*m/35));0;0;0;(I_z*den/10)+(11*m*l/210);0;...
          -(6*den*I_z*l/(5*l))+(9*m/70);0;0;0;(I_z*den/10)-(13*m*l/420)];
m_ele(2,2:12)=[(6*den*I_z*l/(5*l)+(13*m/35)),0,0,0,(I_z*den/10)+(11*m*l/210),0,...
          -(6*den*I_z*l/(5*l))+(9*m/70),0,0,0,(I_z*den/10)-(13*m*l/420)];

m_ele(3:12,3)=[(6*den*I_y*l/(5*l)+(13*m/35));0;-(I_y*den/10)-(11*m*l/210);0;0;0;...
          -(6*den*I_y*l/(5*l))+(9*m/70);0;-(I_y*den/10)+(13*m*l/420);0];
m_ele(3,3:12)=[(6*den*I_y*l/(5*l)+(13*m/35)),0,(-I_y*l*l/10)-(11*m*l/210),0,0,0,...
          -(6*den*I_y*l/(5*l))+(9*m/70),0,-(I_y*den/10)+(13*m*l/420),0];

m_ele(4:12,4)=[((1/3)*den*l*(j1));0;0;0;0;0;((1/6)*den*l*(j1));0;0];
m_ele(4,4:12)=[((1/3)*den*l*(j1)),0,0,0,0,0,((1/6)*den*l*(j1)),0,0];

m_ele(5:12,5)=[(2*I_y*l*den/15)+(m*l*l/105);0;0;0;...
          (I_y*den/10)-(13*m*l/420);0;(-I_y*l*den/30)-(m*l*l/140);0];
m_ele(5,5:12)=[(2*I_y*l*den/15)+(m*l*l/105),0,0,0,...
          (I_y*den/10)-(13*m*l/420),0,(-I_y*l*den/30)-(m*l*l/140),0];

m_ele(6:12,6)=[(2*I_z*l*den/15)+(m*l*l/105);0;(-I_z*den/10)+(13*m*l/420)...
          ;0;0;0;(-I_z*l*den/30)-(m*l*l/140)];
m_ele(6,6:12)=[(2*I_z*l*den/15)+(m*l*l/105),0,(-I_z*den/10)+(13*m*l/420)...
          ,0,0,0,(-I_z*l*den/30)-(m*l*l/140)];

m_ele(7:12,7)=[m/3;0;0;0;0;0];
m_ele(7,7:12)=[m/3,0,0,0,0,0];
```

```
m_ele(8:12,8)=[(6*I_z*den/(5*l))+(13*m/35);0;0;0;(-I_z*den/10)-(11*m*l/210)];
m_ele(8,8:12)=[(6*I_z*den/(5*l))+(13*m/35),0,0,0,(-I_z*den/10)-(11*m*l/210)];

m_ele(9:12,9)=[(6*I_y*den/(5*l)+(13*m/35));0;(I_y*den/10)+(11*m*l/210);0];
m_ele(9,9:12)=[(6*I_y*den/(5*l)+(13*m/35)),0,(I_y*den/10)+(11*m*l/210),0];

m_ele(10:12,10)=[(l*den/3)*(j1);0;0];
m_ele(10,10:12)=[(l*den/3)*(j1),0,0];

m_ele(11:12,11)=[(2*I_y*l*den/15)+(m*l*l/105);0];
m_ele(11,11:12)=[(2*I_y*l*den/15)+(m*l*l/105),0];

m_ele(12,12)=[(2*I_z*l*den/15)+(m*l*l/105)];

% The input shaft is considered to be placesalong the horizontal axis,
% hence considering its inclination angle to be zero Transforming the mass
% matrix for theta=0

trans=zeros(12,12);
theta=0;
% Defining the cosine and sine terms of the transformation matrix
c=cos(2*theta*pi/360);
s=sin(2*theta*pi/360);
% Defining the transformation matrix
t1=zeros(3,3);
t1=[c,s,0;-s,c,0;0,0,1];
null=zeros(3,3);
trans=[t1,null,null,null;null,t1,null,null;null,null,t1,null;null,null,null,t1];
m_ele_s=trans'*m_ele*trans;
m_ele_is=m_ele_s;
```

B.2.7. Stiffness input shaft

```
function [k_ele_is]=stiff_ip_shaft(n_is,dia_is,den,l_is,ndof,E,G)
d=dia_is;
n=n_is;
l=l_is;
A=pi*d^2/4;
m=den*A*l;
%^^^^^^^^^^^^^^^^^^
% made some corrections in the moment of inertia from 32 to 64 check
I_y=pi*d^4/64;
I_z=pi*d^4/64;
j1=pi*d^4/32;
%^^^^^^^^^^^^^^^^^^

k_ele=zeros(ndof,ndof);
K=zeros(6*(n+1),6*(n+1));

k_ele(1:12,1)=(E*A/l)*[1;0;0;0;0;0;-1;0;0;0;0;0];
k_ele(1,1:12)=(E*A/l)*[1,0,0,0,0,0,-1,0,0,0,0,0];

k_ele(2:12,2)=((6*E*I_z)/(l^3))*[2;0;0;0;l;0;-2;0;0;0;l];
k_ele(2,2:12)=((6*E*I_z)/(l^3))*[2,0,0,0,l,0,-2,0,0,0,l];

k_ele(3:12,3)=(6*E*I_y/(l*l*l))*[2;0;-l;0;0;0;-2;0;-l;0];
k_ele(3,3:12)=(6*E*I_y/(l*l*l))*[2,0,-l,0,0,0,-2,0,-l,0];

k_ele(4:12,4)=(G*j1/l)*[1;0;0;0;0;0;-1;0;0];
k_ele(4,4:12)=(G*j1/l)*[1,0,0,0,0,0,-1,0,0];

k_ele(5:12,5)=(E*I_y/(l*l))*[4*l;0;0;0;6;0;2*l;0];
k_ele(5,5:12)=(E*I_y/(l*l))*[4*l,0,0,0,6,0,2*l,0];

k_ele(6:12,6)=(E*I_z/(l*l))*[4*l;0;-6;0;0;0;2*l];
k_ele(6,6:12)=(E*I_z/(l*l))*[4*l,0,-6,0,0,0,2*l];

k_ele(7:12,7)=(A*E/l)*[1;0;0;0;0;0];
k_ele(7,7:12)=(A*E/l)*[1,0,0,0,0,0];

k_ele(8:12,8)=(6*E*I_z/(l*l*l))*[2;0;0;0;-l];
k_ele(8,8:12)=(6*E*I_z/(l*l*l))*[2,0,0,0,-l];

k_ele(9:12,9)=(6*E*I_y/(l*l*l))*[2;0;l;0];
k_ele(9,9:12)=(6*E*I_y/(l*l*l))*[2,0,l,0];

k_ele(10:12,10)=(G*j1/l)*[1;0;0];
```

```
k_ele(10,10:12)=(G*j1/l)*[1,0,0];

k_ele(11:12,11)=(4*E*I_y/l)*[1;0];
k_ele(11,11:12)=(4*E*I_y/l)*[1,0];

k_ele(12,12)=(4*E*I_z/l);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The input shaft is considered to be placesalong the horizontal axis,
% hence considering its inclination angle to be zero Transforming the mass
% matrix for theta=0

% Transforming the stiffness matrix for theta=0
trans=zeros(12,12);
theta=0;
c=cos(2*theta*pi/360);
s=sin(2*theta*pi/360);
t1=zeros(3,3);
t1=[c,s,0;-s,c,0;0,0,1];
null=zeros(3,3);
trans=[t1,null,null,null;null,t1,null,null;null,null,t1,null;null,null,null,t1];
k_ele_s=trans'*k_ele*trans;
k_ele_is=k_ele_s;
```

B.2.8 Mass output shaft

```
function [m_ele_os]=mass_op_shaft(n_os,dia_os,den,l_os,ndof,E,G)
d=dia_os;
n=n_os;
l=l_os;
A=pi*d^2/4;
m=den*A*l;
I_y=pi*d^4/64;
I_z=pi*d^4/64;
j1=pi*d^4/32;


% m_ele=zeros(ndof,ndof);
% m_ele(:,1)=[m/3;0;0;0;0;0;m/6;0;0;0;0;0];
% m_ele(1,:)=[m/3,0,0,0,0,0,m/6,0,0,0,0,0];
%
% m_ele(2:12,2)=[(6*den*I_z*l/(5*l)+(13*m/35));0;0;0;(I_z*den/10)+(11*m*l/210);0;...
%               -(6*den*I_z*l/(5*l))+(9*m/70);0;0;0;(I_z*den/10)-(13*m*l/420)];
% m_ele(2,2:12)=[(6*den*I_z*l/(5*l)+(13*m/35)),0,0,0,(I_z*den/10)+(11*m*l/210),0,...
%               -(6*den*I_z*l/(5*l))+(9*m/70),0,0,0,(I_z*den/10)-(13*m*l/420)];
%
% m_ele(3:12,3)=[(6*den*I_y*l/(5*l)+(13*m/35));0;-(I_y*den/10)-
(11*m*l/210);0;0;0;...
%               -(6*den*I_y*l/(5*l))+(9*m/70);0;-(I_y*den/10)+(13*m*l/420);0];
% m_ele(3,3:12)=[(6*den*I_y*l/(5*l)+(13*m/35)),0,(-I_y*den/10)-(11*m*l/210),0,0,0,...
%               -(6*den*I_y*l/(5*l))+(9*m/70),0,-(I_y*den/10)+(13*m*l/420),0];
%
% m_ele(4:12,4)=[((1/3)*den*l*(j1));0;0;0;0;0;((1/6)*den*l*(j1));0;0];
% m_ele(4,4:12)=[((1/3)*den*l*(j1)),0,0,0,0,0,((1/6)*den*l*(j1)),0,0];
%
% m_ele(5:12,5)=[(2*I_y*l*den/15)+(m*l*l/105);0;0;0;...
%               (I_y*den/10)-(13*m*l/420);0;(-I_y*l*den/30)-(m*l*l/140);0];
% m_ele(5,5:12)=[(2*I_y*l*den/15)+(m*l*l/105),0,0,0,...
%              (I_y*den/10)-(13*m*l/420),0,(-I_y*l*den/30)-(m*l*l/140),0];
%
% m_ele(6:12,6)=[(2*I_z*l*den/15)+(m*l*l/105);0;(-I_z*den/10)+(13*m*l/420)...
%               ;0;0;0;(-I_z*l*den/30)-(m*l*l/140)];
% m_ele(6,6:12)=[(2*I_z*l*den/15)+(m*l*l/105),0,(-I_z*den/10)+(13*m*l/420)...
%               ,0,0,0,(-I_z*l*den/30)-(m*l*l/140)];
%
% m_ele(7:12,7)=[m/3;0;0;0;0;0];
% m_ele(7,7:12)=[m/3,0,0,0,0,0];
%
% m_ele(8:12,8)=[(6*I_z*den/(5*l))+(13*m/35);0;0;0;(-I_z*den/10)-(11*m*l/210)];
% m_ele(8,8:12)=[(6*I_z*den/(5*l))+(13*m/35),0,0,0,(-I_z*den/10)-(11*m*l/210)];
```

```
%
% m_ele(9:12,9)=[(6*I_y*den/(5*l)+(13*m/35));0;(I_y*den/10)+(11*m*l/210);0];
% m_ele(9,9:12)=[(6*I_y*den/(5*l)+(13*m/35)),0,(I_y*den/10)+(11*m*l/210),0];
%
% m_ele(10:12,10)=[(l*den/3)*(j1);0;0];
% m_ele(10,10:12)=[(l*den/3)*(j1),0,0];
%
% m_ele(11:12,11)=[(2*I_y*l*den/15)+(m*l*l/105);0];
% m_ele(11,11:12)=[(2*I_y*l*den/15)+(m*l*l/105),0];
%
% m_ele(12,12)=[(2*I_z*l*den/15)+(m*l*l/105)];


m_ele=zeros(ndof,ndof);
m_ele(:,1)=[m/3;0;0;0;0;0;m/6;0;0;0;0;0];
m_ele(1,:)=[m/3,0,0,0,0,0,m/6,0,0,0,0,0];

m_ele(2:12,2)=[(6*den*I_z*l/(5*l)+(13*m/35));0;0;0;(I_z*den/10)+(11*m*l/210);0;...
          -(6*den*I_z*l/(5*l))+(9*m/70);0;0;0;(I_z*den/10)-(13*m*l/420)];
m_ele(2,2:12)=[(6*den*I_z*l/(5*l)+(13*m/35)),0,0,0,(I_z*den/10)+(11*m*l/210),0,...
          -(6*den*I_z*l/(5*l))+(9*m/70),0,0,0,(I_z*den/10)-(13*m*l/420)];

m_ele(3:12,3)=[(6*den*I_y*l/(5*l)+(13*m/35));0;-(I_y*den/10)-(11*m*l/210);0;0;0;...
          -(6*den*I_y*l/(5*l))+(9*m/70);0;-(I_y*den/10)+(13*m*l/420);0];
m_ele(3,3:12)=[(6*den*I_y*l/(5*l)+(13*m/35)),0,(-I_y*l*l/10)-(11*m*l/210),0,0,0,...
          -(6*den*I_y*l/(5*l))+(9*m/70),0,-(I_y*den/10)+(13*m*l/420),0];

m_ele(4:12,4)=[((1/3)*den*l*(j1));0;0;0;0;0;((1/6)*den*l*(j1));0;0];
m_ele(4,4:12)=[((1/3)*den*l*(j1)),0,0,0,0,0,((1/6)*den*l*(j1)),0,0];

m_ele(5:12,5)=[(2*I_y*l*den/15)+(m*l*l/105);0;0;0;...
          (I_y*den/10)-(13*m*l/420);0;(-I_y*l*den/30)-(m*l*l/140);0];
m_ele(5,5:12)=[(2*I_y*l*den/15)+(m*l*l/105),0,0,0,...
          (I_y*den/10)-(13*m*l/420),0,(-I_y*l*den/30)-(m*l*l/140),0];

m_ele(6:12,6)=[(2*I_z*l*den/15)+(m*l*l/105);0;(-I_z*den/10)+(13*m*l/420)...
          ;0;0;0;(-I_z*l*den/30)-(m*l*l/140)];
m_ele(6,6:12)=[(2*I_z*l*den/15)+(m*l*l/105),0,(-I_z*den/10)+(13*m*l/420)...
          ,0,0,0,(-I_z*l*den/30)-(m*l*l/140)];

m_ele(7:12,7)=[m/3;0;0;0;0;0];
m_ele(7,7:12)=[m/3,0,0,0,0,0];

m_ele(8:12,8)=[(6*I_z*den/(5*l))+(13*m/35);0;0;0;(-I_z*den/10)-(11*m*l/210)];
m_ele(8,8:12)=[(6*I_z*den/(5*l))+(13*m/35),0,0,0,(-I_z*den/10)-(11*m*l/210)];
```

```
m_ele(9:12,9)=[(6*I_y*den/(5*l)+(13*m/35));0;(I_y*den/10)+(11*m*l/210);0];
m_ele(9,9:12)=[(6*I_y*den/(5*l)+(13*m/35)),0,(I_y*den/10)+(11*m*l/210),0];

m_ele(10:12,10)=[(l*den/3)*(j1);0;0];
m_ele(10,10:12)=[(l*den/3)*(j1),0,0];

m_ele(11:12,11)=[(2*I_y*l*den/15)+(m*l*l/105);0];
m_ele(11,11:12)=[(2*I_y*l*den/15)+(m*l*l/105),0];

m_ele(12,12)=[(2*I_z*l*den/15)+(m*l*l/105)];

% The input shaft is considered to be placesalong the horizontal axis,
% hence considering its inclination angle to be zero Transforming the mass
% matrix for theta=0

trans=zeros(12,12);
theta=0;
% Defining the cosine and sine terms of the transformation matrix
c=cos(2*theta*pi/360);
s=sin(2*theta*pi/360);
% Defining the transformation matrix
t1=zeros(3,3);
t1=[c,s,0;-s,c,0;0,0,1];
null=zeros(3,3);
trans=[t1,null,null,null;null,t1,null,null;null,null,t1,null;null,null,null,t1];
m_ele_s=trans'*m_ele*trans;
m_ele_os=m_ele_s;
```

B.2.9 Stiffness output shaft

```
function [k_ele_os]=stiff_op_shaft(n_os,dia_os,den,l_os,ndof,E,G)
d=dia_os;
n=n_os;
l=l_os;
A=pi*d^2/4;
m=den*A*l;
I_y=pi*d^4/64;
I_z=pi*d^4/64;
j1=pi*d^4/32;

k_ele=zeros(ndof,ndof);
K=zeros(6*(n+1),6*(n+1));

k_ele(1:12,1)=(E*A/l)*[1;0;0;0;0;0;-1;0;0;0;0;0];
k_ele(1,1:12)=(E*A/l)*[1,0,0,0,0,0,-1,0,0,0,0,0];

k_ele(2:12,2)=((6*E*I_z)/(l^3))*[2;0;0;0;l;0;-2;0;0;0;l];
k_ele(2,2:12)=((6*E*I_z)/(l^3))*[2,0,0,0,l,0,-2,0,0,0,l];

k_ele(3:12,3)=(6*E*I_y/(l*l*l))*[2;0;-l;0;0;0;-2;0;-l;0];
k_ele(3,3:12)=(6*E*I_y/(l*l*l))*[2,0,-l,0,0,0,-2,0,-l,0];

k_ele(4:12,4)=(G*j1/l)*[1;0;0;0;0;0;-1;0;0];
k_ele(4,4:12)=(G*j1/l)*[1,0,0,0,0,0,-1,0,0];

k_ele(5:12,5)=(E*I_y/(l*l))*[4*l;0;0;0;6;0;2*l;0];
k_ele(5,5:12)=(E*I_y/(l*l))*[4*l,0,0,0,6,0,2*l,0];

k_ele(6:12,6)=(E*I_z/(l*l))*[4*l;0;-6;0;0;0;2*l];
k_ele(6,6:12)=(E*I_z/(l*l))*[4*l,0,-6,0,0,0,2*l];

k_ele(7:12,7)=(A*E/l)*[1;0;0;0;0;0];
k_ele(7,7:12)=(A*E/l)*[1,0,0,0,0,0];

k_ele(8:12,8)=(6*E*I_z/(l*l*l))*[2;0;0;0;-l];
k_ele(8,8:12)=(6*E*I_z/(l*l*l))*[2,0,0,0,-l];

k_ele(9:12,9)=(6*E*I_y/(l*l*l))*[2;0;l;0];
k_ele(9,9:12)=(6*E*I_y/(l*l*l))*[2,0,l,0];

k_ele(10:12,10)=(G*j1/l)*[1;0;0];
k_ele(10,10:12)=(G*j1/l)*[1,0,0];

k_ele(11:12,11)=(4*E*I_y/l)*[1;0];
```

```
k_ele(11,11:12)=(4*E*I_y/l)*[1,0];

k_ele(12,12)=(4*E*I_z/l);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The input shaft is considered to be placesalong the horizontal axis,
% hence considering its inclination angle to be zero Transforming the mass
% matrix for theta=0

% Transforming the stiffness matrix for theta=0
trans=zeros(12,12);
theta=0;
c=cos(2*theta*pi/360);
s=sin(2*theta*pi/360);
t1=zeros(3,3);
t1=[c,s,0;-s,c,0;0,0,1];
null=zeros(3,3);
trans=[t1,null,null,null;null,t1,null,null;null,null,t1,null;null,null,null,t1];
k_ele_s=trans'*k_ele*trans;
k_ele_os=k_ele_s;
```

B.2.10 Transformation_1

```
function [trans]=trans_45(theta)
trans=zeros(12,12);
% Defining the sine and cosine terms for the transformation matrix
c=cos(2*theta*pi/360);
s=sin(2*theta*pi/360);
% Defining the transformation matrix
t1=zeros(3,3);
t1=[c,s,0;-s,c,0;0,0,1];
null=zeros(3,3);
trans=[t1,null,null,null;null,t1,null,null;null,null,t1,null;null,null,null,t1];
```

B.2.11 transformation_2

```
function [trans]=trans_45_cph(theta)
trans=zeros(12,12);
% Defining the sine and cosine terms for the transformation matrix
c=0;
s=sin(2*theta*pi/360);
% Defining the transformation matrix
t1=zeros(3,3);
t1=[c,0,s;0,1,0;-s,0,c];
null=zeros(3,3);
trans=[t1,null,null,null;null,t1,null,null;null,null,t1,null;null,null,null,t1];
```

B.2.12 transformation_3

```
function [trans]=trans_45_cpv(theta)
trans=zeros(12,12);
% Defining the sine and cosine terms for the transformation matrix
c=0;
s=sin(2*theta*pi/360);
% Defining the transformation matrix
t1=zeros(3,3);
t1=[c,s,0;-s,c,0;0,0,1];
null=zeros(3,3);
trans=[t1,null,null,null;null,t1,null,null;null,null,t1,null;null,null,null,t1];
```

B.2.13 Transformation_4

```
function [trans]=trans_45_hz(theta)
trans=zeros(12,12);
% Defining the sine and cosine terms for the transformation matrix
c=cos(2*theta*pi/360);
s=sin(2*theta*pi/360);
% Defining the transformation matrix
t1=zeros(3,3);
t1=[c,0,s;0,1,0;-s,0,c];
```

```
null=zeros(3,3);
trans=[t1,null,null,null;null,t1,null,null;null,null,t1,null;null,null,null,t1];
```

B.2.14 Graphical User Interface

```
function varargout = Joint_GUI_MOD(varargin)
% JOINT_GUI_MOD M-file for Joint_GUI_MOD.fig
%      JOINT_GUI_MOD, by itself, creates a new JOINT_GUI_MOD or raises the
existing
%      singleton*.
%
%      H = JOINT_GUI_MOD returns the handle to a new JOINT_GUI_MOD or the
handle to
%      the existing singleton*.
%
%      JOINT_GUI_MOD('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in JOINT_GUI_MOD.M with the given input
arguments.
%
%      JOINT_GUI_MOD('Property','Value',...) creates a new JOINT_GUI_MOD or
raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Joint_GUI_MOD_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Joint_GUI_MOD_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 Joint_GUI_MOD

% Last Modified by GUIDE v2.5 06-Jun-2005 22:18:41

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Joint_GUI_MOD_OpeningFcn, ...
                   'gui_OutputFcn',  @Joint_GUI_MOD_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(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 Joint_GUI_MOD is made visible.
function Joint_GUI_MOD_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 Joint_GUI_MOD (see VARARGIN)

% Choose default command line output for Joint_GUI_MOD
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

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


% --- Outputs from this function are returned to the command line.
function varargout = Joint_GUI_MOD_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;



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
```

```
handles.edit1
guidata(hObject, handles);
edit1=get(hObject,'string');
yoke_beam_height=str2double(edit1)
save yoke_beam_height
% --- 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 && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
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
handles.edit2
guidata(hObject, handles);
edit2=get(hObject,'string');
len_is=str2double(edit2)
save len_is

% --- 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 && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
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
handles.edit3
guidata(hObject, handles);
edit3=get(hObject,'string');
len_os=str2double(edit3)
save len_os

% --- 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 && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
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
handles.edit4
guidata(hObject, handles);
edit4=get(hObject,'string');
len_cp=str2double(edit4)
save len_cp

% --- 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 && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function edit5_Callback(hObject, eventdata, handles)
% hObject    handle to edit5 (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 edit5 as text
%        str2double(get(hObject,'String')) returns contents of edit5 as a double
handles.edit5
guidata(hObject, handles);
edit5=get(hObject,'string');
len_y_cp=str2double(edit5)
save len_y_cp

% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit5 (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 && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function edit6_Callback(hObject, eventdata, handles)
% hObject    handle to edit6 (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 edit6 as text
%        str2double(get(hObject,'String')) returns contents of edit6 as a double
handles.edit6
guidata(hObject, handles);
edit6=get(hObject,'string');
dia_is=str2double(edit6)
save dia_is

% --- Executes during object creation, after setting all properties.
function edit6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit6 (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 && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
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
handles.edit7
guidata(hObject, handles);
edit7=get(hObject,'string');
dia_os=str2double(edit7)
save dia_os

% --- 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 && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
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
handles.edit8
guidata(hObject, handles);
edit8=get(hObject,'string');
dia_cp=str2double(edit8)
save dia_cp

% --- 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 && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
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
handles.edit9
guidata(hObject, handles);
edit9=get(hObject,'string');
yoke_beam_width=str2double(edit9)
save yoke_beam_width

% --- 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 && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
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
handles.edit10
guidata(hObject, handles);
edit10=get(hObject,'string');
den=str2double(edit10)
save den

% --- 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 && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
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
handles.edit11
guidata(hObject, handles);
edit11=get(hObject,'string');
den_cp=str2double(edit11)
```

```
save den_cp

% --- 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 && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
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
handles.edit12
guidata(hObject, handles);
edit12=get(hObject,'string');
mod_E=str2double(edit12)
save mod_E

% --- 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 && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
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
handles.edit13
guidata(hObject, handles);
edit13=get(hObject,'string');
mod_E_cp=str2double(edit13)
save mod_E_cp

% --- 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 && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
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
handles.edit14
guidata(hObject, handles);
edit14=get(hObject,'string');
poi=str2double(edit14)
save poi

% --- 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 && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
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
handles.edit15
guidata(hObject, handles);
edit15=get(hObject,'string');
poi_cp=str2double(edit15)
save poi_cp

% --- 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 && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

function edit22_Callback(hObject, eventdata, handles)
% hObject    handle to edit22 (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 edit22 as text
%        str2double(get(hObject,'String')) returns contents of edit22 as a double
handles.edit22
guidata(hObject, handles);
edit22=get(hObject,'string');
num_ele_is=str2double(edit22)
```

```
save num_ele_is

% --- Executes during object creation, after setting all properties.
function edit22_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit22 (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 && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

function edit23_Callback(hObject, eventdata, handles)
% hObject    handle to edit23 (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 edit23 as text
%        str2double(get(hObject,'String')) returns contents of edit23 as a double

handles.edit23
guidata(hObject, handles);
edit23=get(hObject,'string');
num_ele_os=str2double(edit23)
save num_ele_os
% --- Executes during object creation, after setting all properties.
function edit23_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit23 (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 && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

function edit24_Callback(hObject, eventdata, handles)
% hObject    handle to edit24 (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 edit24 as text
%        str2double(get(hObject,'String')) returns contents of edit24 as a double

handles.edit24
guidata(hObject, handles);
edit24=get(hObject,'string');
num_ele_y=str2double(edit24)
save num_ele_y
% --- Executes during object creation, after setting all properties.
function edit24_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit24 (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 && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
function edit25_Callback(hObject, eventdata, handles)
% hObject    handle to edit25 (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 edit25 as text
%        str2double(get(hObject,'String')) returns contents of edit25 as a double
handles.edit25
guidata(hObject, handles);
edit25=get(hObject,'string');
num_ele_cp=str2double(edit25)
save num_ele_cp
% --- Executes during object creation, after setting all properties.
function edit25_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit25 (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 && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
```