2004

A DEVELOPMENT OF A COMPUTER AIDED GRAPHIC USER INTERFACE POSTPROCESSOR FOR ROTOR BEARING SYSTEMS

Pavan Kumar Arise
University of Kentucky, pavan@engr.uky.edu

ABSTRACT OF THESIS

A DEVELOPMENT OF A COMPUTER AIDED GRAPHIC USER INTERFACE POSTPROCESSOR FOR ROTOR BEARING SYSTEMS

Rotor dynamic analysis, which requires extensive amount of data and rigorous analytical processing, has been eased by the advent of powerful and affordable digital computers. By incorporating the processor and a graphical interface post processor in a single set up, this program offers a consistent and efficient approach to rotor dynamic analysis.

The graphic user interface presented in this program effectively addresses the inherent complexities of rotor dynamic analyses by linking the required computational algorithms together to constitute a comprehensive program by which input data and the results are exchanged, analyzed and graphically plotted with minimal effort by the user. Just by selecting an input file and appropriate options as required, the user can carry out a comprehensive rotor dynamic analysis (synchronous response, stability analysis, critical speed analysis with undamped map) of a particular design and view the results with several options to save the plots for further verification. This approach helps the user to modify the design of turbomachinery quickly, until an efficient design is reached, with minimal compromise in all aspects.

KEYWORDS: Rotordynamic Analyses, Graphical User Interface, Post Processor, Cubic Spline, Visual Basic.

Pavan Kumar Arise

09/24/2004

A DEVELOPMENT OF A COMPUTER AIDED GRAPHIC USER INTERFACE POSTPROCESSOR FOR ROTOR BEARINGS SYSTEMS

By

Pavan Kumar Arise

Dr. Keith E. Rouch

(Director of Thesis)

Dr. George Huang

(Director of Graduate Studies)

09/24/2004

RULES FOR THE USE OF THESIS

Unpublished thesis submitted for the Master's degree and deposited in the University of Kentucky Library are as a rule open for inspection, but are to be used only with due regard to the rights of the authors. Bibliographical references may be noted, but quotations or summaries of parts may be published only with the permission of the author, and with the usual scholarly acknowledgements.

Extensive copying or publication of the thesis in whole or in part also requires the consent of the Dean of the Graduate School of the University of Kentucky.

THESIS

Pavan Kumar Arise

The Graduate School

University of Kentucky

2004

A DEVELOPMENT OF A COMPUTER AIDED GRAPHIC USER INTERFACE POSTPROCESSOR FOR ROTOR BEARINGS SYSTEMS

THESIS

A thesis submitted in partial fulfillment of the
requirements for the degree of Master of Science in Mechanical Engineering in the
College of Engineering
at the University of Kentucky

By

Pavan Kumar Arise

Lexington, Kentucky

Director: Dr. Keith E. Rouch, Professor of Mechanical Engineering

Lexington, Kentucky

2004

Dedication

To my family, especially my mother for being such a wonderful person and all that she has done for me.

ACKNOWLEDGEMENTS

This thesis would not have been possible without the guidance and support of Dr.Keith Rouch, the director of this thesis. He was always ready to spare his valuable time for any trivial questions I had. With an exceptional knowledge and expertise in the field of turbomachinery rotor dynamics, he was an ever accessible resource to me. I thank him for the resources he has provided me until the completion of this work. This thesis would not have been complete without the reviews and comments of many people, especially Dr.Carlo Roso, for sharing his thoughts and suggestions to improve the program in all aspects. I would also like to thank Dr.David Herrin and Dr.John Baker for being in my thesis committee.

I must thank my parents, sisters and brother for their unwavering support throughout the period of my work on this thesis. I cannot forget their encouragement at each step of my life and the motivation that they have provided, for me to reach this level. I am indebted to them for all they have done to me.

Finally, I take the opportunity to thank each one of my friends who have made my stay at the University of Kentucky so wonderful. My special thanks to Bhavana Parvathareddy for her support and cooperation.

TABLE OF CONTENTS

Acknowledgements…….....vii
List of Tables…….xi
List of Figures…….....xii
Nomenclature……xv
List of Files……...xvi

Chapter Page
1. Introduction and Background …..……1
1.1 Introduction……..……...2
1.2 Turbomachinery Rotordynamics Historical Background…..…….....4
1.3 Modeling of Rotor-Bearing Systems……..8
1.4 Rotordynamic Analyses…….10
1.4.1 Undamped Critical Speed Analysis……13
1.4.2 Synchronous Response Analysis……17
1.4.3 Stability Analysis……19
1.4.4 Transient Analysis……..23
1.5 Computational Approaches……24
1.5.1 RotBrg© Processor Program……25
1.6 Scope of Thesis……..27
2. Graphical User Interface and Applications.....……28
2.1 Introduction……...….....29
2.2 Graphic User Interface ……...……29
2.2.1 User-Centric Design …….……...30
2.2.2 Event-Driven Programming …….…….....30
2.2.3 Portability of Applications……...31
2.2.4 Template Usage……32
2.3 GUI Options Available……...…….34
2.3.1 C/C++ ...…….....34
2.3.2 Visual Basic……..34
2.3.3 Visual basic.NET …….35

Chapter Page

2.3.4 C# ……...36
2.3.5 Java ……..37
2.4 Selected Option……...……...39
3. Plotting and Curve Fitting……..……..40
3.1 Introduction……41
3.2 Plotting Options……..42
3.2.1 C/C++ ……..42
3.2.2 Visual Basic……42
3.2.3 Visual Basic.NET/ C# ……..43
3.2.4 Java…….....…….43
3.2.5 SciPlot..... ……..43
3.2.6 GnuPlot….……...44
3.2.7 TecPlot…….45
3.2.8 Mathematica.. ……..45
3.2.9 Fortner Dataplot……...46
3.2.10 Excel/Spreadsheet……..46
3.3 Selected Option……...47
3.4 Curve Fitting……...48
4. Program Schematic and Implementation……..51
4.1 File Structure……...52
4.2 Program Schematic of Rotordynamic Post-Processor……54
4.3 Program Installation...……...….....74
4.4 Sample Plots……78
4.4.1 Undamped Critical Speed Analysis Sample Plots..……..….....78
4.4.2 Stability Analysis Sample Plots……82
4.2.3 Synchronous Response Sample Plots……...88
5. Conclusions and Future Work…….....93
5.1 Conclusions……..93
5.2 Future Work…….……94

Appendix…… 95
A1. Visual Basic Cubic Spline Subroutine Code..……95
References……...97
Vita…….103

LIST OF TABLES

Table **Description** **Page**

Table 4.1 : Synchronous Response Analysis Output Files……53
Table 4.2 : Stability Analysis Output Files……53
Table 4.3 : Critical Speed Analysis Output Files……...53

LIST OF FIGURES

Chapter	Figure	Description	Page
1	Figure 1.1 :	Rotor-bearing-foundation model for rotordynamic analysis…….	10
	Figure 1.2 :	Sequential flow diagram of various rotor-bearing system calculations. ……....	11
	Figure 1.3 :	Logic diagram of a typical design procedure…….	12
	Figure 1.4 :	A typical critical speed map……...	14
	Figure 1.5 :	A typical normal mode shape in stability analysis…….	15
	Figure 1.6 :	A typical synchronous response analysis : Phase angle Vs Speed…….	18
	Figure 1.7 :	A typical synchronous response analysis : Amplitude Vs Speed……...	18
	Figure 1.8 :	A typical stability map at constant operating speed……	21
	Figure 1.9 :	A typical whirl speed map……...	22
3	Figure 3.1 :	A Typical Cubic Spline Curve……	48
4	Figure 4.1:	RotPlot© Program schematic overview……...	55
	Figure 4.2 :	Program loading……..	56
	Figure 4.3 :	Program background……...	57
	Figure 4.4 :	Help tips at startup……...	57
	Figure 4.5 :	Main menu showing the processor and post-processor overview……	58
	Figure 4.6 :	Main menu showing program help……..	58
	Figure 4.7 :	Program version and copyright information……	59
	Figure 4.8 :	Processor menu……	59
	Figure 4.9 :	Processor file input menu…….	60
	Figure 4.10 :	Program working message…….	60
	Figure 4.11 :	Rotor dynamic analyses options menu……..	61
	Figure 4.12 :	Synchronous response specific option menu…….	62
	Figure 4.13 :	Synchronous response analysis sub menu…….	62
	Figure 4.14 :	Synchronous response specific file input menu…….	63
	Figure 4.15 :	Plot settings menu for response plot……..	64

Chapter	Figure	Description	Page

4 Figure 4.16 : Synchronous response default setting changing menu…….65

Figure 4.17 : Plot setting menu for phase plot……...66

Figure 4.18 : File input menu for bearing response plot……67

Figure 4.19 : Plot setting menu for bearing response…….67

Figure 4.20 : Stability analysis menu…….68

Figure 4.21: Stability analysis file input error handler……...68

Figure 4.22 : Stability analysis file input menu……..69

Figure 4.23 : Stability analysis defaults changing menu……69

Figure 4.24 : Critical speed analysis options menu……70

Figure 4.25 : Critical speed analysis file input error handler…….70

Figure 4.26 : Critical speed analysis selection error handler……..70

Figure 4.27 : Critical speed analysis file input menu……..71

Figure 4.28 : Critical speed map input file menu……71

Figure 4.29 : Critical speed analysis mode plotting options menu…….72

Figure 4.30 : Program exit confirmation…….72

Figure 4.31 : Sample plot with options for the user……73

Figure 4.32 : Program installation start up screen……..74

Figure 4.33 : Program installation wizard …….75

Figure 4.34 : Program installation setup option…….75

Figure 4.35 : Program Installation settings conformation……..76

Figure 4.36 : Program installation progress……76

Figure 4.37 : Program installation confirmation…….77

Figure 4.38 : Critical speed mode shape for stiffness of 1E4……..……..78

Figure 4.39 : Critical speed mode shapes for stiffness of 1E5……..…….....79

Figure 4.40 : Critical speed mode shapes for stiffness of 1E6……..…….79

Figure 4.41 : Critical speed mode shapes for a stiffness of 1E7……..……...80

Figure 4.42 : Critical speed mode shapes with grid option turned on……80

Figure 4.43 : Critical speed mode shapes with rotor model and bearing locations…81
superimposed

Chapter	Figure	Description	Page

4 Figure 4.44 : Critical speed map……..……..……..……..…….81

Figure 4.45 : First damped mode shape……..……..……..……82

Figure 4.46 : Second damped mode shape……..……..……..……82

Figure 4.47 : Third damped mode shape……..……..……..……...83

Figure 4.48 : Fourth damped mode shape……..……..……..…….83

Figure 4.49 : Fifth damped mode shape……..……..……..……84

Figure 4.50 : Sixth damped mode shape……..……..……..……84

Figure 4.51 : Seventh damped mode shape……..……..……..……...85

Figure 4.52 : Damped mode shape with rotor dynamic model superimposed……85

Figure 4.53 : Damped mode shape with rotor and bearings location superimposed...86

Figure 4.54 : Damped mode shape with grid option turned on……..…….86

Figure 4.55 : Damped mode shape with specific position parameters selected……..87

Figure 4.56 : Synchronous response of the rotor at first station location……88

Figure 4.57 : Synchronous response of the rotor at second station location……88

Figure 4.58 : Synchronous response of the rotor at third station location……...89

Figure 4.59 : Synchronous response of the rotor with grid option turned on……..89

Figure 4.60 : Synchronous response plot with specific location parameters……...90
highlighted.

Figure 4.61 : Phase response plot……..……..……..……..……90

Figure 4.62 : Bearing response plot at first bearing location……..…….91

Figure 4.63 : Bearing response plot at second bearing location……..……91

Figure 4.64 : Bearing phase plot……..……..……..……..…….92

NOMENCLATURE

The nomenclature in general use in this thesis is listed below. Isolated usage of symbols from references is explained at the time of use and may not be included hereafter.

Variable

D	Complex quantity
F	Force magnitude
I_p	Polar moment of Inertia
I_d	Diametral moment of inertia
K	Stiffness
m	Mass
M	Moment
N_j	J^{th} Nodal location
n	Number of points to be plotted
S_j	J^{th} Station (element)
$\bar{q}$	Vector of nodal displacement
U	Unbalance mass
λ	Real part of comple eigenvalue (growth factor)
ω	Complex part of eigenvalue – frequency (radians)

Subscript

x	Along x-axis
y	Along y-axis
z	Along z-axis

Matrix, Vector and Derivative Notation

.	Indicates derivative with respect to time
..	Indicates second derivative with respect to time
[ ]	Matrix quantity
{ }	Vector quantity

LIST OF FILES

Filename	File size
1.PavanA.pdf	1.90MB

CHAPTER 1

INTRODUCTION AND BACKGROUND

This chapter introduces the basic concepts of turbomachinery rotor dynamics and related analysis methodology, commonly employed to design, verify and diagnose the high speed rotating equipment. A brief study delving into the historical developments in the field of rotor dynamics is provided. Also included in the chapter, is a review of computational approaches pertaining to rotor dynamics, including the one used in this thesis. The chapter concludes with the motivation and scope of effort presented in the thesis.

1.1 INTRODUCTION

Industrial machinery is said to be optimized when it results from a state-of-the-art mathematical simulation and design, comprehensive prototype, preproduction testing and manufacturing with minimal costs. A design is stated to be satisfactory if it satisfies the high technical requirements and low production costs simultaneously. Hence the machinery design involves an element of compromise in the many requirements of secondary importance. The success of a design, is not simply achieved by looking at one parameter in isolation, but is a complex process, with various parameters in interaction. Generally, as the objectives of the design can be formulated in terms of a few global variables, there may be a great many variables also defined and handled within each subsystem. The evaluation of how the variables of primary importance affect the overall quality of the design is of significant importance. Choosing an optimal design based on the particular requirements of the application and at the same time keeping in view the manufacturing costs is a challenge for any design, Roso, C. (1997). Choosing the best combination of design parameters from the innumerable options is of critical importance.

Apart from the above mentioned objectives, the other equally vital parameter in a profitable machine design is time. Though a designer may achieve a most optimal design, its success also depends on the amount of time taken, which is almost always kept constant. Hence regardless of whether the other parameters are varied or not, the time assigned for the development activity of new machinery is invariably constant. A general industry dictum is that a greater degree of inflexibility to the ‘time parameter’ leads to more rapid returns on the investment, Roso, C. (1995). Hence it seems straightforward that the time factor for design activities must be managed effectively to achieve a machine configuration that best confirms the requirements of a given application without requiring exorbitant production costs. Thus it can be said that a designer always has time, as an objective function, that is required to be kept at a minimum.

Machine design is customarily subdivided in phases namely: conceptualization, modeling, analysis and verification. Thus in computer terminology, design activity can be roughly divided between input data generation, computation and interpretation of analysis

results, Roso, C. (1997). The advent of faster digital computers has seen a significant reduction in computational time with the enhanced hardware capabilities making it possible to see enhanced graphical interpretation of results. So, when data preparation is improved and the non-computational time reduced, more time is available to the designer to explore alternative solutions and hence improve overall design quality, without infringing on improved time constraint.

An analysis of rotor bearing dynamics is critical to the design of any high-speed machinery that has rotating parts. Such analysis has taken a giant leap forward with the development of standard computer programs for determining various system characteristics. When the tedious and error prone modeling process is automated and data input is consistently shared among the analysis processors and graphic postprocessors, the analysis would be significantly improved.

During the discussion that follows, the reader should be better able to appreciate the need for an improved graphical interface for computer analysis of rotor dynamics.

1.2 TURBOMACHINERY ROTORDYNAMICS HISTORICAL BACKGROUND

In recent years, there has been a marked increase in the average operating speed and power output of rotating machinery. The need for more compact and lighter-weight engines and power transmission systems, for improved engine performance, has provided the thrust for this trend. The industrial turbo machinery has benefited by this advancement in the aerodynamic knowledge which led to a substantial redesign, thereby increasing the efficiency.

Efficient rotating machinery requires high design speeds, which can result in a number of rotor bearing system dynamic issues such as prediction and control of rotor response, balancing and rotor bearing stability, Roso, C. (1997). Rigid rotor balancing methods are not adequate for balancing the new generation of high-speed rotors. Thus, a prediction of the flexible rotors during the transition through critical speeds is important for the design. Also, the sources of self excited instabilities need to be predicted as well as understood and instability mechanisms controlled.

The ability to mathematically simulate the behavior of the rotating machinery leads to the development of systems with high-speed performance and low production costs. Furthermore, it helps in better understanding of the problems associated with the design of such high-speed machinery. The evolution of the analysis of the rotor bearing system began with a single degree of freedom, expanding to multi degrees of freedom, constrained only by the capacity and performance of digital computers, Roso, C. (1995).

Early in the development of rotor dynamics, it was a commonly prevailing notion that operation above the first critical speed was impossible. Nascent stages of rotor dynamic study began with Rankine (1869), whose work resulted in his conclusion that a shaft would be stable under the first critical speed and would always be dynamically unstable above that. Thus, the unbounded increase in the vibration in the vicinity of the critical speed was seen as an unstable condition. This misconception was corroborated by Greenhill (1883), who stated that the shaft inertia contributes to its buckling, thus reinforcing Rankine's concept. Later Dunkerley (1895), using the Reynold's eigen value concept, could calculate critical speeds of wide variety of shaft disc systems. The turn of the century saw the strong endorsement of Rankine's concept by

renowned researchers such as Chree,C (1904) and Kerr,W.(1916) although Delaval,G. claimed to have built a steam turbine that could operate beyond the "instability region" in 1883.

A major breakthrough in rotor dynamics, which later inspired the studies of many researchers, was made by Jeffcott,H.H. (1919), whose more comprehensive model consisted of a flat disc supported by a uniform, massless flexible shaft supported, at its ends, by rigid frictionless bearings. Jeffcott's analysis, which included damping, led to the formulation and approaches to the problem of minimizing amplitudes of synchronous whirl.

Following Jeffcott's theory, a number of publications came forth investigating the shortcomings and steadily resolved a number of issues. Rodgers,C. (1922) introduced the concept of Coriolis forces in the computation of resonant frequencies. He studied the behavior of vertical and horizontal undamped rotor systems with dissimilar lateral stiffness. Newkirk,B.L. (1924) addressed the concept of non-synchronous motion and of shaft whirling. He later studied the oil whirl in hydrodynamic journal bearings. Kimball,A.L. (1925) came out with a significant understanding and solution to internal friction. He iterated that the bending in shafts was the cause of the shaft to whirl when rotating above the first critical speed. Several studies were made by Robertson,D. (1933), on the transient whirling of a rotor disturbed from its steady state condition and on hysteretic whirling of rotors. Around the same time Smith,D.M. (1933) completed a comprehensive study analyzing the characteristic features of most of the important problems in rotor bearing dynamics from the properties of the governing equations. The study mainly focused on the unbalance whirl and stability of flexible rotors in flexible bearings.

The Jeffcott model was basically a particle or point mass representation and was inadequate to explain the phenomena that arose due to rigid body characteristics of flexible rotating equipment. The Stodola (1927) and Green (1928) model consisted of a rigid disc to examine the influence of rigid body parameter on the rotor's natural frequencies, critical speeds and synchronous response. From a practical viewpoint, the Stodola-Green mode could be used to explain many of the dynamic consequences of an over hung turbine wheel design for rotating equipment.

An experimental study of shaft unbalance whirling was conducted by Downham,E. (1950-1957). Kellenburger,W. (1958) made a detailed study of response and stability of a shaft

with distributed mass and elasticity having dissimilar stiffness properties, rotating in rigid end bearings. Yamamoto,T. (1964) examined the critical speeds and forced vibrations of a shaft carrying an asymmetrical rotating body. The experimental study, of the various rates of acceleration through a critical speed zone, by Lewis,R.M (1960) led to the conclusion that a rapid transition restricts the maximum amplitude, and a slow transient allows large transient whirl amplitudes. His results used a simple rotor-spring-mass system. He along with Dimentberg,F.M. (1961) showed that a harmonic vibration was induced with reverse precession near critical speed, due to the effects of flexible supports, which results in dynamic shaft stresses.

Due to the increasing speed and memory of the digital computers, the prediction of stiffness and damping characteristics of bearings and foundation received greater attention, since it directly affects the accuracy of rotor dynamics of complex machine system. In this path, a detailed study was conducted by Lund,J.W. and Sternlicht,B. (1961) on the response of flexible single disk rotor operating in plain cylindrical fluid film bearings with direct and cross coupled stiffness and damping properties.

A study involving rotor response to the unbalance of a flexible single disk rotor operating in bearings with radial stiffness and damping was made by Hagg,A.C. (1965). A comprehensive computer code was developed on the formulation of equations for unbalance response and rotor instability presented by Lund,J.W. (1965). Smith,D.M. (1966) investigated the system vibration to the other stationary components of machinery.

The focus of consequent research was shifted to the influence of rotor support bearings in the rotor response. The computer analysis helped in the analysis of hydrodynamic and antifriction bearings.

The validity of theoretical bearing stiffness and damping coefficients was examined by Orcutt,F.K. and Arwas,E.B. (1967). With the availability of fast computers, they were able to investigate bearing configuration better suited to produce a stable behavior of high speed machinery. The solution for plain journal bearings were presented by Lund,J.W. and Stenlich,B. (1962), based on finite difference method. Lund,J.W. along with Thompson,K. (1978) applied the technique to a variety of geometries including partial arc and tilting pad bearings. Reddi,M. (1969) developed the finite element solution of incompressible lubrication problem. The finite

element method, being more generic and accommodating to abrupt geometrical changes, is seen to have advantages over the finite difference approach.

Nicholas,J (1977) studied the finite element analysis of pressure dam and tilted pad bearing. Rouch,K.E. (1977) developed an approach to include both pad mass and pitch inertia in predicting the dynamic performance of large pivoted pad journal bearings customarily utilized in high power rotating equipment. Subsequently, Nicholas,J. (1986-1988) and Barrett,L. (1985) analyzed and recognized the failure to include the influence of bearing support structure in rotor dynamic analysis, as one of the main reasons for the difference between analytical prediction and practical data.

Additional sources of instability in the operation of turbomachinery are the clearance excitation destabilizing forces, which was studied by Thomas,H. (1958) followed by Alford,J. (1965). Further contribution towards the modeling and measurement of destabilizing effect on compressor impellers was made by Urlichs,K. (1977) and Brennen,C. (1980)

This compendiary overview of the development in the field of rotordynamics shows that a significantly large amount of knowledge in the field has been accumulated and the thirst to master the field is still continuous. Reference books by Vance,J.M. (1988), Childs,D.W. (1993) and Ehrich,F.F. (1998) provide an insight into the analysis required to design rotor bearing system. With the knowledge and experience required to produce rotor bearing system, a designer can fulfill the requirements of the application in an optimal fashion.

But as the number of complexities of rotordynamic analyses have increased, the need to develop a comprehensive computational environment that comes up with an optimal design has surfaced, as will be clear during the presentation of this effort.

1.3 MODELING OF ROTOR BEARING SYSTEMS

For any analysis, it becomes necessary for the physical phenomena to be represented in mathematical terms. A rotor bearing system specifically requires consideration of structural and fluid dynamics to describe the rotor and bearing behavior. A very brief review of the two modeling approaches in rotor dynamics viz., transfer matrix technique and finite element method is given below. The detailed description is beyond the scope of this thesis and could be found in the references.

In the transfer matrix method, the system model is formulated as a matrix product of transfer matrices of individual rotor segments, discs and bearings with appropriate boundary conditions. Here the equations for each element relate the state variables at one end to those at the other end in a matrix form, Childs, D. (1993). The solution of the system damped natural frequencies and mode shapes follows from the system equations. Assuming the rotor motion to be synchronous, the response calculations can be made. The modal analysis uses the eigen value-eigen vector results to uncouple the modes.

Finite element method is a technique for solving an equation by approximating continuous quantities as a set of quantities at discrete points, often regularly spaced into a so-called grid or mesh. Because finite element method can be adapted to problems of great complexity and unusual geometry, it is an extremely powerful tool in the solution of important problems in heat transfer, fluid mechanics, and mechanical systems. Furthermore, the availability of fast and inexpensive computers allows problems which are intractable using analytic methods to be solved in a straightforward manner. It is based on formulation of energy functional in a sub-regional or element basis solution of unknown nodal displacements and provides a lower bound of the system strain energy and an upper bound on potential energy. The usage of finite element method has numerous advantages like modeling irregularly shaped bodies quite easily, modeling bodies composed of several different materials (since the element equations are evaluated individually), handling unlimited number and different kind of boundary conditions, varying the size of the elements to make it possible to use small elements wherever necessary, including dynamic effects and ability to handle nonlinear behavior existing with large deformations and non linear materials, Logan, D.L. (2001).

The finite element technique has been used in the processor discussed in this thesis, to model the rotor bearing system. Extensive information about the finite element analysis of rotor-bearing systems with matrix reduction can be found in Rouch, K.E.(1977) Ph.D dissertation reference.

1.4. ROTORDYNAMIC ANALYSES

Any rotor dynamic analysis is possible only after modeling the shaft system effectively, usually by interconnecting elements and stations. The description of mass, inertia, internal and external damping and forcing phenomena on a local or global scale are provided within a station or shaft element definition. Mass is represented either as distributed or lumped into rigid elements depending on the other system components. Disc flexibility is one issue that also needs to be included in some cases. The bearings, seals and other destabilizing effects are accounted by stiffness, damping and cross coupling coefficients respectively to appropriate stations of the rotor model. Foundations are represented by stiffness and damping coefficients typically in series with the bearings.

A representative model for rotordynamic analysis of an industrial turbomachinery rotor-bearing-foundation system is as shown in the Fig 1.1., Roso, C. (1995).

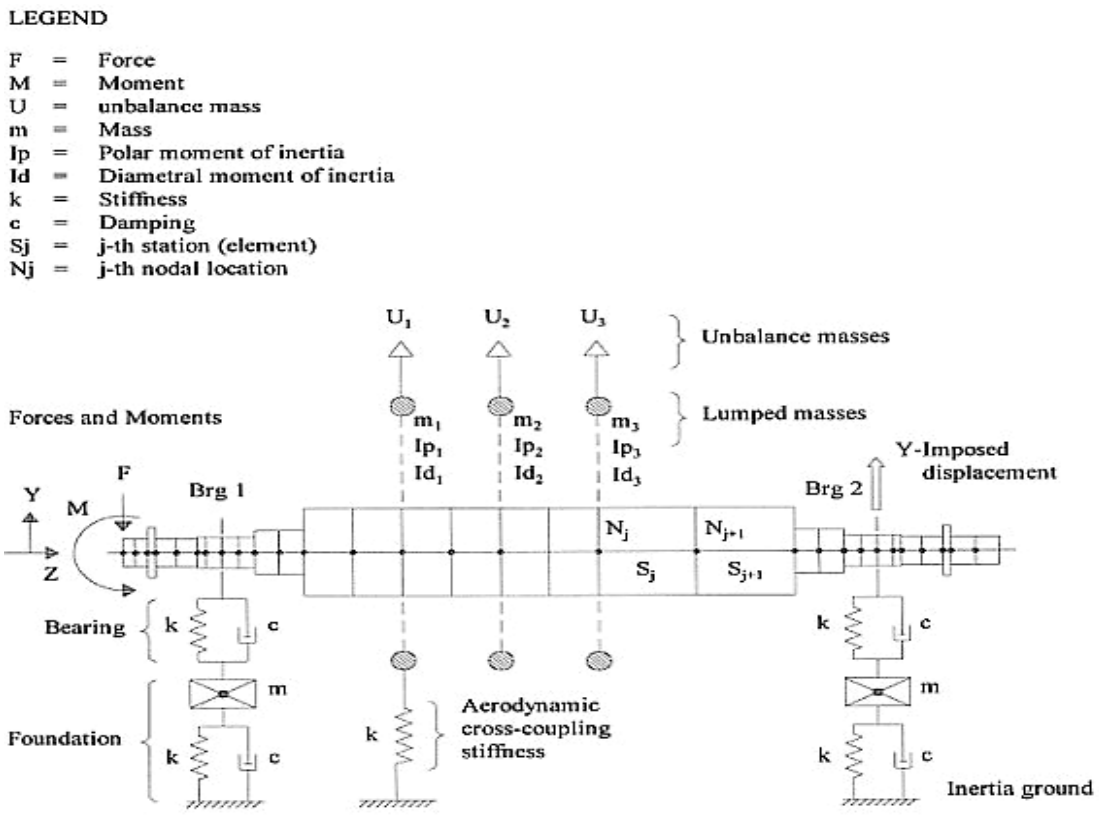


Fig 1.1. Rotor-Bearing-foundation model for rotordynamic analysis.

One common geometrical model can be used to perform synchronous response, critical speed and stability analysis of the physical system. The rest of this section deals with the summary of these analyses and their common results. The flow diagram in Fig 1.2 shows the sequence that is generally followed when using computer programs to perform various rotor-bearing system calculations, Rieger, N. F (1976). It shows that bearing design is to be dealt first due to its impact on the calculation of rotor and foundation characteristics. However, the bearing selection, its design and discussion of their parameters is itself a big subject to deal with and is out of the scope of this thesis work.

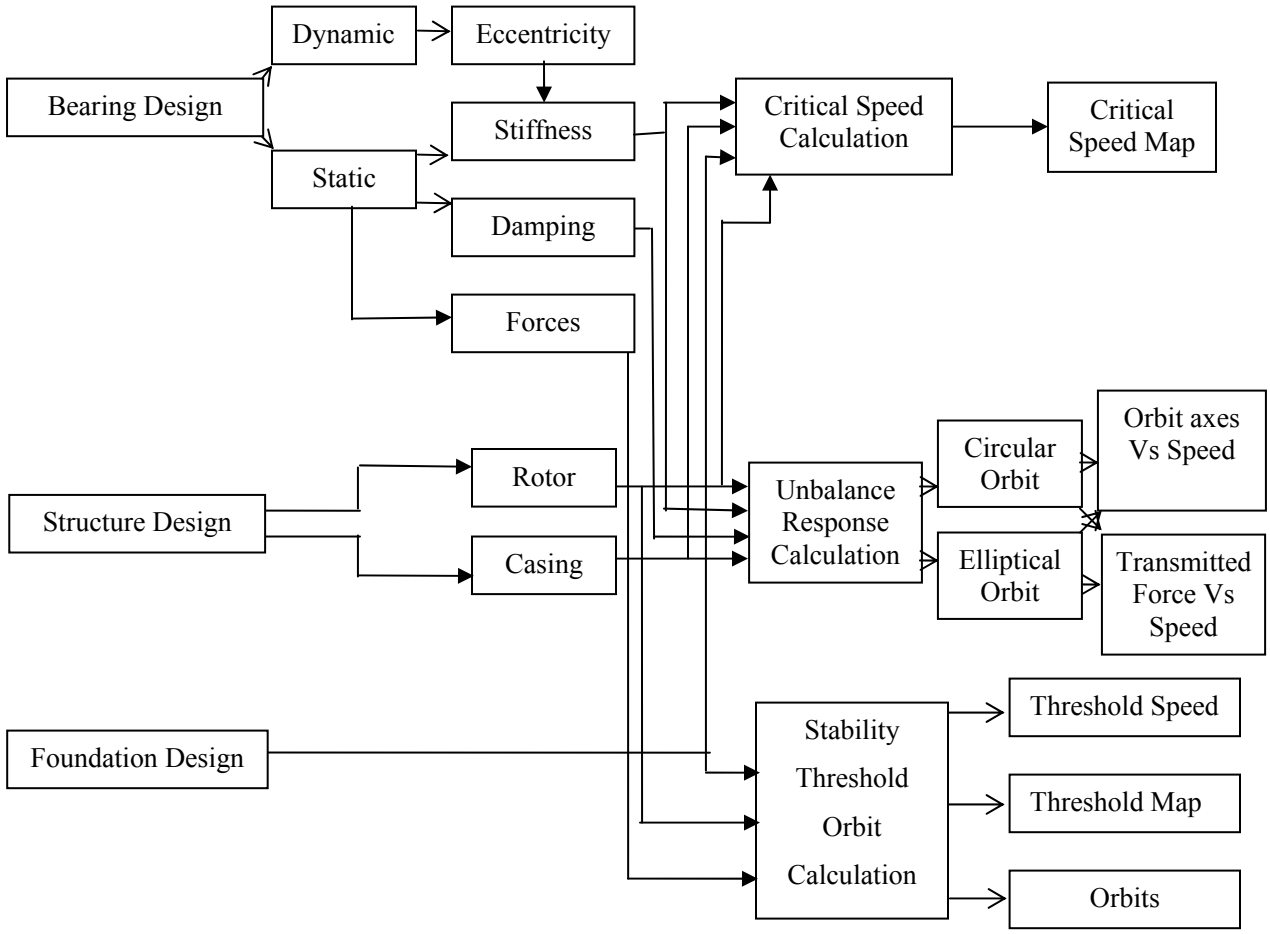


Fig.1.2. Sequential flow diagram of various rotor-bearing system calculations.

The relationship between the rotor-bearing system dynamics and the overall design procedure involving several other parameters is illustrated in the Fig.1.3, Malanoski, S. B.

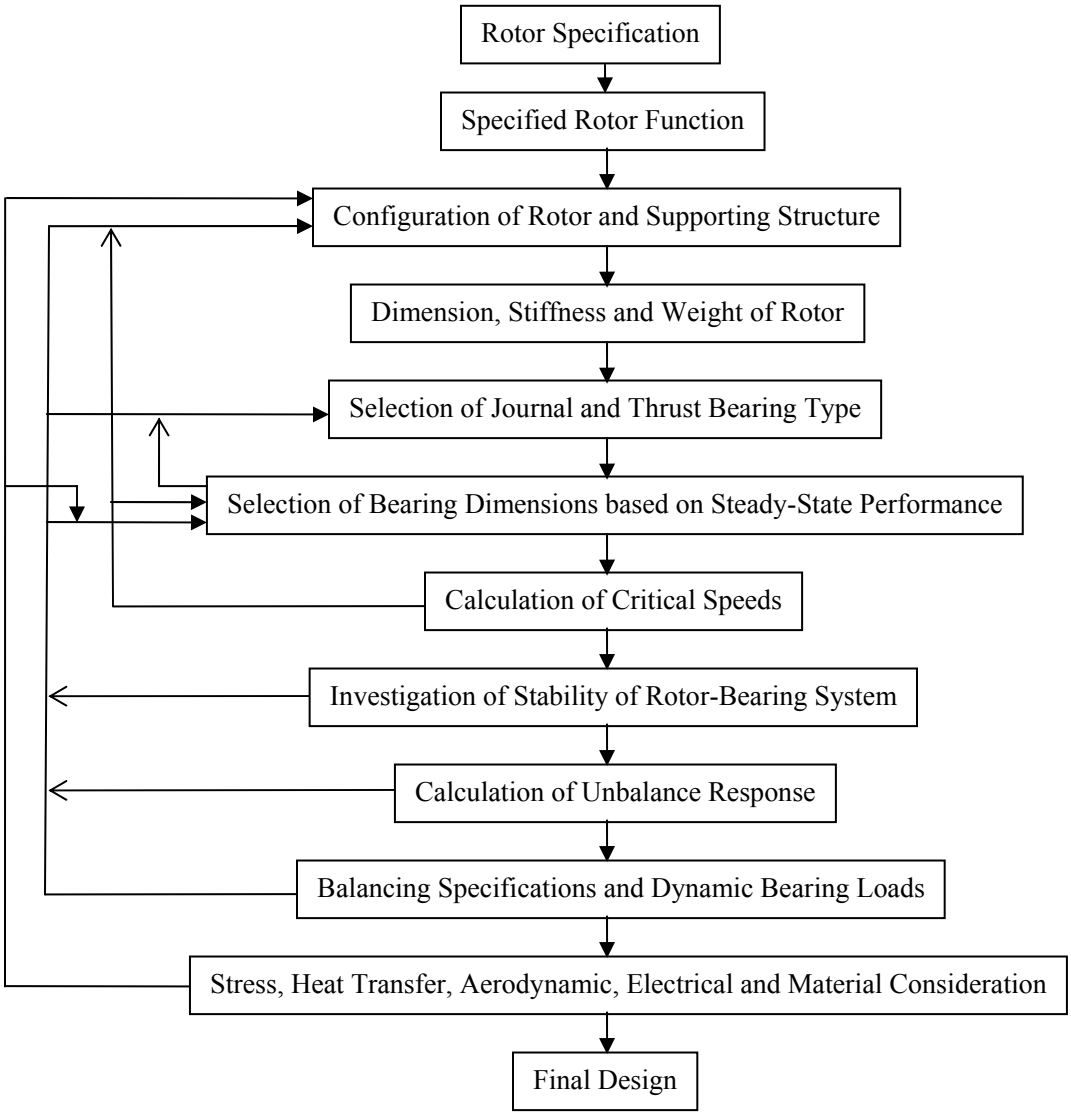


Fig.1.3. Logic diagram of a typical design procedure

The rest of this section deals with the summary of the comprehensive rotor dynamic analysis and their common results.

1.4.1 UNDAMPED CRITICAL SPEED ANALYSIS

Any rotor-bearing system has a number of discrete natural frequencies associated with it. Corresponding to each natural frequency is a mode shape which can be thought of as a snapshot of the rotor deflection curve at any instant of maximum strain during vibration, Vance, J.M. (1988). When one of the natural frequencies is excited by the rotor imbalance, synchronous with the shaft speed, the shaft speed coinciding with the natural frequency is called a critical speed. Hence, the critical speeds can be interpreted as resonant responses to excitation forces at frequencies which coincide with the natural resonant frequencies. These are often computed with damping neglected.

Due to the damping offered by the bearings, critical speed and response analyses are primarily to provide a preliminary approximation of the design and performance of a flexible rotor-bearing system. The system is modeled by developing the appropriate set of second order linear differential equations of motion and by solving them in the neighborhood of an equilibrium solution. The assumption of circular orbit for critical speed calculations is justified by some advantages, such as the rotor being considered in one dimensional representation and the resulting stiffness matrix being symmetric, Rouch, K.E. (1977). The system equation would be as below:

$$[M]\ddot{\bar{q}} + [K]\bar{q} = 0$$

For

$$\bar{q} = \bar{q}_o\, e^{st} = q_o\, e^{j\omega t}$$

$$\left([K] - \omega^2\,[M]\right)\bar{q} = 0$$

This leads to the solution of a number of natural frequencies and the possible motion of the model's degree of freedom. Corresponding mode shapes are determined by obtaining amplitude ratios for specific natural frequencies. The mode shape amplitudes are in relative terms due to the equations being linearly dependent. If any mode is excited at natural frequency by an external harmonic force, the system would be stable as long as the effective damping is sufficient. Conversely, with negative effective damping, the response would be seen to be increasing with time without bound, implying the motion is unstable.

Due to the damping characteristics of some of the fluid film bearings, the effective damping to rotor modes tends to zero when the motion is some fraction of operating speed, Vance, J.M. (1988). Considering a linear rotor bearing system, as the speed of the rotor increases, at some point the natural frequency of the system coincides with the fractional frequency at which the effective damping reduces to zero. This condition is called the whirl threshold speed of instability. A further rise in rotor speed would result in rapid growth of whirl displacement amplitude. But in practice, due to some non linearity in the model, the actual system would exhibit a bounded motion.

As discussed, the rotor-bearing design and analysis process is an iterative process. Many constraints on the mechanical design result from process requirements and these are often conflicting. In this design process, some procedure for assessing the resonance characteristics of a rotor as a function of bearing stiffness can give the designer a directive in the iterative process. Such a procedure, a critical speed map, is useful in judging rotor flexibility and in indicating the effectiveness of possible bearing damping. This presents on a log-log graph the critical speed location as a function of support stiffness. This technique shows the rigid rotor or rigid shaft behavior as a straight line with a slope of half for the first two critical speeds. Thus the rigid rotor and flexible rotor regions are readily apparent. Superposition of the bearing stiffness versus speed characteristic on the map gives the location of the undamped rotor natural frequency as shown in Fig 1.4., Roso, C. (1995).

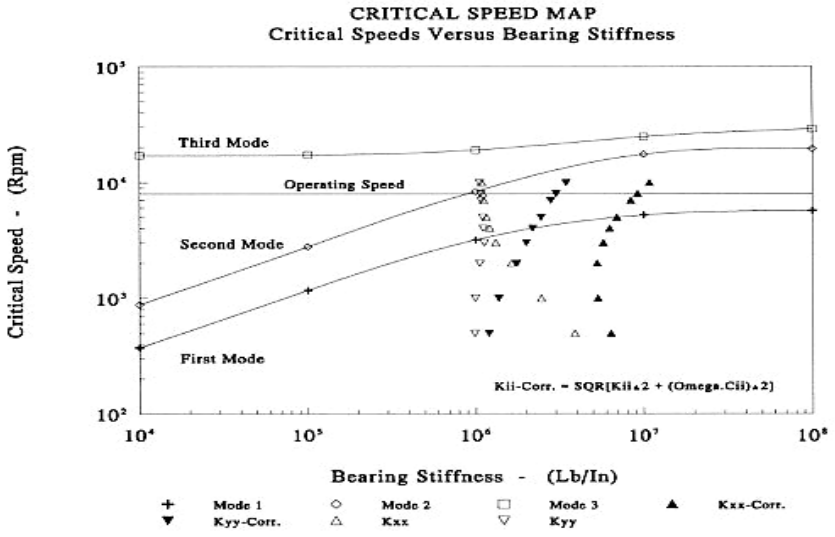


Fig.1.4. A typical critical speed map.

An agreement between lateral speed analysis and the damped synchronous response would be obtained when the effect of damping is included in the computation of the actual bearing as shown in the figure above.

Another output of this type of analysis is the mode shapes of the rotor at critical speeds. This further indicates the degree of rotor bending and presents information relative to natural nodes of the rotor. This further gives a qualitative indication about the effectiveness of the bearing damping. For example, if a bearing is located at a nodal location, very little bearing motion can result to utilize the inherent damping of the system, implying, amplitude at critical speeds would be unacceptably high. A minor bearing location change would result in improvement in the overall performance. Also, as the bearing stiffness is progressively increased from low to high values, the form of the two lowest modes change from that of rigid rotor translatory and conical whirl respectively to that of flexible beam supported on progressively stiffer supports. A sample mode shape plot can be seen in Fig.1.5, Roso, C. (1995).

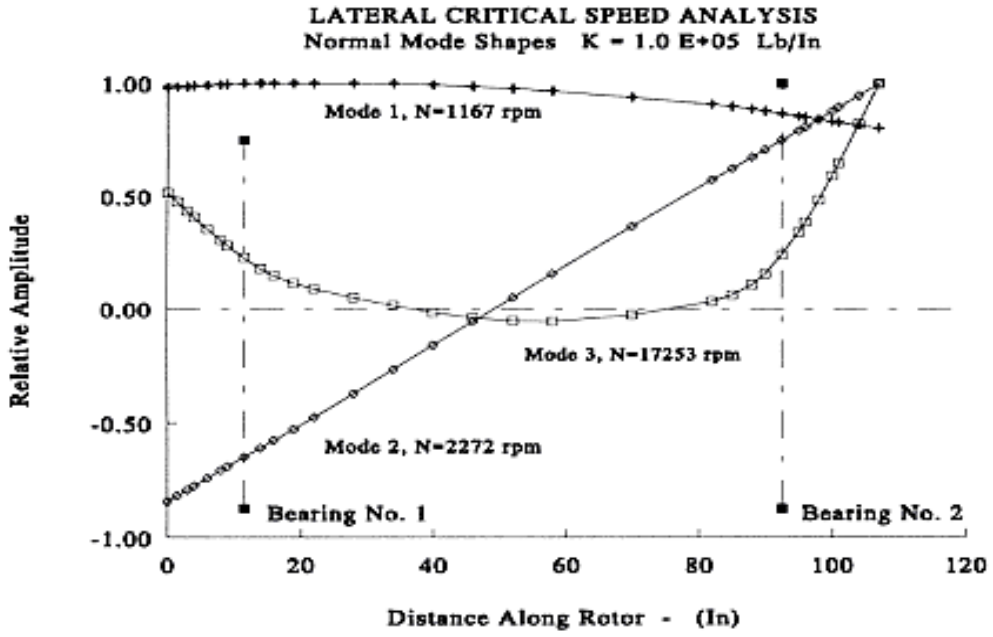


Fig.1.5. A typical normal mode shape in stability analysis.

Undamped Critical speeds are not in itself any indication of the system instability, but rather, a condition of possible large resonant response inherent to unbalance forces. Hence, the undamped critical speed analysis would give the designer a feel of the areas in which the system could reach unacceptable amplitudes under the conditions of synchronous excitation and marginal damping. This analysis, although it can be useful as a guide, is rarely used exclusively, as discussed in the next section on rotor response to unbalance. It is also possible to calculate the damped critical speeds, but this is normally done as proof of stability analysis.

1.4.2 SYNCHRONOUS RESPONSE

The most common form of harmonic motion is that of the unbalance or synchronous response, in which, the deviation of the center of mass of the rotor from the geometric center results in an unbalance force vector rotating with the rotor. This analysis helps the designer to know the magnitude of the displacement of whirl motion while crossing critical speeds or the rotor motion in a predefined speed with certain amount of unbalance distribution. The subject analysis determines the amplitude of response for each of the rotor locations as a function of rotor speed. Another result from this analysis is the bearing dynamic forces and system dissipated energy level.

This type of analysis can be used as a sensitivity analysis. In this case, a unit imbalance can be located at locations coinciding the components such as couplings, impellers, balance pistons, etc and that which is particularly influential in generating high amplitudes of vibration at critical locations is determined. Another value of this analysis is that it can establish the effect of small modification in design such as component weight reduction, reduction of overhangs and also, as discussed, bearing location or bearing span.

The unbalance response is calculated by assuming a general synchronous elliptical orbit motion. The linear differential equation of motion, which includes the effects of damping, is written in a non-homogeneous form. The substitution of the assumed solution designated as steady state, and of the calculated value of excitation in the differential equation of motion, produces a system of algebraic equation with complex coefficients that can be solved with respect to complex displacement vectors.

After the displacement vector is known, the phase angle by which response lags the excitation, can be computed. The forces transmitted to the rotor support can also be obtained once the displacement vector has been computed using force-displacement relationship. The results of the synchronous response analysis are presented for each nodal location of the model. Fig.1.6. and Fig.1.7 show representative results of a typical synchronous response of a rotor bearing system, Roso, C. (1995)

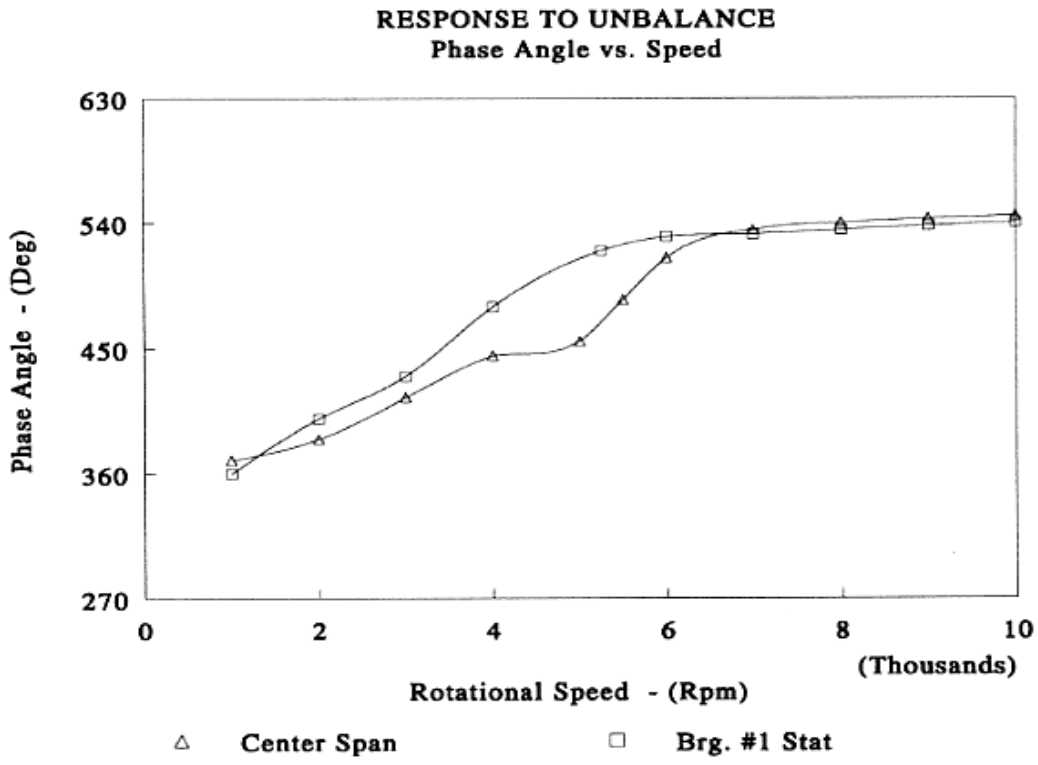


Fig.1.6. Synchronous Response Analysis : Phase angle Vs Speed

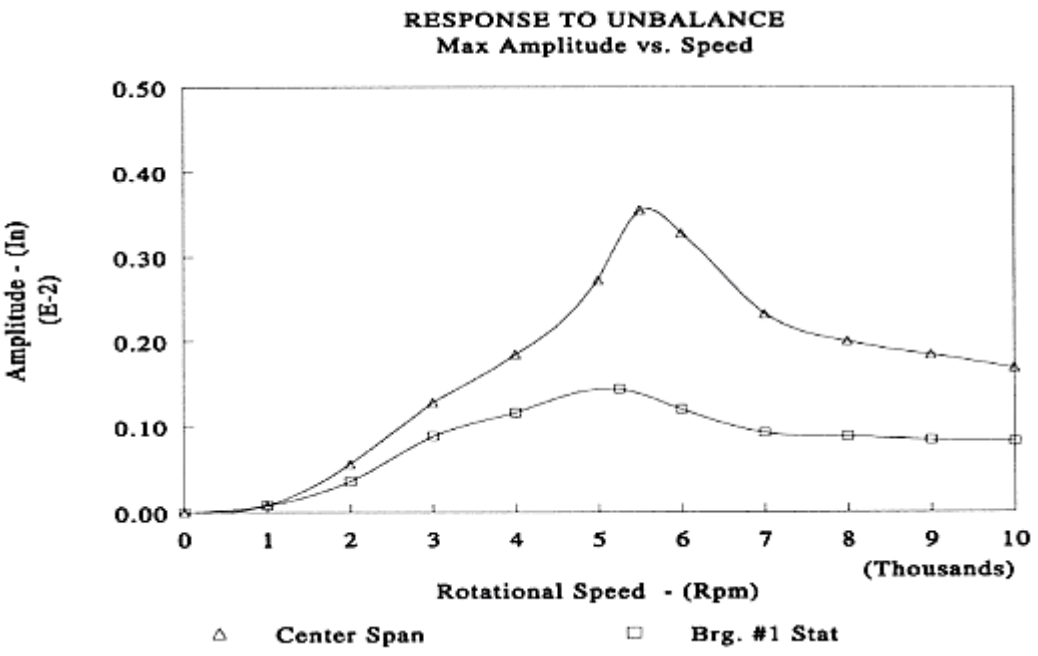


Fig.1.7. Synchronous Response Analysis : Amplitude Vs Speed

1.4.3 STABILITY ANALYSIS

The evaluation of rotor-bearing system stability utilizes the same mass elastic model, bearing spring and damping coefficients, as well as other system elements which can exhibit cross-coupling or destabilizing behavior. A system is termed as stable, pertaining to rotor dynamics, when "the motion following a sufficiently small disturbance remains within prescribed bounds" and is termed as asymptotically stable when the system "tends to resume its original position with time", Huseyin,K. (1976).

Stability analysis provides information on the behavior of the rotor-bearing system verifying if the whirl orbit of rotor increases or decreases with time from the equilibrium position after some disturbing action. A stable system would return to its original whirl orbit with a damped oscillation motion, otherwise would continue to grow with time until restrained by some other non rotating component, following removal of disturbance. The orbit form is the result of all the collective forces acting on the rotor, such as the inwardly directed radial forces tending to maintain dynamic equilibrium of the system and the tangential forces acting opposite to the whirling motion, which confer stability to the system at rotor critical speeds, Poritsky,H. (1965).

As outlined by Lund,J.W. (1975) some sources of destabilization of the system are:

- Hydrodynamic bearing and seal forces
- Interaction with process fluid flow forces
- Internal rotor damping
- Rotor asymmetry

If such a source of instability is present, some degree of external damping, generally in bearings or supports, needs to be present to maintain stability, Rouch, K.E. (1977).

As previously described, the behavior of a rotor-bearing system is represented by linear differential equations of motion which reflect a condition of dynamic equilibrium. The general solution of equations is a combination of linearly independent solutions in a form of exponential time dependent function with generally complex exponents ($\lambda+j\omega$). The real part (λ) of the

exponent corresponds to the rate of growth or decay, while the imaginary part (ω) is the damped natural frequency corresponding to the frequency components of the response. Huseyin,K. (1976) defines four cases.

a) All the λ_k (λ=1,2,3,…n)<0 indicates asymptotic stability.

b) One or more λ_k >0 shows instability

c) If some λ_k = 0 while remaining λ's < 0, state is critical; may be stable but not asymptotically stable

d) If all λ_k = 0 (ω not repeated) state is stable.

A divergent instability is classified as the case where $\lambda_k > 0$ and $\omega_k = 0$. When $\lambda_k > 0$ and $\omega_k \neq 0$, a dynamic situation arises in which the system oscillates at increased amplitudes.

The procedure generally followed is to calculate the system damped natural frequencies and determine the output at these frequencies in the form of a log decrement. In summary, the results of this analysis are the system damped critical speeds and the sensitivity of the system towards supporting non-synchronous vibrations. Generally the first undamped critical speed is deemed to be of primary interest in determining the stability margin of the system. However each of the eigen values will be accompanied by a system log decrement to guide this interpolation.

At any operating speed of a rotor-bearing system, there is a system natural frequency ν (in general, nonsynchronous with the rotational speed) which can be excited and which has an amplitude decay component. Amplitude growth is described by the expression '$e\lambda t$', where t is the time. In the commonly referred to as logarithmic decrement, $\delta=-2\lambda/\nu$, if δ becomes zero, and then negative, the rotor instability threshold speed becomes equal to the running speed. The rotor then becomes unstable and highly susceptible to any internal or external excitation forces.

Normally, the basic rotor model is first evaluated with the bearing system alone. This is valuable in that most available field data is more complete in the definition of rotor model and bearing system, whereas the definition of other elements in the system is often unknown or unattainable. According to Malanoski.S.B, log decrement values above 0.5 portend to be a stable

system; negative values indicate definite instability while positive values below 0.25 indicate marginal behavior. Values between 0.25 and 0.5 represent a gray area which must be more thoroughly evaluated in terms of the other destabilizing forces of the system. Once having ascertained the sensitivity of the system, estimates are made for the effects of oil-buffered seals, aerodynamics, etc.

Generally, the whirling motion in rotating machine assemblies does not follow the exponential growth rate predicted by linear analysis because non linear system energy is dissipated more rapidly. Thus a steady state of motion is reached after a sharp increase in whirl amplitude with rotational speed. The high amplitude of the rotor can be tolerated unless it causes a functional damage to the system.

A typical stability analysis is summarized graphically by plotting stability parameter i.e. the growth factor or logarithmic decrement versus an increasing value of destabilization excitation as shown in Fig.1.8, Roso, C. (1995). The behavior of the rotor bearing system can then be predicted for the estimated value of destabilizing excitation.

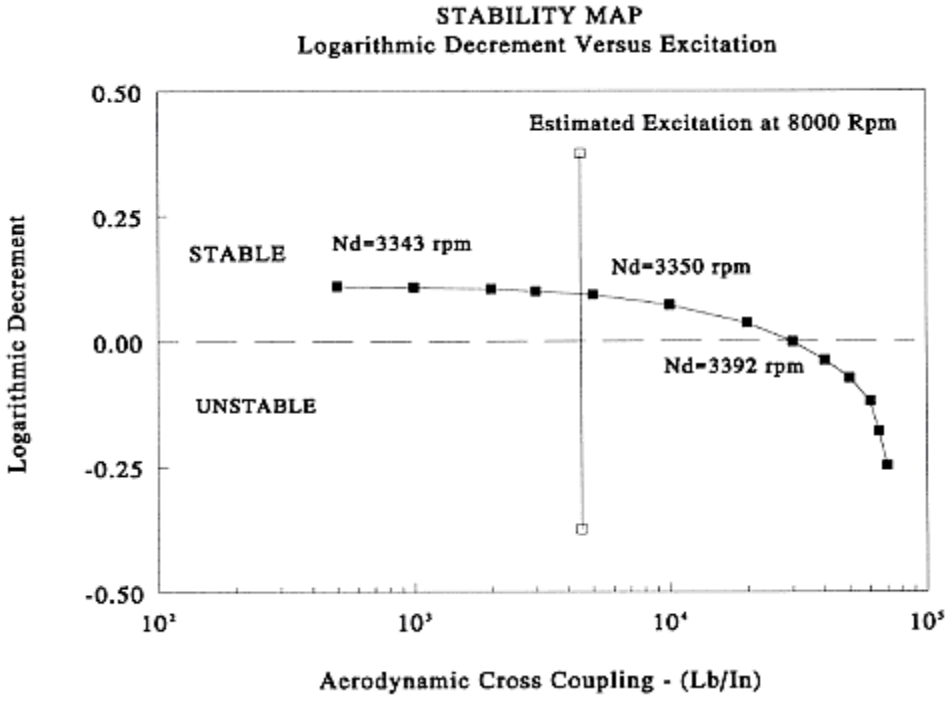


Fig.1.8. A typical stability map at constant operating speed.

When a linear plot of damped natural frequency versus shaft rotational speed is constructed, the plot is called 'whirl speed map'. Superimposition of excitation forcing frequency versus rotational speed identifies the potential critical speeds as indicated in Fig.1.9, Roso, C. (1995). The logarithmic decrement values are also indicated in the plot to have an idea of the sensitivity of the rotor to the assumed excitation.

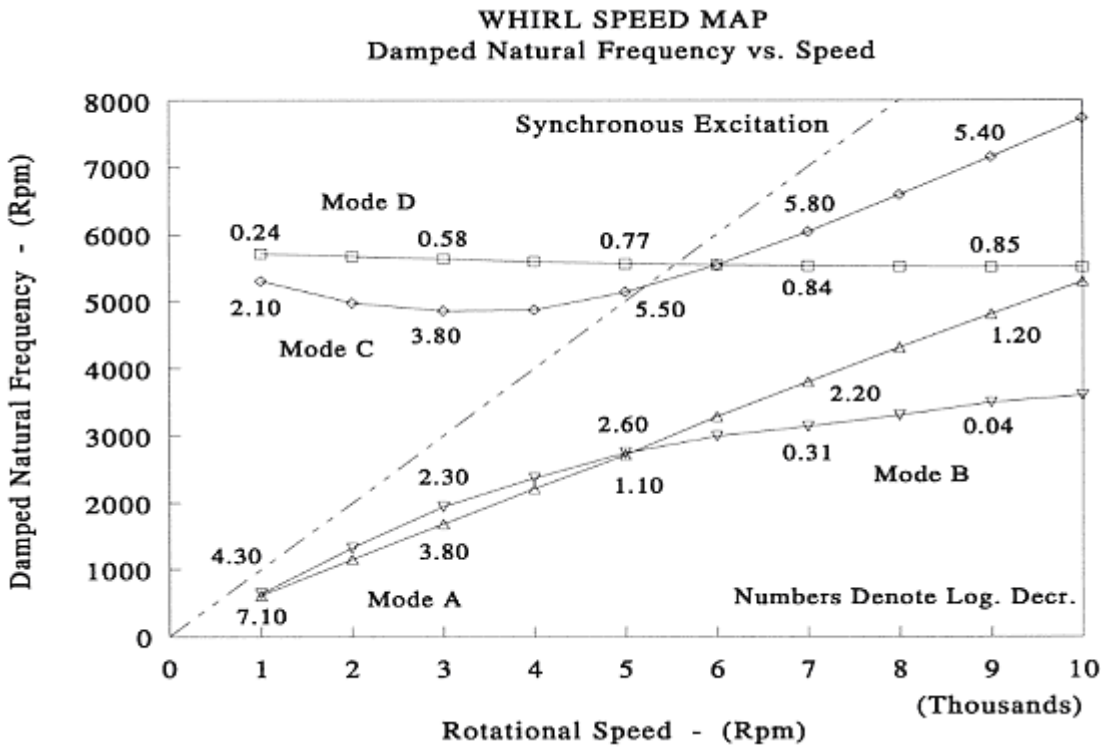


Fig.1.9. A typical whirl speed map.

The various individual mode shapes corresponding to various natural frequencies are also computed as part of stability analysis.

1.4.4 TRANSIENT RESPONSE

The transient response analysis is used to analyze the behavior of complex rotor-bearing systems. This analysis is carried out by observing the trend of amplitude response with time. Here the system of differential equations of motion is integrated with respect to time allowing for the most general solution, including possible non-linearity. Due to the large time and space steps required to maintain numerical stability, a large computer storage and fast processor are required for this analysis. This thesis, however, does not provide for the transient response of the rotor bearing system.

1.5 COMPUTATIONAL APPROACHES

For any rotor dynamic analysis, an extensive amount of data is needed. The modeling of a rotor requires specifications pertaining to its geometry, physical and mechanical properties, support and loading characteristics and bearing properties. Also, different analyses require different kind of information. The rotor design may be altered considering other geometric preferences, manufacturing issues and operating conditions. As a result for a particular analysis, many data files need to be generated containing both input and output information.

The application of this thesis is directed after the development of a comprehensive program, which develops efficiently and accurately the rotor dynamic analysis data. The customary approach first involves the modeling of the rotor bearing system. This involves the division of the rotating shaft into interconnecting stations or elements. Properties of lumped masses representing the large discs or impellers, mass imbalances and natural boundary conditions are to be incorporated into the model. The bearing and foundation supports are to be modeled in terms of their dynamic stiffness and damping coefficients. The forces and moments, if any, are to be made part of the computed file describing the rotor bearing system geometry. This approach would enable a generic file which could be edited according to the specific needs.

After the geometry of the system has been defined, the input-output file needs to be specific for the kind of analysis intended. Such a file system would make sure that different analyses are carried out on the same geometric model or the same analysis could be carried out just by changing the geometry of the model. Studies can be conducted for the influence of projected operating conditions on the dynamic behavior of the system.

The following section gives an overview of one such computer program, "RotBrg©", which this thesis work uses as a processor, with a modified version to suit the requirements of the post-processor. The basic computer program and theory were the subject of Ph.D dissertation of Rouch,K.E.(1977), the comprehensive documentation of which is beyond the scope of this thesis. However, a brief overview is presented in the following section.

1.5.1 RotBrg© PROCESSOR PROGRAM

As stated in the previous section, this thesis work uses a modified version of RotBrg© as its processor. This computer program, which is based on finite element formulation, is used for modeling the Rotor-Bearing-foundation system and run for various analyses effectively thus providing data for the postprocessor.

A key element in RotBrg© is the use of matrix reduction of the rotor element matrices. This allows greater freedom for the analysis in describing the rotor geometry and greater efficiency because only essential degrees of freedom need to be retained for the solution, Rouch, K.E. (2001). The finite element technique is better suited than the transfer matrix technique, which occasionally develops numerical problems, for the structural, thermal and other physical systems. It provides a greater degree of accuracy with fewer stations than the transfer matrix approach, but with some penalty on the computer storage, which is not a limitation on present day computers.

Dynamic reduction is a common method for reducing the size of a finite element problem. It is especially applicable in rotor dynamics because in this application, only rotor matrices are reduced and bearing and foundation matrices are added to the reduced rotor matrices. Thus this application of reduction can be considered sub-structuring or "super element" approach, with the rotor being the substructure. Shaft characteristics can be generated, placed in reduced matrix form, and used repeatedly, in combination with varying bearing properties across a speed range. It also has the advantage for the analyst to be flexible in preparing as detailed a rotor model as is convenient and then retaining only sufficient degrees of freedom to describe the rotor characteristics of interest. In summary, by reduction, a transformation is established between the slave degrees of freedom and retained degrees of freedom based on the principle of minimum potential energy, Rouch, K.E. (2001). This transformation, applied to the system matrices, results in reduced matrices which approximate system characteristics. The reduction process is carried out in conjunction with rotor element assembly to minimize core requirements. Since the mass and damping matrices are modified by this reduction process, it is much more accurate than the static condensation sometimes used in dynamic analysis.

For the calculation of critical speed, the rotor is condensed to user-selected degrees of freedom in the Y-Z plane and natural frequency and mode shape calculations performed at a series of bearing stiffness values. Each bearing may have different values of stiffness, but the usual practice being to use identical stiffness for this analysis. Gyroscopic effects are included for a circular orbit assumption. A second option is a search procedure which iteratively updates bearing properties at each of the critical speeds until convergence is obtained.

For the calculation of synchronous response, for a given unbalance distribution, the rotor response is calculated by assuming synchronous (elliptical) motion, Rouch, K. E. (2001).

$$[\ddot{M}]q + [\dot{\bar{C}}]q + [K]q = F$$

$$\dot{\bar{q}} = j\Omega q$$

$$-\{[M]\Omega^2 + j\omega[C] + [K]\}q = F$$

$$[\bar{D}]\bar{q} = F$$

[D], q and F are complex quantities, but the equation is otherwise equivalent to the standard linear system of equations, and can be solved accordingly. The calculated complex displacements are converted to orbit parameters.

For the stability evaluation, if damping is included in the rotor system and a general sinusoidal motion postulated: $q=q_o e(\lambda+j\omega)t$ where $S_i= \lambda+j\omega$, is the complex frequency. The resulting set of equations is a damped eigen value problem with roots S_i. The mode shapes are then calculated. The real part of the root is called the growth factor, a negative value indicating a stable rotor system.

1.6. SCOPE OF THESIS

Turbomachinery design and analyses requires a rigorous analytical approach. A coherent sequencing of analyses and verifications is required to accomplish the design objectives in a timely and reliable manner, Roso, C. (1995). The ability to analyze the rotor-bearing system at off-design operating and manufacturing conditions enhances the quality of the equipment at its roots. Hence the design of machinery and its operational analysis must posses an inherent stability which insures that the configuration which is manufactured sufficiently close to a known specification will perform according to the design objectives and requirements for a predetermined period of time. Hence, an effective designing program that could perform most of the fundamental analyses by utilizing a common or slightly edited input file, backed up by a post-processor program, can result in an efficient rotor-bearing system. The development in recent years of advanced object oriented programming techniques has eased the design of computer user-friendly interface programs specifically dedicated to the needs of rotor dynamic analyses. This has further facilitated the easy user friendly and improved graphical interface and plotting capability of any kind of rotor dynamic analysis.

The aim of this thesis has been to develop a comprehensive user friendly post processing program to facilitate the analyses of a rotor bearing system. The effort involved a set up program package, which includes the processor for an input file, creation of separate output files for later analysis purpose and the post processor display of the results with convenience of saving as a hard copy or print out the plots and results, thus efficiently handling the designer's development and verification of turbomachinery.

CHAPTER 2

GRAPHICAL USER INTEFACE

A graphical user interface (abbreviated as GUI) is a method of interacting with a computer through the use of graphical images and widgets in addition to text. It can be seen as a program interface that takes advantage of the computer's graphics capabilities to make the program easier to use. This chapter provides an overview of the graphic user interface application. The general advantages and features offered by a GUI package are briefly discussed. A brief description of various GUI options available at hand, is presented. The chapter ends with the selection of the best option to suit the requirements and graphical representation of the work in this thesis. All the packages and softwares discussed in this chapter are trademarks or registered trademarks of their respective companies.

2.1 INTRODUCTION

An individual rotor dynamic analysis of a computer rotor-dynamic system requires an extensive amount of data and comprehensive analysis requires the data to be displayed in the form of plots. The geometry of the rotor may be altered as a consequence of analysis iterations, based on the observations of the plot by the designer. Hence, separate data output files for various rotor dynamical analyses become essential, as also a graphical output feature. The designer would have a chance to browse through the vast amount of data and choose that which he intends to analyze. Given the development of compact, large capacity computer storage devices capable of retaining sizeable amounts of information, the retaining of information is not a concern, as is instead the need to properly channel the data as per requirements and visualization. Hence the necessity to use a GUI arises for the comprehensive post-processor analysis of a rotor bearing system.

2.2 GRAPHICAL USER INTERFACE

Well-designed graphical user interfaces can free the user from learning complex command languages and avoid the need for preparing input in a text-based format. GUIs are used extensively for interaction with computers because of the ease, superiority, efficiency, user friendliness and robustness as they allow the user to interact by manipulation of graphical objects, web resource (http://www.wikipedia.org).

The first graphical user interface was designed by Xerox Corporation's Palo Alto research center in the1970s, but it was not until late1980s and the emergence of the Apple Macintosh that graphical user interfaces became popular. One reason for their slow acceptance was the fact that they require considerable CPU power and high quality monitor, which until then were prohibitively expensive, Alistair D. N. Edwards., (http://www.rit.edu)

In addition to their visual components, graphical user interfaces also make it easier to move data from one application to another. A true GUI includes standard formats for representing text and graphics. Because the formats are well defined, different programs that run under a common GUI can share data. This makes it possible, for example, to copy a graph

created by GUI application into a document created by a word processor. If functionality is what a program actually does, then the interface is how the user interacts with the program and provides input and retrieves output. The hallmark of GUI programming lies in its graphical control features, such as toolbar buttons or icons. Some vital points of a successful GUI are discussed below.

2.2.1 User-Centered Design

In early days of computing, a good software program was one that worked. A great program was one that worked and expended the fewest computing resources. Computer programmers designed programs for use by other computer professionals and not the general public, so a generic text based interface permitting expeditious user input was the order of the day, Cortes, L. (1997). But in today's world, in which computer resources are abundant and computer users are usually non-programmers and computer neophytes, a good program is one that not only works but also is easy to learn. Hence the focus shifted onto user centric applications. Unlike problem-centered programming, which has first and foremost focus on the task, user centered design begins with observing how users confront present manual and automated methods. The goal, under this design, would be to understand users' work tasks, their mental models of those tasks, and the tools already familiar to them.

2.2.2 Event-driven Programming

In event-driven programming, tasks are performed in a predictable fashion, one step following the other. The GUI permits event-driven programming, and therein lies another part of its strength. In event-driven programming, the user controls the program's tasks via GUI events—entries using a keystroke, mouse click, penlight, etc., the variety of events a user can generate are set by the hardware and operating system. Both procedure and event driven programming restrict the user's possible choices, but only the latter gives the user control over several task steps, Cortes, L. (1997). Procedure and event driven programming both arrive at the same calculations, but differ in how the user interacts with the program to create and process the output results. The user experiences a variety of ways to enter data on any user centric form and as a result of the complex algorithms working in the background, perceives the software as user friendly and simple to use.

2.2.3 Portability of Applications

Portability of applications across different platforms is a subject that has attracted a lot of attention for some time. Until recently, it has been quite difficult or expensive for an application with a graphical user interface (GUI) to be truly portable, Cortes, L. (1997). Portability requires a certain amount of conformance of operating systems, programming languages and tools. This means in most cases that a GUI is limited to a certain environment, for example, to the Windows platform and Linux. In addition, a new version of any of the underlying ingredients may break the original GUI. Things become more difficult if the application must be portable across very different platforms, such as UNIX and Microsoft Windows. One approach is to make the interface layer as 'thin' as possible. The vast majority of the code is in the fully portable 'engine' and the interface code is an easily separable module, with a version for each of the supported platforms, Cortes, L. (1997). This does require separate source code for each supported platform, with the risk that all code will not be maintained in step, leading to divergence of versions.

If a new application is being written, rather than an existing one being ported, the most attractive method of creating a portable GUI is to use a portable library. This can then be used to create user interface components on all target platforms. Portability toolkits tend to take either a 'lowest common denominator' approach, providing only those interface components that appear on all platforms, or attempt to provide all widgets on all platforms, often by supplying their own widget set on all platforms, Cortes, L. (1997). One example of the former approach is the XVT toolkit. This is supplied as a set of libraries that must be linked with the application code, which can be C or C++. The application makes calls to the same XVT functions, whatever the destination platform. The code is then linked with the library for the appropriate platform. Hence, proper care must be taken before a GUI is designed to have the portability incorporated.

2.2.4 Templates Usage

Rather than individually customizing each screen, global templates could be created, that would convert screens in large to GUI format, which still leaves with the opportunity to customize individual screens when necessary. The more powerful GUI tools also let the developer customize the screen by adding visual and functional enhancements such as pop-up windows, radio buttons, list boxes, pull-down menus, toolbars, background graphics and logos. Colors and fonts can be changed. Function keys can be emulated with buttons or other controls. The new displays are typically much more intuitive and easier to use than the original text displays. To this point, GUI tools have focused on transforming existing text screens to look and function like a program developed from scratch for the Web. The new generation of GUI tools provide scripting tools, which make it possible to go one step further, by improving the productivity of the user interface, web resource (http://www.webopedia.com). Scripts can be developed to enter data and perform the functions of another application on an automated basis. The commands can be activated and controlled by the user through a more intuitive graphical interface while the text screens are hidden from view. The script can perform repetitive tasks and make decisions based on predetermined rules. It can exchange information with other applications such as databases, spreadsheets and word processors.

In general, some common objectives ought to be met, to lead to a good GUI design. The user must be able to anticipate a widget's behavior from its visual properties (Principle of consistency at widget level). Widgets in this context refer to visual controls such as buttons, menus, check boxes, scroll bars and rulers. The user must be able to anticipate the behavior of the program using knowledge gained from other programs (Principle of consistency at platform level). This refers to abstractions such as mouse gestures, placement of menus, icons and toolbar glyphs. A good GUI would help users enter appropriate data. If the program requires formatted data (numeric range or alphanumeric only), bounded input widgets could be used to appropriately limit the user's input choices. If a certain program step cannot be legitimately performed until the user completes other steps, the dependent steps can be disabled until all its dependencies are satisfied, Bernard J. J. (1998), (http://jimjansen.tripod.com). Every screen should be so designed that a novice can easily tell what steps, especially critical ones, have been performed. A good GUI application should be self evident, with required help materials.

Incorporating suitable warning and error messages force the user to address the critical issues before returning to the task. Finally, it is important to design the interface so that the users can accomplish their tasks while being minimally aware of the interface itself.

However, in spite of having a good GUI capability, a software is deemed of not having a good functionality if it does not support extensive computer graphics. Some of the vital features of a GUI with good graphics feature are:

- Render plotting
- Interactivity
- Real-time manipulation
- Scientific visualization
- Storage of plots/images in memory or on disk

All the above graphical features coupled with a good interface make a good package for general industrial usage environment. The next step would be to find a suitable programming language/package that fulfils all the above requirements with respect to Graphical user interface.

2.3 GRAPHICAL USER INTERFACE OPTIONS AVAILABLE

There are a lot of programming packages available currently, which offer a wide range of capabilities in terms of speed, extensive graphics, security, performance and compatibility. The following section covers some of the options available for building a comprehensive Graphical interface along with other embedded features.

2.3.1 C/C++

Although building graphics in C language is feasible, it requires a lot of effort on the programmer's part for that. C has a rich collection of libraries and a variety of toolkits for building a graphical interface. These toolkits are particularly valuable for C and add ever-expanding functional richness to the language. Most of the GUI functionalities can be incorporated using C graphics like the pop up menu, location cross hair, etc., Johnson, N. (1987).

Graphics programming in C involves, perhaps more than non-graphics programming, a great deal of memory management. A lot of structured declaration is needed to keep track of memory, which is not easily comprehended. A lot of syntactic declaration and coding is required for a simple interface. The language is known for high performance.

The same explanation holds true even in the case of C++, except for its enhancements over its predecessor C with respect to usage of object oriented features. However, there had been no significant improvement in the graphics it offered, Stevens, A. (2000). Because C++ blended the high efficiency and stylistic elements of C with the object-oriented paradigm, it was a language that could not be used to create a wide range of programs.

2.3.2 Visual Basic

As the name suggests, a major portion of the programming with Visual Basic (VB) is accomplished visually. This means that during design time, the designer would be able to see how the program will look during runtime. This is a great advantage over other programming languages, because the designer is able to change and experiment with the design until satisfied with the colors, sizes, images and other features included in the program. Visual Basic is completely graphically oriented, so it enables to directly create windows, menus, buttons, etc.,

and what need to be written are the codes that process system messages and events. Several designer friendly features like the capability of developing 'Hot Keys', automatic compiling for 'Reserve words' during design time, etc., have made it a qualified option. Already enhanced with the power and elegance of many high-level languages, VB's most outstanding feature is the ability to generate with ease a Windows graphical user interface - an interface that is not restricted to pure alphanumeric display. Perhaps the most powerful feature of Visual Basic is its capability of incorporating third-party controls, Gurewich, N., Gurewich, O. (1997), (known as OCX ActiveX controls), which extend the add-on features like multimedia and extensive graphics. With the introduction of a compiler and its enhanced controls, Visual Basic applications can hold their own against C++ applications for speed and extensibility, Donald, R.P., Oancea, G. (1999). Even though Visual Basic (version 6.0) can create blazing fast applications, it is best known for its capability to create Internet and client/server applications very quickly. Visual Basic can create programs that interact with files, databases, interact with the Internet, and even interact with hardware. The industrial-strength development environment is suitable for almost any type of Windows application. The files processing feature of Visual Basic enables creation, manipulation, and storage of large amounts of data, access several sets of data at once, and share data with other programs.

The major disadvantage of Visual Basic is that the implementation of powerful features requires add-ons. These add-ons are called VBX files.

2.3.3 Visual Basic.NET

The VB.NET language is disarmingly simple and is highly expressive when it comes to implementing modern programming concepts. VB.NET includes all the support for structured, component-based, object oriented programming that one expects of a modern language. Object Oriented Programming (OOP) is a vast topic in itself, which is beyond the subject of this thesis. This concept is popular in the current programming due to its techniques to help manage the complexity of long codes. The goal of VB.NET is to provide a simple, safe, object-oriented, internet-centric, high performance language for .NET development, Liberty, J. (2002). It is simple because there are relatively few keywords. This makes it easy to learn and easy to adapt to a programmer's specific needs. One more merit of VB.NET is that it is considered safe because it provides support in the language to find bugs early in the development process itself.

This makes for code, that is easier to maintain and programs that are more reliable. It frees the programmer from having to deal with the mundane complexities of writing windows programs, and instead concentrate on solving problems.

One of the attracting features that VB.NET enjoys is that it has a number of features which makes it backward compatible with Visual Basic 6.0. Other features have been added specifically to adapt visual basic to object oriented programming and to the .NET platform. Its support feature to find bugs early in the development process makes for the easier maintenance and reliability of the code. It also does not support many features like pointers that make for unsafe code. It goes beyond traditional windows programming to facilitate creating web applications, quickly and easily. VB.NET can be used to develop three types of applications to be run on a windows computer, Reynolds, M., Blair, R., Crossland, J. (2003):

A. Console applications displaying no graphics

B. Windows applications using the standard windows interface

C. Web applications that can be accessed using a browser

The main drawback of VB.NET, however, is that it is a version of Visual Basic specifically written for .NET. While .NET is developed to become cross-platform, the overwhelming majority of these programs are written to run on a machine running Windows. The .NET platform is web-centric.

2.3.4 C#

C# is a programming language that was developed specifically for the purpose of writing applications for the .NET platform. It embodies most of the positive features of C++, Java and Visual Basic. C# is used to create the same applications as described previously for VB.NET. Since the language was built on the shoulders of C++ and Java, all the features such as modern object oriented and component based programming concepts are enabled. To build and run C# programs, Windows 2000, IE5.5, Microsoft .NET SDK, text editor and an optional Visual Studio .NET are required, Drayton, P., Albahari, B., Neward, T. (2002).

The goal of C# is to provide a simple, safe, object oriented, internet-centric, high performance language for the .NET development. C# is considered simple due to its relatively few keywords, which make sit easy to learn and easy to adapt to the programmer's specific needs. Like VB.NET, C# also is considered safe because it provides support in the language to find bugs early in the development process making it easier in maintenance and more reliable. C# is considered as a next major step in the evolution of component-based development languages, Liberty, J. (2002).

The demerits associated with C# are the same as VB.NET. C# was designed for developing web and web-aware programs. In other words, it is most useful for creating web applications.

2.3.5 Java

The difficulties with C and C++ (and most other languages) being designed to be compiled for a specific target, led to the creation of Java. Java has gained a wide and rapidly growing popularity and acceptance as the programming language of the World Wide Web since its first release in 1995 by Sun Microsystems, Naughton, P., Schildt, H. (1999). Java is an object-oriented language that is secure, robust, portable, simple, multi-threaded and distributable. Java comes with a large library of tools useful for many areas of distributed and internet computing. Java is, more than just a language for writing applets (small programs that are downloaded from a server and execute in a client browser, independent of the server's and client's platforms), Naughton, P., Schildt, H. (1999). It is a general programming language whose library contains a rich set of tools to create GUIs. The first versions of Java provided only basic GUI tools like lists, radio buttons and the like, but the latest release of the Java libraries has a much larger set of tools that include tables and trees, various forms of buttons for toolbars, sliders, progress bars, etc. Java rivals the capabilities of existing GUI interfaces (Visual Basic on Windows, X/Motif on Unix) but with the distinct advantage of being portable. In many cases Java is also faster than competing products as compiler technology takes advantage of the built-in safety features of Java.

A special feature of Java is that the programmer, independently of the underlying platform, can choose the “look-and-feel” of the GUI. Earlier versions of Java GUIs always reflected the platform; the same Java application would use the Microsoft Windows "look-and-feel" on Windows platforms while it appeared like an X/Motif GUI on Unix machines. With the most recent Java version one can choose either a Windows, X/Motif, a special Java or even a user-defined "look-and-feel" on all platforms, Naughton, P., Schildt, H. (1999). The AWT (Abstract Window Toolkit) package, used to create applet windows, is used for the creation of stand-alone windows that run in a GUI environment.

While the Java windowing environment is portable across all platforms, its performance and capabilities are at present limited to least-common-denominator functionality, Naughton, P., Schildt, H. (1999). This means it is not possible to provide a fully functional, industrial strength GUI interface at present, by using Java. This is a serious limitation, since experience shows that the last few percent in functionality of a GUI interface has a very large effect on the perceived ease of use. The main disadvantage of Java is speed. Java, and the AWT, probably does not provide the performance required for "real-time" displays as required, for example, by rapidly updating histograms and plots. Although Java's ability for producing portable, architecturally neutral code is desirable, the method used to create this code is inefficient. Once Java code is compiled into byte code, an interpreter called a Java Virtual Machine (which accounts for Java’s versatile portability), specifically designed for a computer architecture, runs the program, which makes the application slow. Java, being an interpreted system, is currently an order of magnitude slower than C.

2.4.1 SELECTED OPTION

Building a post-processor graphical interface for a comprehensive rotor dynamic analysis requires taking into account certain important features and neglecting certain inconsequent features. A post processor program would need to access the output files from the processor, for further analysis. The program would be required to have all the regular widgets such as buttons, pull down menus, choice buttons, suitable plotting output, printing capability, feature for saving the plots, etc. Taking into consideration all the above-discussed facts about the graphical user interface requirements and various available software/package options, it is concluded that the optimal option would be using Visual Basic 6.0 for the purpose.

As previously discussed, C and C++ are poor in the capabilities of building graphic interface. Also, the implementation time would be high. Due to the requirement of a secondary scientific plotting package, this option was ruled out. Visual Basic.NET, though advanced and conforming to all the requirements in terms of simplicity and features, does not suit well for a normal industry environment package. Also, the .NET platform is a disadvantage in itself along with its exorbitant cost. C#, with almost the same features as Visual Basic.NET was eliminated on the same basis. Java, with its extensive set of toolkits and graphic libraries is a good option for the GUI. But, its speed, when taken into account the file accessing, and the code involved in building the whole interface, made it a difficult choice. The scientific plotting capabilities of Java, which are not advanced, have ruled it out of contention.

On the other hand, Visual Basic 6.0 with its rich set of controls and built in windows toolkits made it a frontrunner. The sequential file format of the output files from the preprocessor makes it easier for Visual Basic interpretation. The plotting feature of Visual Basic, though not very advanced, was easily used for the purpose, thus eliminating the need for any other plotting package. The rapid development of the interface is a positive feature. Further significant advantage of Visual Basic over other programming packages is the ability to modify and change the interface during design time itself. Utilizing the ActiveX controls, some of the advanced features were easily incorporated.

Hence, the selection of Visual Basic 6.0 for the purpose of creating the user interface proved to be extremely satisfactory.

CHAPTER 3
PLOTTING AND CURVE FITTING

For the rotor dynamic analyses, the postprocessor needs to have the capability to make plots, display them on the screen and save to a file in a compatible format. A further enhancement would be to print the plots to a default printer. It would not be an efficient use of manpower to write a custom set of plotting tools if the designer could find a package that is available that satisfies the need. The selection of a suitable tool for plotting the results for various analyses is discussed in this chapter.

In the plotting of the mode shapes in the stability analysis and the critical response analysis, one of the most critical issues would be to select the curve fitting technique. A discussion on the appropriate curve fitting is discussed to the end of this chapter. All the packages and softwares discussed in this chapter are trademarks or registered trademarks of their respective companies.

3.1 INTRODUCTION

It seems quite clear that the choice of package is intimately related to the scope of what and how much of detail the designer wants to have. For example, if the designer intends to have the tools to just put up plots and images with minimal interactive analysis then the choice would only be limited to some basic Image and Plotting Software (IPS) or if the need is to have more complicated interactions with the user and more extensive image manipulation (e.g. rotatable images), then something more like a graphic toolkit would be desirable. Some of the basic requirements of IPS for science analysis graphics as listed by the user interface committee are as follows:

- The IPS must be available on all supported platforms.
- Any external libraries used by the IPS must be well supported, tested, and free.
- The IPS must be extensible, allowing the creation of custom widgets.
- The IPS must be simple to integrate or install, preferably through a binary distribution for end-users.
- The IPS must be able to generate publication quality images, including PostScript.
- The IPS must allow export of images into standard browser supported formats (e.g., BMP, GIF, JPEG, etc.)
- Plots shall be modifiable, rather than forcing regeneration of plots from scratch.
- The IPS must support overlaying of images, contours and plotted points.
- The display rate should be less than about 2 seconds for tasks that are not data intensive.
- The IPS must provide for returning interactive graphical inputs such as cursor position.

In accordance with the plotting requirements of various rotor dynamic analyses, it is deemed not necessary to include graphic toolkits. Hence, the need for the post processing analysis would be a basic image and plotting package. The following section deals with various options of plotting packages. Their merits and demerits have been discussed accordingly.

3.2 PLOTTING OPTIONS

As discussed previously, the selected post processing plotting package option must satisfy the basic norms. Henceforth, some of the plotting options would be discussed along with their applicability for rotor dynamic analysis.

3.2.1 C/C++

As discussed in the previous chapter, C and C++ languages share a low level of computer graphics and plotting capability relative to other specifically developed packages. Rather, there are a lot of packages developed that could be interfaced with C/C++ for graphic and plotting enhancement, which also provide capability to I/O files, Johnson, N. (1987). The requirement of long syntactic declaration, coding and exhaustive memory management makes this option hard to manage.

With respect to rotor dynamic postprocessor, the analyses requires good I/O file management, plotting compatibility with GUI and plot output streaming. Hence, this option was ruled out for the purpose.

3.2.2 Visual Basic

Visual Basic, with its rich collection of graphic interface tools has good plotting capability. Though it does not posses full scientific image and plotting resources, suitable coding can help achieve the purpose. With the help of necessary toolkits and library routines, Visual Basic can be fully extended to the range of a satisfactory plotting tool. The easy I/O file handling is a plus, Donald, R.P., Oancea, G. (1999). After the plotting is completed, it can be channeled into an editable file or printer. Hence, all its functionalities make it a versatile plotting option.

Using Visual Basic for rotor dynamic plotting purpose could be a satisfactory solution, except that fairly long codes are needed for simple plotting. Since there would be no need for a highly advanced image manipulation and 3-D plotting, Visual Basic offers a comprehensive choice for plots. It also offers a cost effective solution with its many advantages.

3.2.3 Visual Basic.NET/ C#

The features discussed previously for Visual Basic holds good for Visual Basic.NET and C#. The only difference and drawback is that both these options are specific for .NET environment. The web centric software can be suitably used for plotting with the usage of appropriate APIs, Reynolds, M., Blair, R., Crossland, J. (2003).

Visual Basic.NET or C# could be used as a plotting option for rotor dynamic postprocessor analysis, but its web centric approach would not make it a plausible solution for industrial applications. A lot of capabilities of these software packages are deemed unnecessary when it comes to its application for rotor dynamic analyses.

3.2.4 Java

Java has a large library of tools for plotting. Its graphic interface capability coupled with the sets of plotting functions makes Java a good choice. The inbuilt routines for various functionalities like plotting, printing, exporting files and various saving formats has a great edge over other options. Java has a rich collection of in built objects typically used for any graphic user interface, Naughton, P., Schildt, H. (1999). Though a bit long lines of code are required for plotting purpose, Java quite effectively serves the purpose.

Using Java for plotting, though serves the purpose, the speed of the application is reduced quite a bit. Also, this internet centric language does not seem to hold a satisfactory option for an industrial application like rotor dynamic analyses.

3.2.5 SciPlot

The SciPlot toolbox provides a graphic building block for any scientific/engineering development platforms for numerical or statistical analysis and publication of scientific result applications. The SciPlot library provides a basic set of graphics routines for scientific and engineering plotting. Graphs can be generated interactively in a client window for quick preview. Final hardcopy plots can be produced on any windows-supported and graphics capable printer. The various functions provided by SciPlot are axis tilting, numeric label generation, number conversion, data array plotting, 3D surface contour plotting and plot annotation routines to generate plots with linear and log axis scales. SciPlot supports graphic application development

through the use of the Microsoft Windows API. Both static and dynamic linked applications are supported, MicroGlyph/SciPlot™ Graphics Library Documentation Manual.

SciPlot has its demerits in its applicability. One of the main drawbacks is its non-compatibility with some C/FORTRAN compilers. Also, its limited support of color modes makes it a less sought option. The need to include a separate graphics library for the plotting purpose, which could be eliminated with some other options, makes it a disadvantage.

3.2.6 Gnuplot

Gnuplot is a commonly available plotting package. It is a free X window application with command line interface, web resource (http://www.ucc.ie). With a reasonable control of plot appearance, its output quality is quite satisfactory. Gnuplot is commonly used for producing 2D and 3D plots, although it doesn't have as many features as some of the other commercial mathematical software available. It is not as complex to use as packages such as Mathematica or Matlab. It is ideal for users who only need to a plot a graph and don't want to learn a major tool. A nice thing about doing the plot interface this way is that it is not too difficult to maintain or extend. It can be used with the output for many printers and other plotting devices, like PostScript, LaTeX, HPGL, MetaFont, etc., web resource (http://scv.bu.edu)

Some of the disadvantages of using Gnuplot are that the plotting must be done with temporary files, since Gnuplot cannot read data from standard input and lacks some important features, primarily multiple plots per page. Interactive plotting with Gnuplot requires the operating system to be multi-tasking.

Gnuplot can be satisfactorily used for plotting rotor dynamic analyses plots. But, its lacking applicability with other front-end softwares such as Visual Basic or Java makes it a less sought option.

3.2.7 Tecplot

Tecplot has powerful capabilities in three key areas: plotting, data managements, and the user interface. These provide more control to explore, analyze and communicate results. The various merits of Tecplot in plotting are its ability in line plots, log-log plots, in-built spline and curve fits. Tecplot has various plotting options like symbol properties, labels, grid spacing, axes, colors, titles, legends, and fonts and multiple horizontal and vertical axes. It has features like data text loader, supplement auxiliary data information and image import. Customized options to output data and plots in various formats is a plus point for Tecplot. Its graphic user interface has an added advantage, web resource (www.tecplot.com).

The main disadvantage of using Tecplot is its cost, when using as a supplement for plotting purpose, with any other graphical user interface designed specifically for rotor dynamic analysis. Its various advanced functionalities are not needed for rotor dynamic specific analyses. The requirement for setting up of different software for plotting purpose only makes it a serious setback.

3.2.8 Mathematica

Mathematica is also an X windows application with command line interface and good output plot quality, web resource (http://scv.bu.edu). Mathematica seamlessly integrates a numeric and symbolic computational engine, graphics system, programming language, documentation system, and an advanced connectivity to other applications. Some of the advantages of using Mathematica come by its handling complex symbolic calculations that often involve hundreds of thousands or millions of terms, loading, analyzing, and visualizing data, Solving equations, differential equations, and minimization problems numerically or symbolically and doing numerical modeling and simulations, ranging from simple control systems to galaxy collisions, financial derivatives, complex biological systems, chemical reactions, environmental impact studies, and magnetic fields in particle accelerators, web resource (http://www.wolfram.com).

Although Mathematica is very powerful, it is difficult to use. Since the rotor dynamic analysis does not need the full-fledged features, it would not be worth the time investment to learn and use Mathematica without doing a significant amount with it. Hence, this option is, rather, taken off contention.

3.2.9 Fortner Dataplot

DataPlot, a plotting package produced by Fortner Research LLC, is an easy-to-use X Windows application that can produce line graphs, including multiple graphs per plot, error bars, and log axes, as well as parametric plots and scatter plots. The point-and-click driven interface allows easy changes to titles, graph annotation, axis and tick-mark labeling, labeling, font selection, and color. There are various ways to bring data into the program, which can be kept and manipulated in a spreadsheet style database or file. Data is selected from the rows and/or columns of this file, and graphs are produced in X Windows. The results can be saved as PostScript files and printed on printers. Plots can also be saved as raster image file. The highlight of Dataplot being its easiness to learn or use, and obtain high quality postscript output for the most common and simple graphs, web resource (http://www.scv.bu.edu).

Dataplot proves to be an option that has far more features than are required for a normal rotor dynamic analysis. The compatibility with other graphic interface and cost make it a less preferred option.

3.2.10 Excel/Spreadsheet

Excel is an extremely versatile and useful tool with its chief advantage over other commercially available softwares being that it can be easily adapted for a variety of purposes and that it is available on most PCs at no extra cost. It has certain advantages like, being able to plot data with curve fit, a useful searchable database, simulations and interactive worksheets. Excel can incorporate multiple data sets onto the same plot for comparison. Templates can be created in Excel to automatically perform calculations on the data that are retrieved. The runtime context sensitive help makes it a good option.

Excel can only plot data values from tables. This means that, to plot any data set or any function, the information must first be inserted or generated in tabular form, although once a table of data exists, the plotting of the data is straightforward. Hence, a completely different set up for plotting and saving makes it unpractical to merge it with any other user interface program. Also, the various specific plots in rotor dynamic analysis cannot be realized effectively in Excel.

3.3 SELECTED OPTION

The plotting of data for various rotor dynamic analyses requires certain basic functionalities. The processor, which this thesis uses, outputs the data results into separate formatted files for the respective analyses. Any plotting option would need to use these files, perform some conditional branching, and plot the appropriate plots with curve fitting, if necessary. The user would also be needed to give options to either save it in one of the standard formats or print it out for further analysis. Also, embedding the processor with the graphical user interface to run the input files would cut unnecessary steps for a separate processor run. Along with these requirements, the plotting functionality within the graphical user environment would be an interesting feature. Taking all the above requirements along with cost constraint and other features like plotting controls, and an easy setup installation wizard, makes Visual Basic the best of all the choices.

As discussed previously, C and C++ options have a lower functionality with plot controls and a very native graphic interface. C# and Visual basic.NET being more of web centric applications and its unnecessary advanced features, with respect to rotor dynamic post processing, has made it a demerit in itself. Also, the .NET platform, with high cost, ruled them out for the purpose. Java, for its low speed and web centric was ruled out. SciPlot and Gnuplot, being impressive for plotting purpose but having limited user options as an interface and compatibility, web resource (http://scv.bu.edu), with other programs, were ruled out. Tecplot, though advanced, was not considered due to its graphic interface not confirming for rotor dynamic post processing application. Fortner Dataplot, Mathematica and Excel spreadsheet, though share to be a very good plotting option, the user interface is not suited for rotor dynamic analyses and also cannot be combined with other graphic interface.

Visual Basic has a rich plotting controls set. Its ability to have plotting options combined into the graphical interface to give a wholesome look for rotor dynamic analyses makes it a better option. Visual Basic has file writing and accessing features, which makes it suitable to use it in conjunction with RotBrg processor, which this thesis uses, and its output files. Lastly, the plot saving and printing features, using the active X controls, makes it the best of all the options. There are various other features that are well suited for a rotor dynamic analyses post processor.

Hence, Visual basic 6.0 has been chosen for both graphic user interface and plotting purposes and has proved to be extremely satisfactory.

3.4 CURVE FITTING

Applications of numerical techniques in engineering often involve curve fitting of experimental data. Its importance comes in data handling because it produces a mathematical model of the data that can compactly contain and represent its primary properties, Kincaid, D., Cheney, W. (1996). With respect to rotor dynamic application, curve fitting is necessary for its data to be analyzed in the form of plots. For a designer to modify or obtain an optimum rotor-bearing model, the fitting of a polynomial curve to the set of result data points becomes necessary.

For the purpose of curve fit between the data points, a cubic spline method has been chosen. The spline logic, described below, has been dissolved into computer equations in a separate subroutine. It is attached as an appendix to this thesis.

A spline consists of polynomial pieces on subintervals joined together with certain continuity conditions. A cubic spline is a spline constructed of piecewise third-order polynomials which pass through a set of ‘m’ control points. It is as shown in the figure below, Weisstein, E.W. (1999).

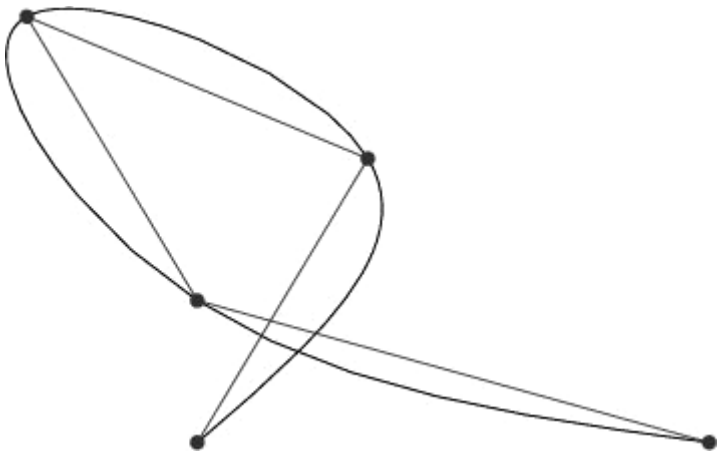


Fig 3.1. A typical cubic spline curve

The second derivative of each polynomial is commonly set to zero at the endpoints, since this provides a boundary condition that completes the system of ‘m-2’ equations. This produces a so-called "natural" cubic spline and leads to a simple tridiagonal system which can be solved easily to give the coefficients of the polynomials, Mathews, J. H., Fink, K. F. (1999). However, this choice is not the only one possible and other boundary conditions can be used instead.

Considering a 1-dimensional spline for a set of n+1 points (y_0,y_1,…,y_n), the i^{th} piece of spline can be represented as

$$Y_i(t) = a_i + b_i t + c_i t^2 + d_i t^3 \quad (1)$$

where t is a parameter $t \in [0,1]$ and i =0,1,…,n-1.

$$Y_i(0) = y_i = a_i \quad (2)$$

$$Y_i(1) = y_{i+1} = a_i + b_i + c_i + d_i. \quad (3)$$

Taking the derivative of $y_i(t)$ in each interval then gives

$$Y_i'(0) = D_i = b_i \quad (4)$$

$$Y_i'(1) = D_{i+1} = b_i + 2c_i + 3d_i \quad (5)$$

Solving equations (2)-(5) for a_i ,b_i , c_i and d_i gives

$$a_i = y_i \quad (6)$$

$$b_i = D_i \quad (7)$$

$$c_i = 3(y_{i+1} - y_i) - 2D_i - D_{i+1} \quad (8)$$

$$d_i = 2(y_i - y_{i-1}) + D_i + D_{i+1} \quad (9)$$

Now it requires that the second derivatives also match at the points, so

$$Y_{i-1}(1) = y_i \quad (10)$$

$$Y_{i-1}'(1) = Y_i'(0) \quad (11)$$

$$Y_i(0) = y_i \quad (12)$$

$$Y_{i-1}''(1) = i''(0) \quad (13)$$

for interior points, as well as end points satisfy

$$Y_0(0) = y_0 \quad (14)$$

$$Y_{n-1}(1) = y_n \quad (15)$$

This give s a total of 4(n-1)+2 =4n-2 equations for the 4n unknowns. To obtain two more conditions, it requires that the second derivatives at the end points be zero. This is one of the five end-point constraints chosen for the purpose. Hence

$$Y_0''(0) = 0 \quad (16)$$

$$Y_{n-1}''(1) = 0 \quad (17)$$

Rearranging all these equations leads to the following symmetric tridiagonal system:

$$\begin{bmatrix} 2 & 1 & & & & & 1 \\ 1 & 4 & 1 & & & & \\ & 1 & 4 & 1 & & & \\ & & 1 & 4 & 1 & & \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\ & & & & 1 & 4 & 1 \\ 1 & & & & & 1 & 4 \end{bmatrix} \begin{bmatrix} D_0 \\ D_1 \\ D_2 \\ D_3 \\ \vdots \\ D_{n-1} \\ D_n \end{bmatrix} = \begin{bmatrix} 3(y_1 - y_n) \\ 3(y_2 - y_0) \\ 3(y_3 - y_1) \\ \vdots \\ 3(y_{n-1} - y_{n-3}) \\ 3(y_n - y_{n-2}) \\ 3(y_0 - y_{n-1}) \end{bmatrix}$$

The above equation applies to an open curve, which is the case in rotor dynamic analysis. The above system is strictly diagonally dominant and has a unique solution. The coefficients $D_0, D_1, D_2, \ldots, D_{n-1}, D_n$ and other spline coefficients are computed in a subroutine. The above system is broke up to make suitable equations for computer calculation, thus giving coefficients for a smooth cubic spline curve. The cubic spline subroutine written in Visual Basic 6.0, has been attached in the appendix section for reference.

CHAPTER 4
PROGRAM SCHEMATIC AND IMPLEMENTATION

This chapter provides an overview and schematic of the post processor program - 'RotPlot©'. The organization and mode of operation of the program are intuitive and user interactive. The salient features of the program in the form of screen shots and the flow chart of the program are presented. It includes the description of the installation of the program, which enables to transfer the code from a removable media to a computer hard disc in a user friendly manner. The chapter ends with a section containing sample plots of different post-processor analyses results, of a particular rotor design.

4.1 FILE STRUCTURE

The rotor dynamic analysis of a complex rotor bearing system requires an extensive amount of data. A comprehensive rotor bearing system may be subject to various analyses, as different design considerations and manufacturing issues must be evaluated with different geometry iterations. Hence, for a particular study, many data files will be generated which may contain both input and output information. Due to availability of large capacity computer storage, the need to keep track of data in an efficiently organized way is more of a concern than of the memory storage requirement.

A rotor bearing foundation is generally subjected to three kinds of analysis viz., undamped critical speed analysis, stability analysis and the synchronous response analysis. The subject program of this thesis has the option to carry out the above mentioned analyses. The program requires a separate input file, with pertinent information such as rotational speeds, station information, loads and other rotor model identification code for each of the analysis intended. But only a slight change in the input file is needed to change the analysis type. The output files generated also depend on the selected analysis type. The evaluation of output analysis files is enhanced by all the related files being identified by the same input file name, but with different extensions, appropriate to the analysis carried out. This facilitates the archiving process of analysis, since all the files of interest can easily be traced to the name of the analysis input file. Since these output files are text based, in format, they can be printed or imported into other compatible programs.

Irrespective of the type of analysis intended, the input file for the program bears the extension ‘IN’. The following tables show the various files generated by the post processor analysis program and a brief description of its content.

Extension	Brief Content Description
OUT	Comprehensive analysis information as output
SYR	Response amplitude, phase and orbit data at various speeds and stations
SYB	Bearing response amplitude and phase at various speed and bearings
SYF	Response amplitude and phase data at foundation level station locations
BRL	Bearing parameters and rotor location data

Table 4.1 Synchronous Response Analysis Output Files

Extension	Brief Content Description
OUT	Comprehensive analysis information as output
STY	Comprehensive Mode parameters and related data for all the specified modes
BRL	Bearing parameters and rotor location data

Table 4.2 Stability Analysis Output Files

Extension	Brief Content Description
OUT	Comprehensive analysis information as output
CRM	Critical speeds and its associated modes information
CTR	Critical speed map data (Bearing stiffness associated at various speeds)
SFN	Bearing stiffness data in study
BRL	Bearing parameters and rotor location data

Table 4.3 Critical Speed Analysis Output Files

4.2 PROGRAM SCHEMATIC OF ROTORDYNAMIC POST-PROCESSOR

Once the post processor program is installed on a computer, from the setup package, the main selection menu is displayed as seen in Fig. 4.5. The analyst can access the available options using the keyboard or any pointing device. Upon selection of an option, subsequent menus are displayed, guiding the user through the process to the ultimate plots. The program automatically sets default parameters such as sub directory selection, default file extension setting depending on the analysis, output plots saving menu and others, relieving the analyst from performing tasks that are peripheral to the main objective of analysis. All the related analyses output files would be automatically generated in the same directory where the input file is located, for the user's convenience for further analysis.

Fig 4.1 illustrates a schematic of the organization and mode of operation of the program developed. Fig. 4.2 through Fig 4.31 graphically illustrate the various steps and options prior to the display of the analysis plots, upon which the user has the opportunity to either save a plot to a convenient file, copy it to the clipboard or print it to a default printer as a hard copy, for future reformatting and analysis. The illustrations are typically from a computer with Windows XP – SP1 operating system environment, the look and shape may vary depending on the operating system and preloaded theme settings.

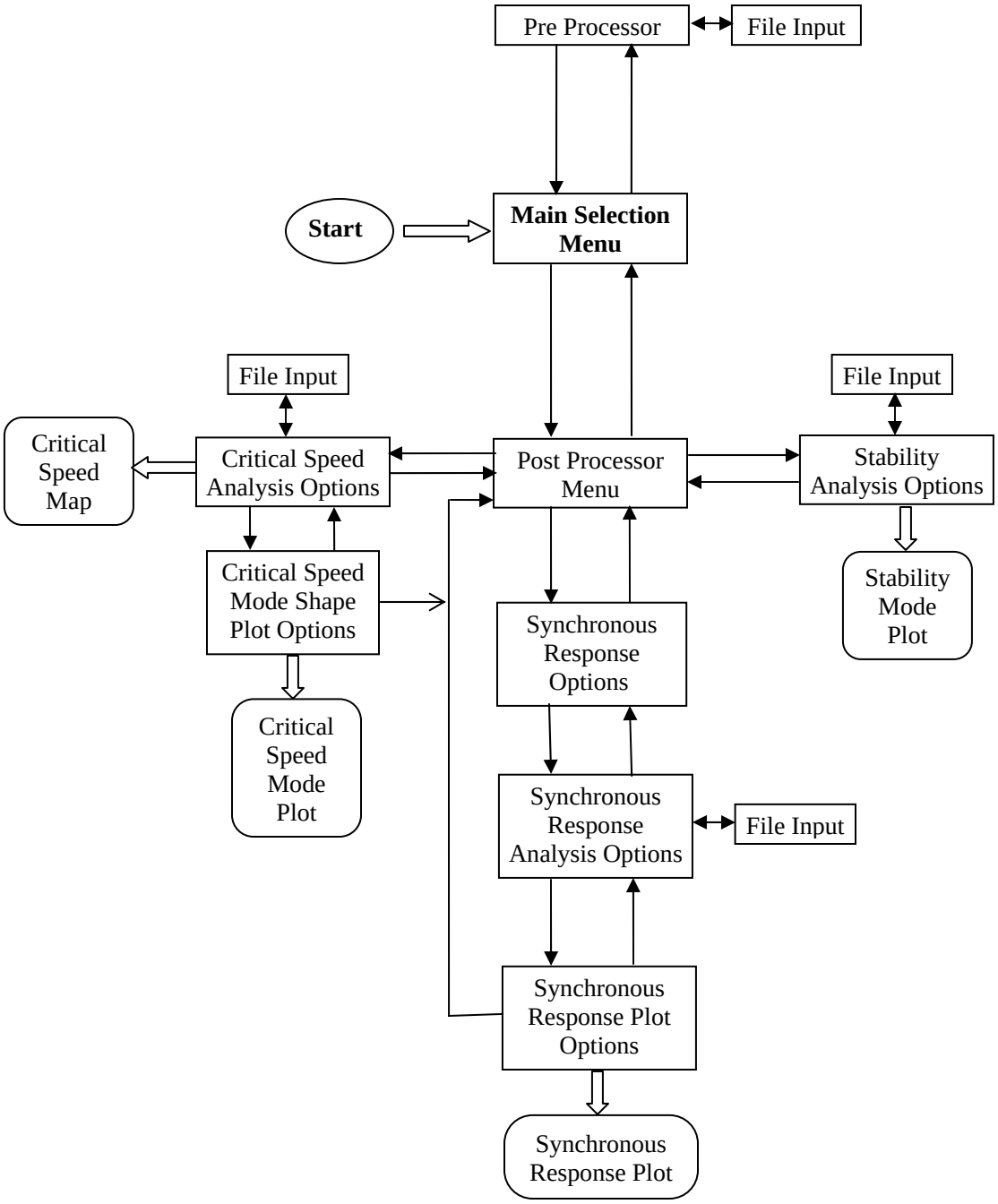


Fig. 4.1 RotPlot© Program schematic overview

Fig 4.2 shows a typical program loading screen whenever the user starts RotPlot. The progressive bar indicates the level of loading completion of the program and the main menu is displayed immediately after the program has completely loaded.

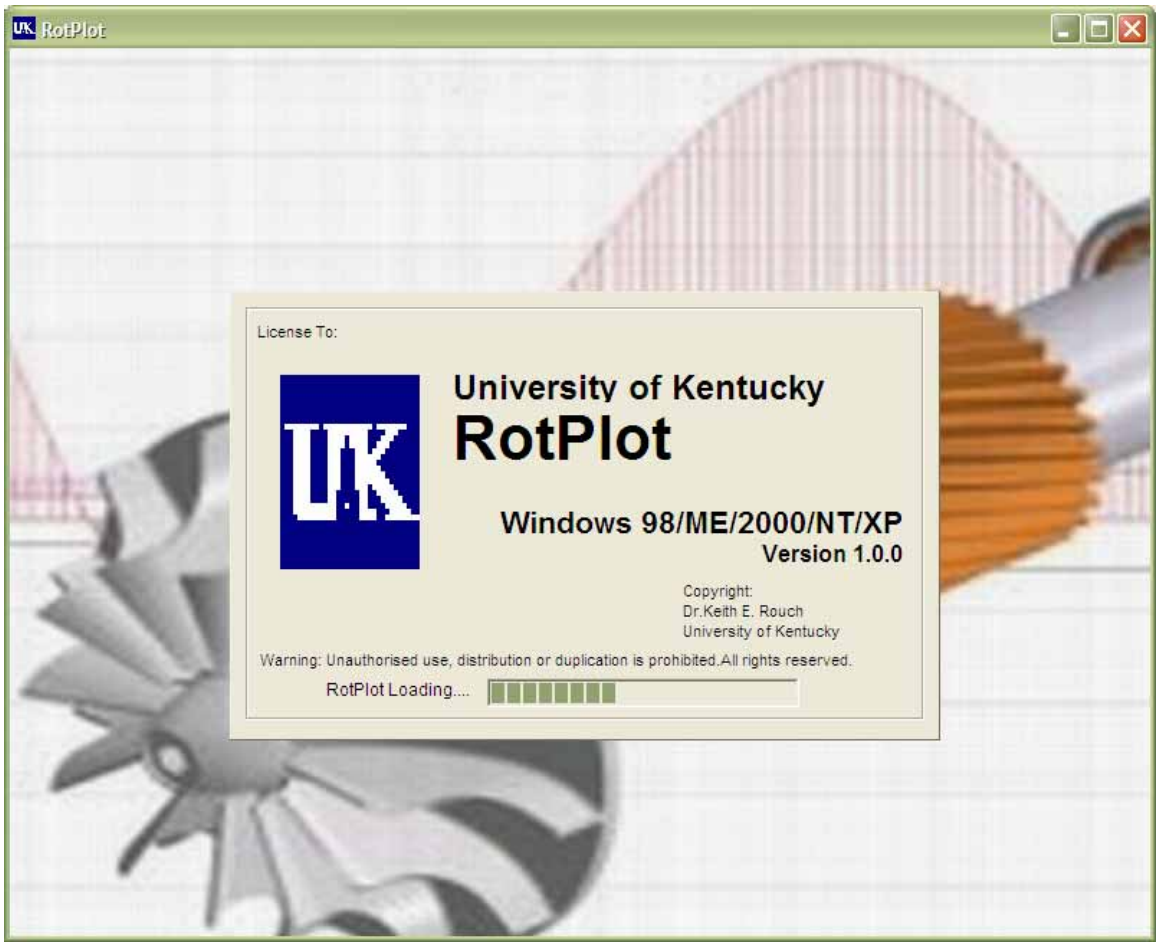


Fig 4.2 Program loading

Fig.4.3 shows the program background and is constant throughout the program execution. Fig.4.4 shows the help tips related to the program, at start up. The option can be turned off at any time.

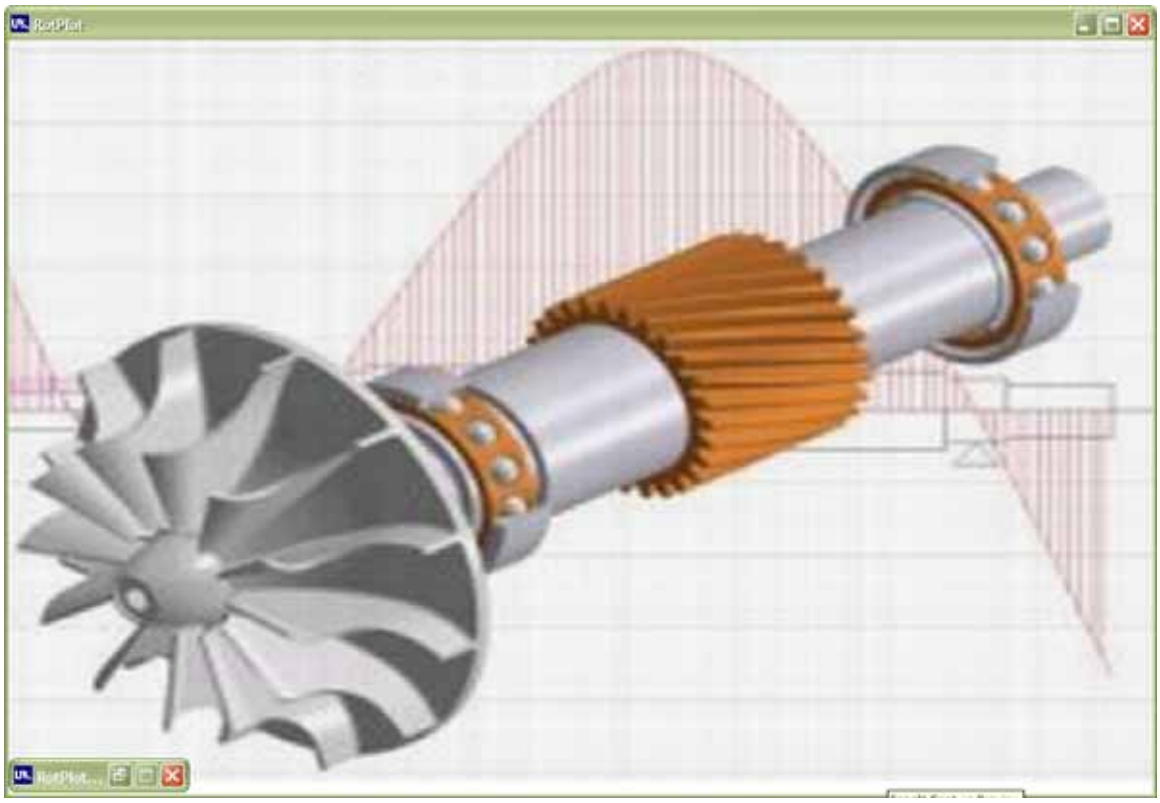


Fig 4.3 Program background

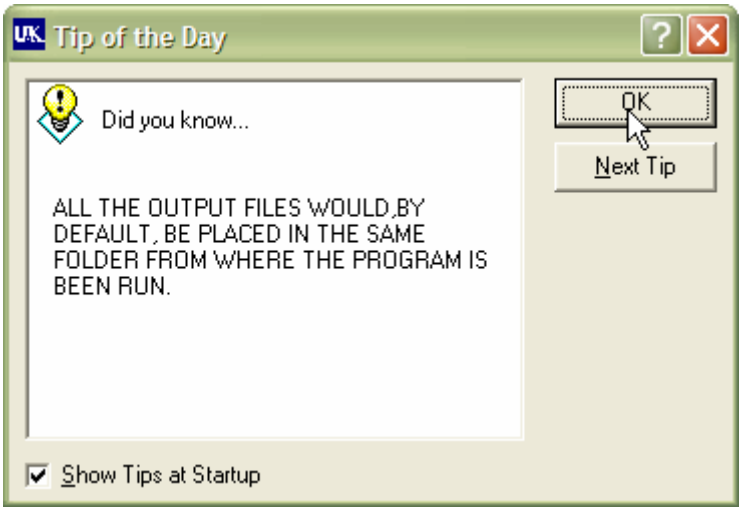


Fig 4.4 Help tips at startup

Fig. 4.5 and Fig. 4.6 show the main menu, from where the user can easily navigate between the post processor and processor. The help and overview toggle button can be used to know about the processor program and post-processor program along with its features. The startup tips can be made to display, by turning on the appropriate option.

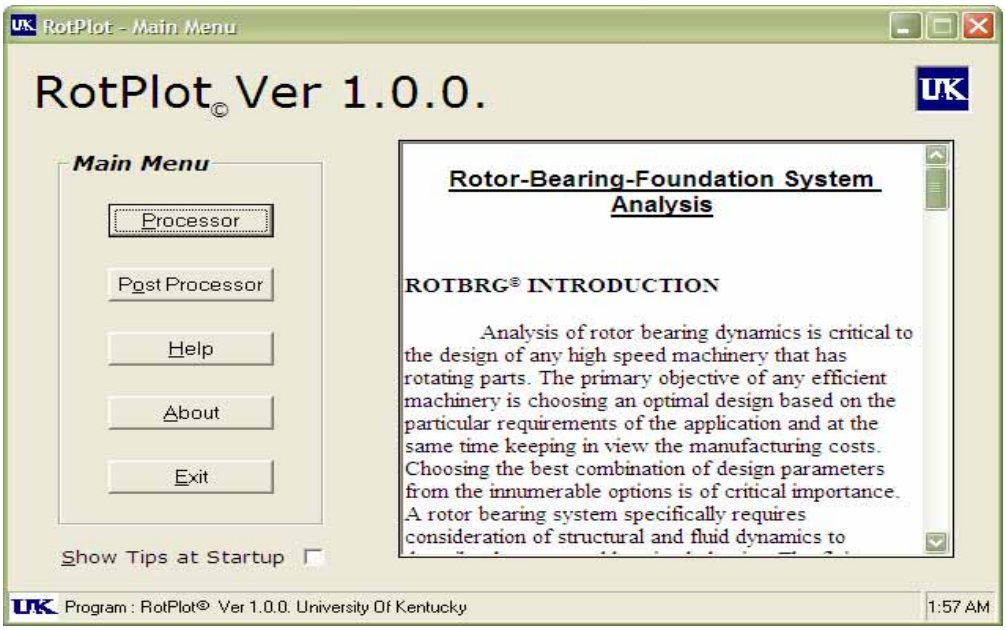


Fig 4.5 Main menu showing the processor and post-processor overview

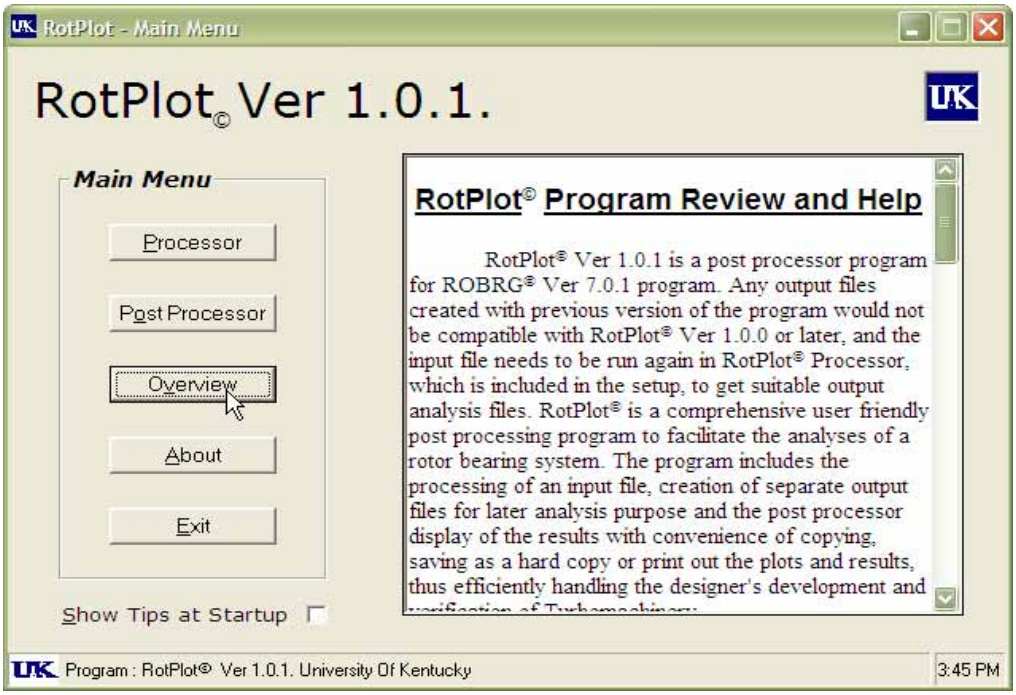


Fig 4.6 Main menu showing program help

Fig 4.7 shows the program information and the related copyrights. Fig 4.8 shows the processor menu. All the compatible input files in the current directory can be viewed by clicking on the 'Browse' button. Upon selecting an input file and clicking the 'Run Processor' button, the necessary analysis files are generated for post processing.

Fig 4.7 Program version and copyright information

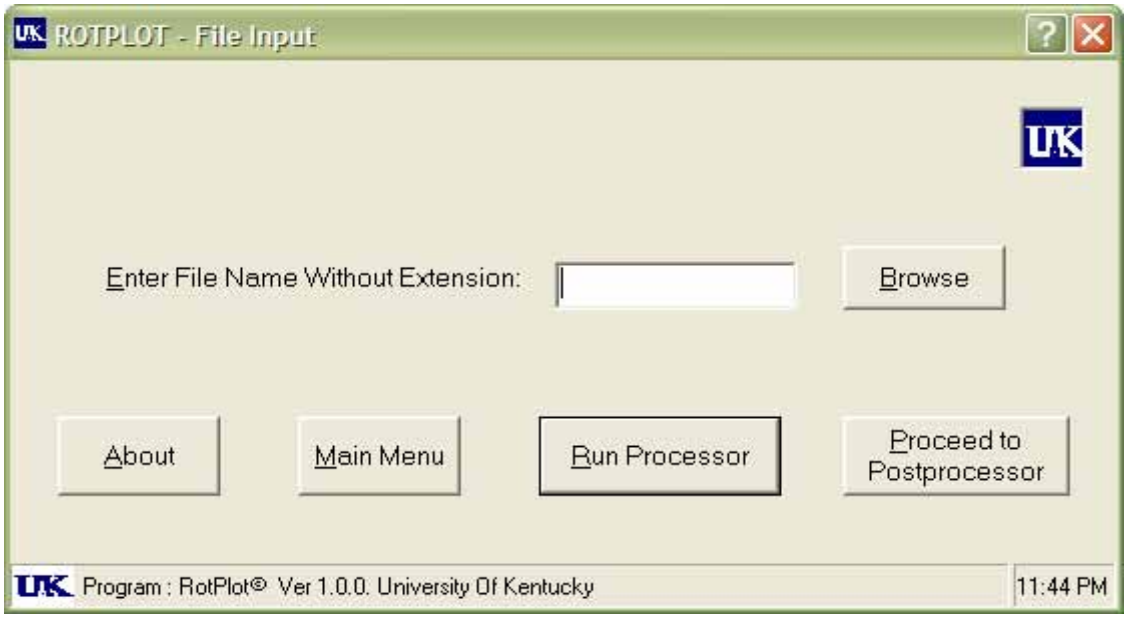


Fig 4.8 Processor menu

Fig 4.9 shows a typical browse form for the processor. Only the input files in the current directory are displayed and any one of them can be selected for processing. Fig 4.10 shows a typical screen where the user is prompted with the status of the program.

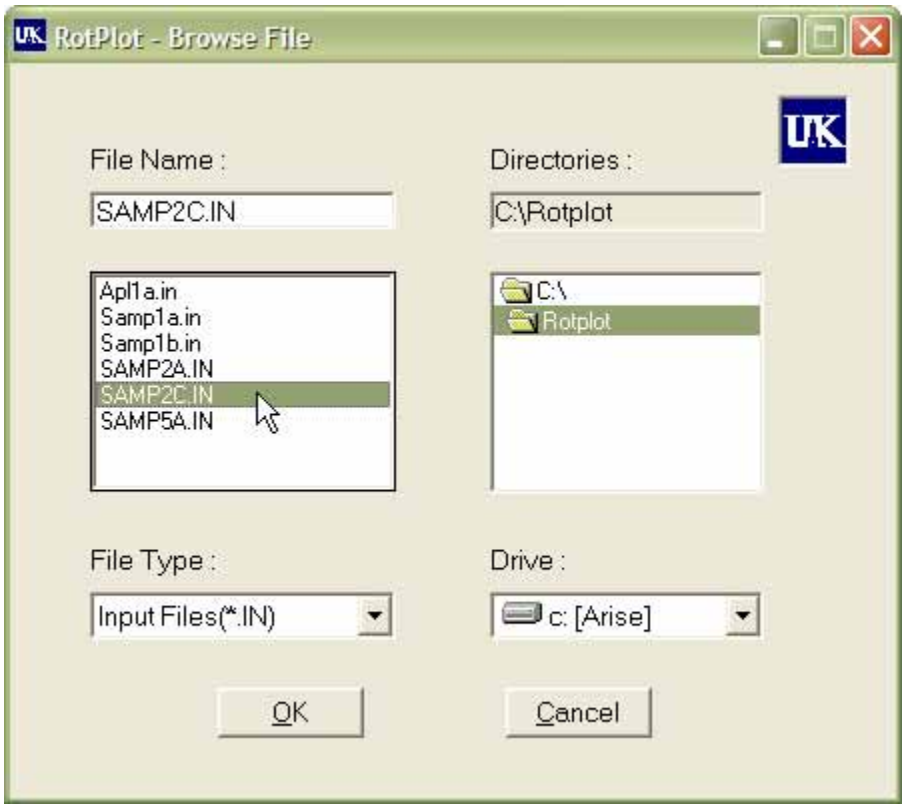
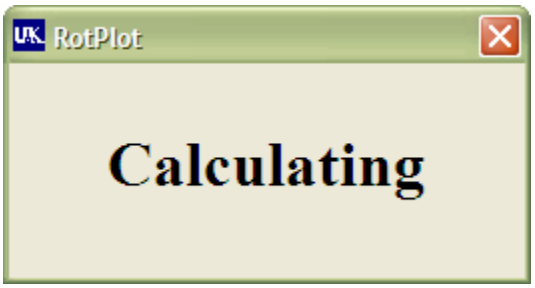


Fig 4.9 Processor file input menu



4.10 Program working message

Fig 4.11 shows the main post processor menu. Each of the buttons indicates the analysis options that can be carried out. The user has the option to switch to the main menu or exit the program from this menu. Each analysis further has options before the final plot is viewed. The user can run the processor once and come back to perform the post processing at a later time, without losing the necessary data.

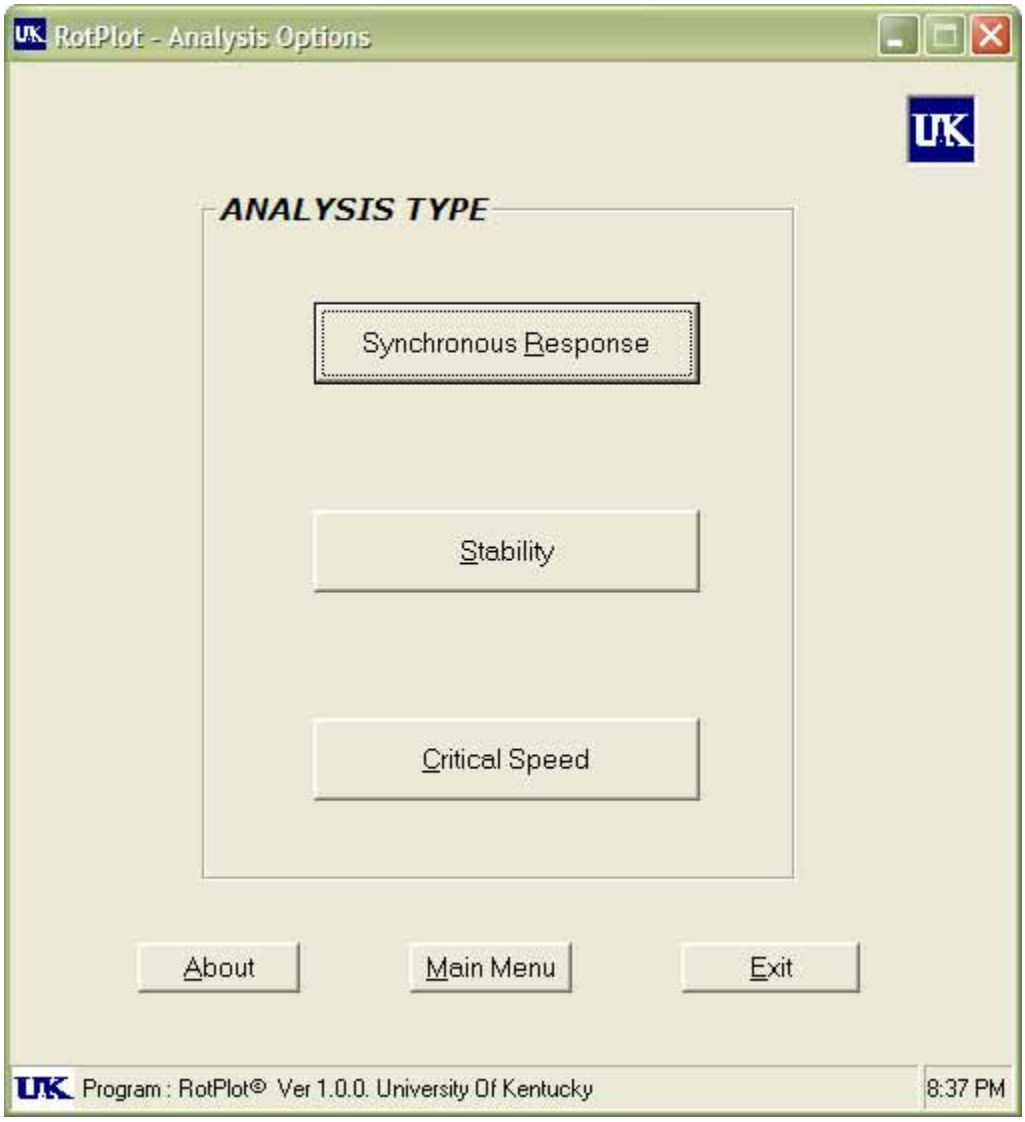


Fig 4.11 Rotor dynamic analyses options menu

Fig 4.12 shows the synchronous response analysis options. The user can proceed further only after selecting a particular option. Fig 4.13 shows the synchronous response plot options. A compatible analysis file can to be selected by clicking the browse button. The user can easily navigate and change options throughout the program using the 'Back' and 'Proceed' buttons. The selected options can be cancelled anytime by clicking 'Cancel' button.

Fig 4.12 Synchronous response specific option menu

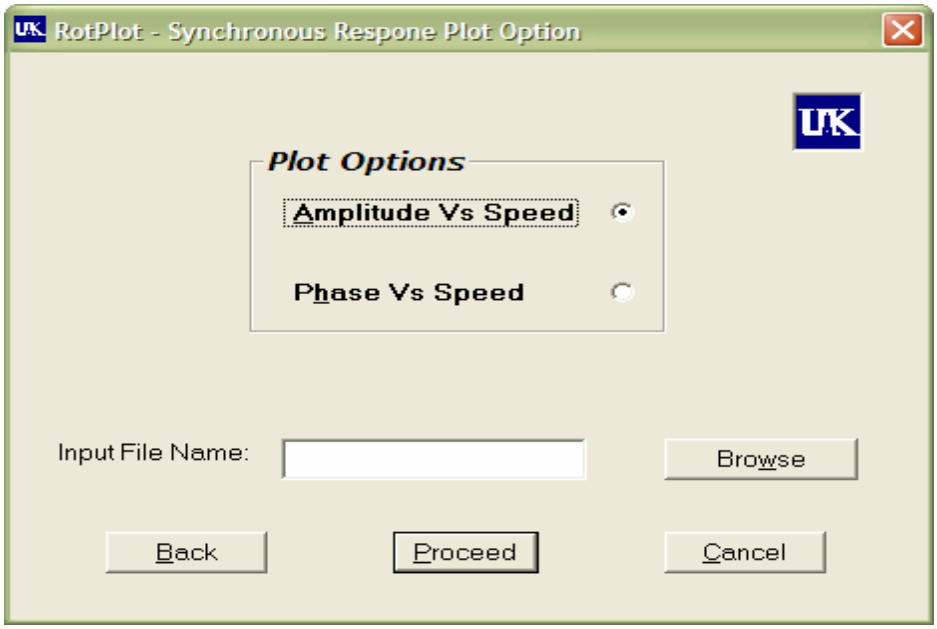


Fig 4.13 Synchronous response analysis sub menu

Fig 4.14 shows a typical form for browsing and selecting a synchronous response analysis file. Only the synchronous response analysis files, with an extension SYR, would be viewed for selection.

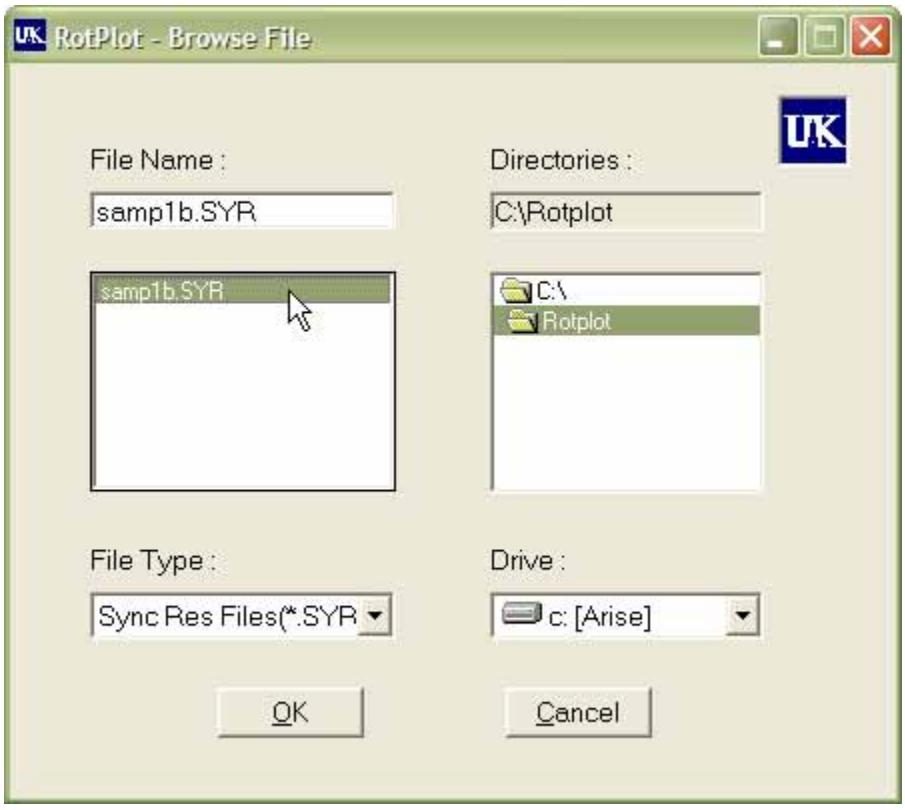


Fig 4.14 Synchronous response specific file input menu

Fig 4.15 shows the plot settings menu. A user can select the increment level on the rotor speed axis by selecting the pre-specified value or entering any other desired value. The amplitude increments on the plot can also be selected or entered similarly. The 'Station to plot' pull down menu has a list of all the stations on file, where the synchronous response is carried out. The user may select any option of choice. The grid option can be checked to have the grid superimposed on the plot. The operating speed(s) can also be marked on the plot, for convenience, by the appropriate check box. The user has the option to switch back to the main menu directly or exit out of the program at this point. The default settings can be changed by an advanced user, to suit the plot requirements.

Fig 4.15 Plot settings menu for response plot

Fig 4.16 shows the default options that could be changed to suit the plot requirements. The ‘Speed Range’ specifies the maximum speed value that the user intends to view on the plot. The default value is the maximum speed specified in the input file. Similarly, a maximum amplitude setting can be entered. The user is given the option to view multiple stations response on a single plot, rather than having them plotted separately. A maximum of three stations response plots can be plotted at once. The user may select to have only the ‘X Amplitudes Vs Speed’ or ‘Y Amplitudes Vs Speed’ plot by checking the appropriate setting. Similarly, the amplitude plots at foundation level can be plotted.

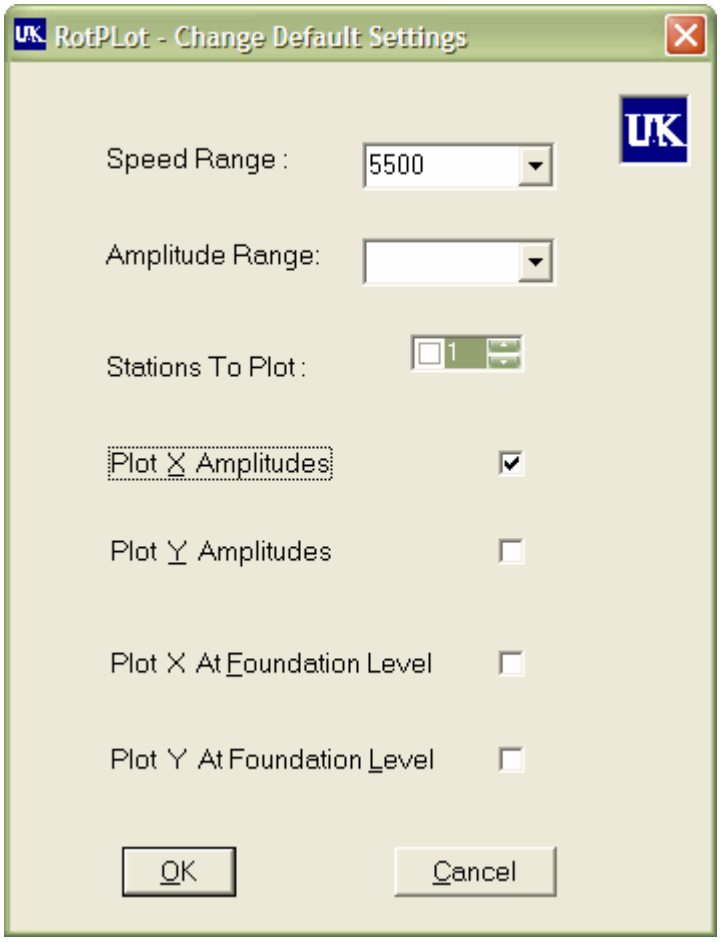


Fig 4.16 Synchronous response default setting changing menu

Fig 4.17 shows the plot setting menu for the Phase Vs Speed plot. The user may select one of the pre defined phase increment values or enter any positive value of choice. The phase axis (typically Y-axis) would be divided in multiples of the value selected by the user. The grid would also be modified accordingly.

Fig 4.17 Plot setting menu for phase plot

Fig 4.18 shows a typical 'Browse File' menu for the bearing response analysis. Only the compatible files with extension 'SYB' would be viewed for selection. Fig 4.19 shows the plot settings menu for bearing response. The 'Bearing to Plot' pull down menu has all the bearings of the input file rotor model, for selection. The user may select any one of them.

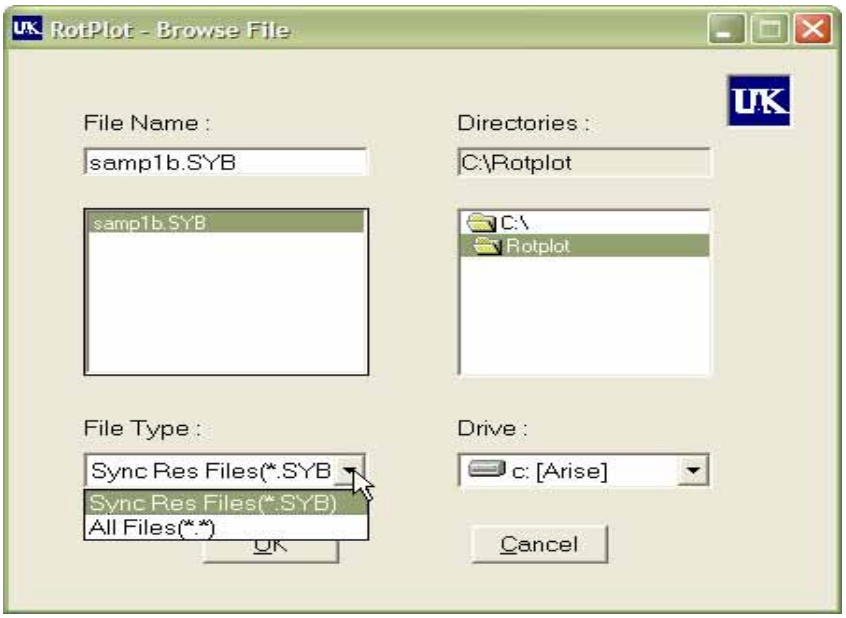


Fig 4.18 File input menu for bearing response plot

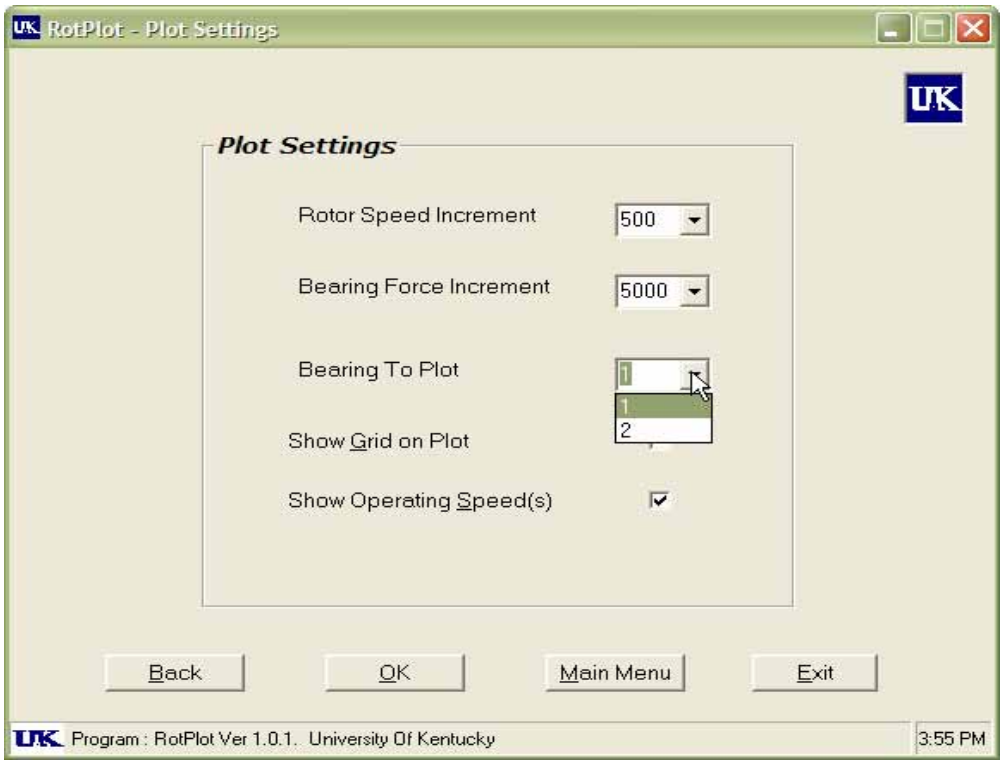


Fig 4.19 Plot setting menu for bearing response

Fig 4.20 shows the stability analysis options. Once the user selects an input file, the mode plotting options are activated for selection. The ‘Mode to be Plotted’ pull down menu has as many modes to plot as indicated in the input file. The ‘Rotor Length Increment’ gives the user the option to select the increment on the rotor length axis, typically X-axis. The rotor model can be superimposed on the mode plot by selecting the appropriate option. The bearing locations can also be plotted for convenience by selecting the option. A grid mesh can be superimposed on the plot by selecting the option. The default settings can be changed by an advanced user to suit the requirements. Fig 4.21 shows the error prompt when the user tries to enter mode plotting options without entering any appropriate input file.

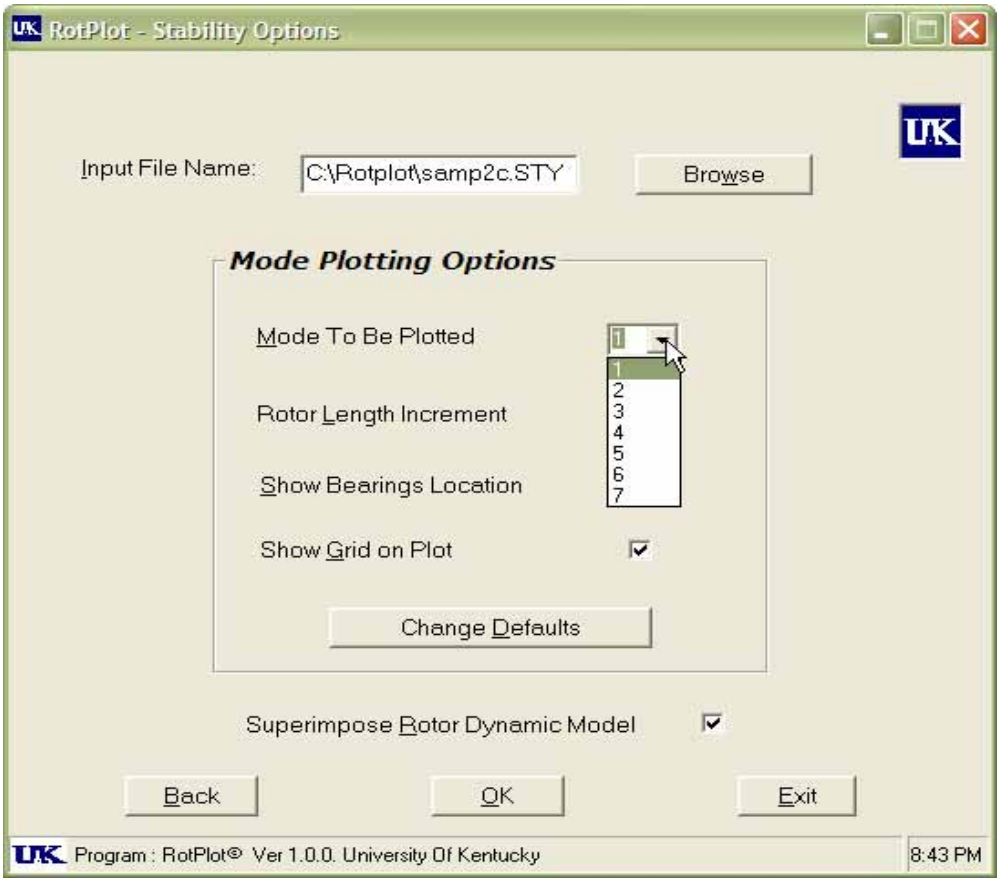


Fig 4.20 Stability analysis menu

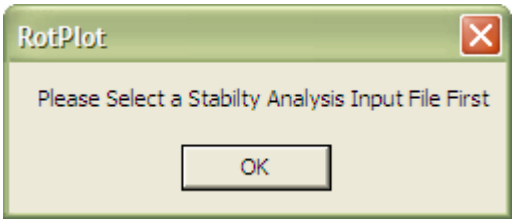


Fig 4.21 Stability analysis file input error handler

Fig 4.22 shows a typical ‘Browse file’ menu for the stability analysis, which would display only the appropriate files with ‘STY’ extension. Fig 4.23 shows the default changing settings by which the user has the option to plot only the X mode shapes or Y mode shapes. Also, multiple modes can be plotted simultaneously by selecting all those modes

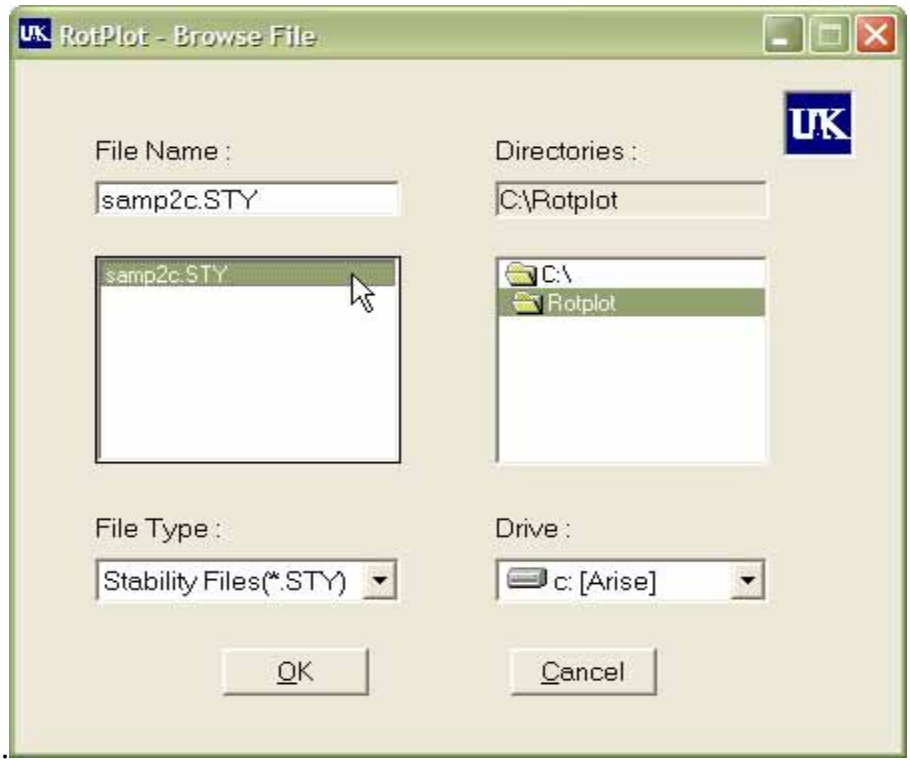


Fig 4.22 Stability analysis file input menu

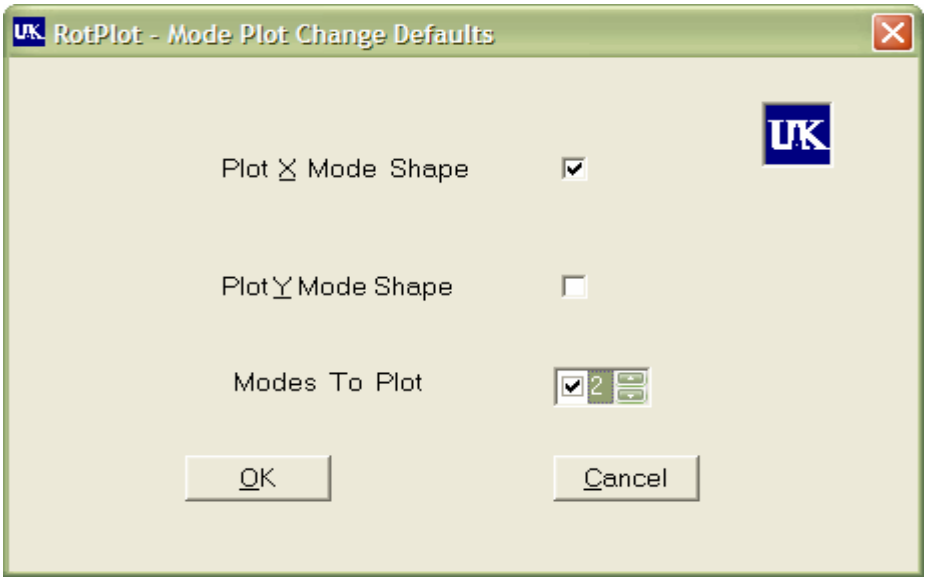


Fig 4.23 Stability analysis defaults changing menu

Fig 4.24 shows the Critical speed analysis option menu. A suitable analysis file can be selected only after selecting an analysis option. Fig 4.25 shows the error prompt when the user tried to proceed to the plot without inputting an analysis file. Fig 4.26 shows the error prompt when the user tried to select a file for input before selecting an analysis option.

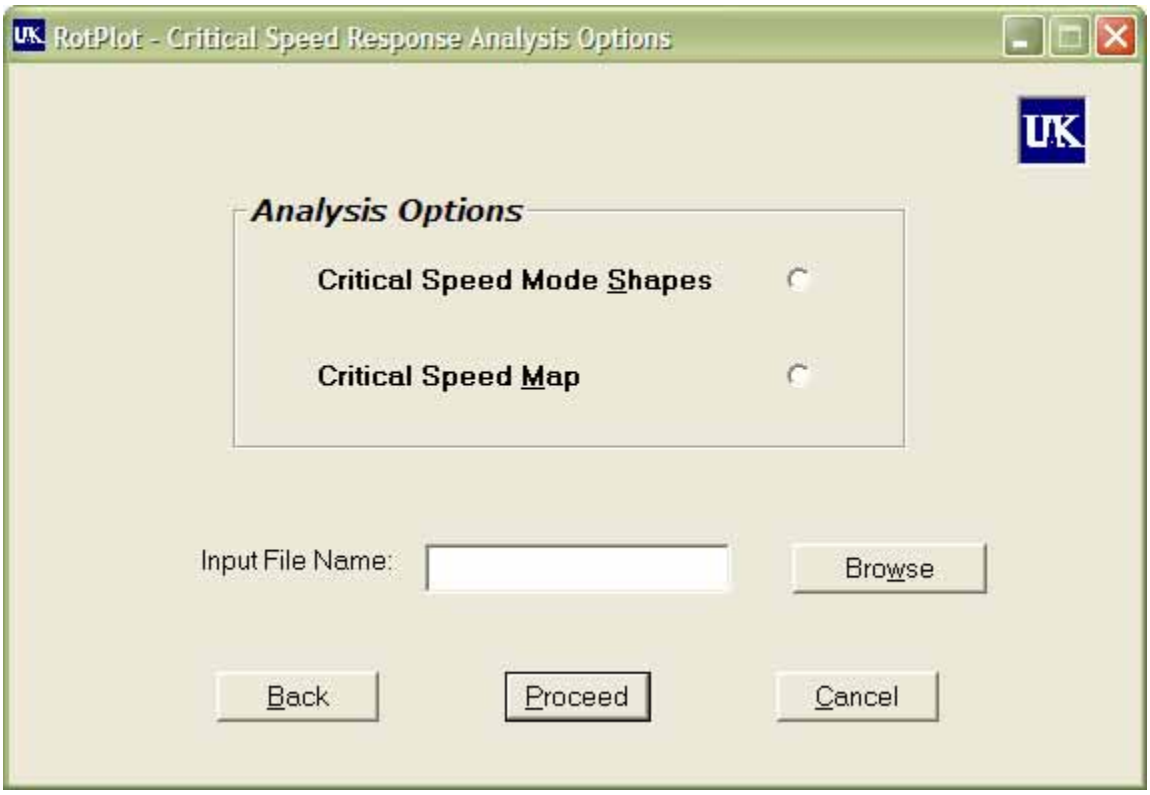


Fig 4.24 Critical speed analysis options menu

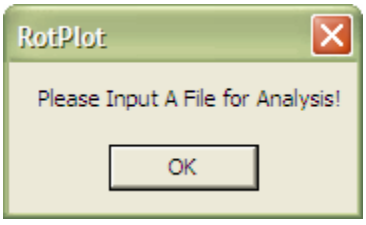


Fig 4.25 Critical speed analysis file input error handler

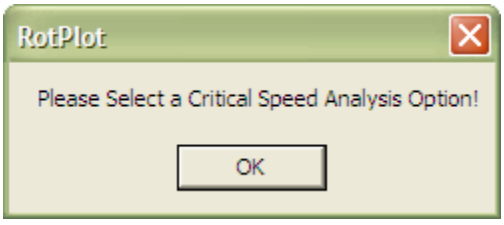


Fig 4.26 Critical speed analysis selection error handler

Fig 4.27 and Fig 4.28 show the 'Browse File' menu depending on the analysis option selected by the user. The type of files displayed for selection also depends on the analysis option selected. Files with extensions 'CRM' and 'CTR' stand for the critical speed mode shape and critical speed map options respectively.

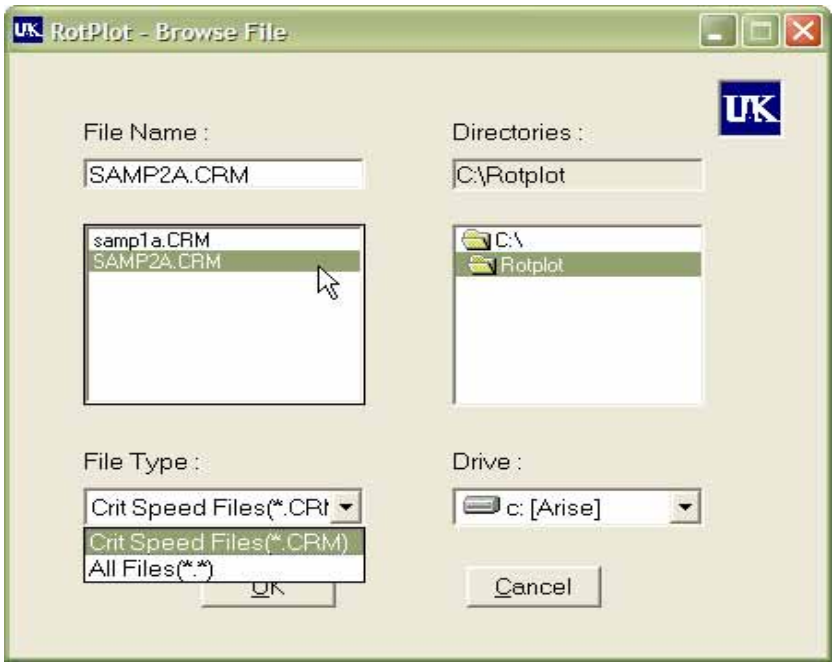


Fig 4.27 Critical speed analysis file input menu

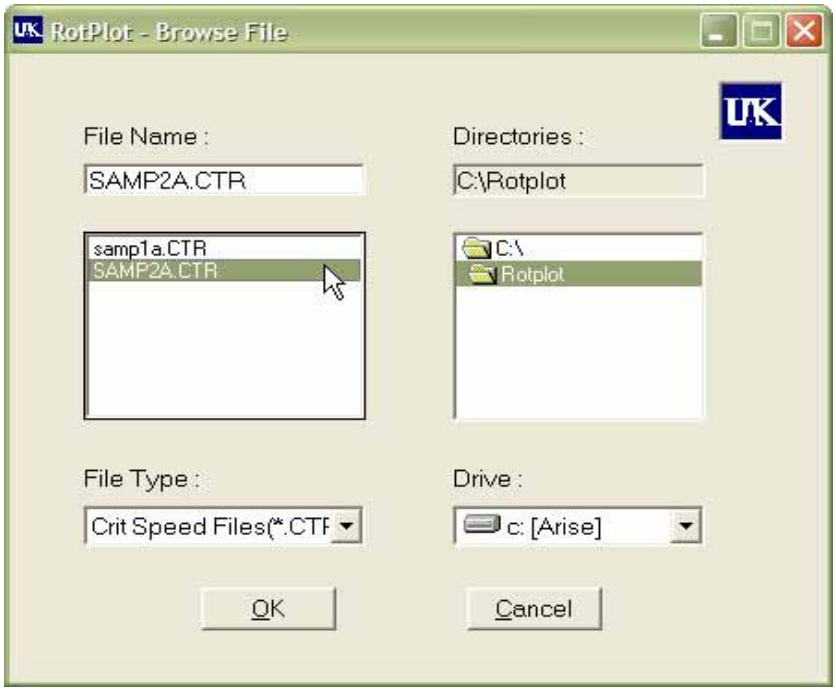


Fig 4.28 Critical speed map input file menu

Fig 4.28 shows the Critical speed mode plotting options menu. The 'Bearing Stiffness of plot' pull down menu has all the cases of bearing stiffness listed in the input file. User may select any case to plot. The rotor length increment option value lets the user set the increment for the rotor length axis, typically X-axis. The rotor model can be superimposed on the mode plot by selecting the appropriate option. The bearing locations can also be plotted for convenience by selecting the option. By checking those options, the corresponding bearing stiffness values are showed on the plot along with the critical speeds, at that particular selected mode. A grid mesh can be superimposed on the plot by selecting the option. Fig 4.29 shows the confirmation prompt to the user whenever the user tries to exit the program.

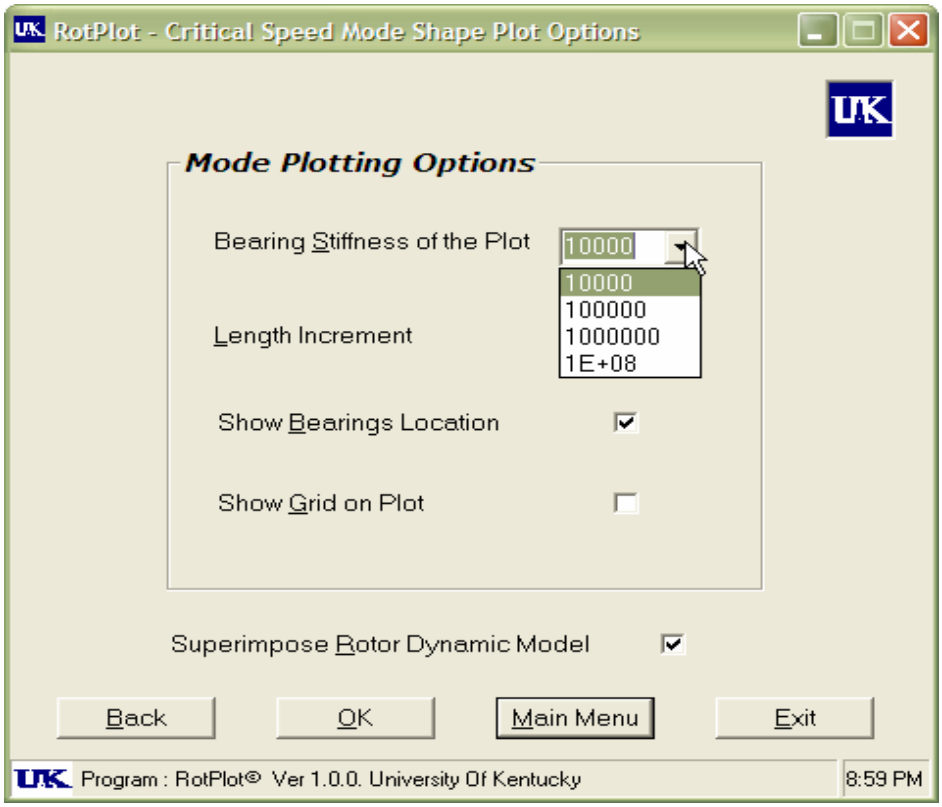


Fig 4.29 Critical speed analysis mode plotting options menu

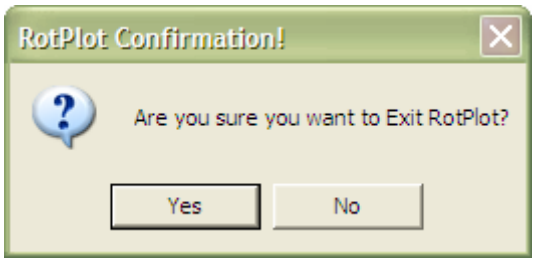


Fig 4.30 Program exit confirmation

Fig 4.31 shows a typical output plot. It shows the title of the plot, as in the input file and the various plot parameters depending on the analysis selected. A right click with the mouse on the plot would show the specific plot parameters at that particular point. The ‘Save’ button would enable to save the plot in any specified directory, in bitmap (BMP) format. The ‘Copy’ button would copy the entire plot to clipboard, giving the option of inserting it later in any appropriate file, for analysis. The ‘Print’ option enables the user to print it to the default printer as a hard copy.

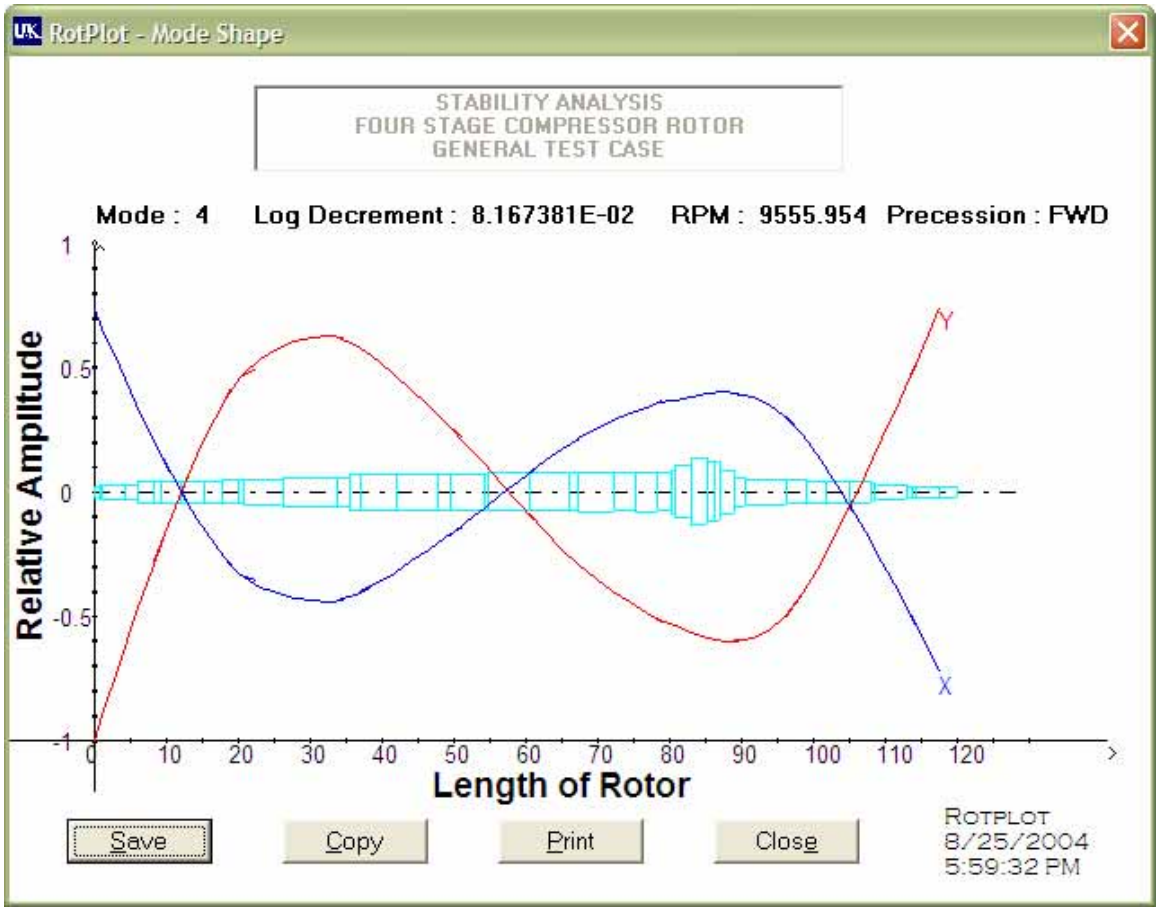


Fig 4.31 Sample plot with options for the user

4.3 PROGRAM INSTALLATION

Effort was made to facilitate the installation of the program with an easy setup package as part of this thesis. This installation requires a minimum of user's effort and is menu driven. The user is given the option to select the destination drive and directory where the program is chosen to be installed. The author has used InstallShied software to build the installation package along with all necessary program controls for compatibility.

Fig 4.32 through Fig 4.37 depicts the setup process in a computer with Windows XP – SP1 operating system environment. All the illustrations are self explanatory with the details being provided at each step.

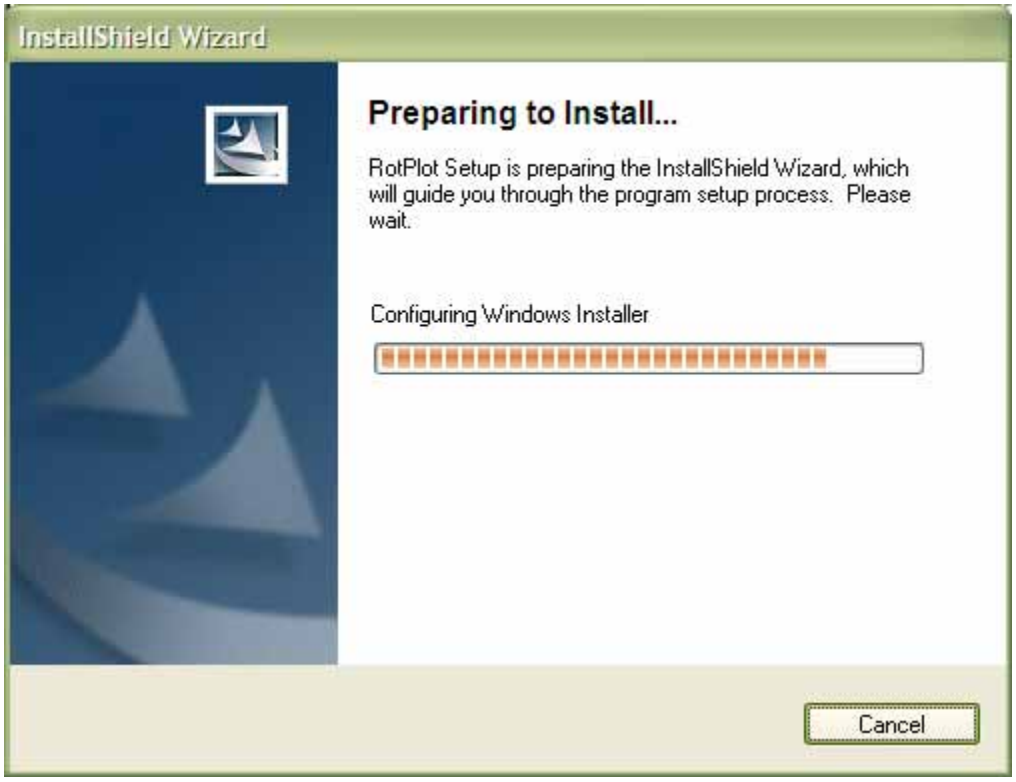


Fig 4.32 Program installation start up screen

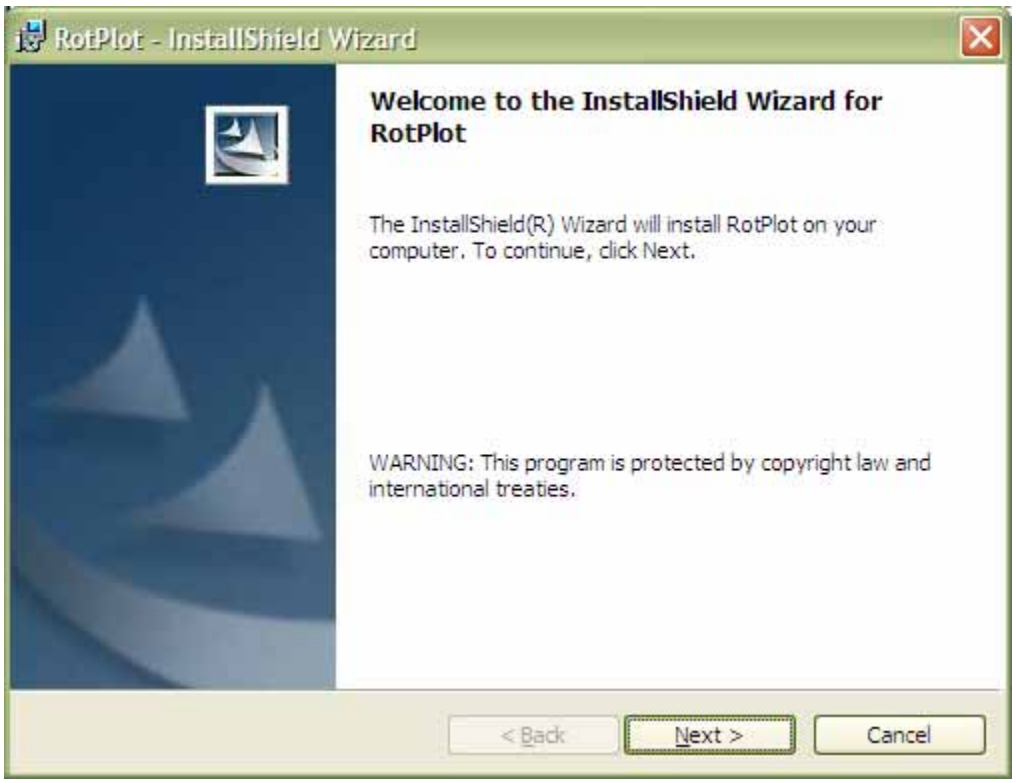


Fig 4.33 Program installation wizard

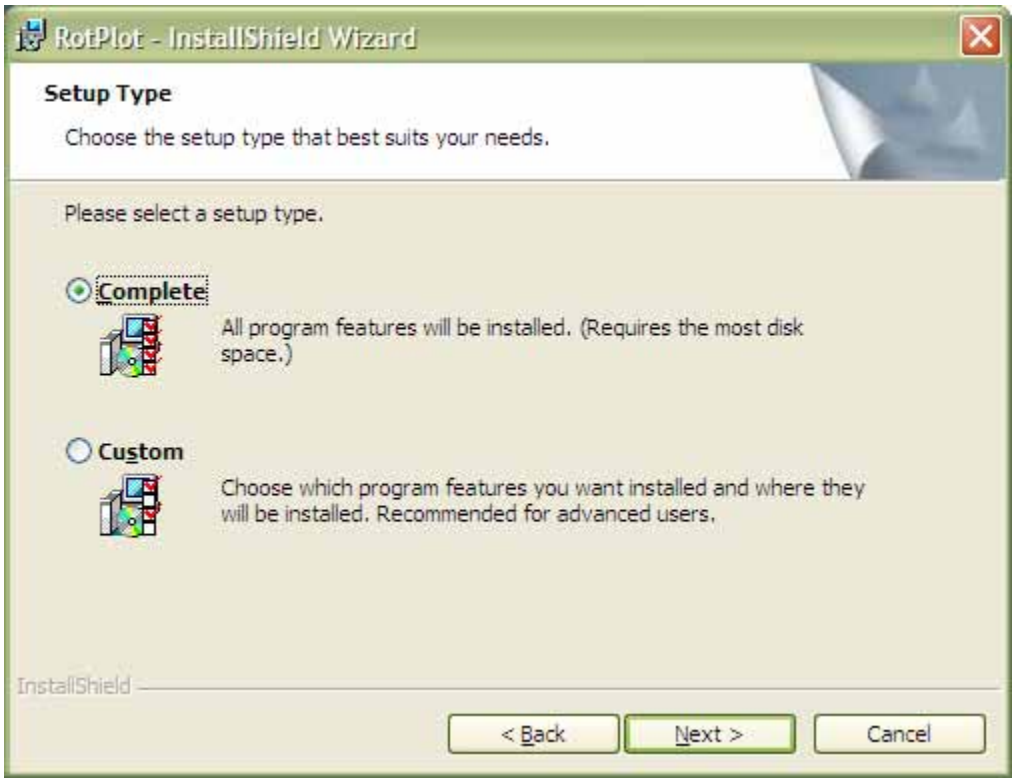


Fig 4.34 Program installation setup option

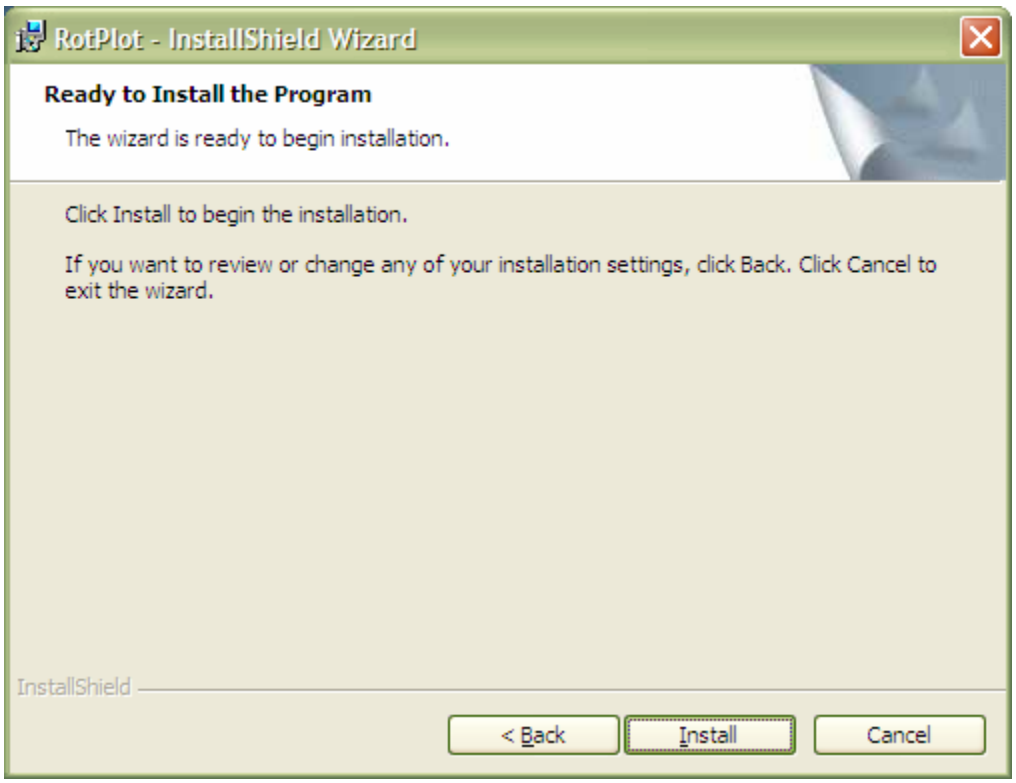


Fig 4.35 Program Installation settings conformation

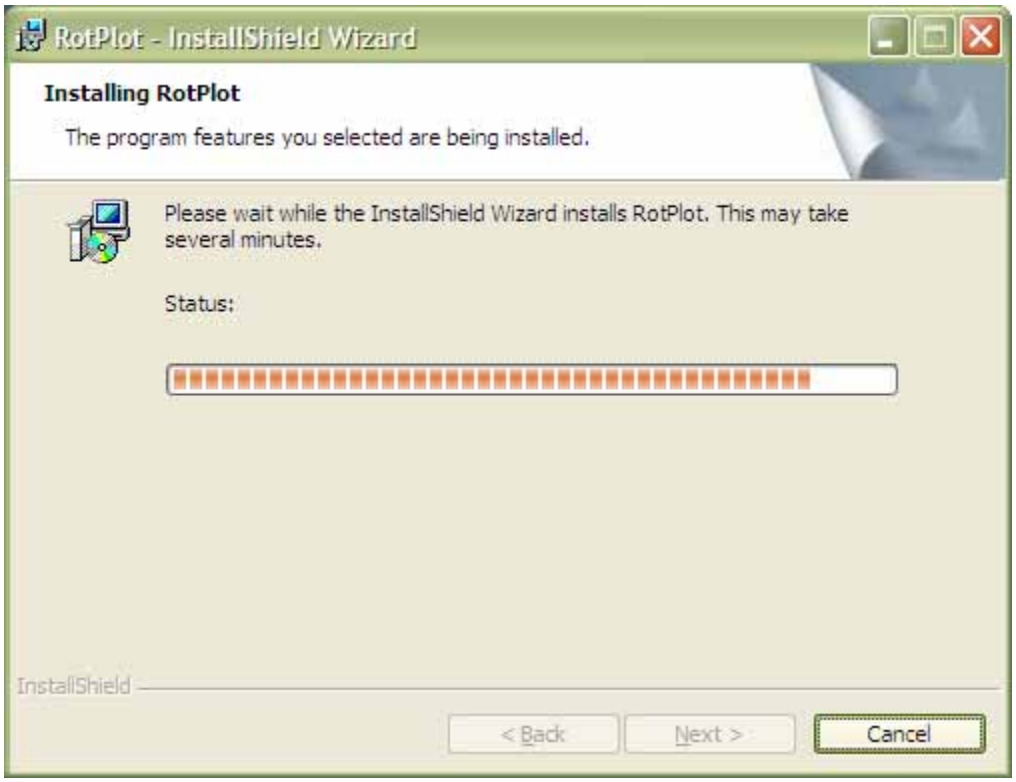


Fig 4.36 Program installation progress

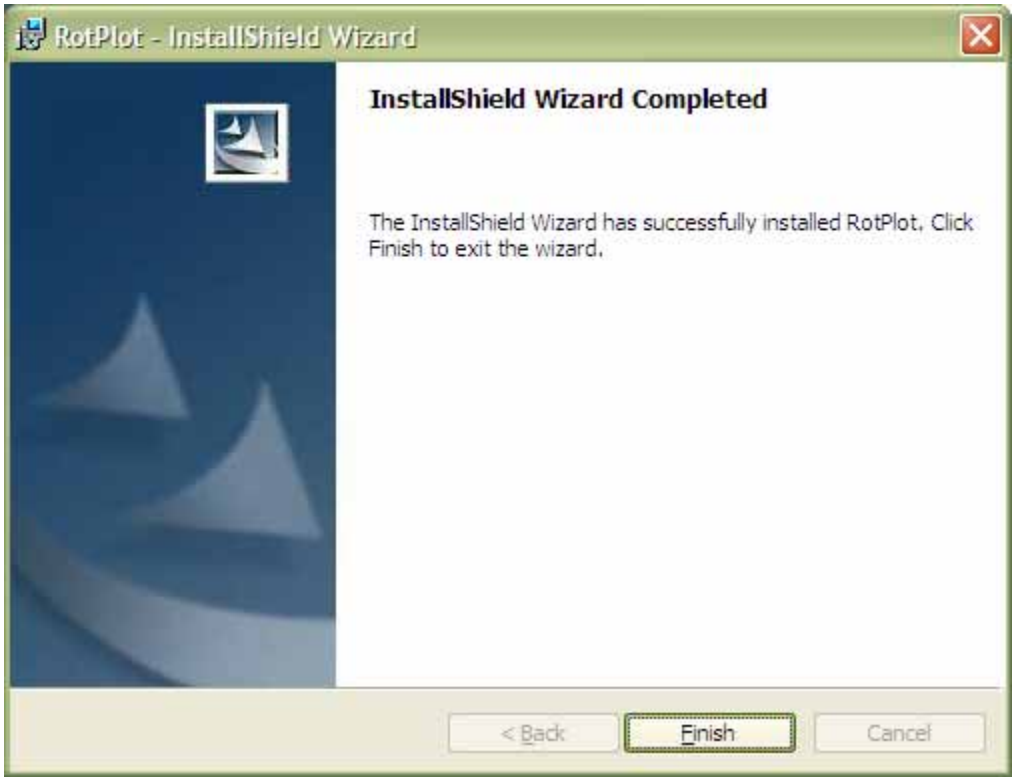


Fig 4.37 Program installation confirmation

After the successful installation of the program onto the hard drive of any computer, the program short cut is seen on the desktop as default and also in the start menu of the operating system. This facilitates the user to directly launch the program from the desktop. The program and all of its associated components can be easily removed from the computer from the control panel's add/remove programs menu. The uninstallation would, however, not remove any analysis files created by the program and need be removed manually by the user.

4.4 SAMPLE PLOTS

In the later sections, some sample plots produced by RotPlot post-processor program are illustrated. The plots are divided into sub sections to distinguish the three rotor dynamic analyses – undamped critical speed analysis, stability analysis and synchronous response analysis. Different colors are used to differentiate the different curves and mode shapes.

4.4.1 Undamped Critical Speed Analysis Sample Plots

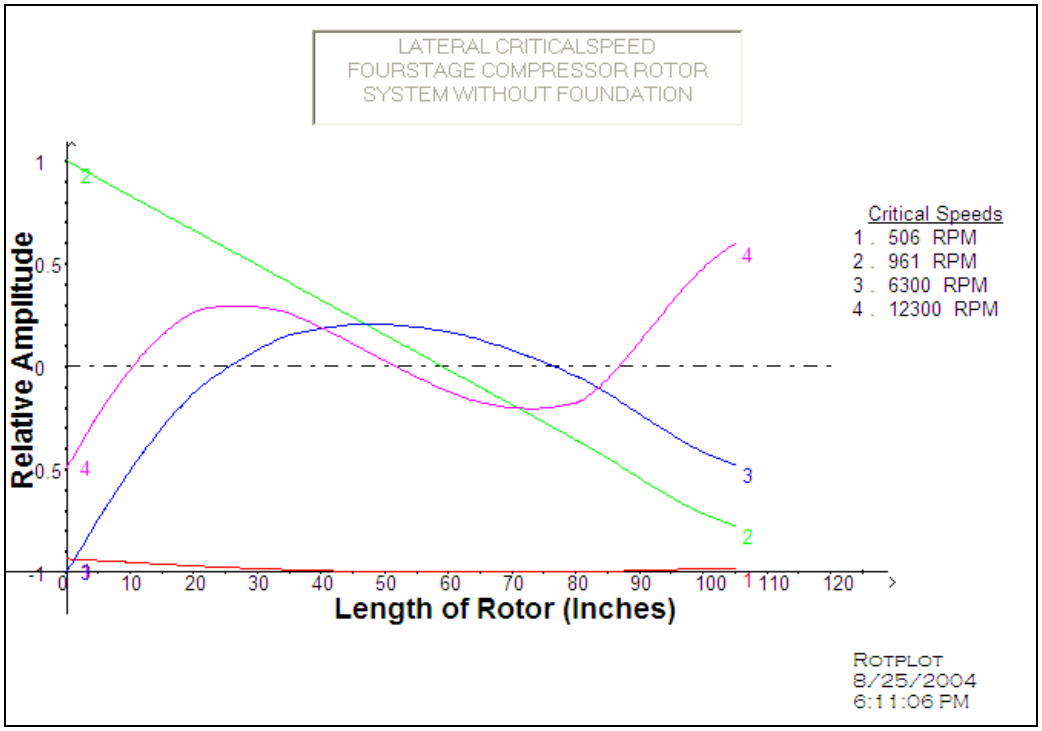


Fig 4.38 Critical speed mode shape for stiffness of 1E4

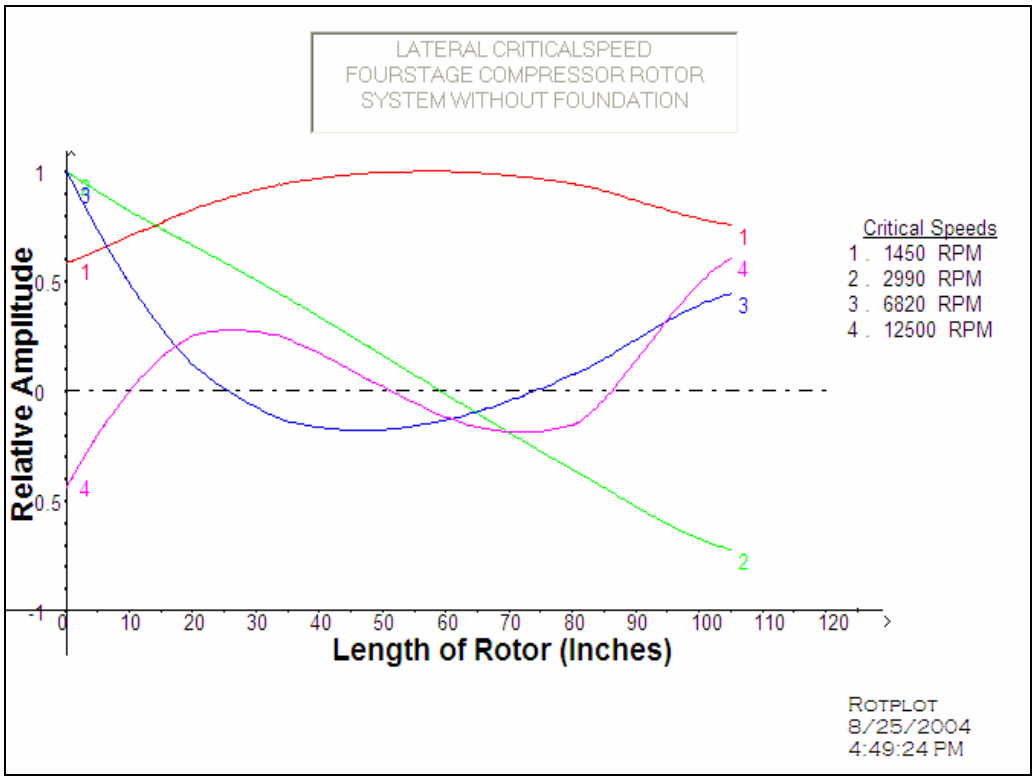


Fig 4.39 Critical speed mode shapes for stiffness of 1E5

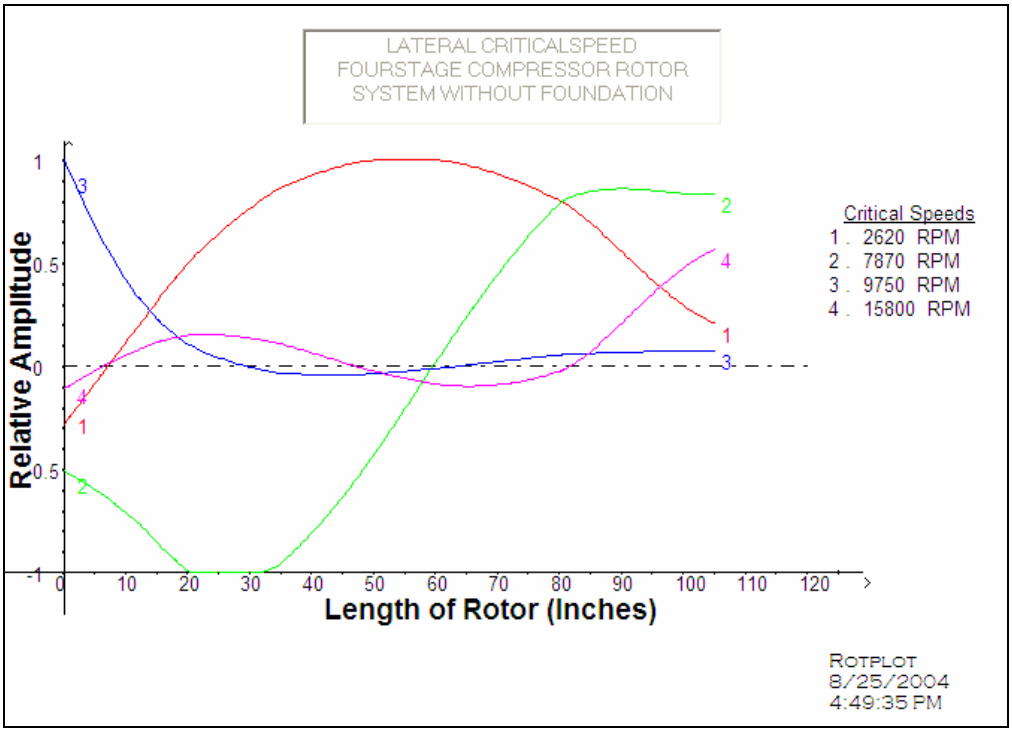


Fig 4.40 Critical speed mode shapes for stiffness of 1E6

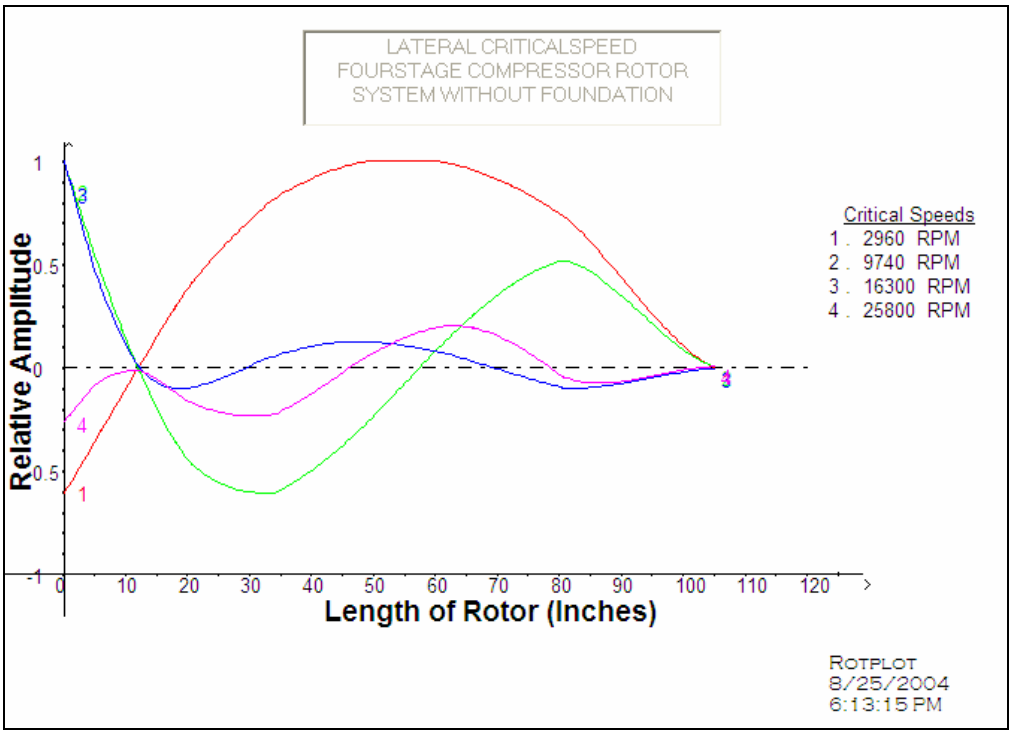


Fig 1.41 Critical speed mode shapes for a stiffness of 1E7

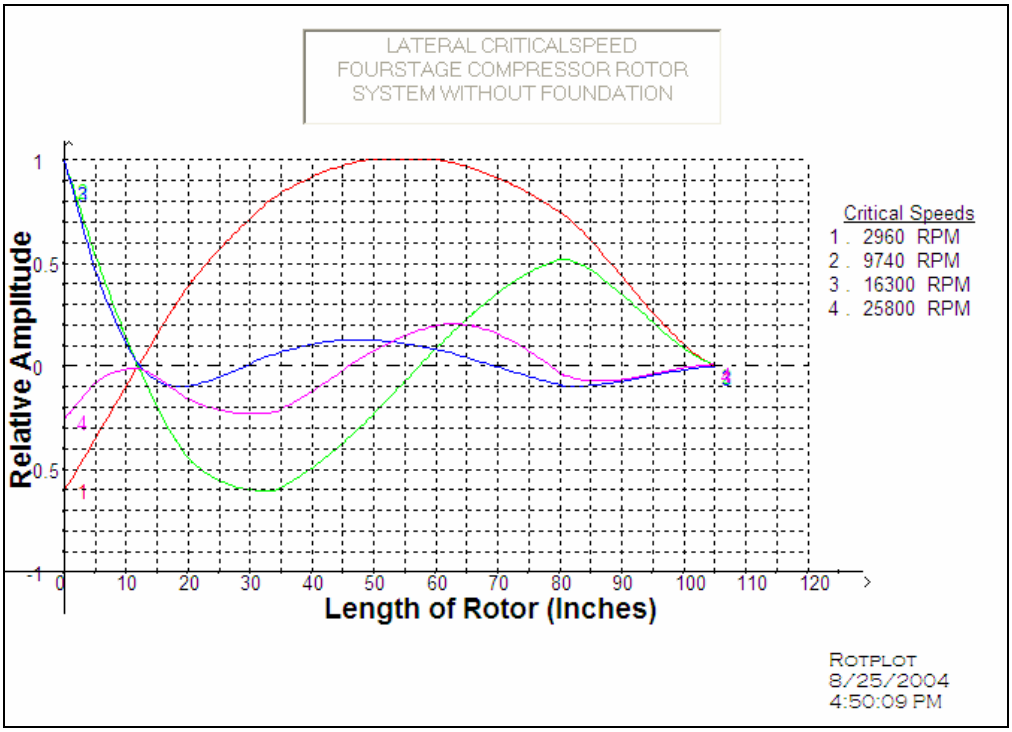


Fig 4.42 Critical speed mode shapes with grid option on

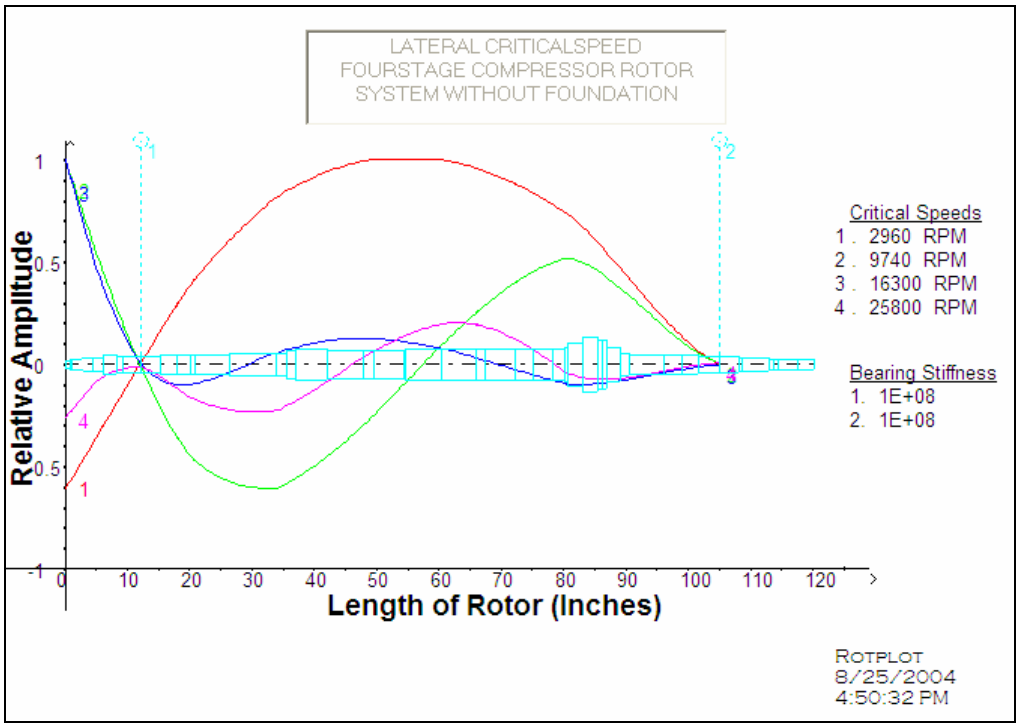


Fig 4.43 Critical speed mode shapes with rotor model and bearing locations superimposed

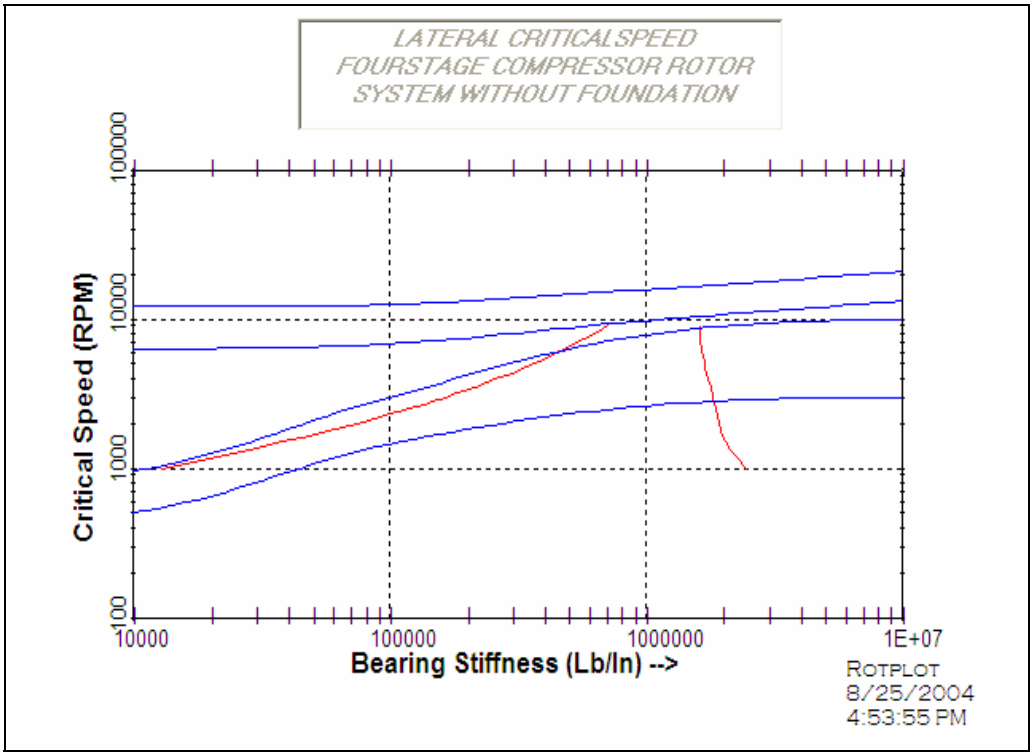


Fig 4.44 Critical speed map

4.4.2 Stability Analysis Sample Plots

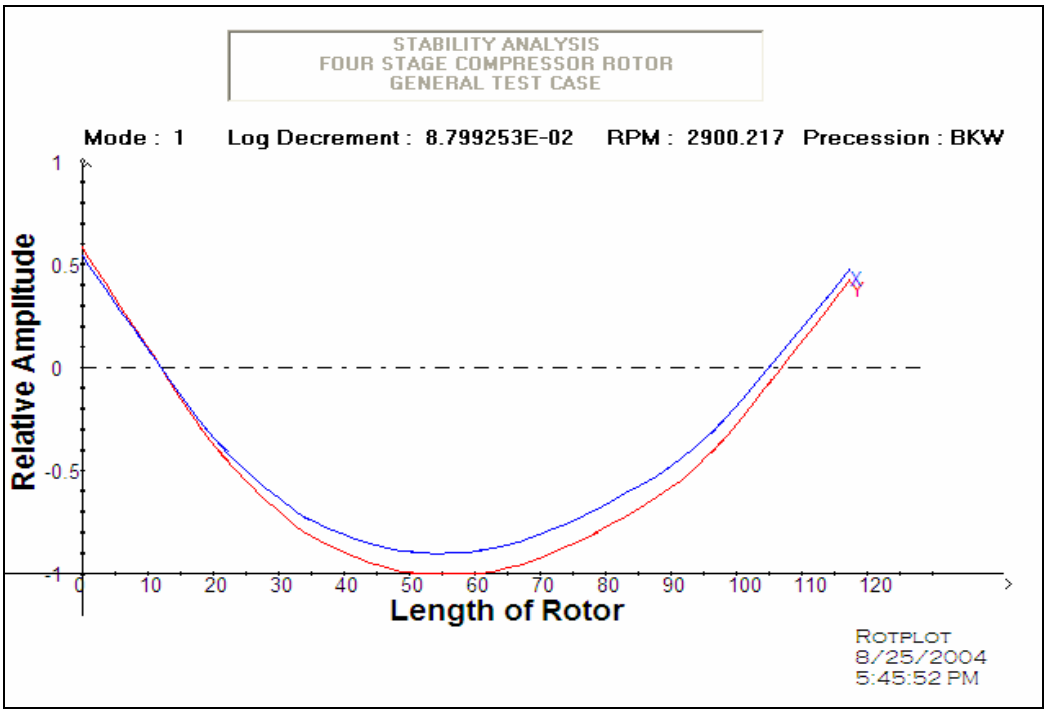


Fig 4.45 First damped mode shape

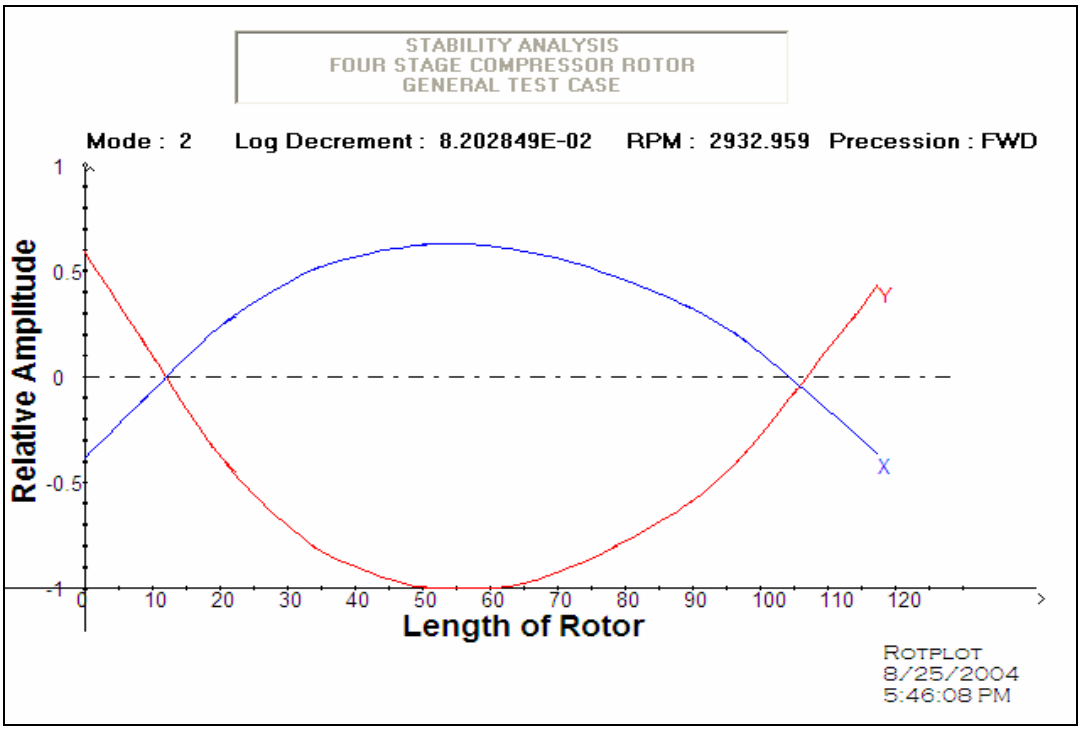


Fig 4.46 Second damped mode shape

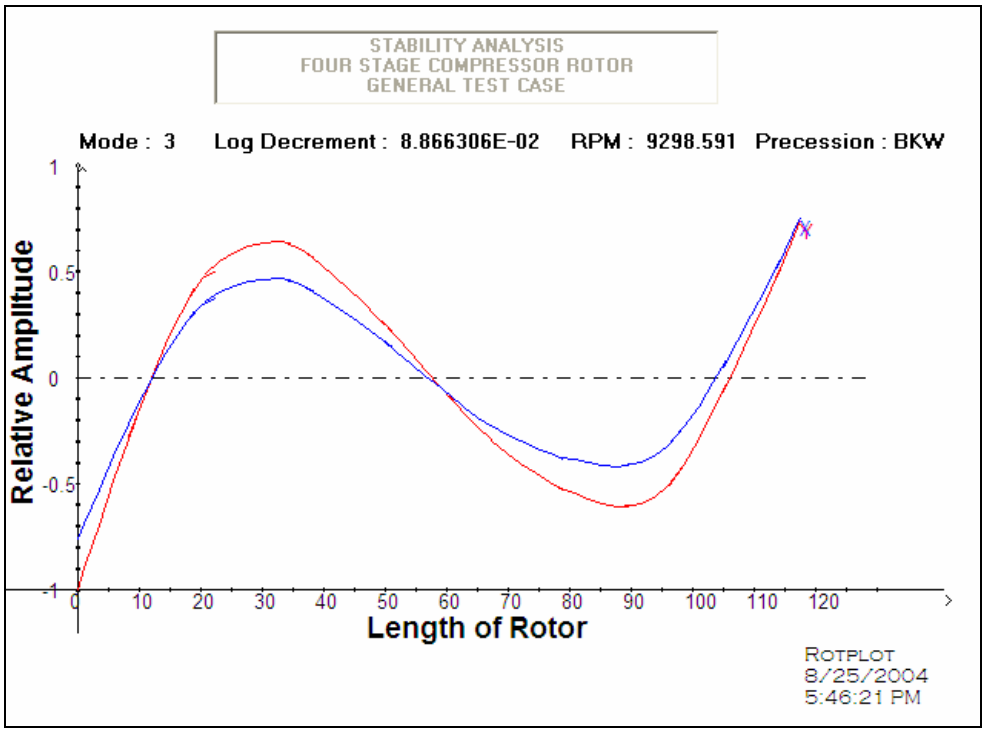


Fig 4.47 Third damped mode shape

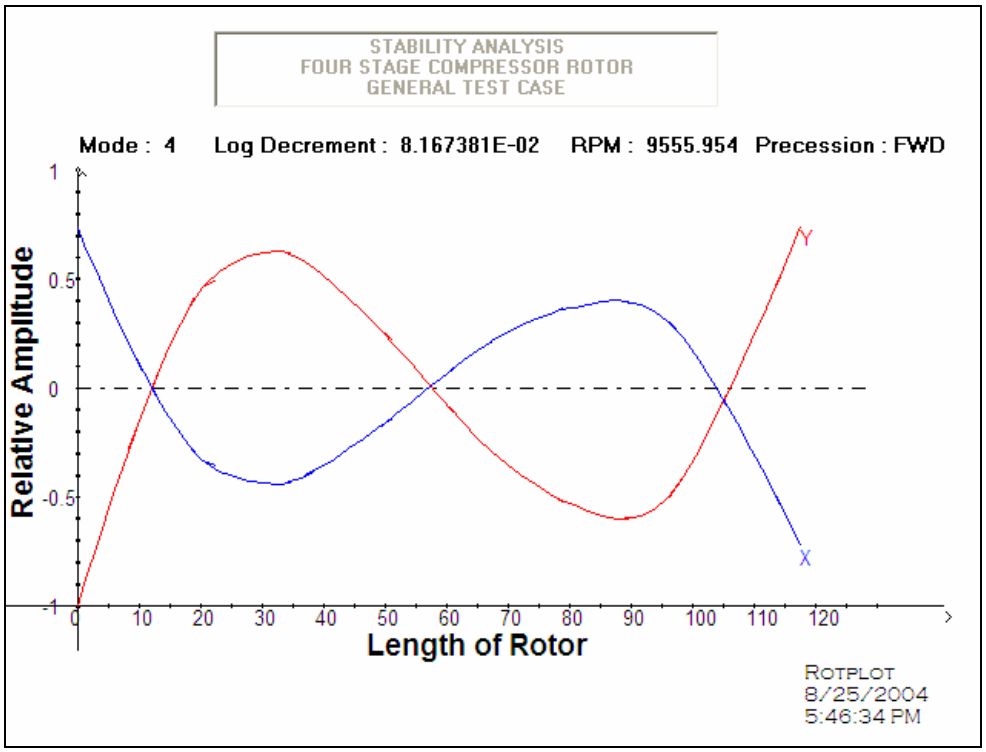


Fig 4.482 Fourth damped mode shape

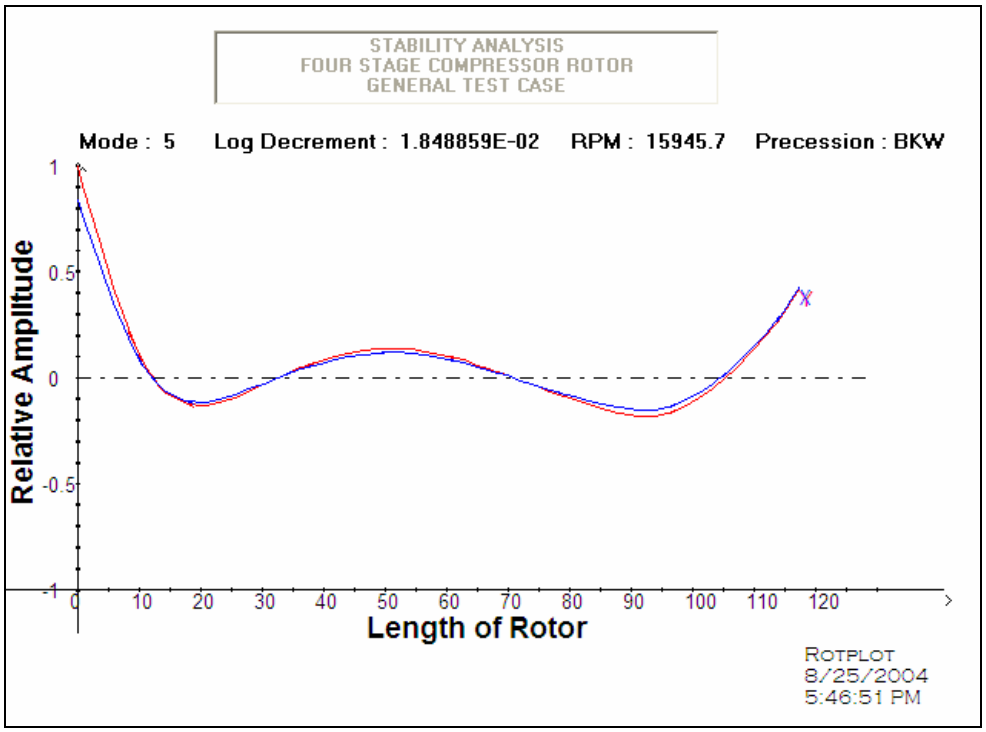


Fig 4.49 Fifth damped mode shape

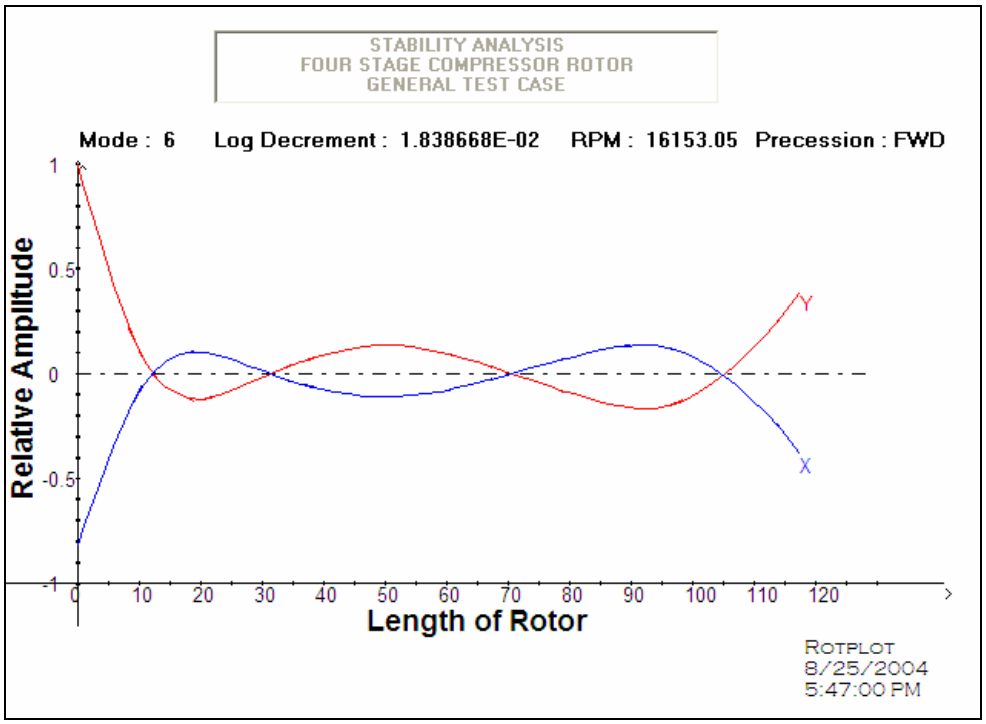


Fig 3 Sixth damped mode shape

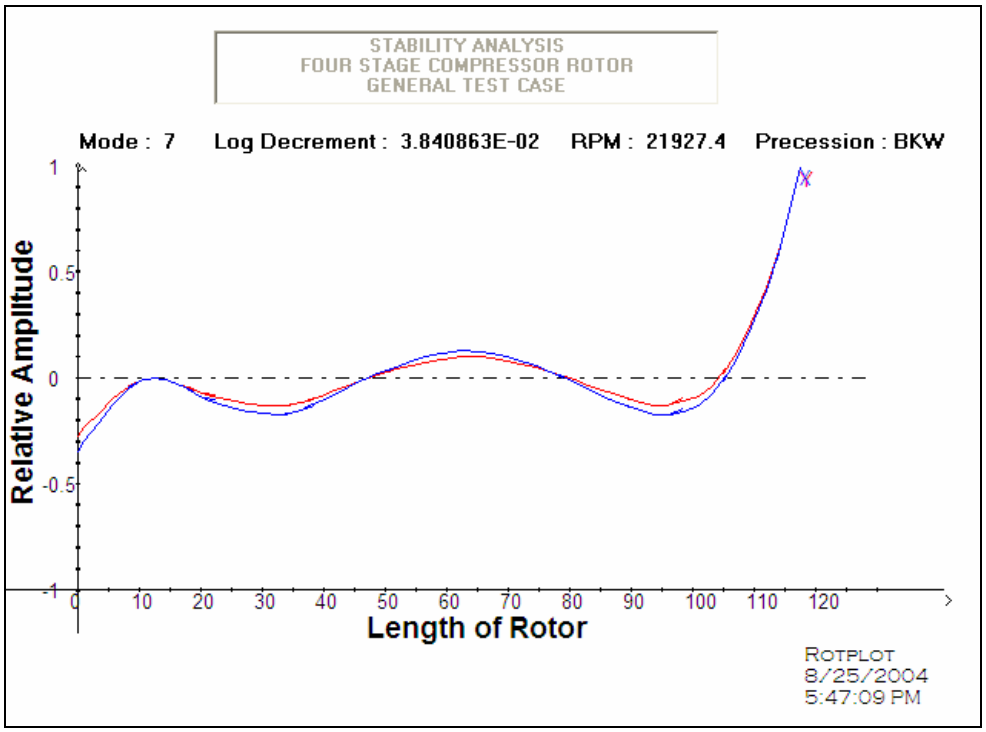


Fig 4.51 Seventh damped mode shape

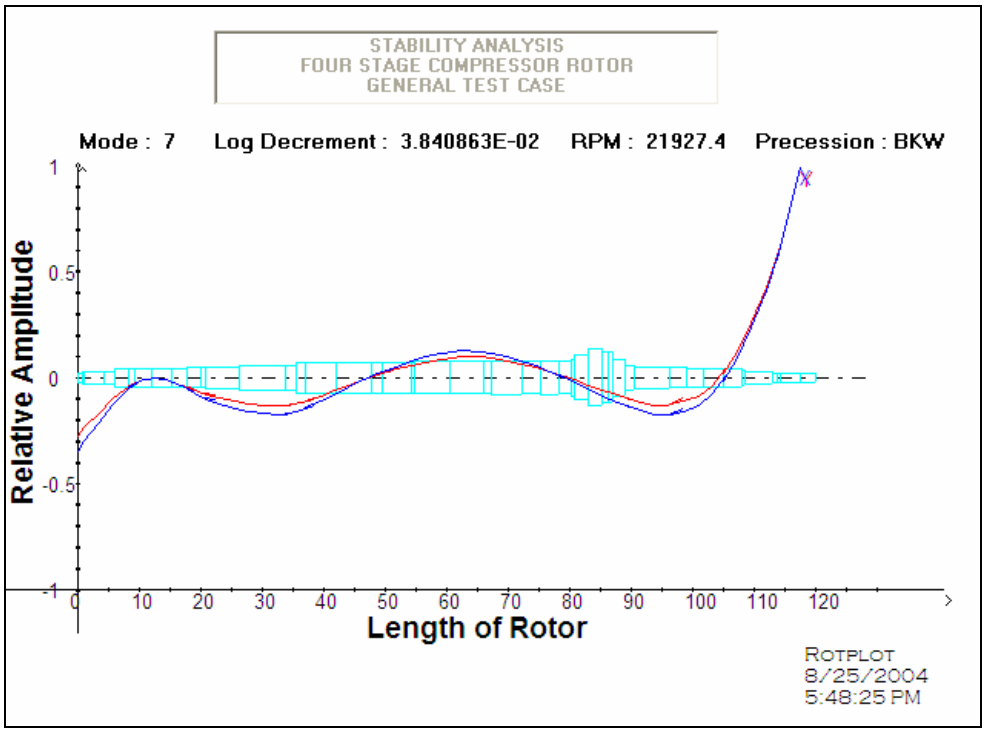


Fig 4.52 Damped mode shape with rotor dynamic model superimposed

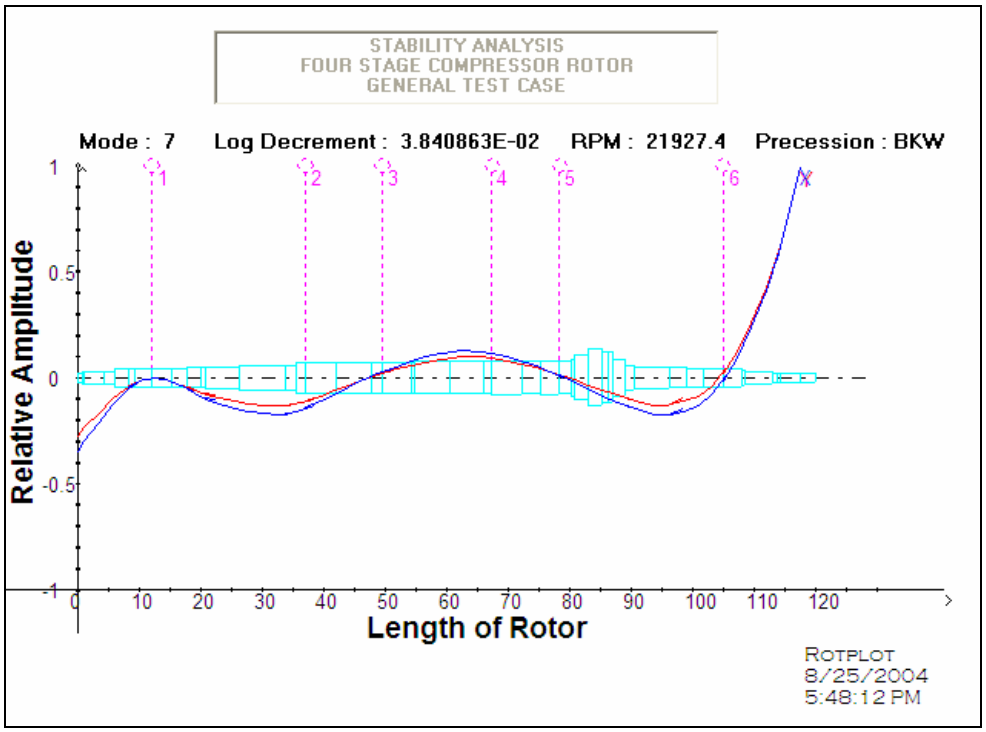


Fig 4.53 Damped mode shape with rotor and bearings location superimposed

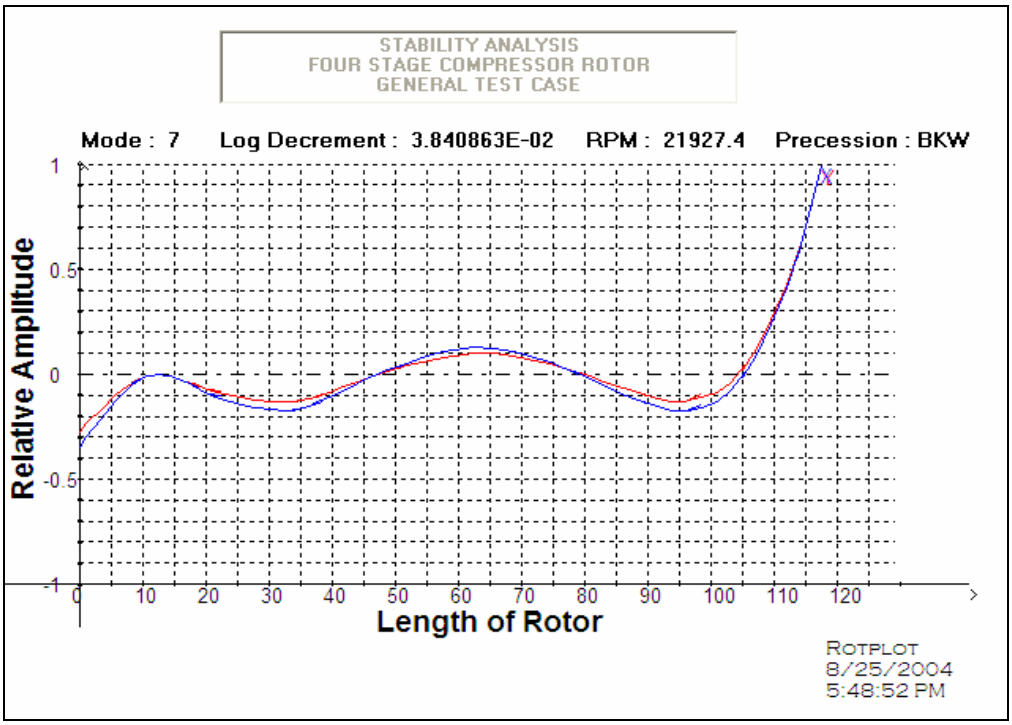


Fig 4.54 Damped mode shape with grid option turned on

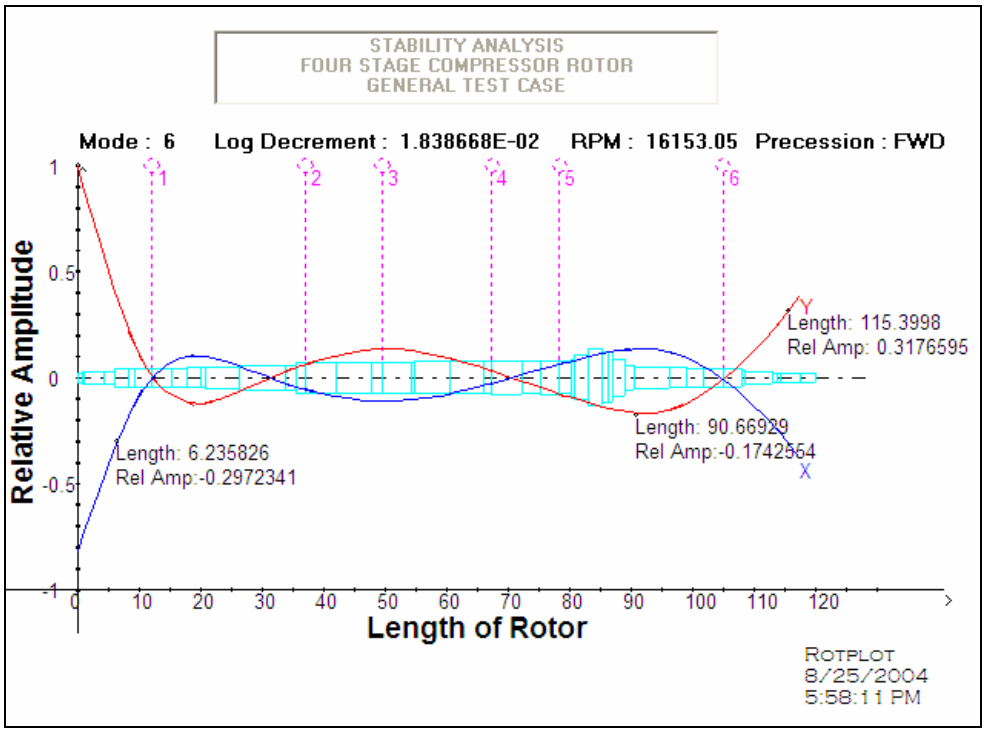


Fig 4.55 Damped mode shape with specific position parameters selected

4.4.3 Synchronous Response Analysis Sample Plots

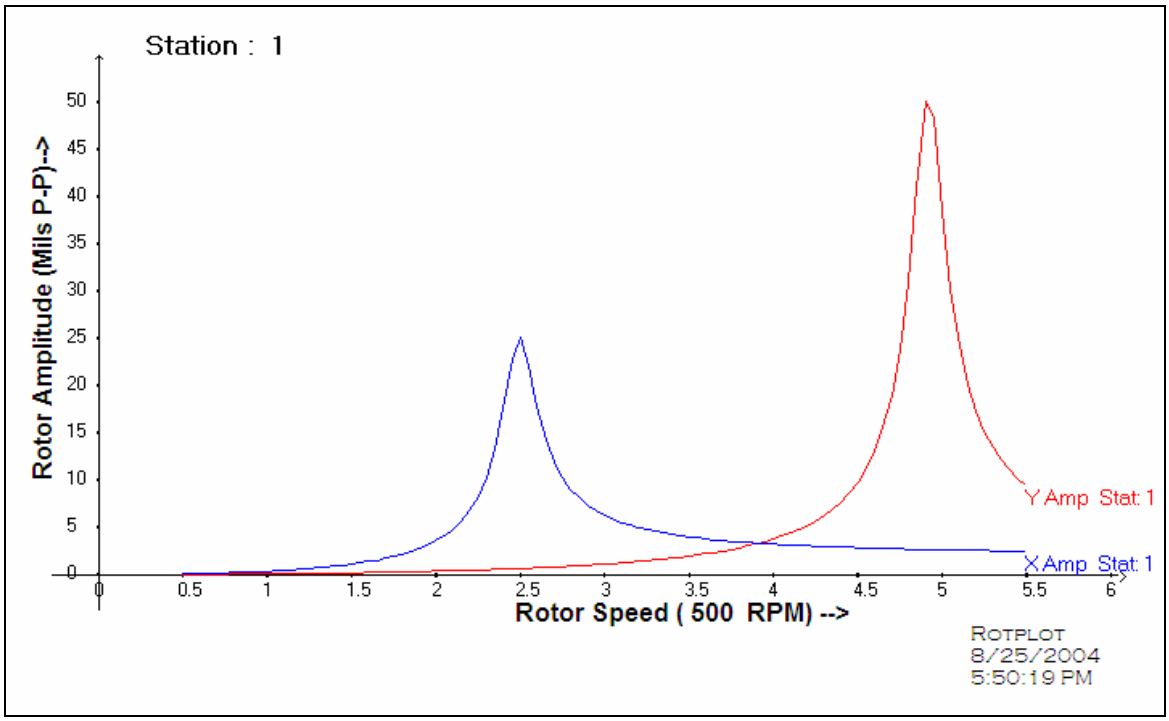


Fig 4.56 Synchronous response of the rotor at first station location

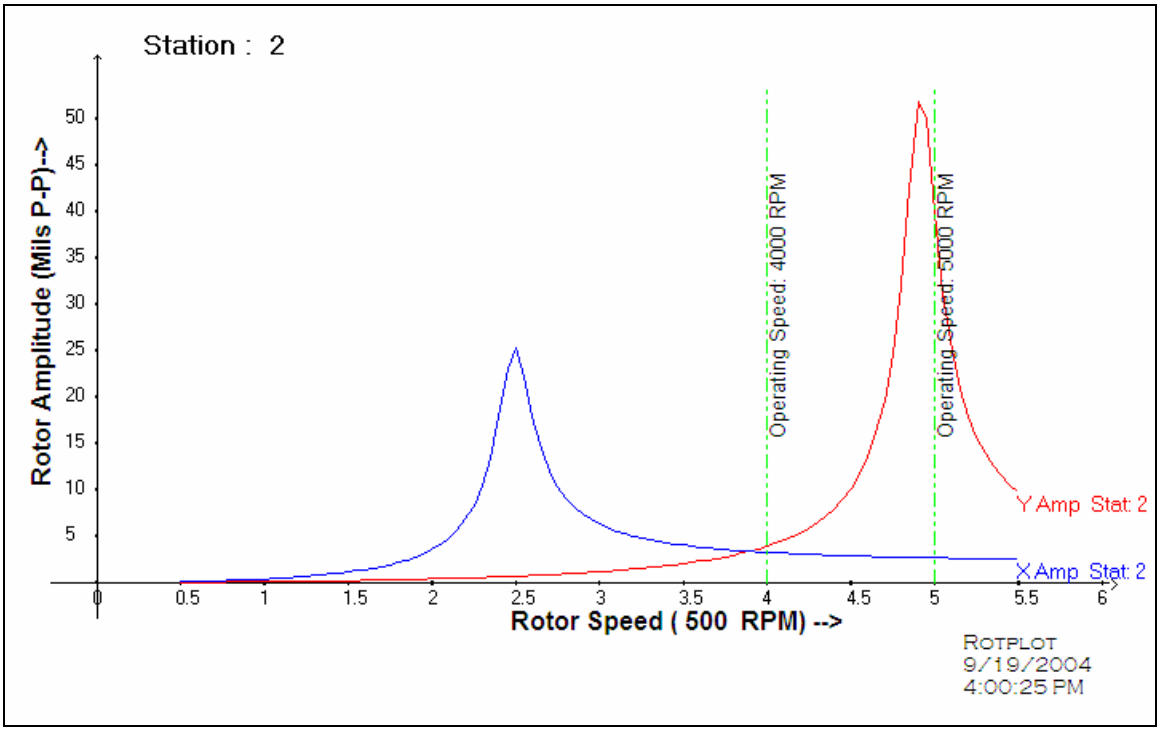


Fig 4.57 Synchronous response of the rotor at second station location

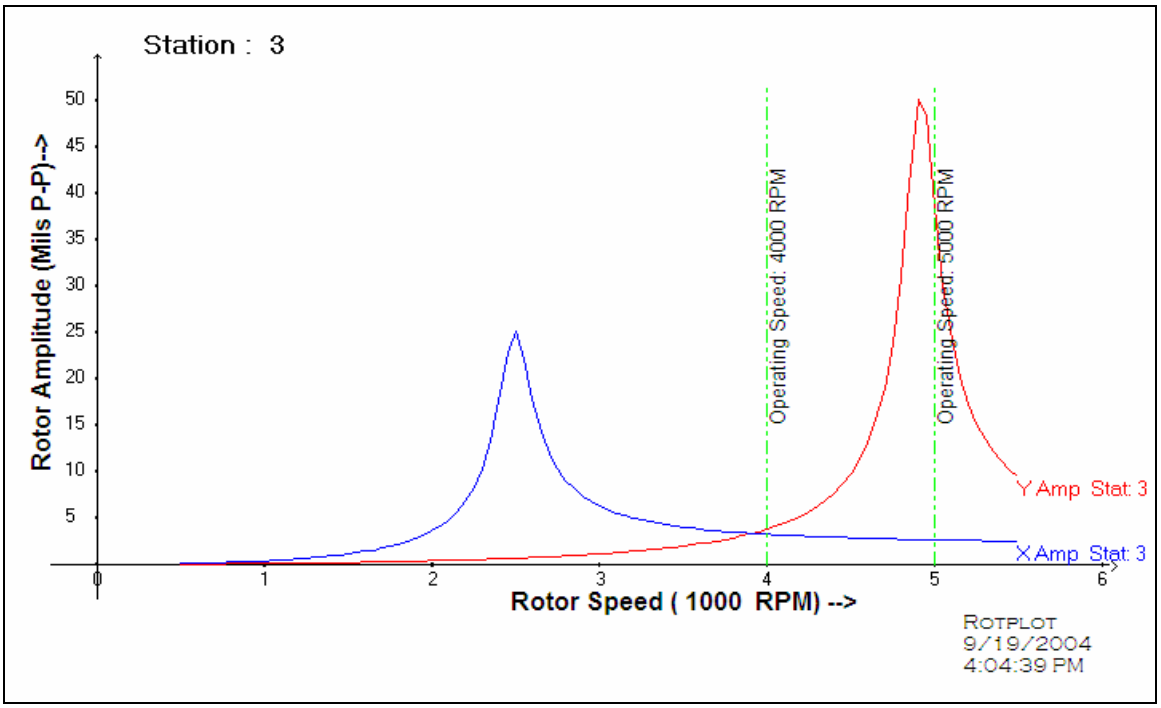


Fig 4.58 Synchronous response of the rotor at third station location

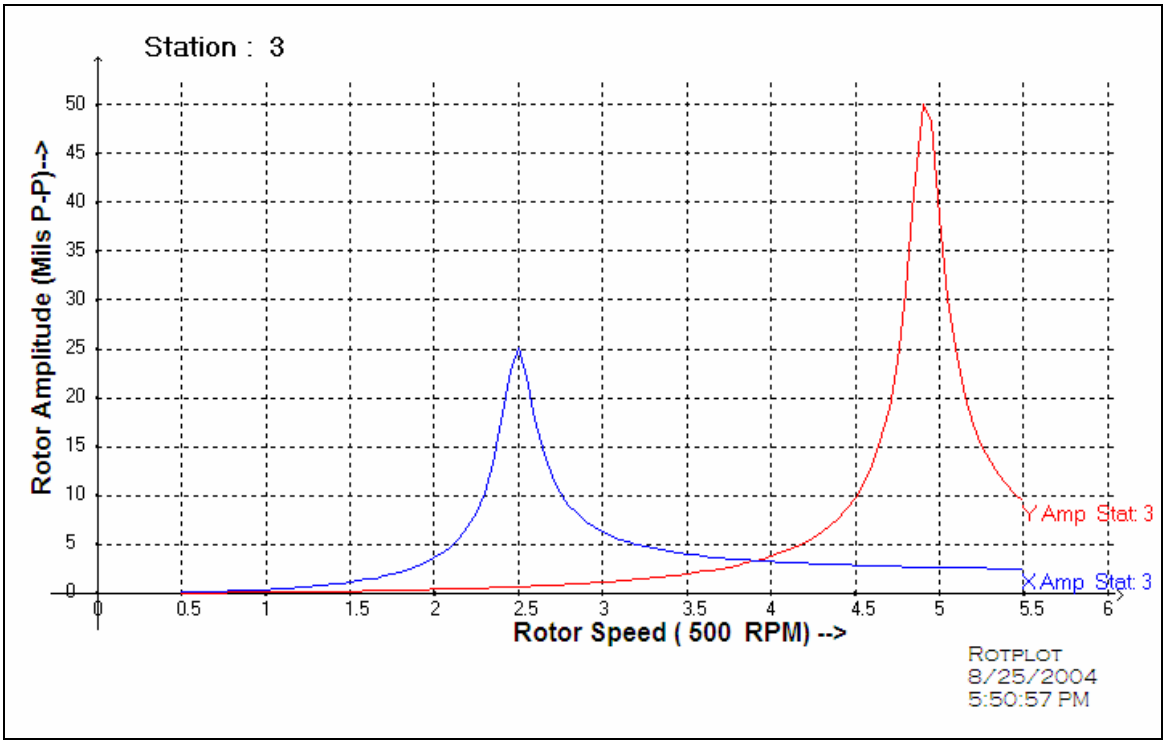


Fig 4.59 Synchronous response of the rotor with grid option turned on

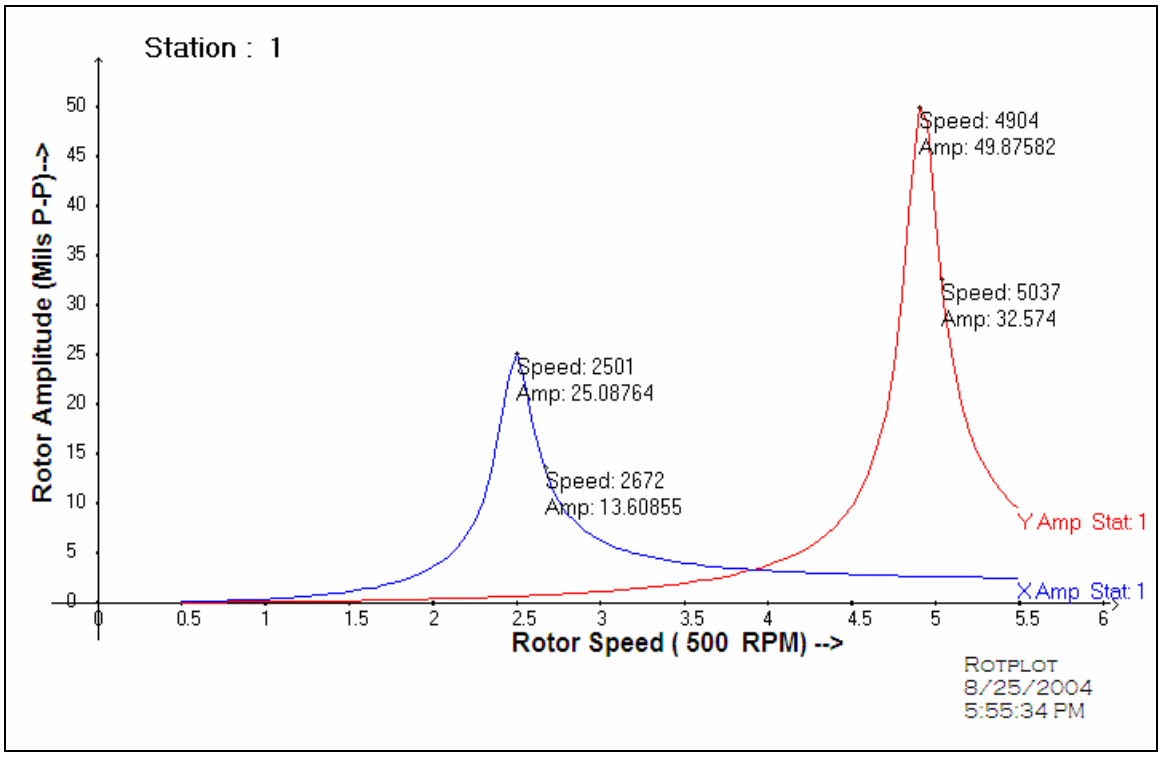


Fig 4.60 Synchronous response plot with specific location parameters highlighted

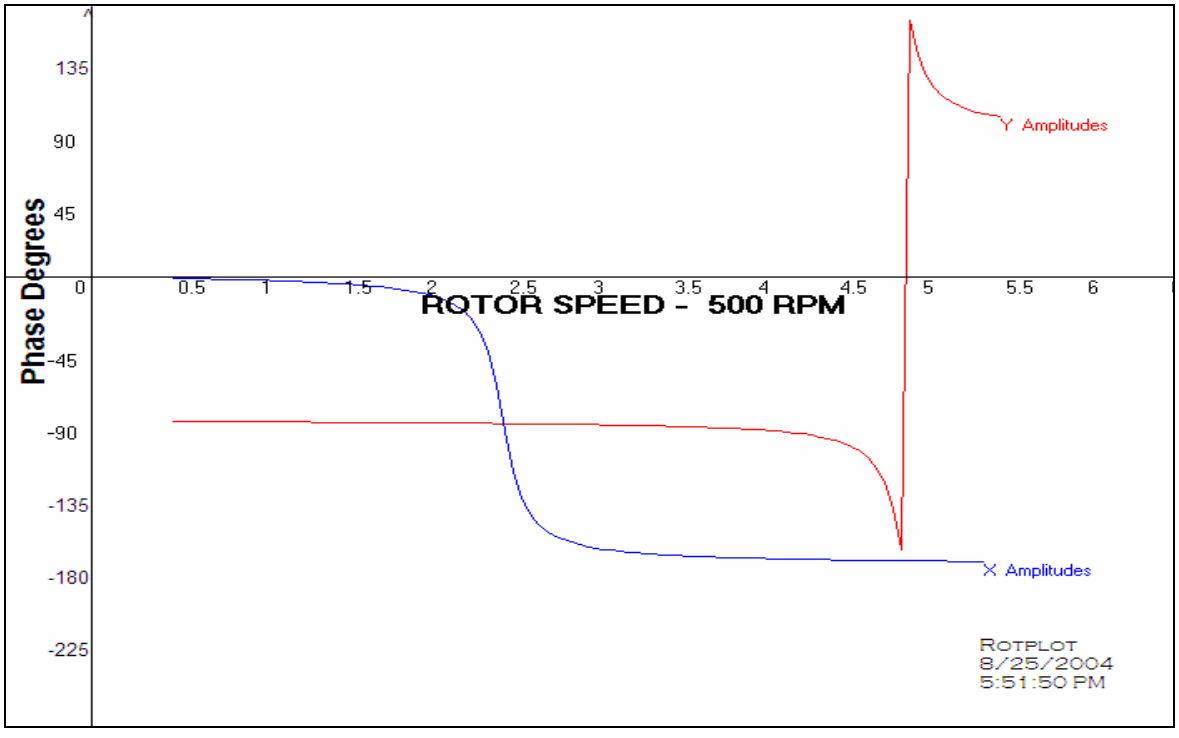


Fig 4.61 Phase response plot

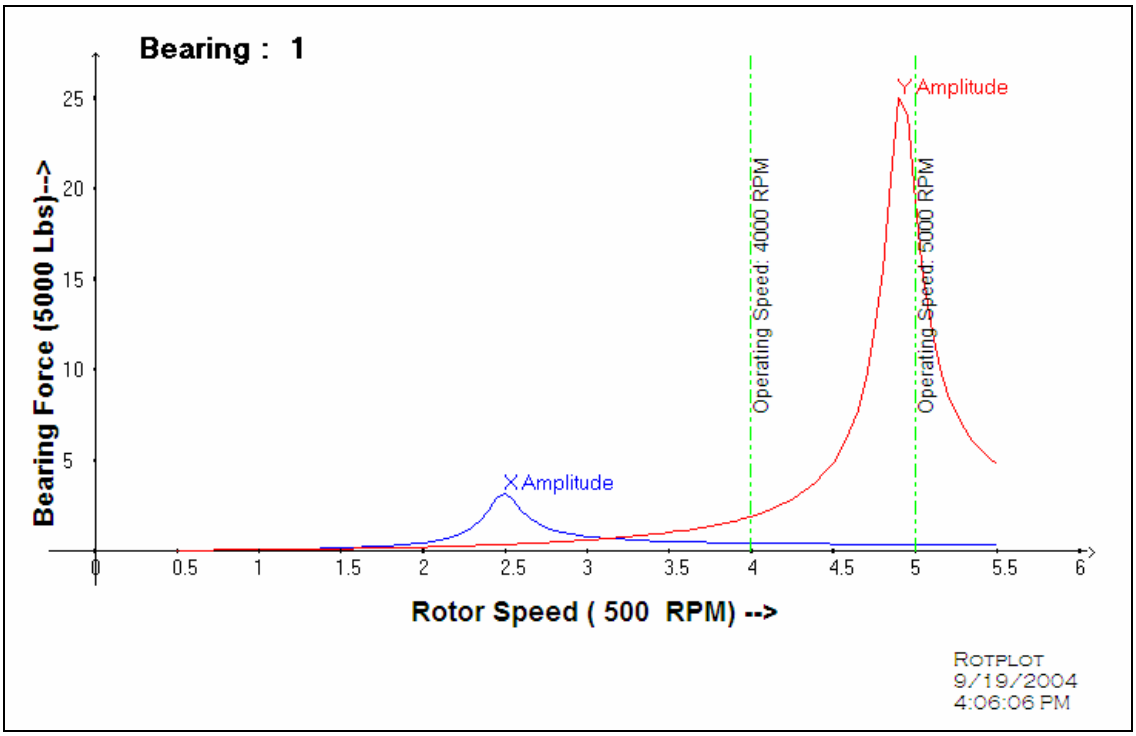


Fig 4.62 Bearing response plot at first bearing location

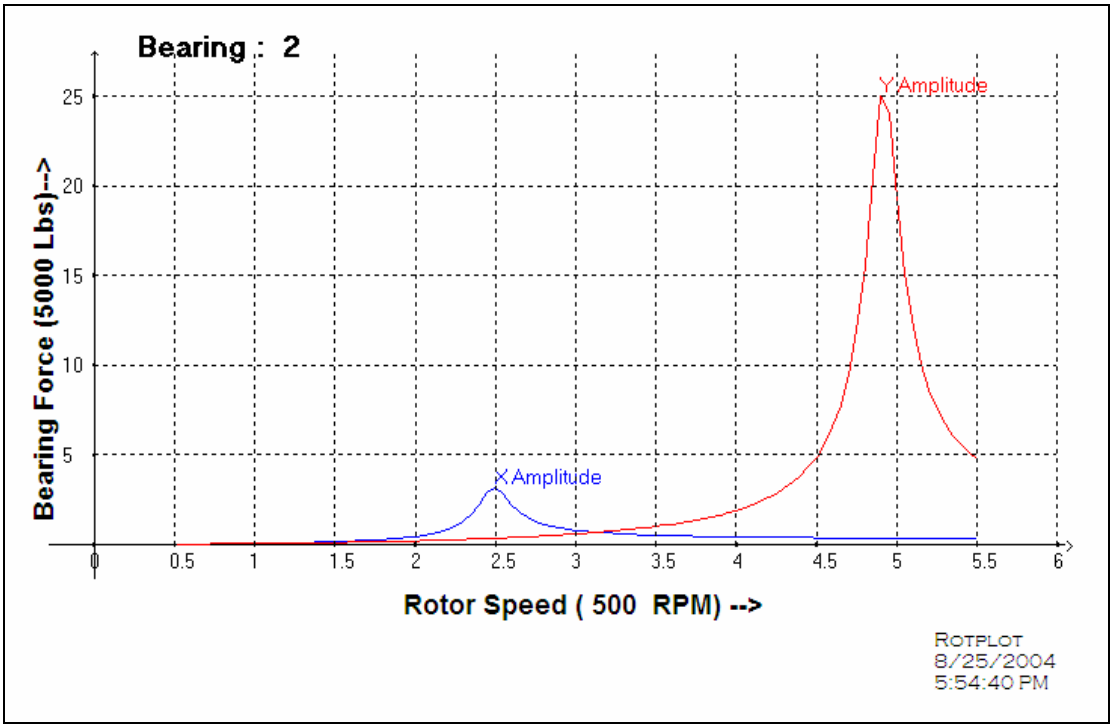


Fig 4.63 Bearing response plot at second bearing location

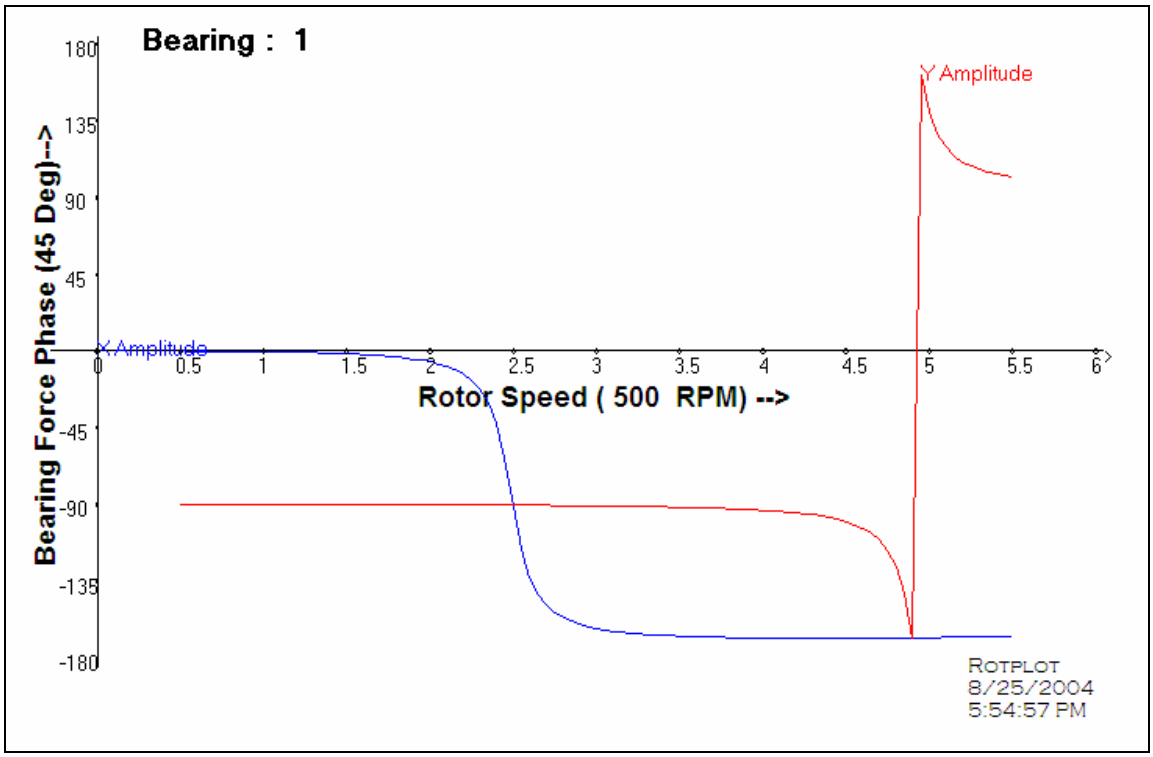


Fig 4.64 Bearing phase plot

CHAPTER 5
CONCLUSIONS AND FUTURE WORK

5.1 CONCLUSIONS

This thesis work involved the development of a computer program that would offer the designer and analyst of a complex rotor bearing system, an opportunity to obtain data and graphic display of rotor dynamic analysis results in an effective and easy way. As was mentioned earlier in this thesis, an efficient rotor bearing design requires the quantitative evaluation of how the interaction of variables of primary importance affects the rotor bearing system behavior in operating conditions. The modified processor, as part of the program, is developed to produce the relevant files of a particular analysis. This approach helps an analyst in verifying the data for a particular design consideration. The ability to quickly change the design of a rotor and view the results instantly will enable the designer to obtain an efficient rotor dynamic system.

The subject program of this thesis has the ability to further post process the analysis files and visualize results in the form of plots, which further enhance the designer's insight of a particular design. Most of its user friendly features such as the default analysis files directory selection, ability to multitask in combination with Windows operating system based operations, etc., make it a powerful tool for rotor dynamic analysis. Furthermore, it provides options for the user to save the result plots to any user selected file in a picture format, copy them to clipboard, access it instantly from any other compatible program or instantly print it to a printer. The built-in-multi-analysis aspect of the program allows the designer to conduct a comprehensive evaluation of the dynamic behavior of the rotor bearing assembly, thus enhancing the mechanical soundness of the system being designed.

The computer software, as being developed, represents a useful tool for designers of modern turbomachinery and promises to contribute to the ever demanding quest of quality in design through comprehensive analytical simulation of the phenomena pertaining to rotor dynamic analysis.

5.2 FUTURE WORK

The scope of the analysis program allows the user to perform the main analyses namely undamped critical speed analysis, stability analysis and synchronous response. A revised version of the program should have the capability to conduct the transient response analysis of a rotor dynamic system.

The program in the present configuration does not offer the capability to show the orbit information for the synchronous response analysis. Future versions could provide this option.

The present program version has an in-built processor and a compatible post processor packed into a single unit. The user is required to manually develop the input file in a format compatible to the processor. At the time of compilation of this thesis, effort was being put by the author to integrate a pre-processor program, initially developed by Dr.Carlo Roso, with the current program, which would enhance the program's capability as a comprehensive rotor dynamic design and analysis program.

The current version of the software is dedicated to a single user operation. Even though at the present time this is not viewed as a significant limitation, consideration should be given in the future to adapt the code for multi user service.

APPENDIX

A1. Cubic Spline Visual Basic Code

```
Public Sub SplineCurve_Modes(X0, Y0, X1, Y1, X2, Y2, X3, Y3, ByVal MinVal As Single, _
ByVal MaxVal As Single)

    ' Spline Logic (Dissolved into computer equations for stability mode plots)

  On Error Resume Next
  Dim MaxX, Px, Py, X, Y
  Dim A1, A2, B1, B2, C1, C2, D1, D2, M1, M2, M3
  Dim A, B, C, D

  A1 = (X1 ^ 3 - X2 ^ 3) - 3 * (X2 ^ 2) * (X1 - X2)
  B1 = (X1 ^ 2 - X2 ^ 2) - 2 * X2 * (X1 - X2)
  A2 = 3 * (X1 ^ 2 - X2 ^ 2)
  B2 = 2 * (X1 - X2)

  M1 = (Y1 - Y0) / (X1 - X0)
  M2 = (Y2 - Y1) / (X2 - X1)
  M3 = (Y3 - Y2) / (X3 - X2)
  D1 = (M1 + M2) / 2
  D2 = (M2 + M3) / 2
  C1 = Y1 - Y2 - D2 * (X1 - X2)
  C2 = D1 - D2

  A = (C1 * B2 - C2 * B1) / (A1 * B2 - A2 * B1)
  B = (C2 - A2 * A) / B2
  C = D2 - 3 * X2 ^ 2 * A - 2 * X2 * B
  D = Y2 - A * X2 ^ 3 - B * X2 ^ 2 - C * X2
```

```
'   FrmModesPlot.Circle (X0, Y0), 0.002, vbBlack    ' Optional plotting of data points on plots
'   FrmModesPlot.Circle (X3, Y3), 0.002, vbBlack
      If  X1 < X2 Then
         X = X1
         MaxX = X2
         Py = Y1
   Else
         X = X2
         MaxX = X1
         Py = Y2
   End If
   Px = X
   '  Drawing cubic curve
   '  Form2.DrawStyle = 2              ‘ Optional setting to change the plot drawing style
   Do While X < MaxX
         X = X + 0.02
         Y = A * X * X * X + B * X * X + C * X + D
         Y = Y
               If Y < MinVal Then              ' Condition to prevent the spline to cross the _
                     Y = MinVal                '  lower bound
               ElseIf Y > MaxVal Then          ' Condition to prevent the spline to cross the _
                     Y = MaxVal                '  upper bound
               End If
         FrmModesPlot.Line (Px, Py)-(X, Y)     ' Main plotting
         Px = X
         Py = Y
Loop
' Form2.DrawStyle = 0                ' Optional setting back the draw settings to original
End Sub
```

REFERENCES

This section contains a listing of published literature relating to the subject of the thesis. The complete literature on the subject of rotordynamics and computer modeling of the associated phenomena is very extensive and is beyond the scope of this thesis. The listing, therefore, has been restricted to those publications that constitute the more significant contributions available.

REFERENCES

Alford, J. (1965), "Protecting Turbomachinery from Self-Excited Rotor Whirl", *Journal of Engineering for Power*, pp. 333-334.

Alistair D. N. Edwards., Web Resource (http://www.rit.edu/~easi/itd/itdv02n4/article3.html)

Bernard J. J. (1998), Web Resource (http://jimjansen.tripod.com/academic/pubs/chi.html)

Childs, D. (1993), "Turbomachinery Rotordynamics", John Wiley & Sons, Inc.

Chree, C. (1904), "The Whirling and Transverse Vibration of Rotating Shafts", *Phil. Mag.* Vol.7, p.504.

Cortes, L. (1997) Web Resource (http://www.medicalcomputingtoday.com/0agui.html)

Dimentberg, F.M. (1964), "Theory of Balancing Flexible Rotors", *Russ. Engg. Journal*, pp.11.

Donald, R.P., Oancea, G. (1999), "Visual Basic 6 from Scratch", Indianapolis, Ind. Que.

Downham, E. (1954), "The Critical Whirling Speeds and Natural Vibrations of a Shaft Carrying a Symmetrical Rotor", Ministry of Supply, ARC R&M No.2854

Drayton, P., Albahari, B., Neward, T. (2002), "C# in a Nutshell", O'Reilly.

Dunkerley, S. (1895), "On the Whirling and Vibration of Shafts", *Phil. Trans. Roy. Soc.*, London, Series A, Vol.185, p.279.

Ehrich, F.F. (1992), "Handbook of Rotordynamics", McGraw-Hill, Inc., New York.

Green,R.B. (1948), “Gyroscopic Effects on the Critical Speeds of Flexible Rotors”, Trans. ASME, *Jrnl. Appl. Mech.* P.369.

Gurewich, N., Gurewich, O. (1997), “Visual Basic 5”, Sams Publishing.

Huseyin, K. (1976), “Vibration and Stability of Mechanical Systems”, *Shock and Vibration Digest*, Vol. 8, No.4, pp. 56-66.

Jeffcott, H.H. (1919), “The Lateral Vibration of Loaded Shafts in the Neighbourhood of a Whirling Speed – The Effect of Want of Balance”, *Phil. Mag.*, Ser. 6, Vol.37, p.304.

Johnson, N. (1987), “Advanced Graphics in C”, Mc Graw Hill Osborne Media.

Kerr, W. (1916), “On the Whirling Speed of Loaded Shafts”, Engineering, Feb. 18, 1916, pp.150, 296, 386, 410, 420.

Kimball, A.L. (1925), “Internal Friction as a Cause of Shaft Whirling”, *Phil. Mag.*, S6, Vol. 49, pp. 724-727.

Kincaid, D., Cheney, W. (1996), “Numerical Analysis”, 2nd Edition, Brooks/Cole Publishing.

Lewis, R.M. (1960), Vibration During Acceleration Through a Critical Speed”, Trans. Amer. Mech. Engrs., Vol. 54, pp 253-261.

Liberty, J. (2002), “Learning C #”, O’ Reilly.

Liberty, J. (2002), “Learning Visual Basic.NET”, O’ Reilly.

Logan, D.L. (2001), “A First Course in the Finite Element Method”, 2nd Edition, Brooks/Cole Publishing.

Lund, J.W. and Sternlicht, B. (1961), "Rotor-Bearing Dynamics with Emphasis on Attenuation", ASME, *Trans. Journal of Basic Engineering*, Ser. D., pp. 491-502.

Lund, J.W. (1975), "Stability and Damped Critical Speeds of a Flexible Rotor in Fluid-Film Bearing", Trans ASME, *Journal of Engr for Industr*, pp. 509-517.

Malanoski, S.B. (1975), "Rotor Bearing System Audit", Proceeding of the 4th Turbomachinery Symposium, Texas A&M University.

Mathews, J. H., Fink, K. F. (1999), "Numerical Methods Using MATLAB", Thord Edition Prentice Hall International, Inc.

MicroGlyph/SciPlot™ Graphics Library Documentation Manual.

Newkirk, B.L. (1924), "Shaft Whipping", General Electric Rev. Vol. 27, pp. 169-178.

Nicholas, J. (1977), "A Finite Element Dynamic Analysis of Pressure Dam and Tilting-Pad Bearings", Ph.D Thesis, University of Virginia, Charlottesville, VA.

Nicholas, J. and Barrett, l. (1985), "The Effect of Bearing Support Flexibility on Critical Speed Prediction", ASLE Transactions, No.29, pp. 329-338.

Naughton, P., Schildt, H. (1999), "Java 2 - A Complete Reference", 3rd Edition, Tata McGraw Hill Publishing Company Limited.

Rankine, W.J. (1869), "On the Centrifugal Force of Rotating Shafts", Engineer, London, Vol.27, p. 249.

Reynolds, M., Blair, R., Crossland, J. (2003), "Beginning VB.NET", 2nd Edition, Wiley Publishing Inc., Indianapolis, IN.

Rieger, N.F. (1976), "A Comprehensive Guide to Computer Programs for Analyzing Rotor Systems", Machine Design, pp. 89-95.

Rodgers,C. (1922), "On the Vibration and Critical Speeds of Rotors", *Phil. Mag.*, Ser. 6, Vol. 44, pp. 122-156.

Roso, C.A. (1995), "A Development of a Computer Aided User Interface for Dynamic Studies of Rotor Bearing Systems", M.S. Thesis, University of Kentucky, Lexington, KY.

Roso, C.A. (1997), "Design Optimization of Rotor-Bearing Systems for Industrial Turbomachinery Applications", Ph.D. Dissertation, University of Kentucky, Lexington, KY.

Rouch, K.E. (1977), "Finite Element Analysis of Rotor-Bearing Systems with Matrix Reduction", Ph.D. Dissertation, Marquette University, Milwaukee, WI.

Rouch, K.E. (2001), User's Manual for RotBrg© Ver. 5.0.0 - A Computer Program for Rotor-Bearing Systems Analysis, University of Kentucky, Lexington, KY.

Rouch, K.E. and Kao, J. (1979), "A Tapered Beam Finite Element for Rotordynamics Analysis", *Journal of Sound and Vibration*, Vol.66, pp. 119-140.

Smith, D.M. (1933), "The Motion of a Rotor Carried by a Flexible Shaft in Flexible Bearings", Proc. Roy. Soc. Series A, Vol. 142, pp. 92-118.

Smith, D. M. (1966), "Vibrations in Turbomachinery", Proc. Inst. Mech Engrs., London.

Stephenson, R. and Rouch, K.E. (1990), "Rotor Modeling Considerations with Shaft Diameter Changes", Dynamics of Rotating Machinery, Hemisphere Publishing.

Stevens, A. (2000), "Teach Yourself C++", 6th Edition, IDG Books Worldwide Inc.

Stodola, A. (1927), “Steam and Gas Turbines”, Vols. I and II, pp. 491, 929, 931, 1122. McGraw-Hill Book Co., Inc., New York.

Vance, J.M. (1988), “Rotor Dynamics of Turbomachinery”, 1st Edition, John Wiley & Sons, Inc., New York.

Web Resource (http://www.wikipedia.org/wiki/Graphical_user_interface)

Web Resource (http://asds.stsci.edu/graphics_pack.html)

Web Resource (http://www.webopedia.com/term/g/graphical_user_interface_GUI.html)

Web Resource (http://www.ucc.ie/gnuplot/gnuplot-faq.html)

Web Resource (http://www.tecplot.com/products/tecplot/tecplot_features.htm)

Web Resource (http://www.wolfram.com/products/mathematica/introduction.html)

Web Resource (http://scv.bu.edu/Tutorials/Plotting/plotting101.html)

Web Resource (http://encyclopedia.thefreedictionary.com/graphical%20user%20interface)

Weisstein, E.W. (1999), “Cubic Spline” Web Resource (http://mathworld.wolfram.com)

Weser,J., Web Resource, (http://kisssoft.gwj.de/index.html)

VITA

Pavan Kumar Arise was born on February 9th 1980, in Andhra Pradesh, India. He did his schooling from St.Ann's high school. He then completed his intermediate education from St.Alphonsa's junior college, Hyderabad. He obtained his Bachelors degree in Mechanical Engineering from University college of Engineering, Osmania University, Hyderabad in May 2001. He is a member of A.S.M.E.