# NAVAL POSTGRADUATE SCHOOL
# Monterey, California

# THESIS

**A COMPUTER CODE FOR RAPID CALCULATION OF BENDING FREQUENCIES OF ROTOR BLADES**

by


Hakki E. Akin


September 2002


Thesis Advisor: E. Roberts Wood
Co-Advisor: Mark A. Couch


**Approved for public release; distribution is unlimited.**

THIS PAGE INTENTIONALLY LEFT BLANK

| REPORT DOCUMENTATION PAGE | | | *Form Approved OMB No. 0704-0188* |
|---|---|---|---|

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.

| **1. AGENCY USE ONLY** | **2. REPORT DATE** September 2002 | **3. REPORT TYPE AND DATES COVERED** Master's Thesis |
|---|---|---|

**4. TITLE AND SUBTITLE**: A Computer Code for Rapid Calculation of Bending Frequencies of Rotor Blades

**5. FUNDING NUMBERS**

**6. AUTHOR(S) Akin, Hakki, E.**

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**
Naval Postgraduate School
Monterey, CA 93943-5000

**8. PERFORMING ORGANIZATION REPORT NUMBER**

**9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES)**
N/A

**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**

**11. SUPPLEMENTARY NOTES** The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

**12a. DISTRIBUTION / AVAILABILITY STATEMENT**
Approved for public release; distribution is unlimited.

**12b. DISTRIBUTION CODE**

**13. ABSTRACT**

This thesis provides a MATLAB® code and graphical user interface (GUI) which rapidly estimates the bending frequencies of rotating beams from the knowledge of bending frequencies of nonrotating beams. The approach is based on the Rayleigh energy method developed by Yntema. The beams may be rotating or nonrotating, cantilever or hinged, uniform or with linear mass and stiffness distributions, with or without root offsets from the axis of rotation; or uniform with or without tip mass. Especially, the frequencies of both nonrotating and rotating cases can be estimated for (a) beams with and without offset which have mass and stiffness distributions which can be approximated by linear relations and (b) beams with uniform mass and stiffness distributions plus a concentrated mass at the tip. Also, as a part of the MATLAB® code, the bending frequencies of rotating beams with arbitrary stiffness and mass distributions can be estimated given that the stiffness and mass distributions as well as mode shapes of the rotating beam are provided. In latter case, the mode shapes of nonrotating beams may be used to get rough estimations. The code also presents the nonrotating bending-mode shapes in conjunction with the bending frequencies. The code and GUI are intended for use as a subprogram of JANRAD computer program developed at the Naval Postgraduate School, but can also be used as a stand-alone MATLAB® program.

**14. SUBJECT TERMS** Bending Frequency, Mode Shape, Rotating Beam, Nonrotating Beam, Rotor Blades, Rayleigh, MATLAB, JANRAD, GUI

**15. NUMBER OF PAGES** 219

**16. PRICE CODE**

| **17. SECURITY CLASSIFICATION OF REPORT** | **18. SECURITY CLASSIFICATION OF THIS PAGE** | **19. SECURITY CLASSIFICATION OF ABSTRACT** | **20. LIMITATION OF ABSTRACT** |
|---|---|---|---|
| Unclassified | Unclassified | Unclassified | UL |

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

# A COMPUTER CODE FOR RAPID CALCULATION OF BENDING FREQUENCIES OF ROTOR BLADES

Hakki E. Akin
First Lieutenant, Turkish Army
B.S., Istanbul Technical University, 1996

Submitted in partial fulfillment of the
requirements for the degree of

**MASTER OF SCIENCE IN AERONAUTICAL ENGINEERING**

from the

**NAVAL POSTGRADUATE SCHOOL**
**September 2002**

Author:
Hakki E. Akin

Approved by:
E. Roberts Wood
Thesis Advisor

Mark A. Couch
Co-Advisor

Max F. Platzer
Chairman, Department of Aeronautics and Astronautics

THIS PAGE INTENTIONALLY LEFT BLANK

# ABSTRACT

This thesis provides a MATLAB® code and graphical user interface (GUI) which rapidly estimates the bending frequencies of rotating beams from the knowledge of bending frequencies of nonrotating beams. The approach is based on the Rayleigh energy method developed by Yntema. The beams may be rotating or nonrotating, cantilever or hinged, uniform or with linear mass and stiffness distributions, with or without root offsets from the axis of rotation; or uniform with or without tip mass. Especially, the frequencies of both nonrotating and rotating cases can be estimated for (a) beams with and without offset which have mass and stiffness distributions which can be approximated by linear relations and (b) beams with uniform mass and stiffness distributions plus a concentrated mass at the tip. Also, as a part of the MATLAB® code, the bending frequencies of rotating beams with arbitrary stiffness and mass distributions can be estimated given that the stiffness and mass distributions as well as mode shapes of the rotating beam are provided. In latter case, the mode shapes of nonrotating beams may be used to get rough estimations. The code also presents the nonrotating bending-mode shapes in conjunction with the bending frequencies. The code and GUI are intended for use as a subprogram of JANRAD computer program developed at the Naval Postgraduate School, but can also be used as a stand-alone MATLAB® program.

THIS PAGE INTENTIONALLY LEFT BLANK

## TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

# LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

# LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

# TABLE OF SYMBOLS

| | |
|---|---|
| $a$ | Nonrotating Beam Bending Frequency Coefficient |
| $EI(x)$ | Lengthwise Bending Stiffness Distribution for Beam |
| $EI_0$ | Bending Stiffness of Beam at Root |
| $K$ | Southwell Coefficient |
| $K_0$ | Zero-offset Southwell Coefficient |
| $\bar{K}_0$ | Nondimensional Zero-offset Southwell Coefficient which is Identical to $K_0$ |
| $K_1$ | Offset-correction Factor for Southwell Coefficient |
| $\bar{K}_1$ | Nondimensional Offset-correction Factor for Southwell Coefficient, $K_1L$ |
| $K_0'$ | Zero-offset Rotating Beam Frequency Coefficient which is Essentially Independent of Beam Mass Distribution |
| $L$ | Beam Length, Measured from Point of Root Fixity to Tip |
| $m(x)$ | Lengthwise Mass-per-unit-length Distribution for Beam |
| $m_0$ | Mass per Unit Length of Beam at Root |
| $\bar{m}(\bar{x})$ | Nondimensional Mass Distribution for Beam, $m(\bar{x})/m_0$ |
| $M_t$ | Mass Concentrated at Tip of Beam |
| $r$ | Nondimensional Tip-mass Ratio for Uniform Beam, $M_t/m_0L$ |
| $T_1$ | Lengthwise-distribution Function for Tension Force |
| $x$ | Spanwise Coordinate along Beam Measured from Root |
| $\bar{x}$ | Nondimensional Spanwise Coordinate, $x/L$ |
| $Y$ | Bending Mode Shape of Nonrotating Beam |
| $y$ | Bending Mode Shape of Rotating Beam |
| $e$ | Offset of Hinge or Point of Fixity from Axis of Rotation |
| $\bar{e}$ | Nondimensional Offset, $e/L$ |
| $\eta, \bar{\eta}$ | Dummy Variables for $x$ and $\bar{x}$ |
| $\theta$ | Characteristic Number for Nonrotating Uniform Beam with Mass at Tip; Identical to square root of Nonrotating Beam Bending Frequency Coefficient |
| $\Omega$ | Rotational Speed of Beam |
| $\omega_R$ | Natural Bending Frequency of Rotating Beam |
| $\omega_{NR}$ | Natural Bending Frequency of Nonrotating Beam |
| $\psi$ | Angle of Attack for the Rotating Beam |

Subscripts:

| | |
|---|---|
| $n$ | The Natural Bending Mode Under Consideration |
| $F$ | Beam Cantilevered or Fixed at Root |
| $0$ | Root of Beam |
| $t$ | Tip of Beam |

Primes:

| | |
|---|---|
| $'$ | First Differentiation with respect to $x$ or $\bar{x}$ |
| $''$ | Second Differentiation with respect to $x$ or $\bar{x}$ |

THIS PAGE INTENTIONALLY LEFT BLANK

# ACKNOWLEDGMENTS

I want to thank Prof. Wood and CDR Couch for their guidance and enthusiasm during the work in performing this study.

I also like to acknowledge my instructors for their support throughout my presence at the Naval Postgraduate School and the Turkish Armed Forces Command for their appreciation.

I dedicate this work to my mother, Fatma, for her support during my distance from home, and to my brother, Emre, for his patience here during this time-consuming thesis work.

THIS PAGE INTENTIONALLY LEFT BLANK

# I. INTRODUCTION

A helicopter blade is essentially a beam that rotates around the axis of rotation and the natural bending frequencies of the blade are of utmost importance for the resonance-free operation of the helicopter rotor system. Although the mode shape of a rotating beam is very similar to the mode shape of the nonrotating beam with same characteristics, the natural bending frequencies of rotating beams are quite different from those of nonrotating beams due to the centrifugal forces encountered in the rotating beams. The centrifugal force makes the blade seem stiffer and therefore causes the natural frequency of a rotating beam to increase appreciably.

Since the mode shapes of rotating and nonrotating beams are similar, according to the well-known Rayleigh principle, they may be used interchangeably for approximate results provided that the mode shape is consistent with the constraints of the system. The report by Robert T. Yntema, “Simplified Procedures and Charts for the Rapid Estimation of Bending Frequencies of Rotating Beams” (1955), is a good example of the application of the Rayleigh principle and can be summarized by the statement: ‘A Rayleigh energy approach utilizing the bending mode results of the nonrotating beam to determine the bending frequency of the rotating beam.’ [Ref. 1] Although dating back to 1955, this report has perhaps been used by almost every helicopter manufacturer especially in most of the preliminary design processes for its ease of use and practicality in giving quick and good results for helicopter blades. A helicopter blade must be designed as free as possible from resonant or near-resonant excitation of the periodic loading on the rotor, and therefore designers need a simple yet reasonably accurate method for determining the bending frequencies of rotating blades.

The main equation of this approach yields an exact result, but it requires that the mode shape of the rotating beam be known and the integrals have simple mathematical expressions for mass and stiffness distributions. By replacing the mode shape of the rotating beam with that of the nonrotating beam, Yntema makes an approximation to the bending frequency of the rotating beam. In addition, computing this main equation for some characteristic types of nonrotating beams, he reduces the whole process to a few Southwell coefficients that can be read from a set of charts. The report provides these quick-reference charts for the rapid estimation of first three bending frequencies of rotating or nonrotating beams. The first set of frequency charts provides the Southwell coefficients for the rotating or nonrotating hinged or cantilever beams with mass and stiffness distributions that can be approximated by linear relations, and with root offset from the axis of rotation. The second set of charts serves for the beams with uniform mass and stiffness distributions but no root offset, plus a concentrated mass at the tip. The uniform beams with a concentrated tip mass is also included in the procedure. For the validation of the method, Yntema uses a more exact but complex method to evaluate and compare with the results of Rayleigh energy approach. Classical information on elasticity and strength of materials can be found in References 2 through 8.

The purpose of this study is to digitize the Yntema method into a computer program for quick preliminary analysis of rotating beam bending frequencies. Southwell plots will be generated for vibration analysis and comparison. Nonrotating beam mode shapes will also be provided as a computer program to approximate the mode shapes of rotating beams since both rotating and nonrotating beams have similar mode shapes. A graphical user interface will be creates for ease of use and practicality.

A computer code in MATLAB® programming language developed by MathWorks, Inc., has been generated for the Yntema method incorporating the Rayleigh approach. The code is intended for use with JANRAD which is a rotorcraft analysis and design software based on the method described by Wood and Gerstenberger [Ref. 9] and developed by students at the Naval Postgraduate School, Monterey, Ca [Ref. 10~14]. The frequency coefficient and Southwell coefficient charts provided in the Yntema report, have been manually read into a MATLAB® file which is used by a MATLAB® function file to determine the first three bending frequencies of rotating or nonrotating hinged or cantilever beams for the mass and stiffness distribution, root offset, concentrated tip mass cases mentioned in the previous paragraph. A rotational speed range vector may be input to the MATLAB® function to get the corresponding frequency values in a vector. This option also allows for rapid demonstration of a variety of Southwell plots. Another

MATLAB® function gives the exact or more accurate Rayleigh method results provided that the mode shape of the rotating beam is known or can be approximated. With another MATLAB® file, mode shapes of the nonrotating beams can be plotted as provided by the Yntema method. A complete set of MATLAB® file scripts as well as the user guides have been provided in Appendices B through D. For information on MATLAB® software, you can refer to References 15 through 20.

Chapter II provides a general review of the Yntema report related to the intentions of this study. Chapter III provides the modeling of the Yntema method and the verification and validation of the software. Concluding remarks and recommendations for future use are provided in Chapter IV. Appendix A provides the user's guides for the computer code and graphical user interface (GUI). Appendices B through D lists the MATLAB® codes for the script files, functions, and GUI respectively. Appendix E gives several codes for the verification and validation of the software.

THIS PAGE INTENTIONALLY LEFT BLANK

# II. BACKGROUND: YNTEMA'S SIMPLIFIED PROCEDURES AND CHARTS FOR THE RAPID ESTIMATION OF BENDING FREQUENCIES OF ROTATING BEAMS

This chapter provides a review of the sections of the Yntema report [Ref. 1], which are essential for the development of the pertinent computer program. Some sections and appendices of the Yntema report, such as *mode-expansion method* solving high-order determinantal equations, which are intended for the evaluation of the Rayleigh approach, are not mentioned in a detailed fashion since they were not needed for the program.

## A. DESCRIPTION

The Rayleigh energy approach of obtaining the fundamental frequencies of vibration of an elastic system is based on the conservation of energy. By equating the kinetic energy at zero displacement to the potential energy of both the bending and centrifugal forces at maximum displacement, the following frequency equation can readily be derived for any vibration perpendicular to the plane of rotation:

$$\omega_{R_n}^2 = \frac{\int_0^L EIy_n''^2 dx}{\int_0^L my_n^2 dx} + \frac{\int_0^L T_1 y_n'^2 dx}{\int_0^L my_n^2 dx}\Omega^2 \tag{1}$$

where $n$ is the mode under consideration, and the lengthwise-distribution function for tension force is given by

$$T_1 = \int_x^L (\eta + e)\, md\eta \tag{2}$$

Equation(1) is the exact value for the n$^{th}$ bending frequency of a beam rotating at the rotational speed, $\Omega$, if the n$^{th}$ natural bending mode shape is known for this value of $\Omega$. The mode shape is usually an unknown as well as the frequency, but the well-known Rayleigh principle may be used for an estimation of the frequency: A mode shape which is consistent with the constraints of the system can be assumed to evaluate the energy integrals which then yield an approximation to frequency. Yntema chooses the nonrotating beam mode shape as the approximation for the rotating beam mode shape and evaluates whether the results are reasonably accurate.

If the n$^{th}$ mode shape of the nonrotating beam, $Y_n$, is substituted into Equation(1) instead of $y_n$, the equation can be rearranged in the following simple form:

$$\omega_{R_n}^2 = \omega_{NR_n}^2 + K_n \Omega^2 \tag{3}$$

where, $\omega_{NRn}$ is exactly the nonrotating beam bending frequency

$$\omega_{NR_n}^2 = a_n^2 \frac{EI_0}{m_0 L^4} \tag{4}$$

and $K_n$ is the *Southwell coefficient*

$$K_n = \frac{\int_0^L Y_n'^2 \left[ \int_x^L (\eta + e) m d\eta \right] dx}{\int_0^L m Y_n^2 dx} \tag{5}$$

The coefficient $a_n$ is referred to as the *nonrotating beam bending frequency coefficient*, and the Southwell coefficient can be written as

$$K_n = K_{0_n} + K_{1_n} e \tag{6}$$

where $K_{0n}$ is the *zero-offset Southwell coefficient*, and $K_{1n}$ is the *offset-correction factor for Southwell coefficient* defined by

$$\left.\begin{aligned} K_{0_n} &= \frac{\int_0^L Y_n'^2 \left[ \int_x^L \eta\, m\, d\eta \right] dx}{\int_0^L m Y_n^2 dx} \\ K_{1_n} &= \frac{\int_0^L Y_n'^2 \left[ \int_x^L m\, d\eta \right] dx}{\int_0^L m Y_n^2 dx} \end{aligned}\right\} \tag{7}$$

By means of preceding equations, Equation(3) may also be written as

$$\omega_{R_n}^2 = a_n^2 \frac{EI_0}{m_0 L^4} + \left( K_{0_n} + K_{1_n} e \right) \Omega^2 \tag{8}$$

By performing an integration by parts on the numerator of Equation(5), Yntema [Ref. 1] gives a slightly different but useful expression for $K_n$, and in nondimensional form it appears as

$$\bar{K}_n = \frac{\int_0^1 (\bar{x}+\bar{e})\bar{m}\left[\int_0^{\bar{x}} Y_n'^2 d\bar{\eta}\right] d\bar{x}}{\int_0^1 \bar{m} Y_n^2 d\bar{x}} \tag{9}$$

where

$$\left.\begin{aligned} \bar{K}_{0_n} &= \frac{\int_0^1 \bar{x}\bar{m}\left[\int_0^{\bar{x}} Y_n'^2 d\bar{\eta}\right] d\bar{x}}{\int_0^1 \bar{m} Y_n^2 d\bar{x}} \\ \bar{K}_{1_n} &= \frac{\int_0^1 \bar{m}\left[\int_0^{\bar{x}} Y_n'^2 d\bar{\eta}\right] d\bar{x}}{\int_0^1 \bar{m} Y_n^2 d\bar{x}} \end{aligned}\right\} \tag{10}$$

Equation(8) can now be written with these new coefficients as

$$\omega_{R_n}^2 = a_n^2 \frac{EI_0}{m_0 L^4} + \left(\bar{K}_{0_n} + \bar{K}_{1_n}\bar{e}\right)\Omega^2 \tag{11}$$

where the relationships between the Southwell coefficients are

$$\left.\begin{aligned} \bar{K}_{0_n} &= K_{0_n} \\ \bar{K}_{1_n} &= K_{1_n} L \end{aligned}\right\} \tag{12}$$

Equation(11) serves as the basis for the charts presented in the Yntema's report. These charts provide values of $a_n$, $K_{0n}$, and $\bar{K}_{1n}$ which, in conjunction with the mass ratio ($m_{tip}/m_0$) and the stiffness ratio ($EI_t/EI_0$) values, estimates the first three bending frequencies of rotating or nonrotating beams.

For the evaluation purposes, a new rearrangement for Equation(3) may be found to be useful:

$$\left(\frac{\omega_{R_n}}{\omega_{NR_n}}\right)^2 = 1 + K_n\left(\frac{\Omega}{\omega_{NR_n}}\right)^2 = 1 + K_n\left(\frac{\omega_{NR_1}}{\omega_{NR_n}}\right)^2\left(\frac{\Omega}{\omega_{NR_1}}\right)^2 \tag{13}$$

where, $\omega_{Rn}/\omega_{NRn}$ is referred to as the *frequency parameter* and $\Omega/\omega_{NR1}$ is referred to as the *rotational-speed parameter*.

Also defined is a new zero-offset rotational beam frequency coefficient, $K'_{0n}$, which will prove to be useful for subsequent use:

$$K'_{0_n} \equiv K_{0_n}\left[\frac{\left(\omega_{NR_1}\right)_F}{\omega_{NR_n}}\right]^2 \equiv K_{0_n}\left(\frac{a_{1_F}}{a_n}\right)^2 \tag{14}$$

where $F$ indicates that $a_1$ is the nonrotating beam frequency coefficient for the beam cantilevered at the root; all other terms are for the actual root condition, i.e., either cantilever or hinged. Because $K'_{0n}$ is found to be essentially insensitive to mass distribution, it should be useful in estimating bending frequencies of families of beams with similar stiffness distributions.

## B. EVALUATION

A series of rotating beam parameters are systematically varied for the calculation of bending frequencies based on Rayleigh approach. The results of a more exact but complex method are compared with these bending frequencies to determine the accuracy and limitations of Rayleigh approach.

Figure 1 shows the cases studied by both methods. Both cantilever and hinged beams are considered for:

- Uniform beams with zero and ten percent root offset.
- Mass and stiffness distributions varying linearly from the value at the root to zero at the tip ("linear") and root offsets of zero and ten percent.
- Uniform beams with a concentrated tip mass.

The results of all three cases are plotted with nondimensionalized axes. The abscissa is the squared *nondimensional rotational-speed parameter* (rotational speed over the first bending frequency of nonrotating beam) and the ordinate is the squared *nondimensional frequency parameter* (the n$^{th}$ bending frequency of the rotating beam over the n$^{th}$ bending frequency of nonrotating beam). Since the first bending frequency of a hinged beam is roughly four times that of a cantilever beam, the abscissa scales vary widely for hinged and cantilever beams. The abscissa scales also vary with tip mass because the fundamental frequency of a nonrotating beam decreases with increase in tip mass. This variation is larger for cantilever beams.

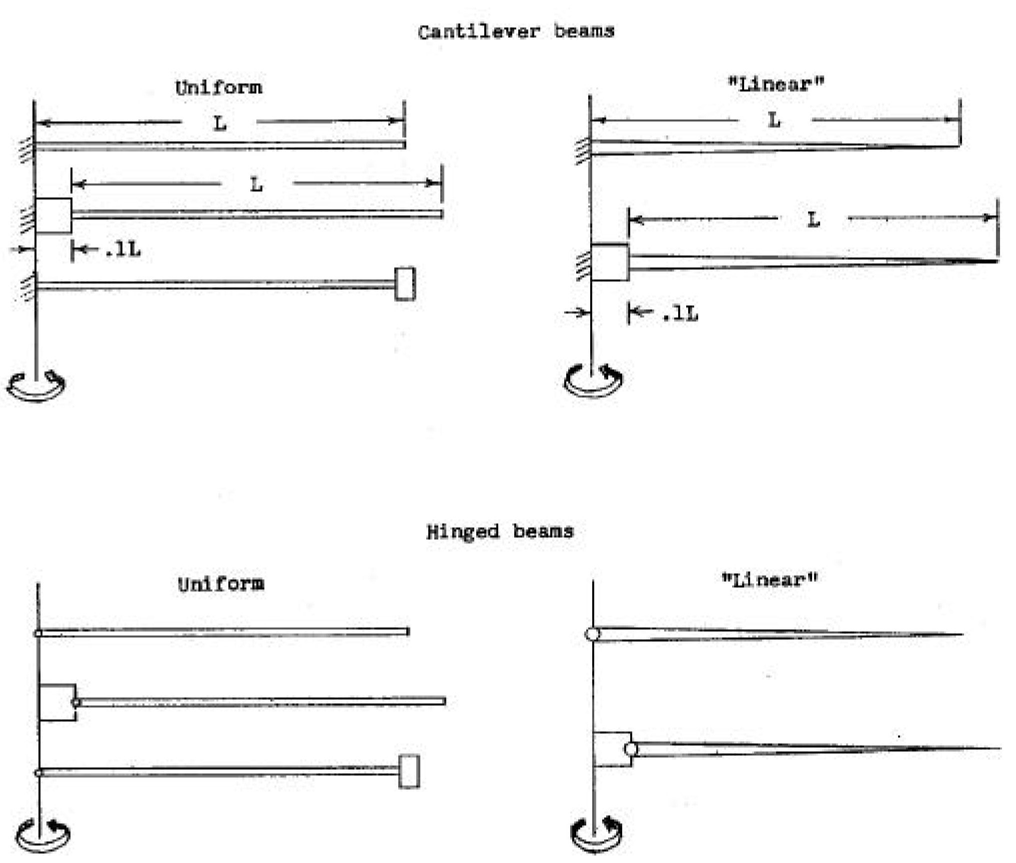



Figure 1. Beams Treated by both the “Exact” and Rayleigh methods. (From: Ref. [1])

### 1. Cantilever Beams Without Tip Mass

The variation of bending frequency for a uniform cantilever beam with root offsets of zero and ten percent is shown in Figure 2. The Rayleigh results are fairly accurate for higher modes, but the maximum error is large for the first mode, about ten percent in the frequency squared. Nevertheless, the effect of root offset is predicted very accurately for all three modes.

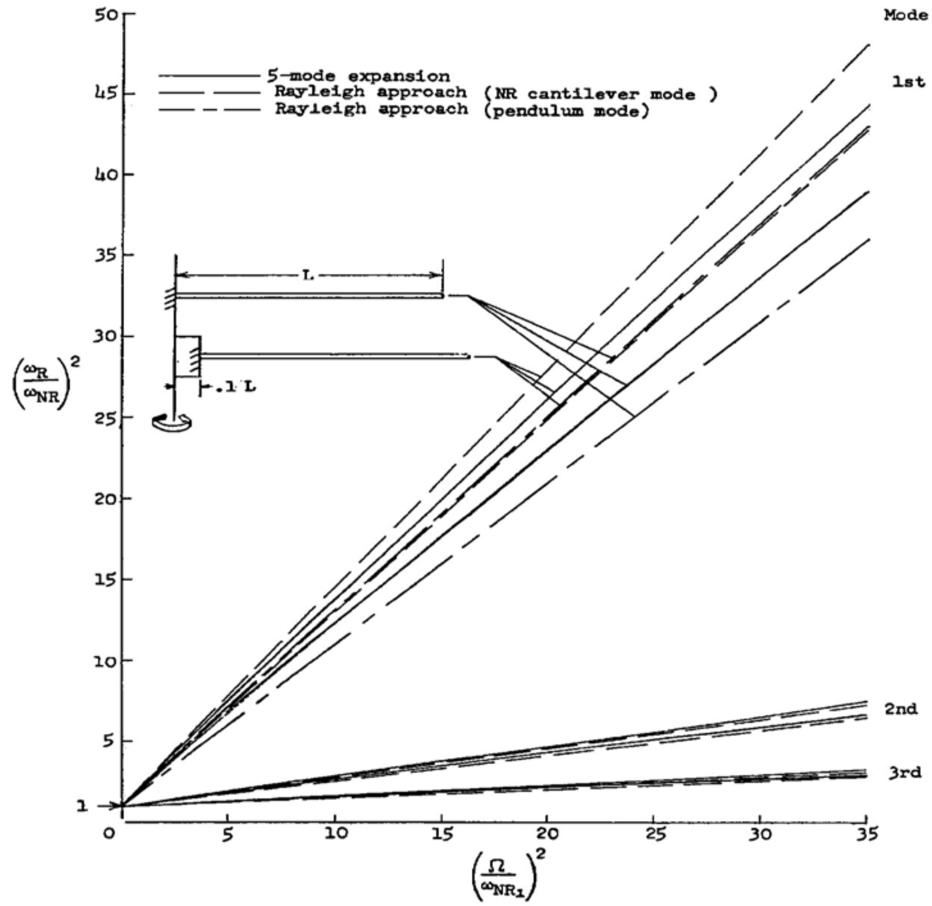



Figure 2. Effect of Rotational Speed on the Bending Frequencies of a Uniform Cantilever Beam. (From: Ref. [1])

For comparison, the results of approximating the first mode of a cantilever beam by the pendulum mode of a hinged beam are also plotted. For higher rotational speeds, the results of the pendulum mode become more and more accurate. But for the lower rotational speeds, the first mode shape of nonrotating beam yields the most accurate results. The root offset effects can be predicted by either the pendulum mode or the first cantilever mode.

In Figure 3, Yntema shows the variation of bending frequency for a cantilever beam with linear mass and stiffness distributions and with root offsets of zero and ten percent. As for the case of uniform cantilever beam, the second and third mode results are very accurate but the first mode result is not so accurate for higher rotational-speed values. Again, the pendulum mode of a hinged beam yields more accurate results for the first mode as the rotational-speed parameter increases. But for lower rotational-speed values, the first cantilever mode is more accurate than the pendulum hinged mode. The effect of root offset is predicted fairly accurately by both methods.

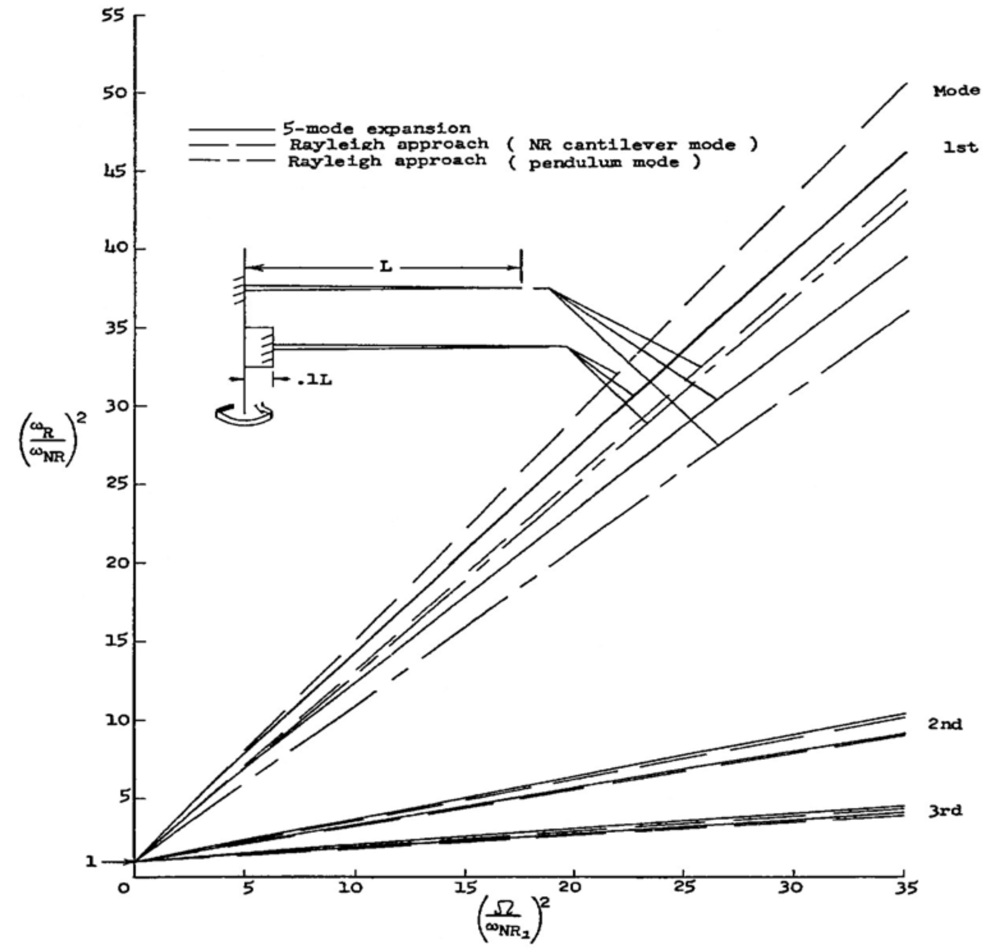



Figure 3. The Effect of Rotational Speed on the Bending Frequencies of a Cantilever Beam with Linear Mass and Stiffness Distributions. (From: Ref. [1])

In Figure 4, Yntema shows a comparison of the bending frequency results for the uniform and linear cantilever beams with zero root offset. The difference for the first mode, which is roughly predicted by the Rayleigh approach, could not be detected by the pendulum hinged mode. The effect of mass and stiffness distribution on higher modes is more visible and suggest that a single value of Southwell coefficient cannot accurately predict the bending frequency variations for beams with appreciably different mass and stiffness distributions.

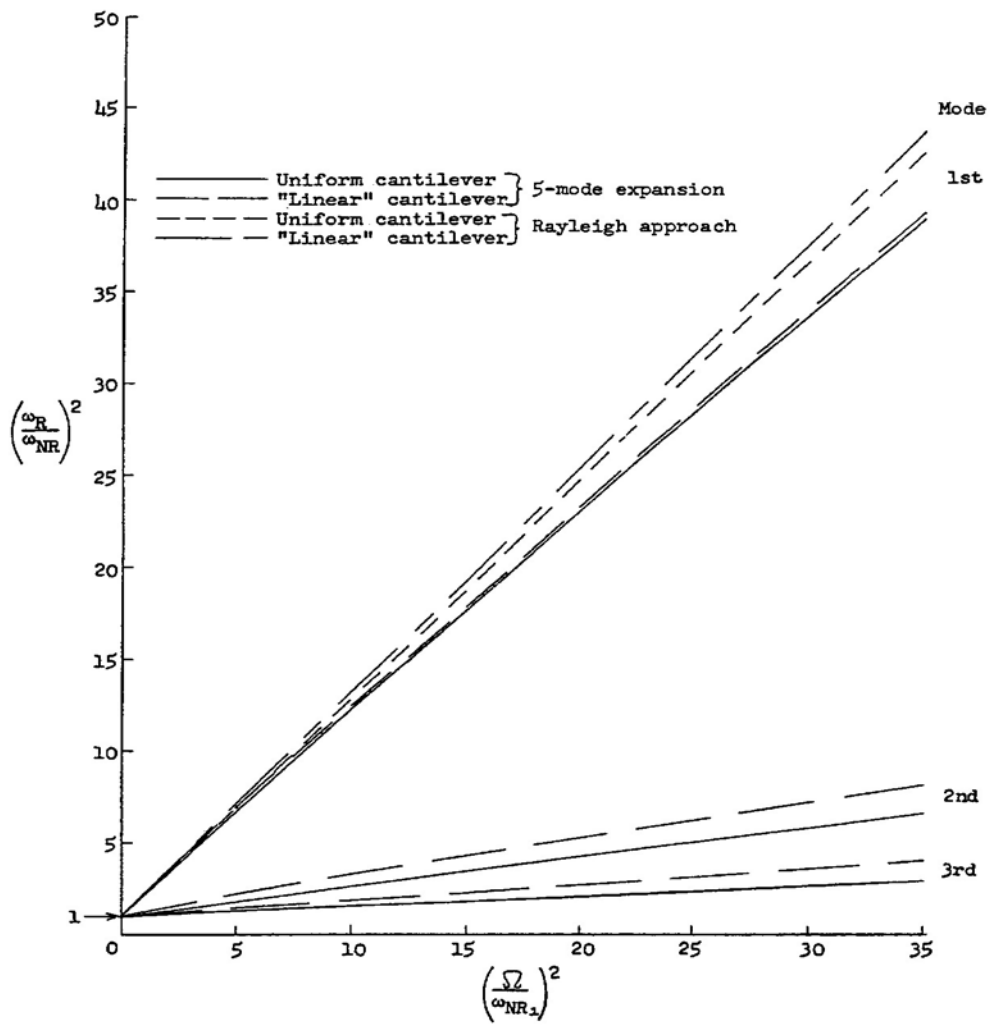



Figure 4. Comparison of Frequencies of Uniform and "Linear" Cantilever Beams with Zero Offset. (From: Ref. [1])

The error of Rayleigh approach in the first mode bending frequency results is almost the same for both uniform and linear beams. Thus, this error seems to be independent of beam mass and stiffness distribution.

### 2. Hinged Beams Without Tip Mass

In Figure 5, Yntema shows the variation of bending frequency for the case of a uniform hinged beam with root offsets of zero and ten percent. The maximum error is about three percent in the frequency squared and approximately the same for all three modes.

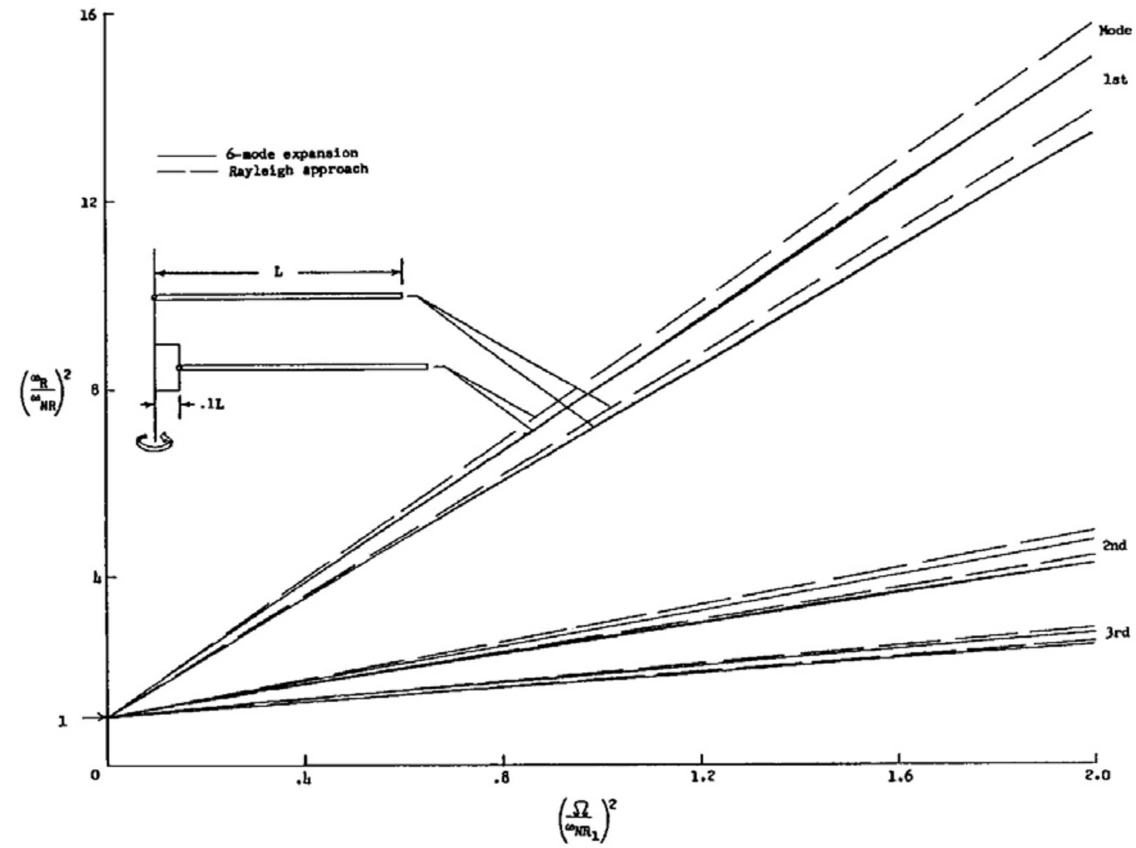



Figure 5. Effect of Rotational Speed on the Bending Frequencies of a Uniform Hinged Beam. (From: Ref. [1])

For the hinged beams with linear mass and stiffness distributions, the bending frequency results are very accurate. In addition, the exact frequency results show that the Southwell coefficient is not largely independent of beam mass and stiffness distributions but it varies appreciably with beam characteristics for all modes, especially for the first mode.

### 3. Cantilever Beams With Tip Mass

In Figure 6, Yntema shows the variation of bending frequency for a uniform cantilever beam with a concentrated tip mass and zero offset for the cases of $r$=1 and $r$=0.5 where $r$ is defined by

$$r = \frac{Tip\_Mass}{Blade\_Mass} = \frac{M_t}{m_0 L} \tag{15}$$

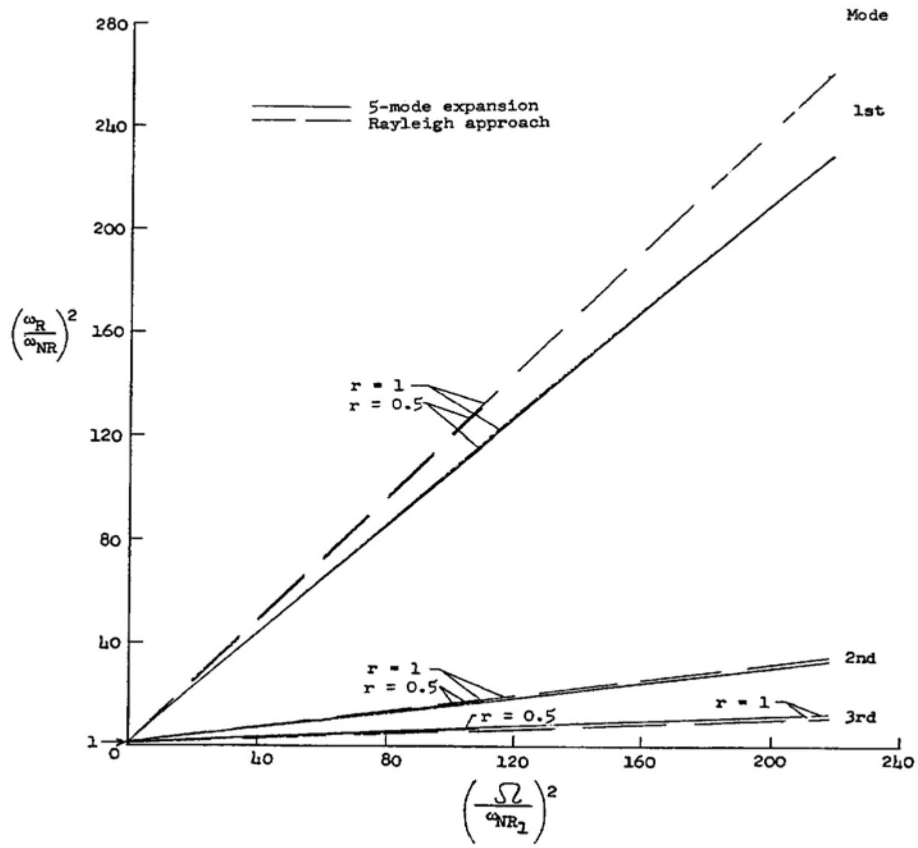



Figure 6. Bending Frequencies of a Uniform Cantilever Beam with a Mass at the Tip and with Zero Offset. (From: Ref. [1])

The error is of the same order of the error in the beam without tip mass: The Rayleigh results are very accurate for higher modes but relatively less accurate for the first mode. The errors for each mode are practically identical for all $r$ values. The impression that the zero-offset Southwell coefficient, $K_{0n}$, is independent of the tip mass value is true for the first mode but misleading for the higher modes where the term $(\omega_{NR1}/\omega_{NRn})^2$ in Equation(14) varies with tip mass.

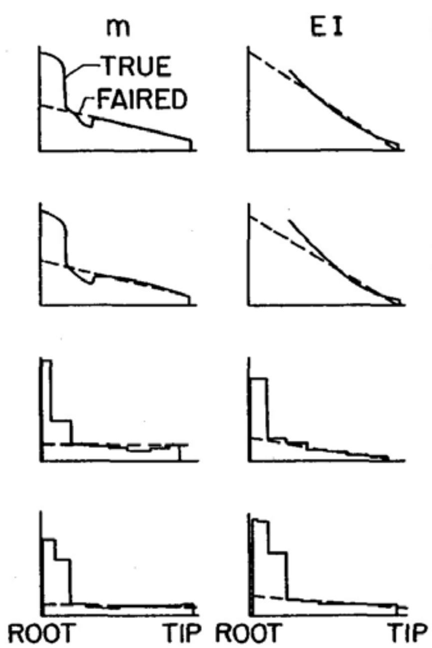
#### 4. Hinged Beams With Tip Mass

The Rayleigh results for the variation of bending frequency for a uniform hinged beam with a concentrated tip mass and zero root offset for the cases of $r$=1 and $r$=0.5, have been plotted and found to be accurate for all three modes over the entire range of variables.

The bending frequency variation of hinged beams are different for different values of tip mass, unlike the cantilever results of Figure 6 which is essentially independent of tip mass. If we plotted the results of the hinged cases as a function of the rotational-speed parameter used for the cantilever cases (the squared ratio of rotational speed to the bending frequency of the beam in the first cantilever mode), then the frequency variation would be seen to be essentially independent of tip mass, as has been noted for the cantilever. Therefore, a new constant, $K'_{0n}$, which is defined by Equation(14) and is insensitive to the beam mass distribution, appears to be justified.

## 5. Validation of Yntema's Work

Table 1 gives the exact and estimated bending frequencies for several manufactured blades. The results are very accurate as can be seen. The Southwell coefficients obtained by utilizing NR beam mode shapes yield reasonably accurate bending frequencies of rotating beams, at least for the range of the rotational-speed parameter encountered in helicopter blades. Error correction efforts may be incorporated for some beams. Southwell coefficients can vary appreciably with beam characteristics.




| MODE | $\omega_{NR}$, RADIANS/SEC EXACT | $\omega_{NR}$, RADIANS/SEC ESTIMATED | $\omega_R$, RADIANS/SEC EXACT | $\omega_R$, RADIANS/SEC ESTIMATED |
|---|---|---|---|---|
| 1st | 17.3 | 17.4 | 49.2 | 47.7 |
| 2nd | 48.5 | 50.0 | 86.8 | 85.7 |
| 3rd | 95.5 | 101.0 | 137.0 | 137.8 |
| 1st | 21.6 | 21.1 | 50.6 | 49.2 |
| 2nd | 58.9 | 60.5 | 92.4 | 92.2 |
| 3rd | 112.1 | 122.0 | 148.0 | 154.0 |
| 1st | 21.9 | 21.1 | 74.0 | 78.3 |
| 2nd | 63.7 | 59.5 | 132.0 | 134.4 |
| 3rd | 126.0 | 125.5 | 200.0 | 207.5 |
| 1st | 13.4 | 14.6 | 37.9 | 37.8 |
| 2nd | 43.7 | 41.6 | 71.0 | 70.3 |
| 3rd | 94.9 | 94.5 | 125.0 | 124.0 |

Table 1. Exact and Estimated Frequencies for Some Manufactured Blades (From: Ref. [1])

## C. YNTEMA'S CHARTS FOR RAPID ESTIMATION OF BENDING FREQUENCIES

A group of charts based on the Rayleigh approach permit the rapid estimation of bending frequencies of rotating and nonrotating beams. These charts give values for $a_n$ ($\theta_n$), $K_{0n}$, and $\bar{K}_{1n}$ (also for $K'_{0n}$) which are used for the determination of rotating beam bending frequencies. All these constants are based on the mode shapes of the nonrotating beam and obtained by standard numerical-iteration procedures.

### 1. Cantilever Or Hinged Beams Without Tip Mass

Equation(11) is the form of the Rayleigh energy equation, which is used to obtain bending frequencies. Equation(12) is also incorporated into the computations. The charts are presented for both hinged and cantilever beams, where the abscissa is the ratio of the beam mass per unit length at the tip to the mass per unit length at the root ($m_t/m_0$). The curves are provided for three different stiffness variations ($EI_t/EI_0$=0;0.5;1). Although each of these curves is faired through only three points, this procedure should involve little error because the variation is almost linear for the Southwell coefficients ($K_{0n}$ and $\bar{K}_{1n}$) and is not entirely arbitrary for the frequency coefficients ($a_n$), which is based on previous references.

The zero-offset Southwell coefficient, $K_{0n}$, for the pendulum mode is always unity regardless of the beam mass and stiffness distributions. $K_{00}$ for a hinged beam is dependent on the mass distribution but the stiffness distribution. There is no zero mode for a cantilever beam.

The $n^{th}$-mode Southwell coefficients for the cantilever beams should be compared with the $(n-1)^{th}$ mode of the hinged beam. From this comparison, the effects of root fixity on the Southwell coefficients are very small and can be neglected for rough approximation in all modes other than the first cantilever mode.

The variation of the Southwell coefficient is relatively insensitive to beam mass distribution. Coupled with the fact that the influence of the Southwell coefficient decreases for higher modes for constant rotational speed, fairly good approximations to the Southwell coefficients for beams with nonlinear stiffness distributions may also be obtainable from these charts.

### 2. Cantilever Or Hinged Beams With A Mass At The Tip

The beams studied for the case of a tip mass are uniform beams with no root offsets. Therefore the offset-correction factors for the Southwell coefficient, $K_{1n}$, are not provided.

#### *a. Uniform Cantilever Beam*

The defining relation for the bending frequencies is given by

$$1+\cos\theta\cosh\theta - r\theta(\sin\theta\cosh\theta - \cos\theta\sinh\theta) = 0 \tag{16}$$

where

$$\omega_{NR_n} = \theta_n^2\sqrt{\frac{EI}{mL^4}} \tag{17}$$

and the mode shapes are

$$y_n(x) = \sinh x - \sin x + (\cos x - \cosh x)\frac{\sinh\theta_n + \sin\theta_n}{\cosh\theta_n + \cos\theta_n} \tag{18}$$

Values of $\theta_n^2$ rather than $\theta_n$ are plotted in the charts because $\theta_n^2$ corresponds to $a_n$ presented previously.

For large values of $r$, $\theta_n^2$ can be fairly accurately computed as:

$$\theta_n^2 \approx \left(\theta_n^2\right)_{r=0}\sqrt{\frac{1}{1+\kappa_n r}} \tag{19}$$

where the constant, $\kappa_n$, can be determined from the frequency results for the largest value of $r$. Equation(19) can also be used for hinged beam as well as any non-uniform beam. Although the $K_{0n}$-curves are faired through three points, they must be fairly accurate because the variation is almost linear.

### b. *Uniform Hinged Beam*

The defining relation for the bending frequency is given by

$$2r\theta + \coth\theta - \cot\theta = 0 \tag{20}$$

and the mode shapes are given by:

$$y_n(x) = \sinh x + \frac{\sinh\theta_n}{\sin\theta_n}\sin x \tag{21}$$

The values of $\theta_n^2$ as well as the zero-offset Southwell coefficients, $K_{0n}$, have been plotted to permit a rapid estimation of bending frequency. For the pendulum mode, $K_{00}$ is always unity.

## 3. Rotating Beams With Nonlinear Mass Distribution And Approximately Linear Stiffness Distribution

A modified form of zero-offset Southwell coefficient, $K'_{0n}$, has been shown to be insensitive to the variations in beam tip mass and is given by Equation(14) for both cantilever and hinged beams. For any stiffness distribution, $K'_{0n}$ is almost constant for each mode, the differences being the same order of magnitude as the errors inherent in the Rayleigh approach. Curves for cantilever and hinged beams have been faired and plotted to give average values for $K'_{0n}$ and thus for $K_{0n}$ for beams with near linear stiffness distributions and with any mass distribution. The first bending frequency of nonrotating beam cantilevered at the root and the n$^{th}$ bending frequency of the nonrotating beam with its actual end fixity (hinged or cantilever) are required to determine $K_{0n}$ from a knowledge of $K'_{0n}$. Although care must be taken using $K'_{0n}$ because it is only demonstrated for a limited number of mass distributions, these charts should be useful in design studies involving rotating beams with nonlinear mass distributions but with approximately linear stiffness distributions.

## 4. Rotating Beams With Mass And Stiffness Distributions Not Representable By Foregoing Approximations

The basic Rayleigh energy method given by Equation(1) can be used for nonlinear variations. All that is required is the frequency and the mode of the nonrotating beam. Also providing the first derivative of the mode is preferable. With these inputs, the integrals of Equation(1) can be computed by numerical methods to give the bending frequencies at any rotational speed.

## 5. Mode-Expansion Method

A more accurate but complex mode-expansion method for determining the bending frequencies and the modes of a rotating or nonrotating beam has been developed to be used in the evaluation of the Rayleigh approach. The method solves a fifth-order determinantal equation for cantilever beams and a sixth-order equation for hinged beams in order to obtain the lowest three bending modes and frequencies.

## 6. Vibration In Planes Other Than Those Perpendicular To Plane Of Rotation

The bending frequency results obtained by Yntema are all for uncoupled bending vibrations perpendicular to the plane of rotation. When the principal axis of the blade airfoils (axis around which the stiffness is a minimum) is not parallel to the plane of rotation, natural bending vibrations having the lowest frequencies will occur perpendicular to the chord.

Vibration frequencies for a blade chord inclined at any angle $\psi$ with the plane of rotation, can be determined from the simple formula, for n$^{th}$ mode:

$$\left(\omega_{R_\psi}^2\right)_n = \left(\omega_{R_L}^2\right)_n - \Omega^2 \sin^2 \psi \tag{22}$$

where $\omega_{R\psi}$ is the frequency of bending vibrations in a plane making an angle $\boldsymbol{\psi}$ with the axis of rotation and $\omega_{RL}$ is the frequency of bending vibrations perpendicular to the plane of rotation.

At large angles of attack, this correction may be significant for the lower modes. But because $\omega_{RL}^2$ is usually five to ten times as large as $\psi^2$, in most cases, the angle of attack of the blade will not largely affect the bending frequency. This indicates that cyclic-pitch changes will not affect the bending frequency appreciably, and therefore the blade frequency may be assumed to be constant.

## D. RESULTS FOR BENDING MODE SHAPES

To determine the bending frequency results, a large number of rotating and nonrotating beam mode shapes have been computed and presented in tabular form for use in analytical and numerical studies.

### 1. Nonrotating Beams

The first three mode shapes, along with their first and second derivatives, for nine cantilever and nine hinged beams with different mass and stiffness distributions are provided in tabular form. These results were obtained by standard numerical-iteration procedures. Ten stations were employed for cantilever beams, and 15 stations are employed for hinged beams because they have one more node than cantilever for the third mode.

### 2. Rotating Beams

The mode shapes, as well as frequencies, have been obtained by the derived *mode-expansion* method explained in previous paragraphs. This method yields mode coefficients which, in turn, give the mode shapes of the rotating beam. These mode coefficients have been normalized to positive tip deflections and provided in tabular form.

### 3. Comparison Of Rotating And Nonrotating Beams

#### *a. Hinged Beams*

For uniform hinged beams, the differences between the rotating and nonrotating mode shapes are relatively small, particularly for the higher modes. For "linear" hinged beams, the difference in mode shapes is very small because the Rayleigh approach has been shown to be very accurate for this case. The mode shapes of uniform and "linear" hinged beams are very different because of the substantial difference between the Southwell coefficients for the two beams. The effect of root offset may be seen to be small for both hinged beams.

#### *b. Cantilever Beams*

For both the uniform and "linear" cantilever beams, the difference in mode shapes changes appreciably with rotational speed, especially for the first and second modes. The effect of root offset is seen to be very small for both uniform and "linear" cantilever beams.

#### *c. Beams With A Mass At The Tip*

For uniform hinged beam with a tip mass equal to the beam mass, the mode shape differences are very small for all modes. This result accounts for the excellent accuracy of Rayleigh approach for this case. For uniform cantilever beam with a tip mass equal to the beam mass, the mode shape differences between rotating beams are slightly different from each other for different values of rotational speed. But the rotating beam mode shape is considerably different from the nonrotating beam mode shape.

## E. SUMMARY REMARKS CONCERNING YNTEMA'S WORK

The Rayleigh approach yields reasonably accurate bending frequencies for rotating hinged or cantilever beams with mass and stiffness distributions which can be approximated by linear relations, with concentrated tip masses for uniform mass and stiffness distributions, at least within the rotational speed limits encountered by helicopter blades. The evaluation also demonstrated that the Southwell coefficients vary appreciably with beam mass distribution and, to a lesser extent, with beam stiffness distribution. Several groups of charts are presented for rapid estimation of the first three bending frequencies of a variety of hinged and cantilever beams vibrating in a plane perpendicular to the plane of rotation of the blade. Bending mode shapes have also been determined for some hinged and cantilever beams, which show the effects of various parameters on the mode shape.

Yntema's method is valuable for application in the following cases:

- To provide the first estimate in more exact detailed analysis methods such as Myklestad, matrix iteration and Rayleigh-Ritz procedure.
- To provide natural frequencies of rotor blades and beams during preliminary design.
- To provide natural frequencies in parametric design studies.
- To provide natural frequencies wherever a rapid, efficient, but approximate method is adequate.
- To provide natural frequencies in combination with Young-Felgar mode shapes in more detailed analysis.

# III. MODELING THE YNTEMA METHOD IN MATLAB®

This chapter describes a computer modeling method to the Yntema method. The statements below are based on the previous chapter and illustrate the planned methodology. MATLAB® computer programming language developed by MathWorks, Inc., has been chosen for the analysis. The computer code includes the parts of the Yntema report that utilizes the Rayleigh approach to determine the first three bending frequencies of rotating or nonrotating beams. The limitations of the computer code is the same as the limitations of the Rayleigh approach demonstrated by Yntema.

## A. DESCRIPTION

The nonrotating beam mode shapes are used to approximate the bending frequencies of rotating beams. The nonrotating beam frequency coefficients ($a_n$), zero-offset Southwell coefficients ($K_{0n}$), and offset-correction factors for Southwell coefficients ($\bar{K}_{1n}$) were manually read from the Yntema report and stored in matrix format. The values for stiffness ratios other than 0, 0.5, and 1 are faired through a second order polynomial. If the mode shape of the rotating beam is known, the exact Rayleigh equation is provided as another MATLAB® function to determine the exact bending frequencies of the rotating beam.

The limitations of Yntema report are assumed, i.e. the mass ratio and the stiffness ratio must be between zero and one. The value of the concentrated tip mass must be less than or equal to the beam mass. For consistency, the units for rotational speed are given in RPM (revolutions per minute), the resulting bending frequencies are given in CPM (cycles per minute). Southwell plots are provided as an option in the function call. Tools are provided to approximate a mass or stiffness distribution as linear.

## B. EVALUATION

The Yntema report is assumed to be accurate enough for most of the cases considered. Error corrections are incorporated only for the cases of:

- All of the first three modes of the uniform hinged beam with a root offset of 0~10 percent for all rotational-speed parameter values. For this case, the maximum error is 1.5% in all modes, for the frequency value.
- The first mode of the uniform cantilever beam with a root offset between zero and ten percent (excluding zero) for the rotational-speed parameter values greater than two. For this case, the maximum error in the first mode is between 4.8% and 4.3% for root offsets between 0% and 5% respectively at the rotational-speed parameter value of two, for the estimated frequency.
- The first mode of the "linear" cantilever beam with a root offset between zero and ten percent (excluding zero) for the rotational-speed parameter values greater than four. For this case, the maximum error in the first mode is between 5.5% and 4.7% for root offsets between 0% and 5% respectively at the rotational-speed parameter value of four, for the estimated frequency.
- The first mode of the cantilever beam with any mass and stiffness distribution and zero root offset for the rotational-speed parameter values greater than two. For this case, the maximum error is 5% in the first mode at the rotational-speed parameter value of two, for the estimated frequency.
- The first mode of the uniform cantilever beams with a concentrated tip mass and zero root offset for *r* values between zero and one and the rotational-speed parameter values greater than two. For this case, the maximum error is 6.5% in the first mode at the rotational-speed parameter value of five, for the estimated frequency.

The corrections are made with the help of Figure 7. The difference (***e***) is given by

$$\begin{aligned} &\left(\frac{w_{R_n}}{w_{NR_n}}\right)^2_{Mode\_Expansion} = \left(\frac{w_{R_n}}{w_{NR_n}}\right)^2_{Rayleigh} - e \\ &\Rightarrow \left(w_{R_n}^2\right)_{Mode\_Expansion} = \left(w_{R_n}^2\right)_{Rayleigh} - e \cdot w_{NR_n}^2 \end{aligned} \tag{23}$$

In Figure 7, ***e+f*** term is the Rayleigh approach result whereas ***f*** is the more exact result for any value of rotational-speed parameter. ***d+c*** term is the referenced Rayleigh approach result whereas '**c**' is the referenced exact result for the value of rotational-speed parameter shown on the corresponding Yntema figures. The error is zero at ***h*** and has its maximum value at ***g***. ***E*** and ***F*** just represent the results corresponding to the rotational-speed parameters greater than the reference rotational-speed. For example, if the maximum error is 10%, then ***d/c*** is equal to 0.10 and ***d+c*** is equal to $1+K_n*(\omega_{NR1}/\omega_{NRn})^2*g$. From this, first ***d*** and then ***e*** can be computed easily.

$$e = \frac{b}{g-h} d = \frac{\left(\dfrac{\Omega}{\omega_{NR_1}}\right)^2_{actual} - h}{g-h} d \tag{24}$$

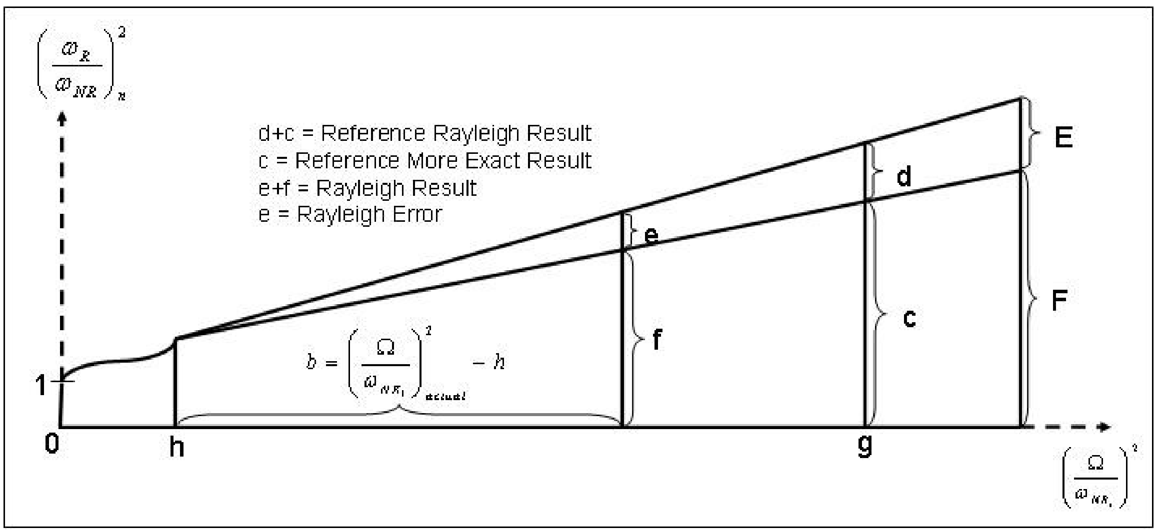



Figure 7. A Geometric Plot for Error Computation

For an example for the uniform hinged beam with a root offset of 0~10 percent, Figure 8 shows the error correction (3% in the frequency squared) for the Rayleigh results for all three modes. For this case, ***g***=2, ***h***=0, ***d/c***=0.03, and ***d+c***=1+2*$K_1$. The plots on the left show the results before the correction and the plots on the right show the results after correction.

For the first mode of the uniform cantilever beams with a root offset between zero and ten (excluding zero), Figure 9 shows the error correction (8.5~9.6% in the frequency squared) for the Rayleigh results. For this case, ***g***=35, ***h***=2, ***d/c***=0.085~0.096, and ***d+c***=1+35*$K_1$.

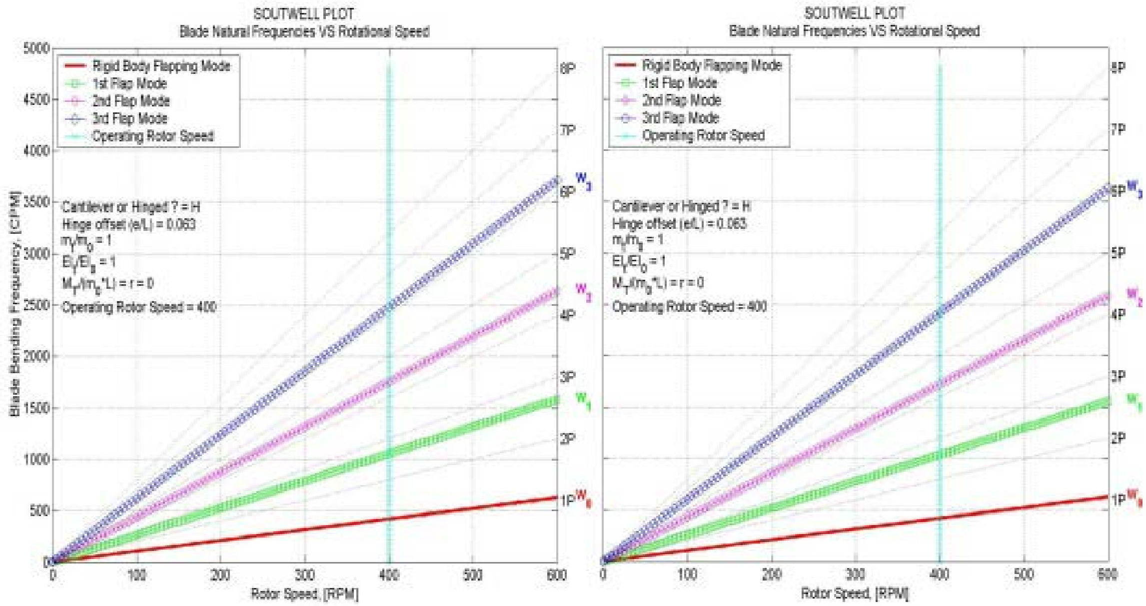



Figure 8. Example Error Correction for Uniform Hinged Beam with a Root Offset between Zero and Ten Percent.

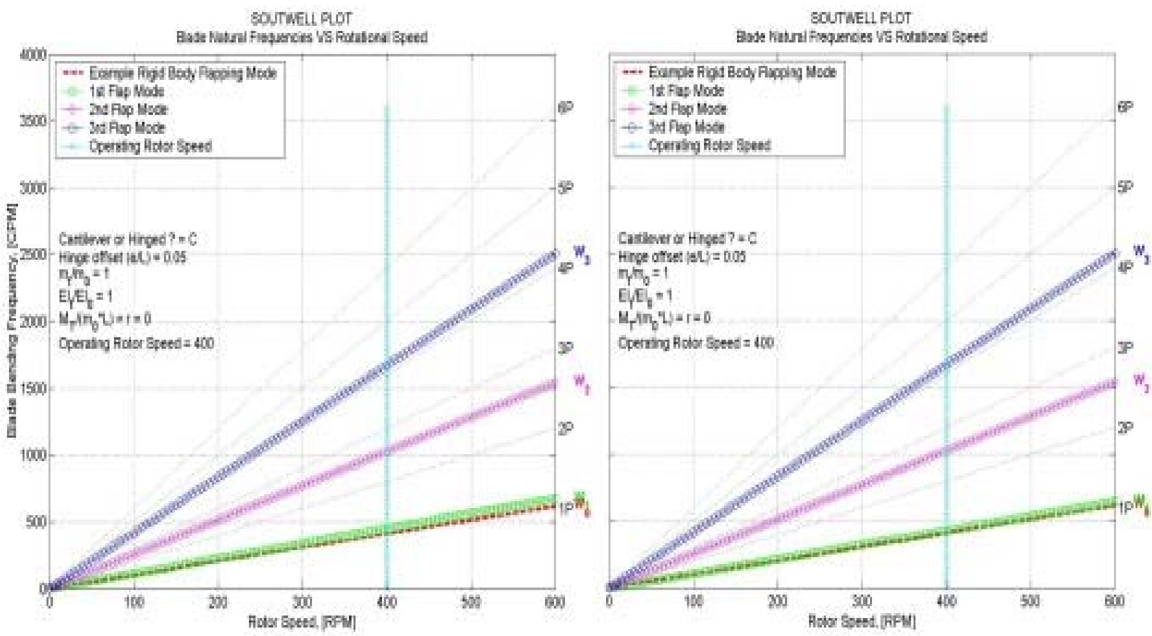



Figure 9. Example Error Correction for Uniform Cantilever Beams with a Root Offset between Zero and Ten Percent (Excluding Zero).

For the first mode of the "linear" cantilever beams with a root offset between zero and ten (excluding zero), Figure 10 shows the error correction (9.4~11% in the frequency squared) results. For this case, ***g***=35, ***h***=4, ***d/c***=0.094~0.11, and ***d+c***=1+35*$K_1$.

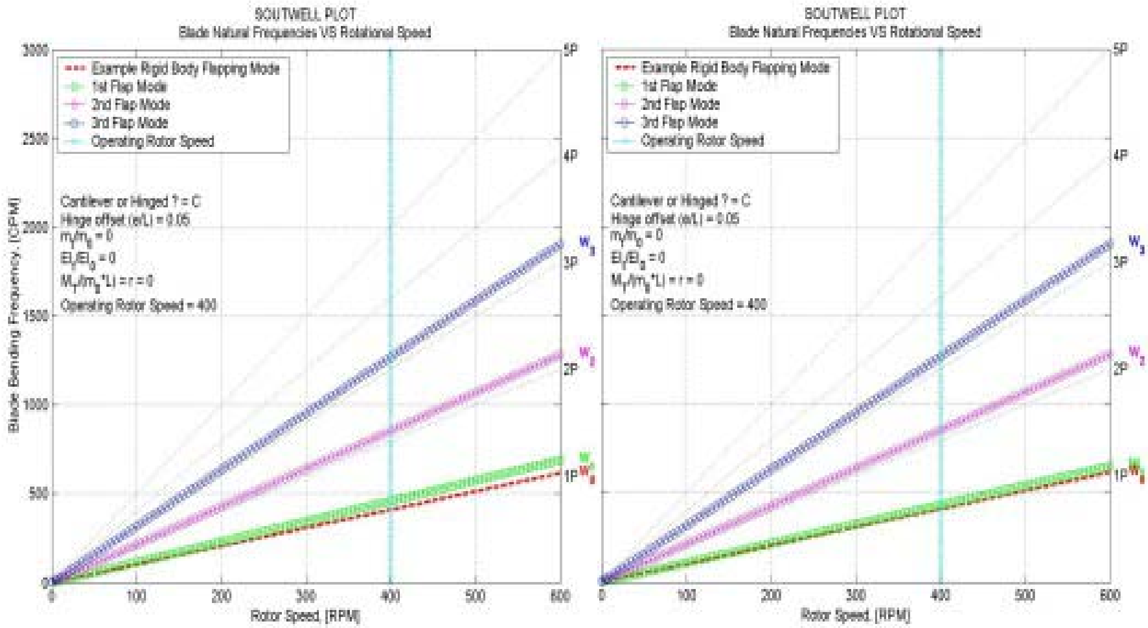



Figure 10. Example Error Correction for the First Mode of "Linear" Cantilever Beams with a Root Offset between Zero or Ten Percent (Excluding Zero).

For the first mode of any cantilever beam with a concentrated tip mass and zero root offset for all values of **r** between zero and one, Figure 11 shows the error correction (10% in the frequency squared) results. For this case, ***g***=35, ***h***=2, ***d/c***=0.10, and ***d+c***=1+35*$K_1$.

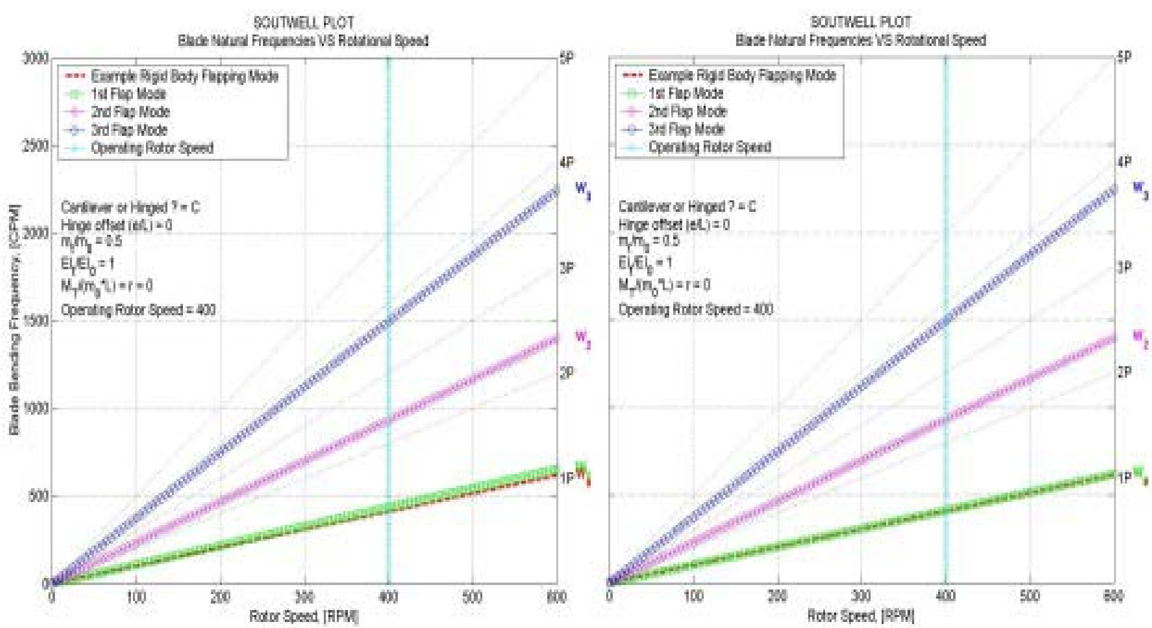



Figure 11. Error Correction for the First Mode of Any Cantilever Beam with a Concentrated Tip Mass and Zero Root Offset.

For the first mode of the uniform cantilever beams with a concentrated tip mass and zero root offset for all values of **r** between zero and one, Figure 12 shows the error correction (13% in the frequency squared) results. For this case, ***g***=200, ***h***=5, ***d/c***=0.13, and ***d+c***=1+200*$K_1$.

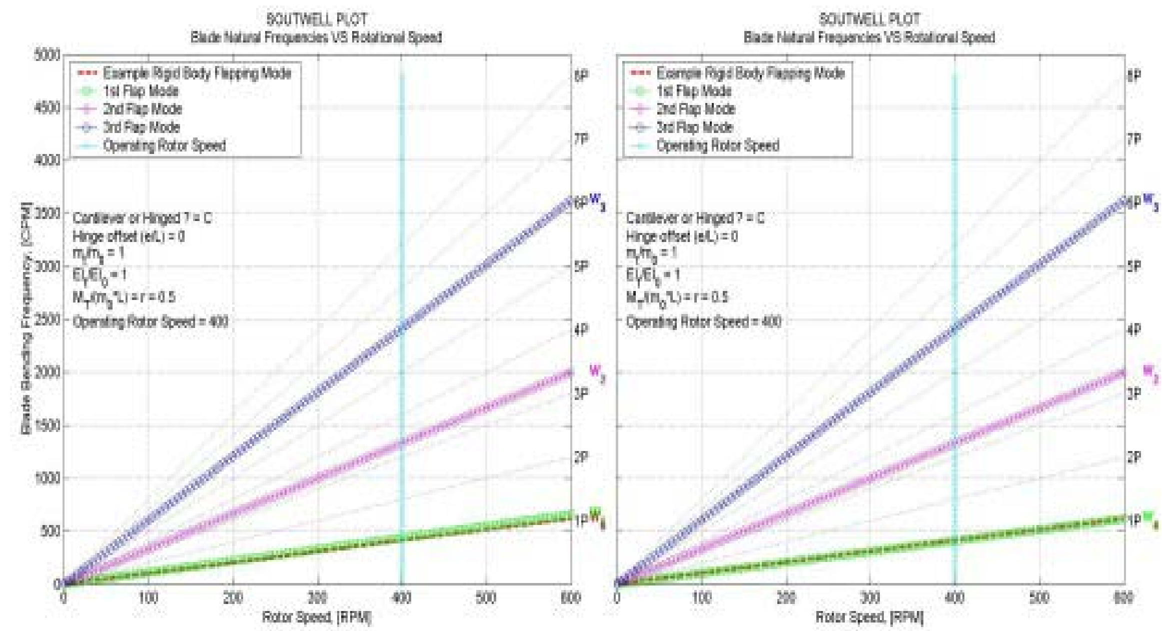



Figure 12. Error Correction for the First Mode of Uniform Cantilever Beams with a Concentrated Tip Mass and Zero Root Offset.

It should be noted that a beam having a mass ratio of greater than 99.5% and a stiffness ratio of greater than 99%, is treated as a uniform beam in the error correction computations. Similarly, a beam having a mass ratio of less than 0.5% and a stiffness ratio of less than 1%, is treated as a "linear" beam. Also, if the root offset is less than 0.1%, it is treated as zero offset. These adjustments have been incorporated for some flexibility in the numerical computations. Because there is limited data in the Yntema report, further adjustments can not be justified.

## C MATLAB® CODES AND GUI FOR RAPID ESTIMATION OF BENDING FREQUENCIES

The curves on the Yntema charts have been measured manually and put into a MATLAB® script file called YNTECOEF.M. This file produces the polynomial coefficients for the Yntema curves. These polynomial coefficients saved in a file called

YNTECOEF.MAT, will be used by the main MATLAB® function file, YNTEMA, to return the bending frequency and Southwell coefficients for a particular beam. YNTECOEF.M script file is listed in Appendix B.

For rotating hinged or cantilever beams with linear mass and stiffness distributions, a main MATLAB® function, YNTEMA, has been created to give the bending frequency results. The YNTEMA function accepts the characteristic values of a beam and then computes its natural bending frequencies by means of the nonrotating beam frequency and Southwell coefficients read from YNTECOEF.MAT file. In case of missing or erroneous inputs, the default values are acquired from the YNTEDFLT.MAT file. Error corrections mentioned in the previous section are incorporated into the computations. The angle of attack correction mechanism of the YNTEMA function may be enabled if an angle of attack value is included in the function call. YNTEMA function returns the values of the blade natural bending frequencies corresponding to each and every value of the rotational speed vector which is specified in the function call. The first element of the resulting bending frequency vector is for the operating rotor speed value. Along with the bending frequencies, YNTEMA function returns the nonrotating beam bending frequency and Southwell coefficients read from YNTECOEF.MAT. If an angle of attack value is specified as input argument, the output arguments include the corrected bending frequency values along with the uncorrected bending frequency values. The Southwell plot corresponding to the input values can be output if needed. A Southwell plot shows the change of the frequency values with respect to the rotational speed. The effect of angle of attack can be demonstrated on a modified kind of Southwell plot which shows the corrected bending frequency values as well as the uncorrected frequency values. A third kind of plot is a 3-dimensional mesh plot which demonstrates the change in bending frequency for a range of rotational speed and a range of angle of attack values. YNTEMA function file is listed in Appendix C and YNTEDFLT script file is listed in Appendix B.

For rotating beams with mass and stiffness distributions not representable by linear relations, another MATLAB® function, YNTERYGH, has been created for beams with any mass and stiffness distributions. This function file computes the basic Rayleigh energy equation, Equation(1), and therefore essentially requires a mode shape result to determine its bending frequency. The function accepts only one mode shape and returns the corresponding bending frequency value along with the nonrotating beam bending frequency and Southwell coefficients computed from the integrations of Equation(1). YNTERYGH function file is listed in Appendix C.

A GUI function file, YNTEMAGUI, has been created for ease of use and flexibility. The GUI function accepts the same inputs and returns the same outputs as the YNTEMA function. YNTEMAGUI function internally calls the YNTEMAG function which is essentially identical to YNTEMA function. The advantage of the GUI is that it is not needed to write all the inputs again if an input value is needed to be changed and that the outputs are altogether displayed on a single window. YNTEMAGUI function file is listed in Appendix D.

There are three utility functions for use in conjunction with the YNTEMA and YNTERYGH functions. The first utility function, MAKELINE, is used to linearize a nonlinear distribution along the blade. The second, LINKMASS function, is used to redistribute the mass distribution vector into a larger vector filling the middle values with zero. The third function, LINKSPOR, redistributes the stiffness distribution vector into a larger vector interpolating the middle values and also returns the order of the polynomial which can be used to fit a curve for the stiffness distribution. MAKELINE, LINKMASS, and LINKSPOR utility functions are listed in Appendix C.

### D. MATLAB® CODE FOR BENDING MODE SHAPES

The first three bending mode shapes, along with their first and second derivatives, for the cantilever and hinged beams provided by Yntema have been tabulated into a MATLAB® script file, YNTEMOSHCOEF.M, for nonrotating hinged or cantilever beams. This file produces the polynomial coefficients of the curves fitted for the deflection and derivative values of the nonrotating beams provided Yntema. Then the polynomial coefficients are saved in the file YNTEMOSHCOEF.MAT to be referred by the MATLAB® function, YNTEMOSHNR. YNTEMOSHNR function takes the mass and stiffness ratios of a hinged or cantilever beam as input arguments, reads the polynomial coefficients from the YNTEMOSHCOEF.MAT file, and interpolates a curve for the mode shape result. The mode shapes can be shown on a single plot or three subplots if needed. YNTEMOSHCOEF.M script file is listed in Appendix B and YNTEMOSHNR function file is listed in Appendix C.

## E. VERIFICATION AND VALIDATION OF THE SOFTWARE

This section provides the verification and validation of the MATLAB® code and GUI generated. All of the codes have been verified for the syntax errors. The curves fitted to the Yntema figures have been validated by visual inspection and manual measurements which are the most suitable procedures for this kind of validation. The function and utility files have been verified and validated by numerous procedures for both usability and their results. The GUI has been verified and validated in a more direct fashion.

### 1. Verification And Validation Of The Fitted Curves

The MATLAB® script files have been debugged to be free of syntax errors.

#### *a. Validation Of The MATLAB® Script File For Calculating Southwell And Nonrotating Beam Bending Frequency Coefficients*

Figures 13 through 23 show the reproduced Yntema figures by the YNTECOEF.M script file. Visual inspection and manual measurements demonstrate that the reproduced figures are very accurate so much that they can be used in place of the Yntema figures. Therefore, the polynomial coefficients produced by YNTECOEF.M and saved in the YNTECOEF.MAT file are accurate enough for use with any function.

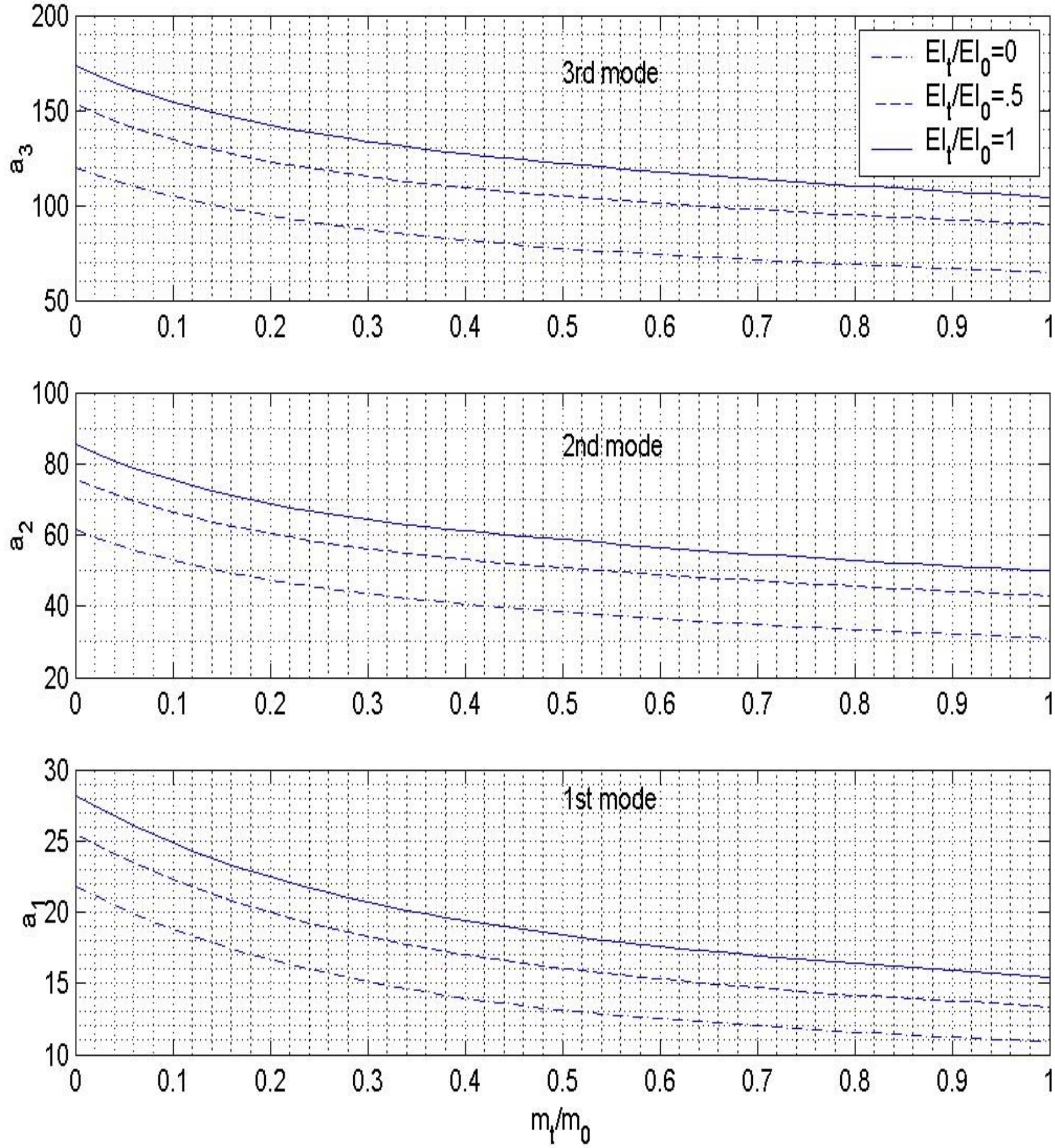



Figure 13. Bending Frequency Coefficients $a_n$ for Hinged Beams with Linear Mass and Stiffness Distributions. (After: Ref. [1])

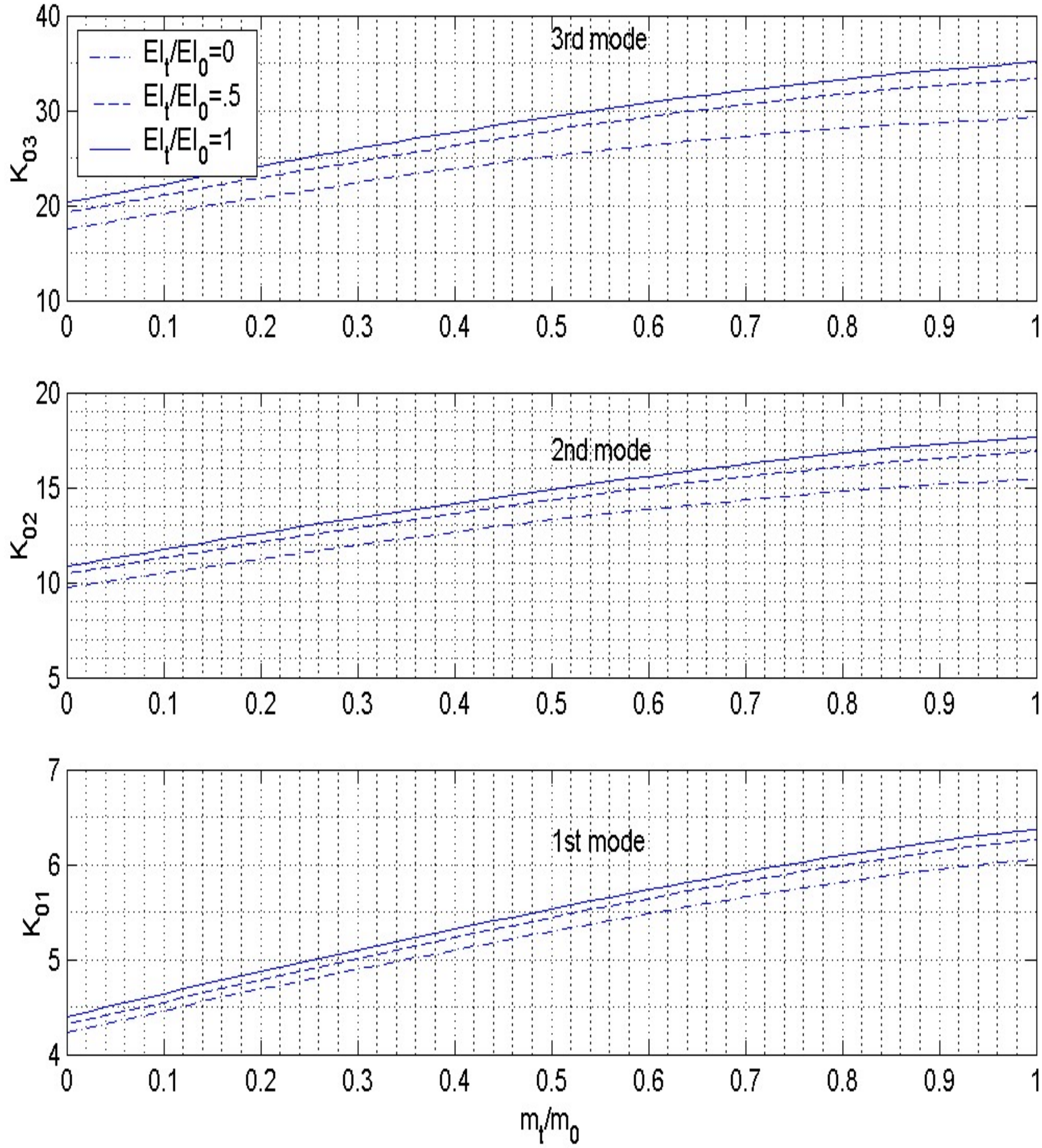



Figure 14. Zero-offset Southwell Coefficient $K_{0n}$ for Hinged Beams with Linear Mass and Stiffness Distributions. (After: Ref. [1])

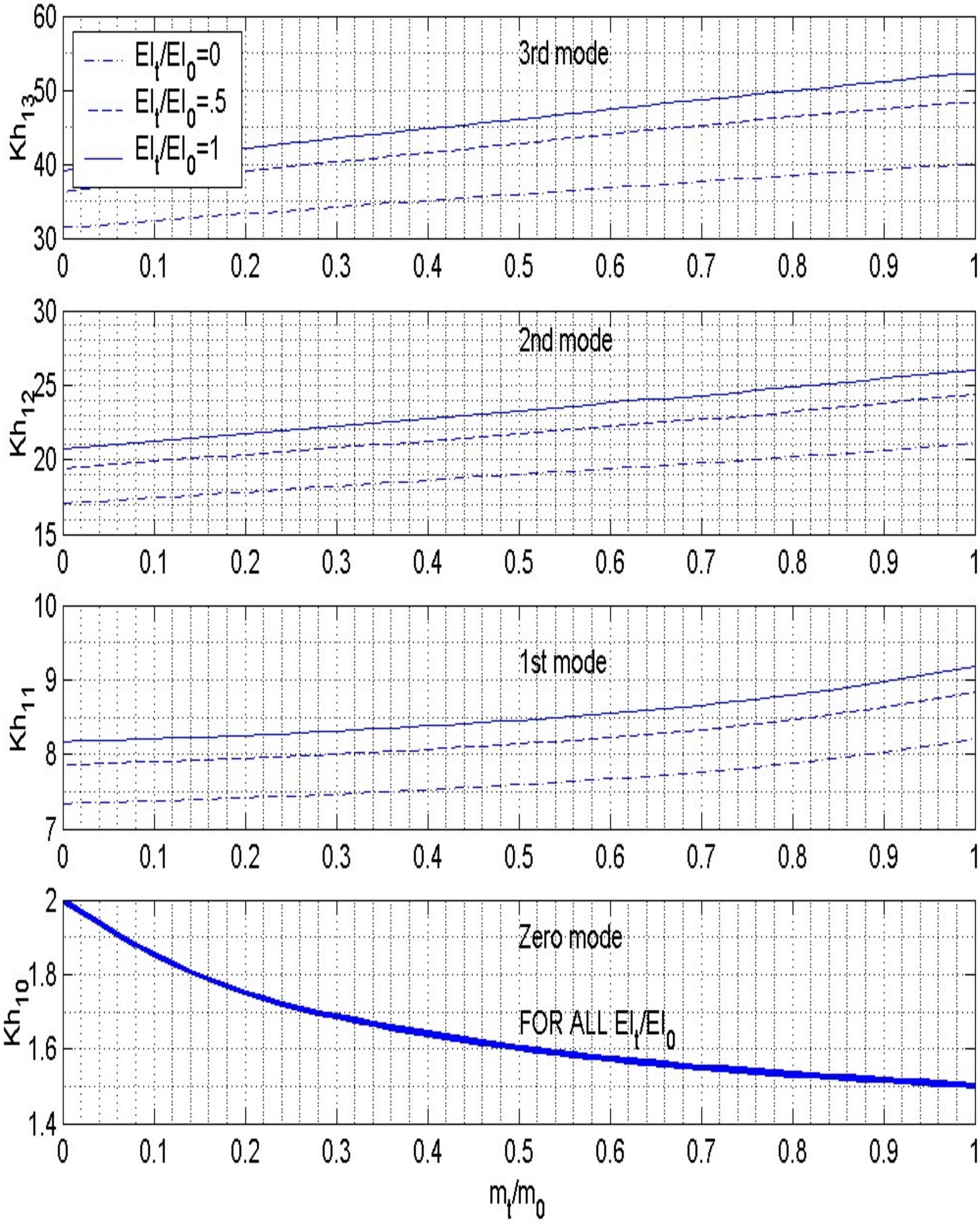



Figure 15. Offset-correction Factors for Southwell Coefficient $\overline{K}_{1n}$ for Hinged Beams with Linear Mass and Stiffness Distributions. (After: Ref. [1])

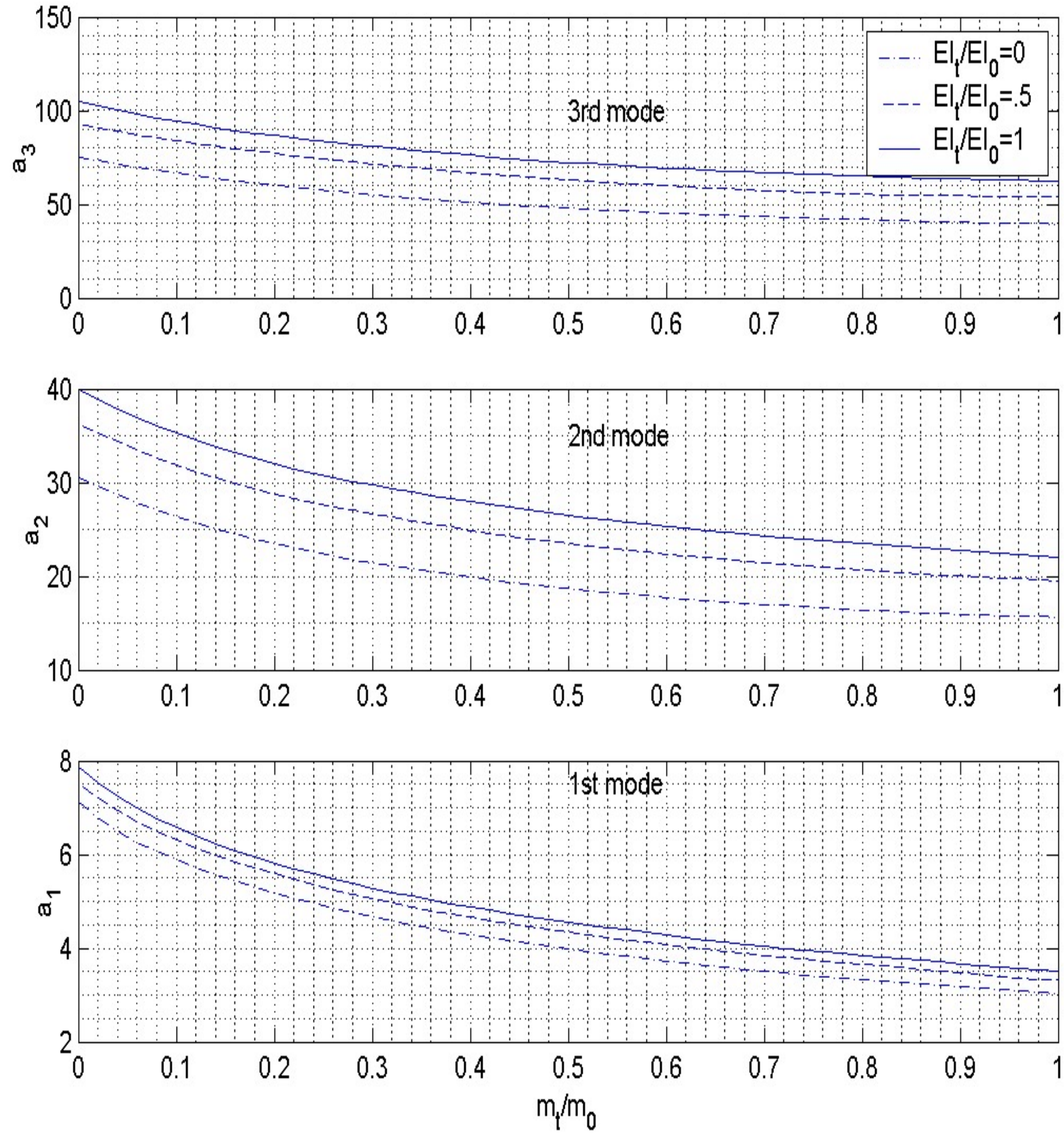



Figure 16. Bending Frequency Coefficients $a_n$ for Cantilever Beams with Linear Mass and Stiffness Distributions. (After: Ref. [1])

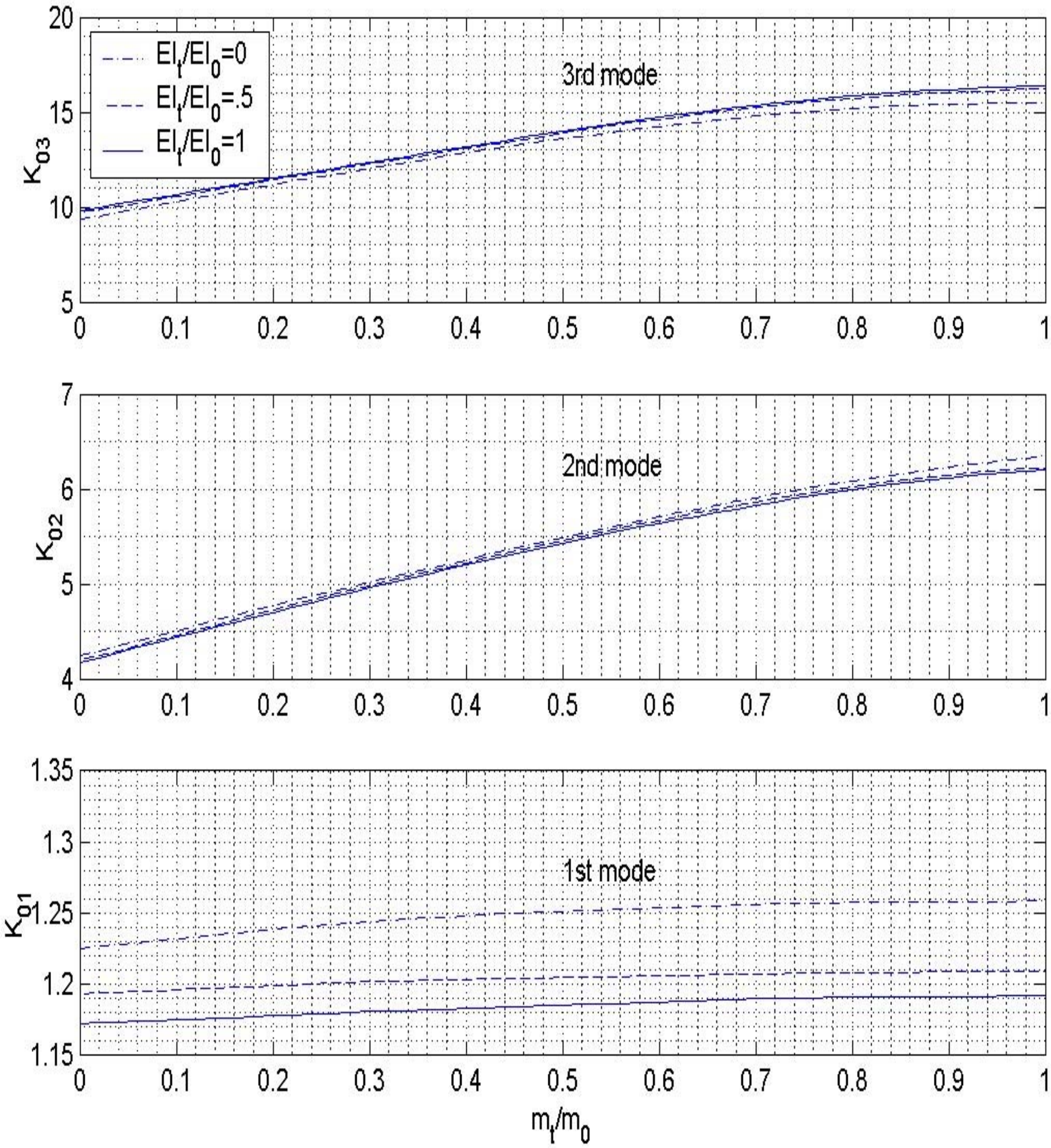



Figure 17. Zero-offset Southwell Coefficient $K_{0n}$ for Cantilever Beams with Linear Mass and Stiffness Distributions. (After: Ref. [1])

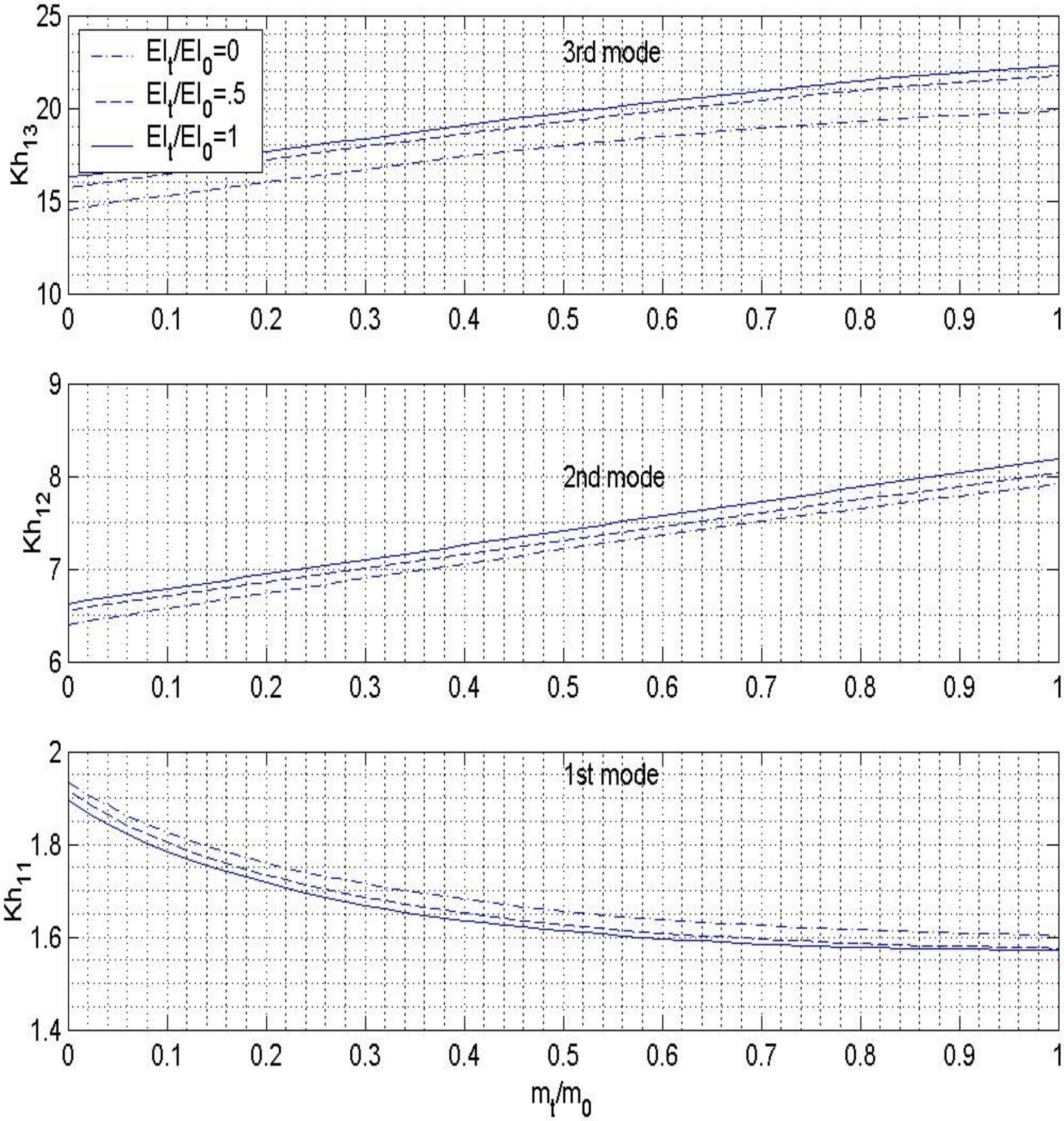



Figure 18. Offset-correction Factors for Southwell Coefficient $\bar{K}_{1n}$ for Cantilever Beams with Linear Mass and Stiffness Distributions. (After: Ref. [1])

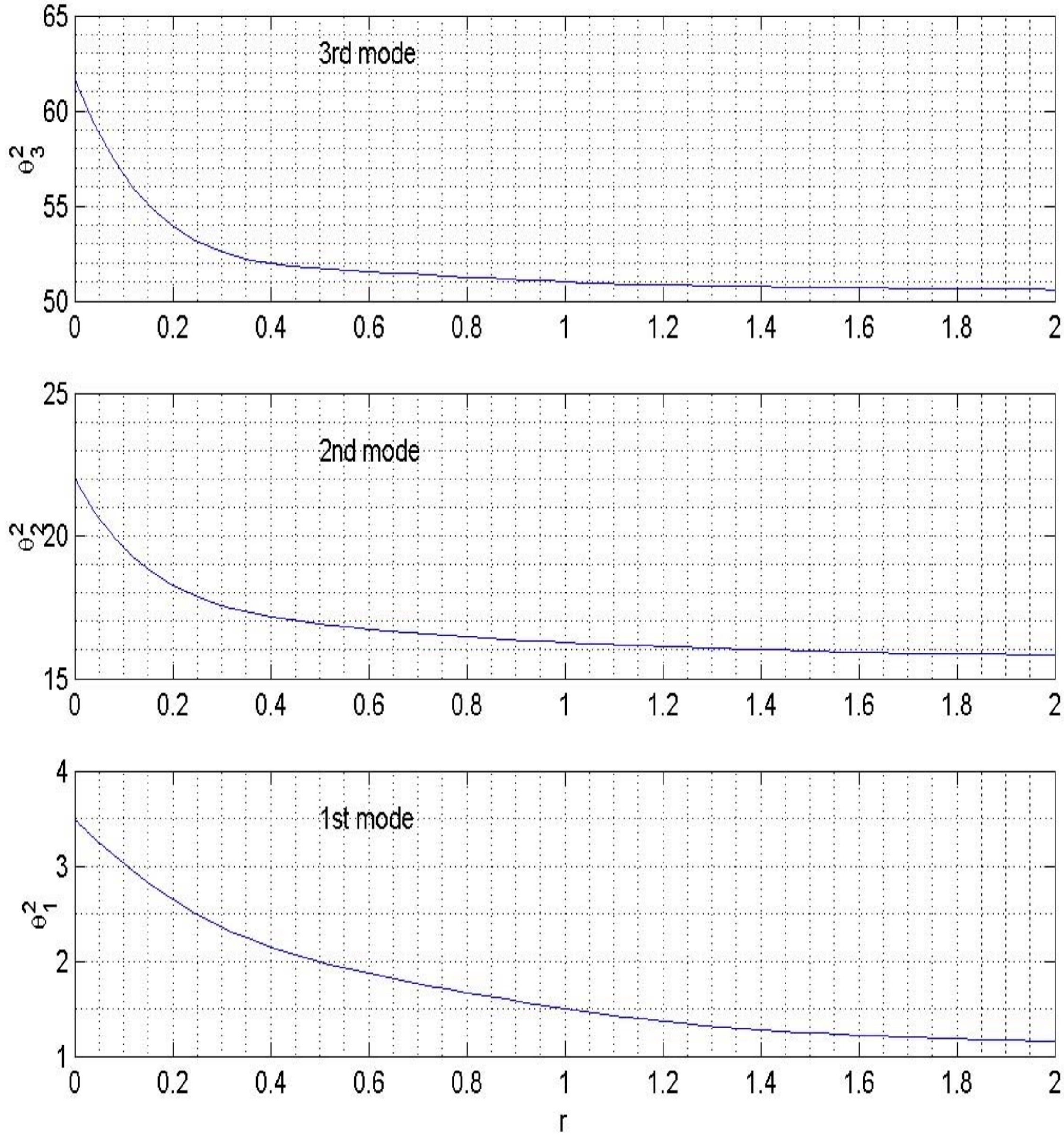



Figure 19. Bending Frequency Coefficients for Nonrotating Uniform Cantilever Beams with a Mass at the Tip. (r = Tip_Mass / Beam_Mass) (After: Ref. [1])

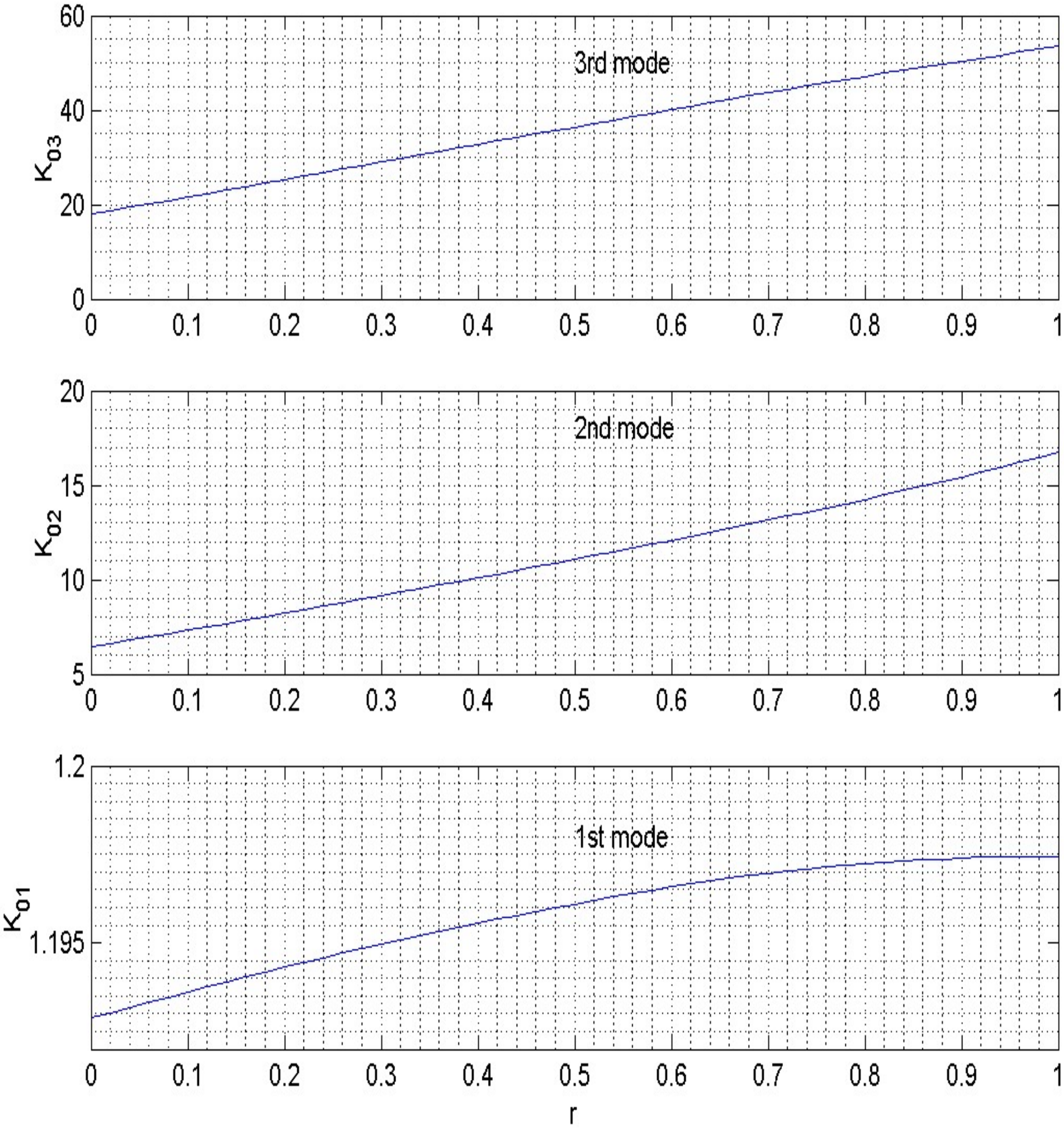



Figure 20. Zero-offset Southwell Coefficients for a Uniform Cantilever Beam with a Mass at the Tip. (r = Tip_Mass / Beam_Mass) (After: Ref. [1])

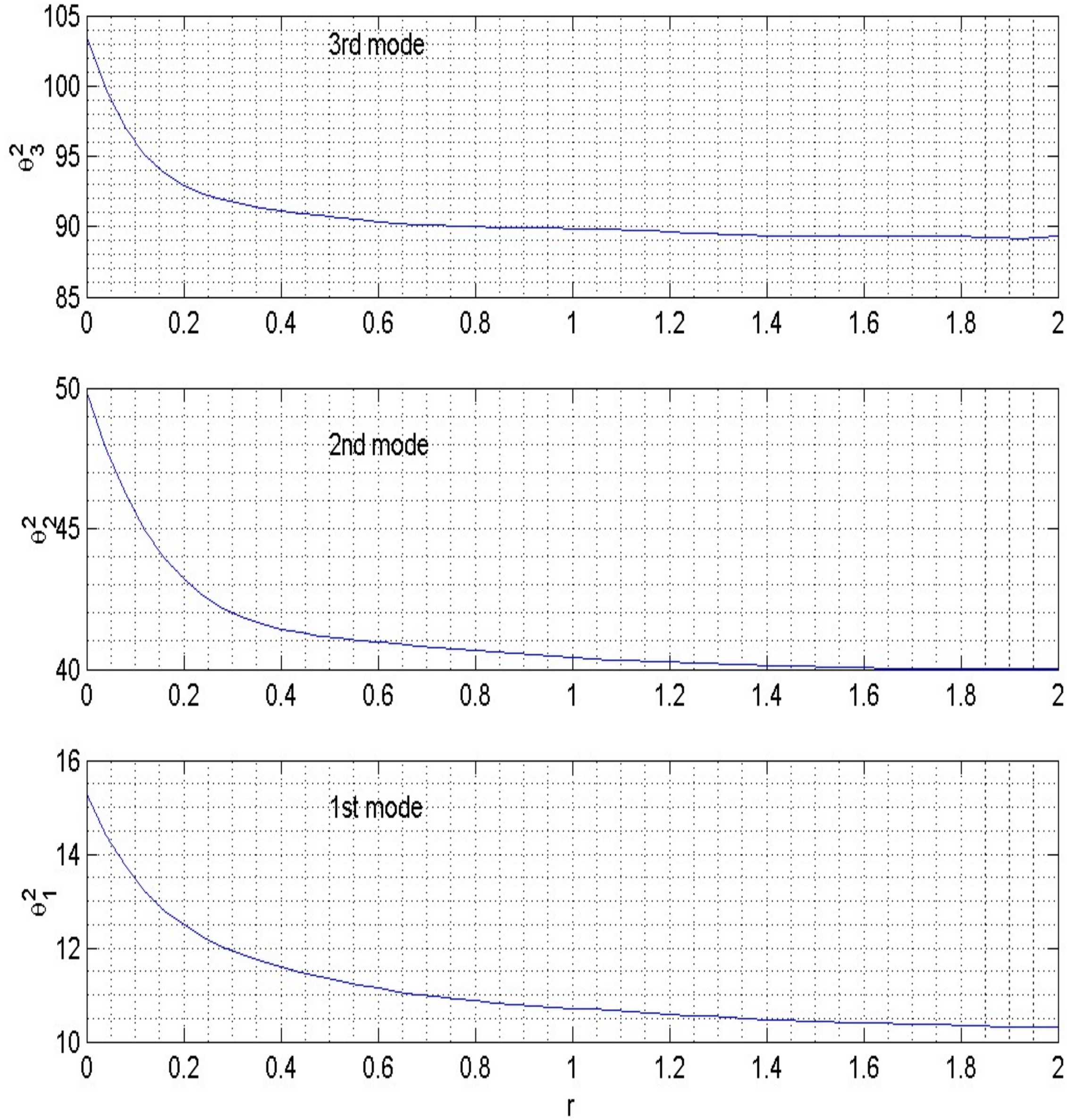


Figure 21. Bending Frequency Coefficients for Nonrotating Uniform Hinged Beams with a Mass at the Tip. (r = Tip_Mass / Beam_Mass) (After: Ref. [1])

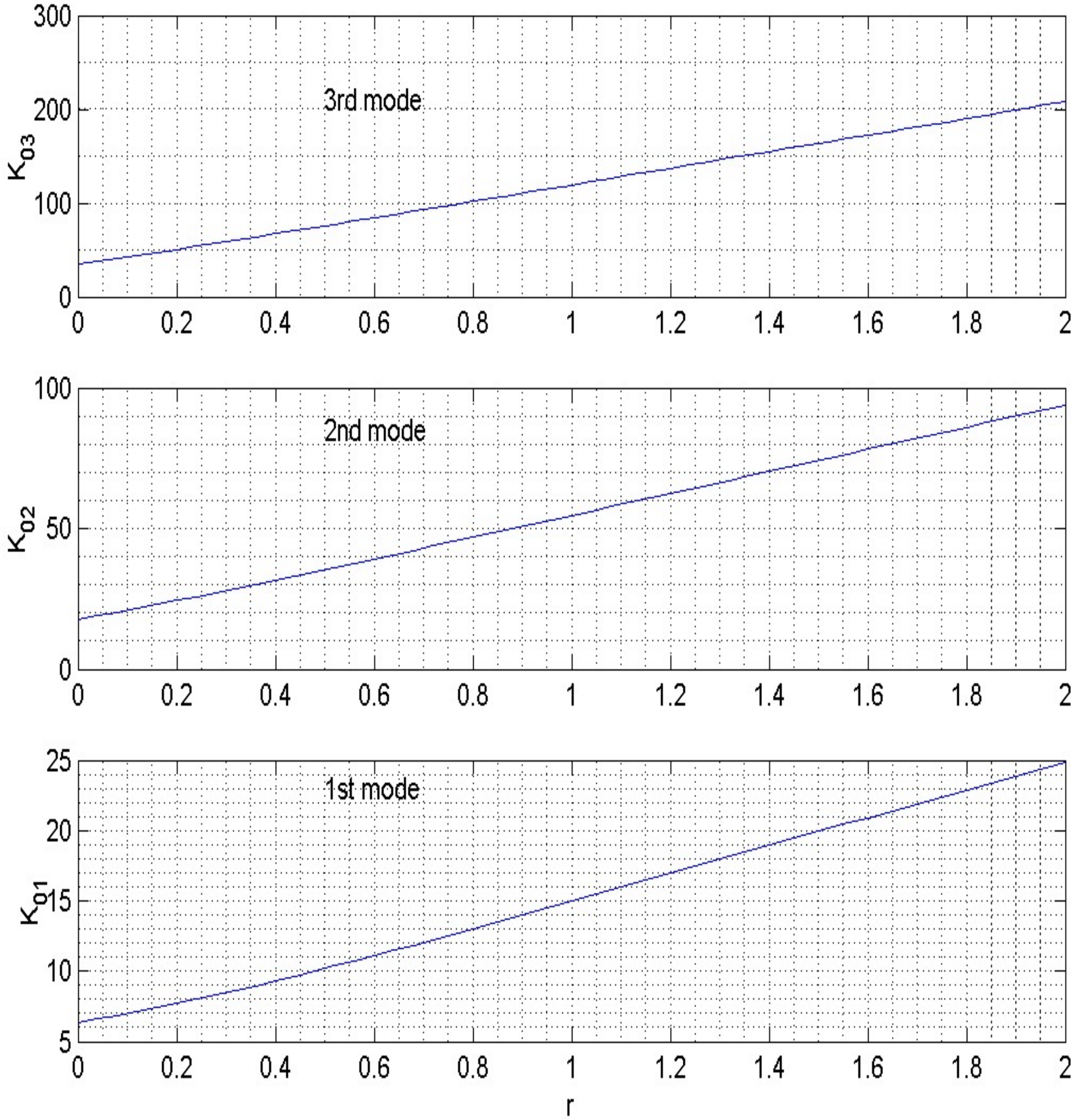



Figure 22. Zero-offset Southwell Coefficients for a Uniform Hinged Beam with a Mass at the Tip. (r = Tip_Mass / Beam_Mass) (After: Ref. [1])

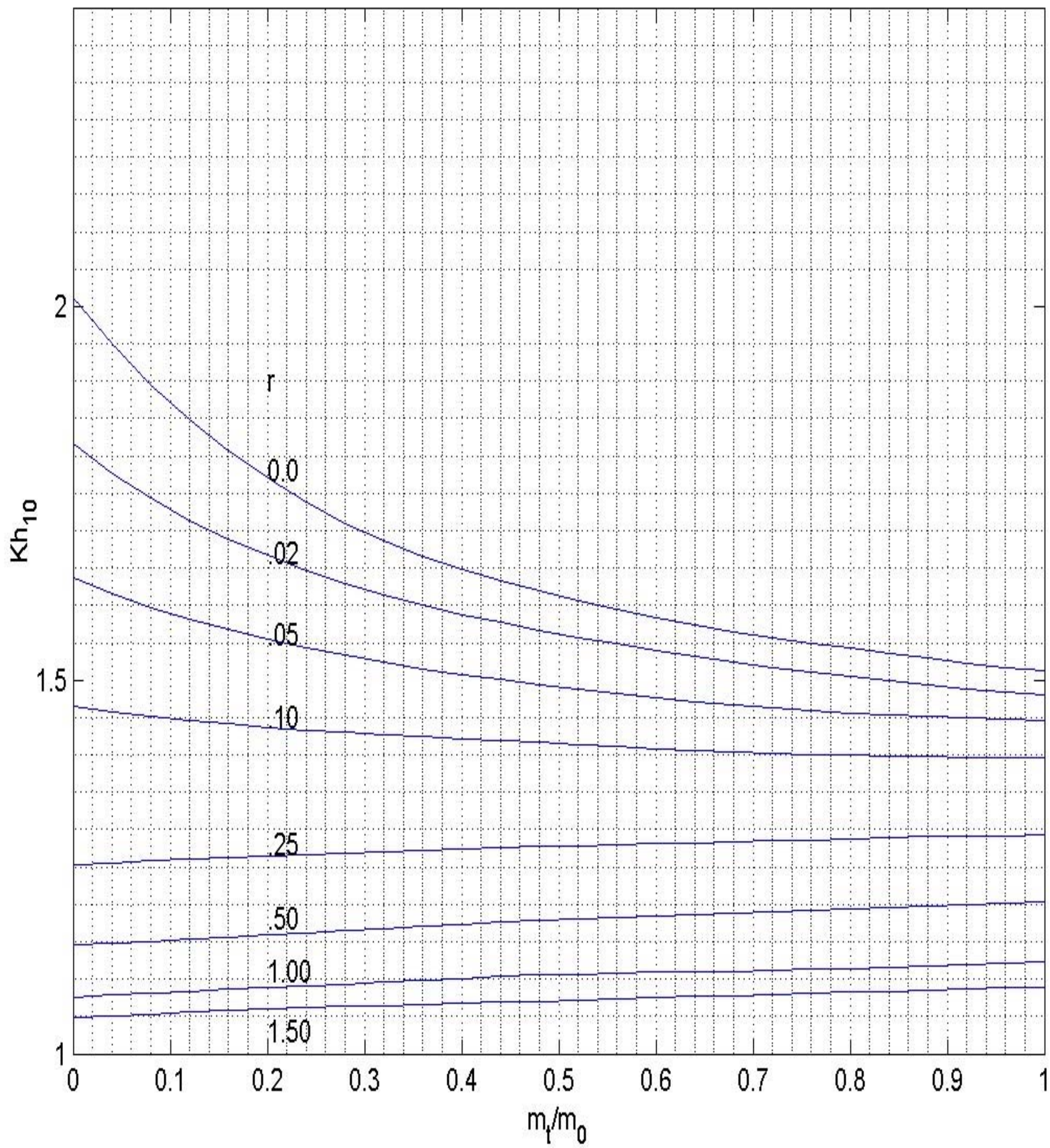



Figure 23. Offset-correction Factors for Southwell Coefficients for the Pendulum Mode of Hinged Beams with Linear Mass Distribution Plus a Mass at the Tip. (After: Ref. [1])

### b. *Validation Of The MATLAB® Script File For Nonrotating Beam Mode Shapes*

Figure 24 through 43 show the reproduced nonrotating beam mode shapes for corresponding Yntema figures. These figures have been generated by YNTEMOSHCOEF.M file. The figures have been visually inspected and manually measured for some reference values and have been found to correspond very well to the Yntema figures. Because both computer-produced and Yntema figures use the same deflection values and there are no manual measurements, they can be accepted as the same. Also in Figures 33 and 43, the first mode shapes of the characteristic beams have been compared for extra evaluations and found to perfectly match the Yntema results. Therefore, the coefficients of YNTEMOSHCOEF.M are accurate representations of Yntema's results.

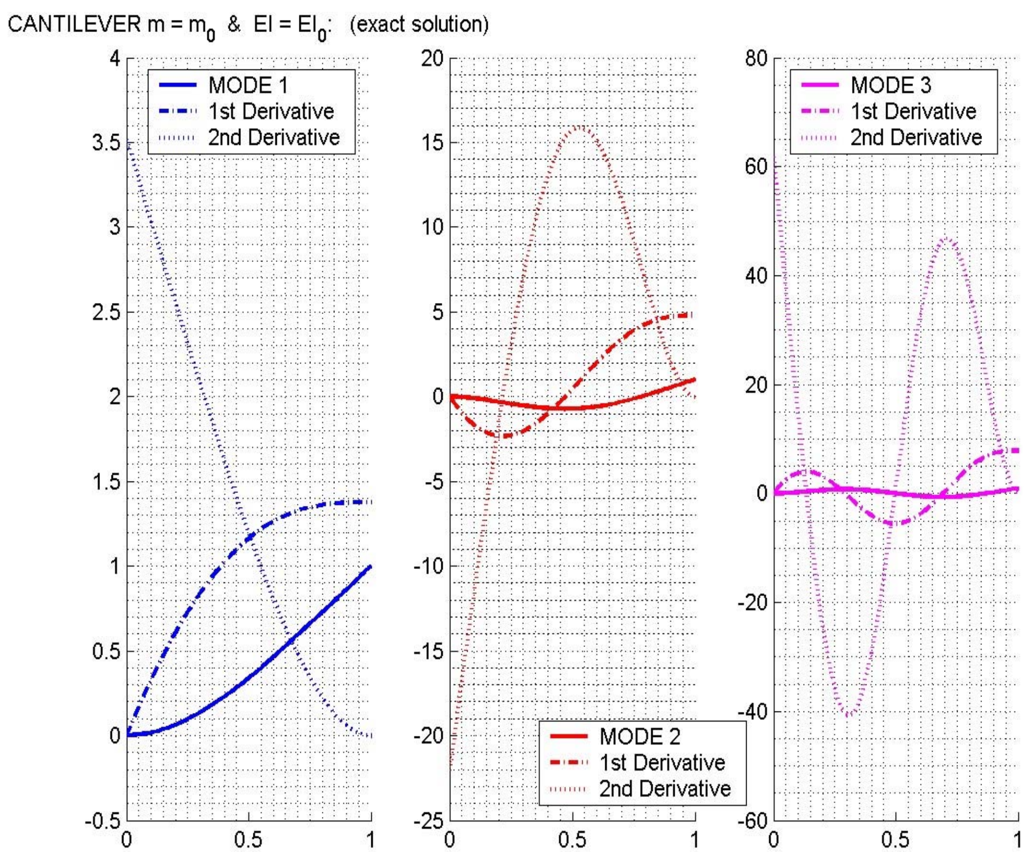



Figure 24. Nonrotating Cantilever Beam Mode Shape. (After: Ref. [1])

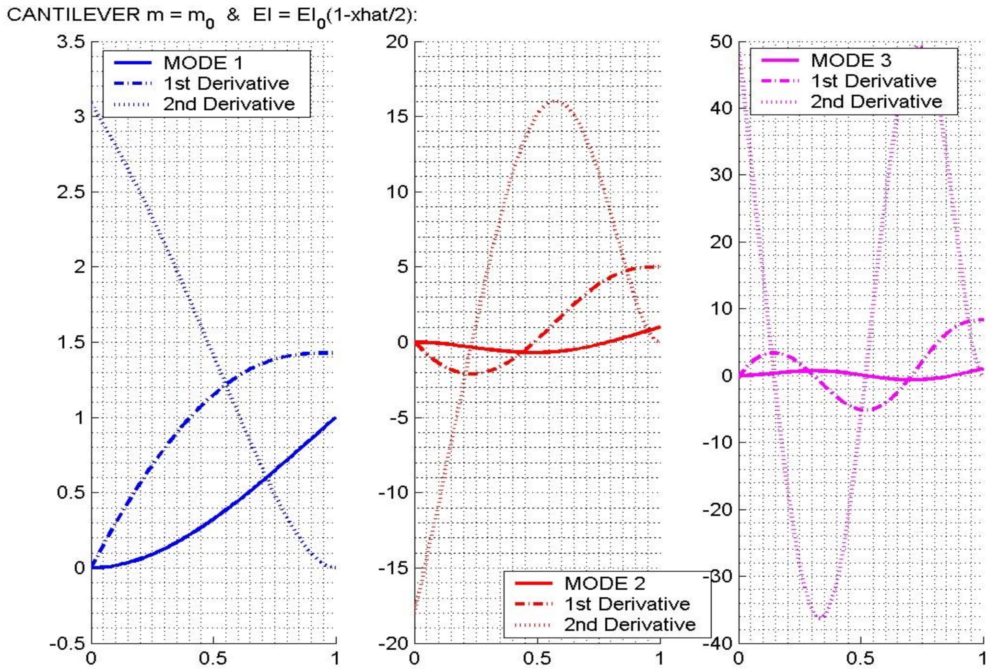



Figure 25. Nonrotating Cantilever Beam Mode Shape. (After: Ref. [1])

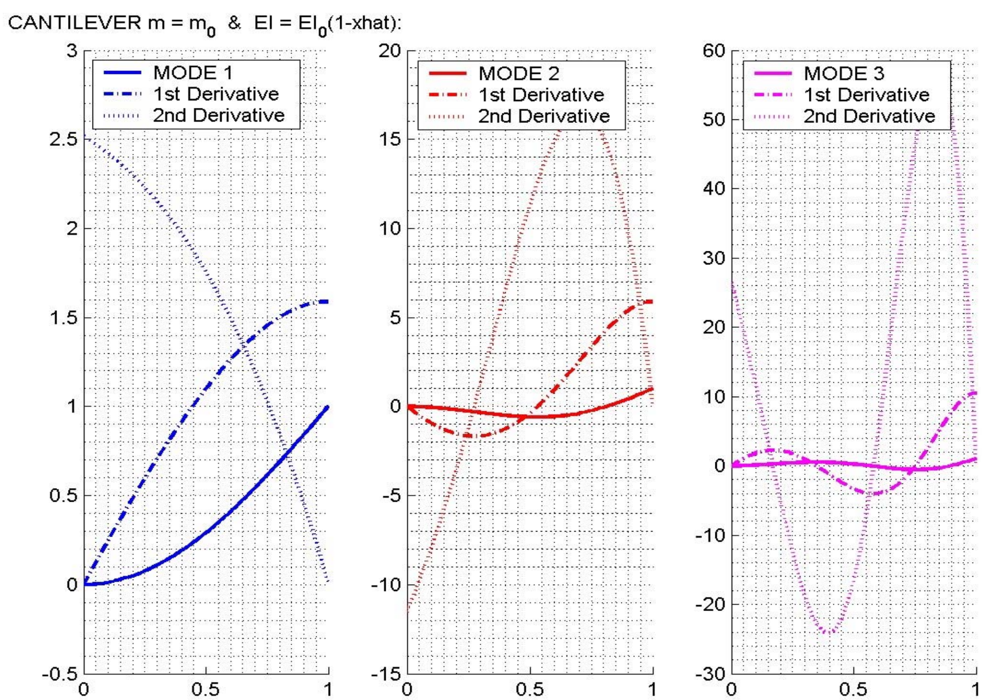



Figure 26. Nonrotating Cantilever Beam Mode Shape. (After: Ref. [1])

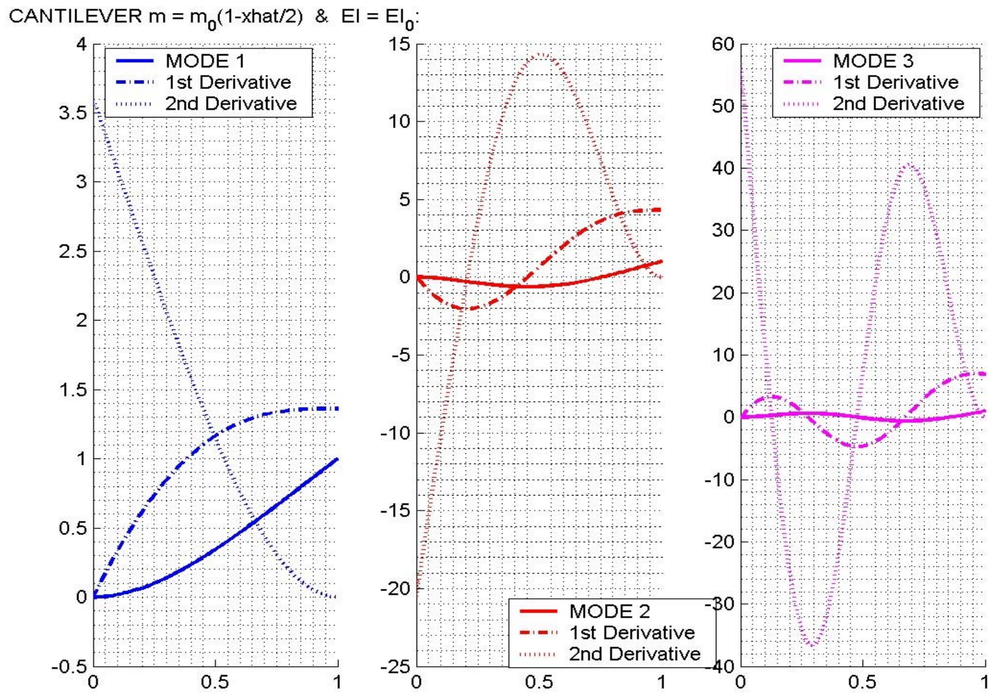



Figure 27. Nonrotating Cantilever Beam Mode Shape. (After: Ref. [1])

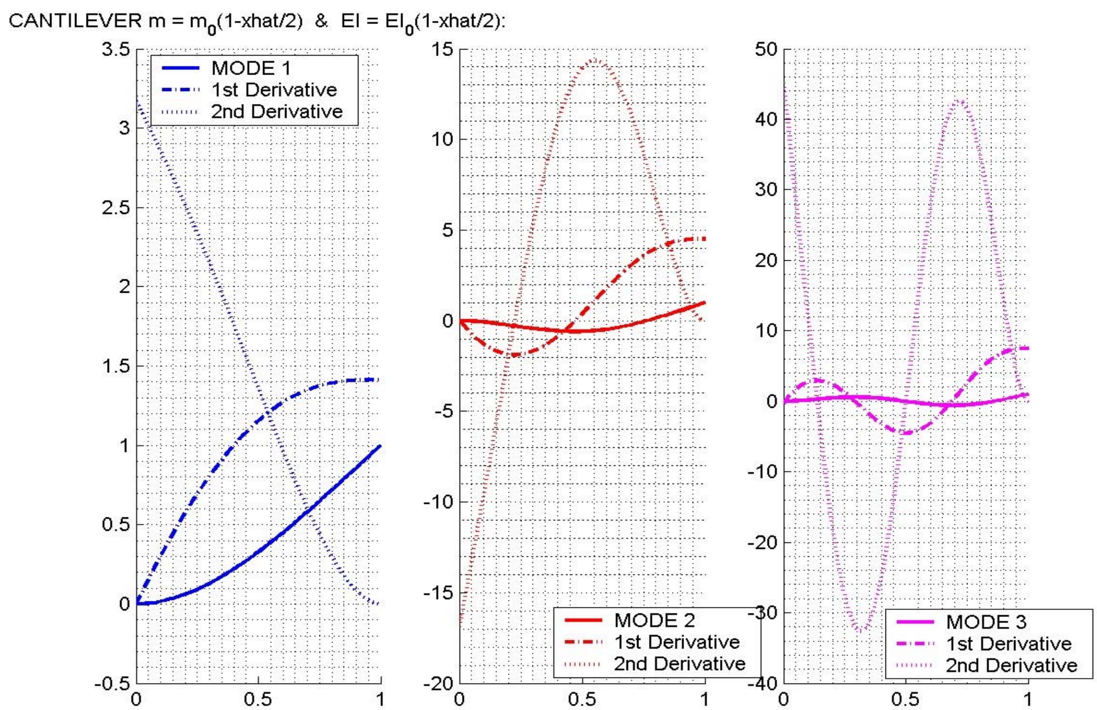



Figure 28. Nonrotating Cantilever Beam Mode Shape. (After: Ref. [1])

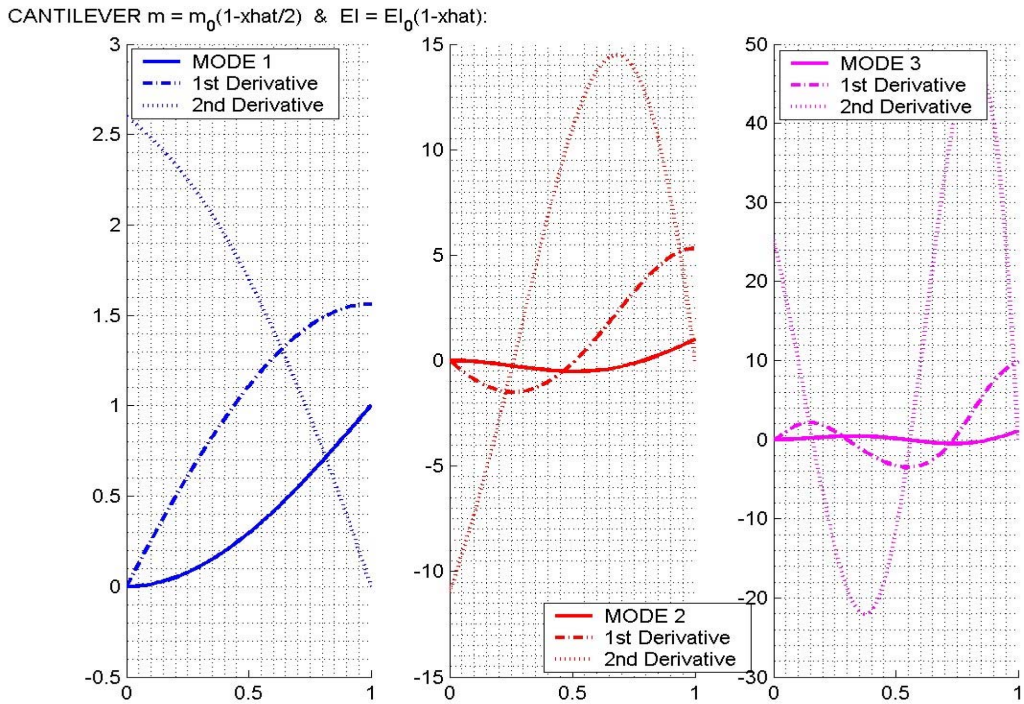



Figure 29. Nonrotating Cantilever Beam Mode Shape. (After: Ref. [1])

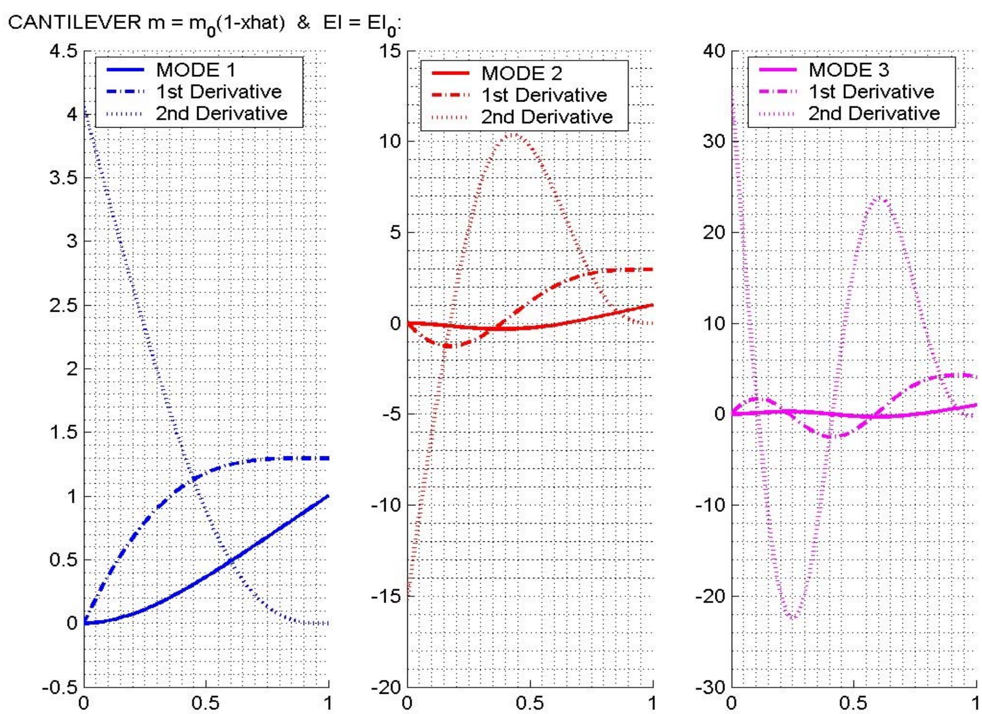



Figure 30. Nonrotating Cantilever Beam Mode Shape. (After: Ref. [1])

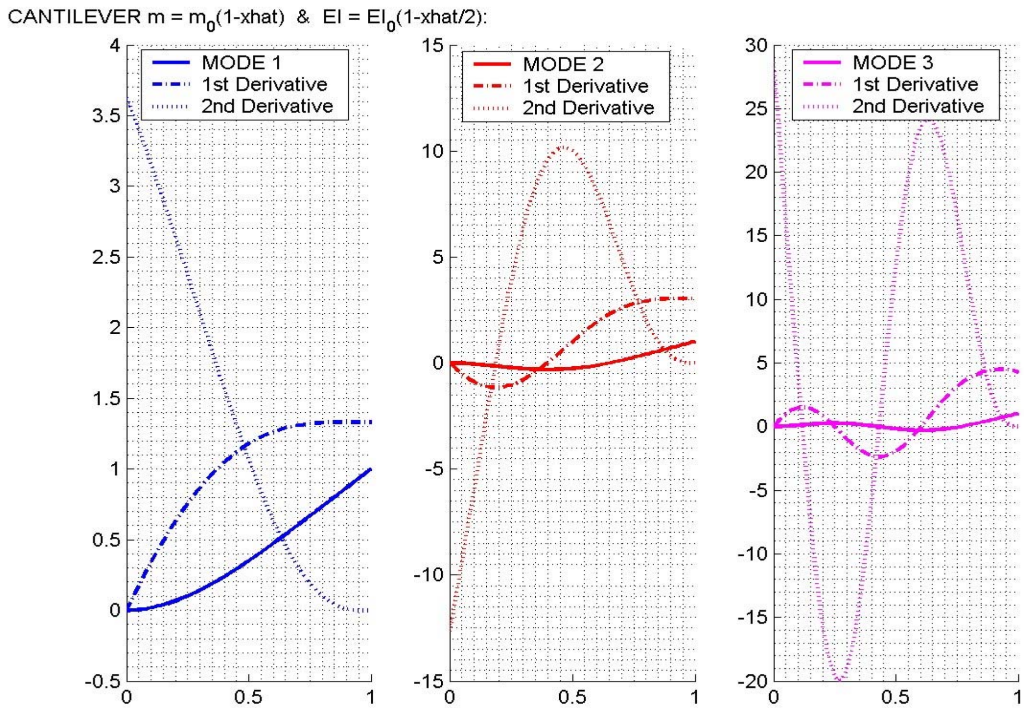



Figure 31. Nonrotating Cantilever Beam Mode Shape. (After: Ref. [1])

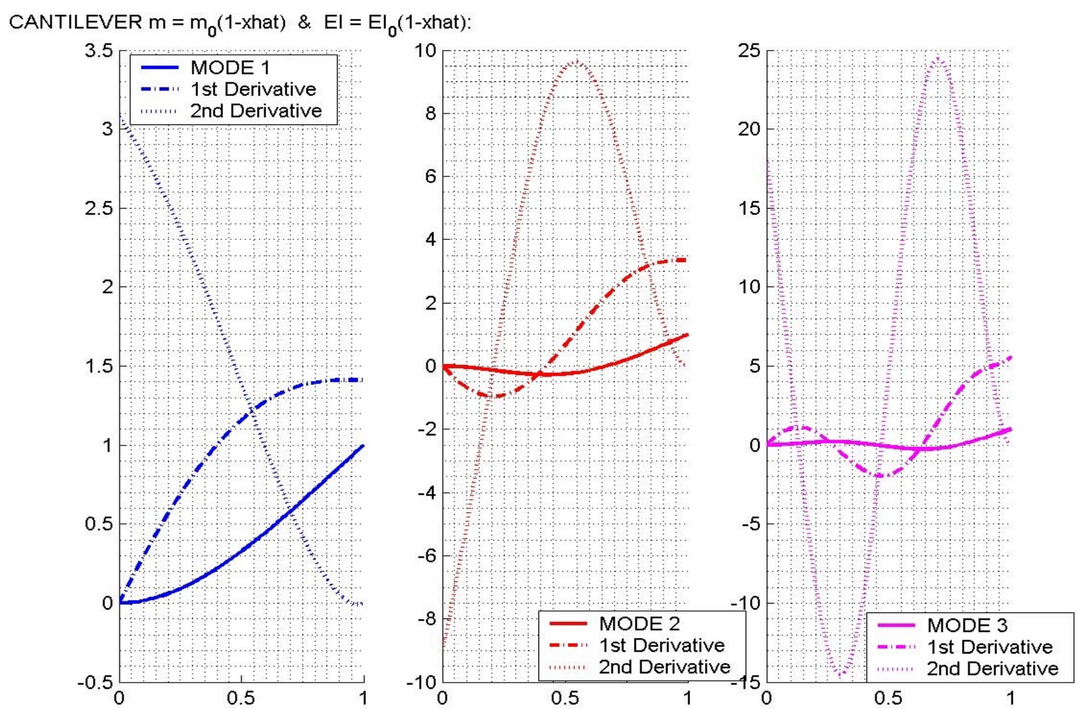



Figure 32. Nonrotating Cantilever Beam Mode Shape. (After: Ref. [1])

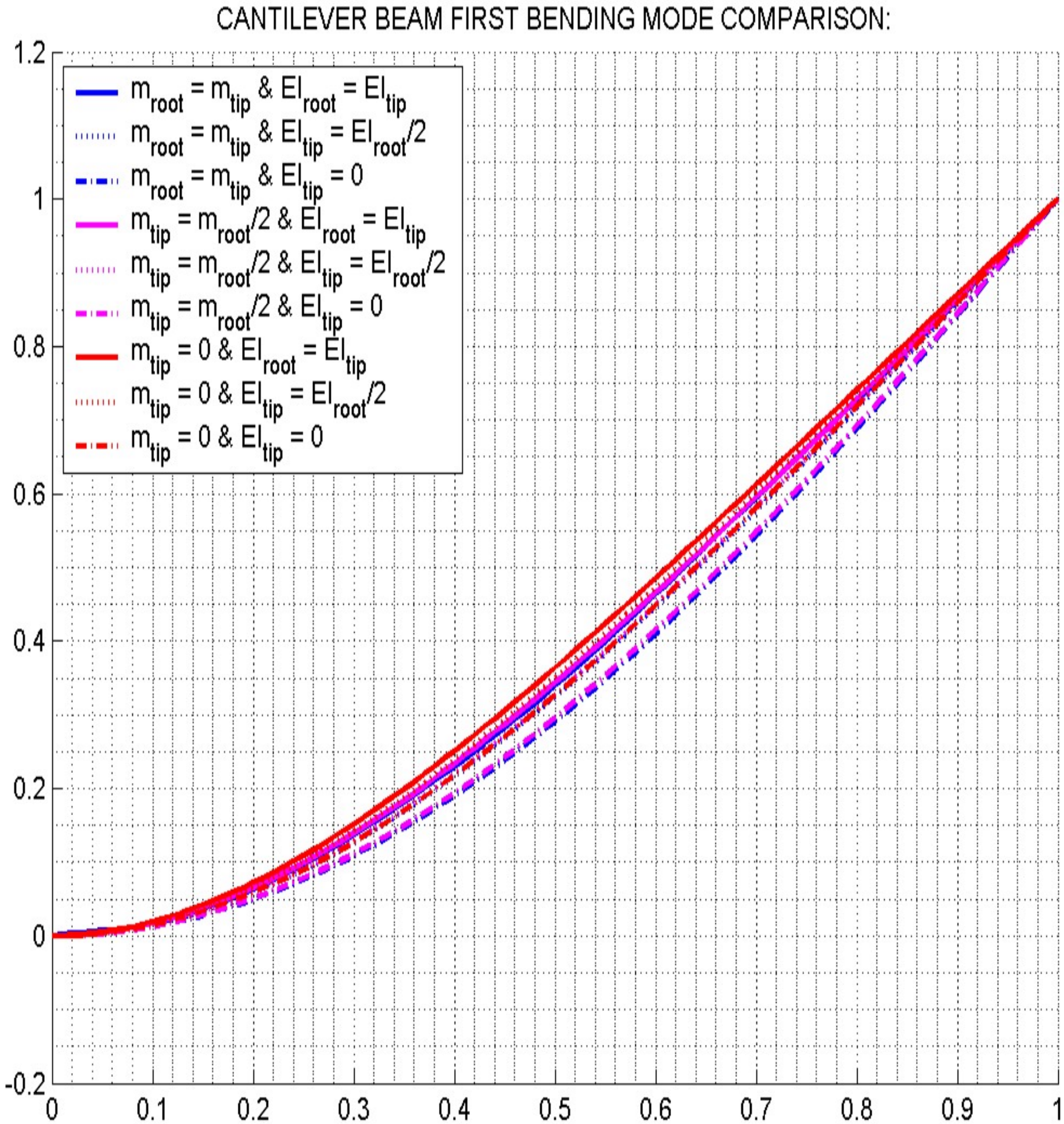



Figure 33. Bending Mode Shape Comparison for the First Mode of Cantilever Beam.

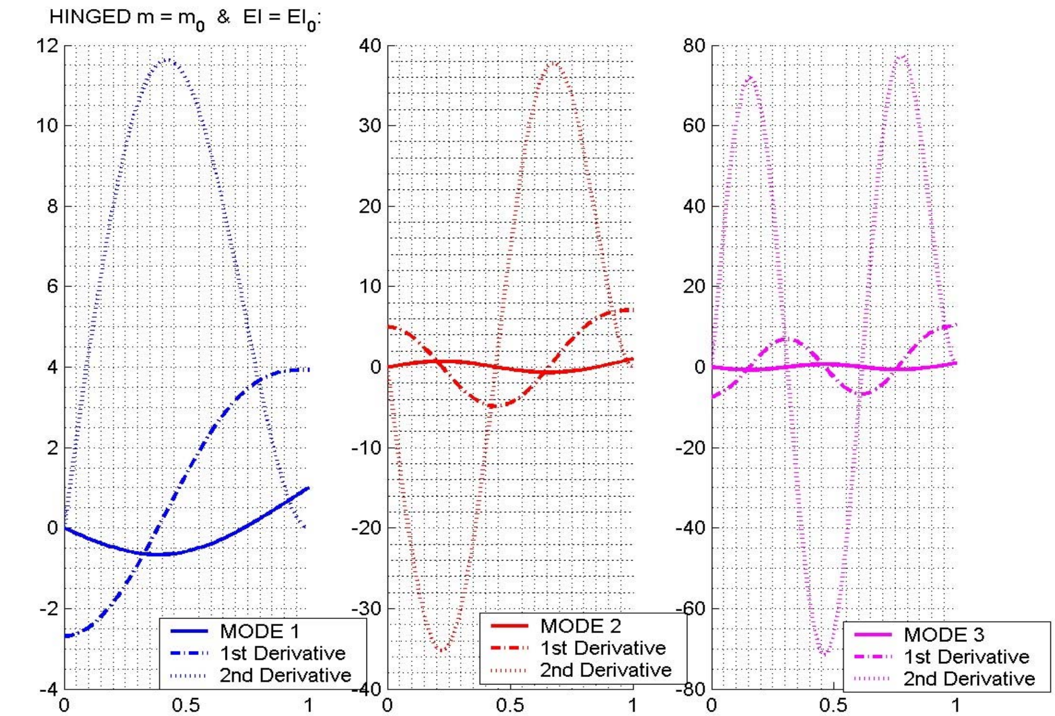



Figure 34. Nonrotating Hinged Beam Mode Shape. (After: Ref. [1])

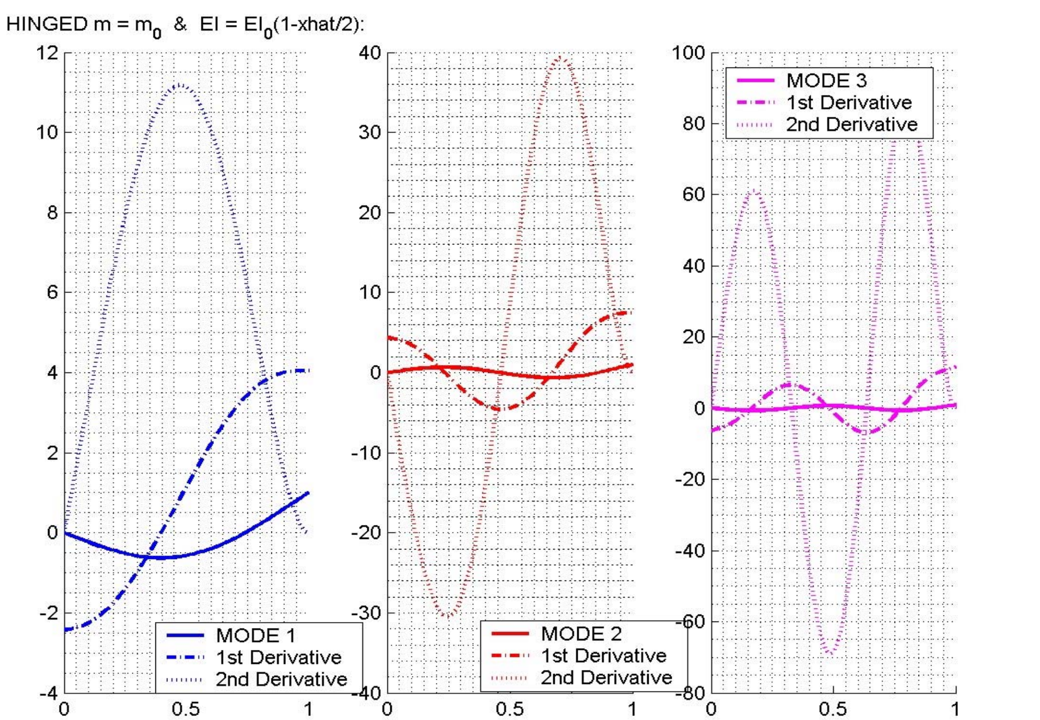



Figure 35. Nonrotating Hinged Beam Mode Shape. (After: Ref. [1])

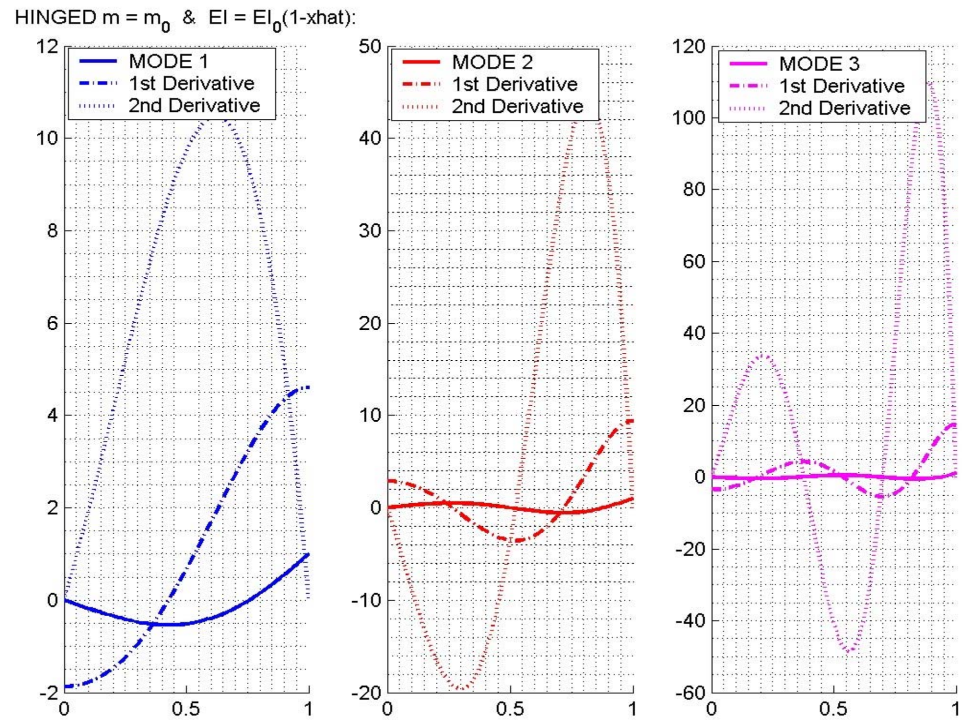



Figure 36. Nonrotating Hinged Beam Mode Shape. (After: Ref. [1])

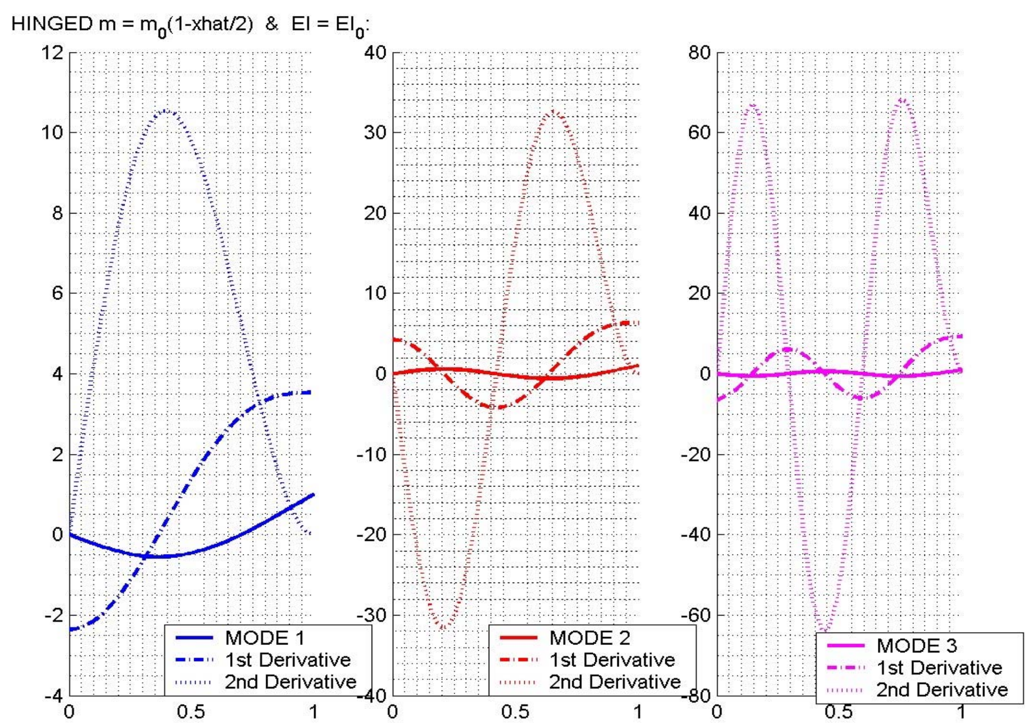



Figure 37. Nonrotating Hinged Beam Mode Shape. (After: Ref. [1])

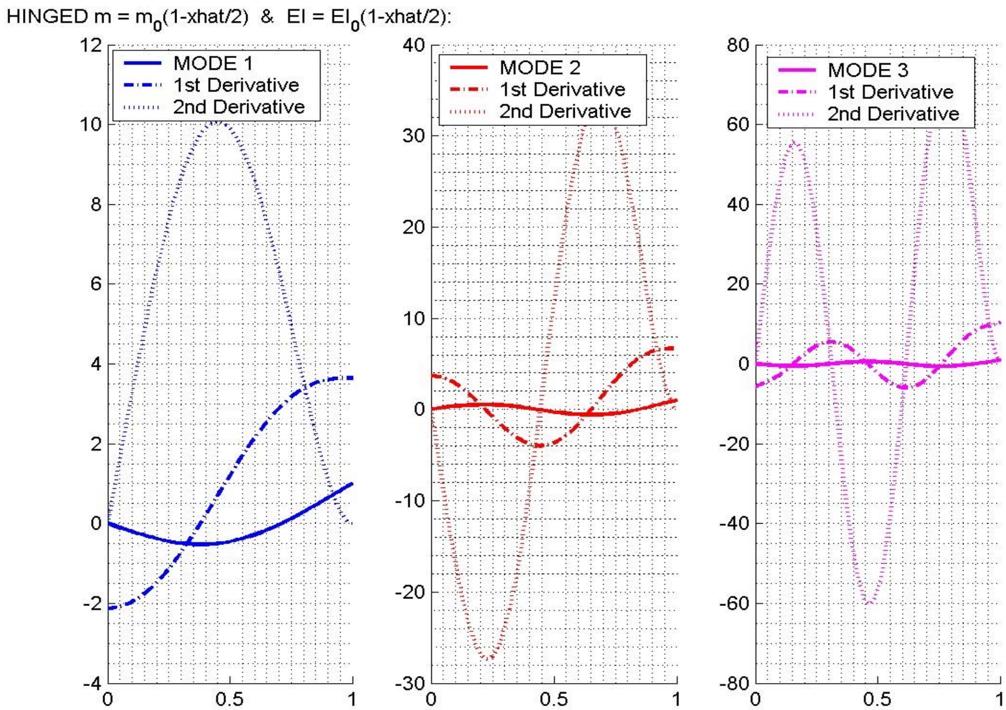



Figure 38. Nonrotating Hinged Beam Mode Shape. (After: Ref. [1])

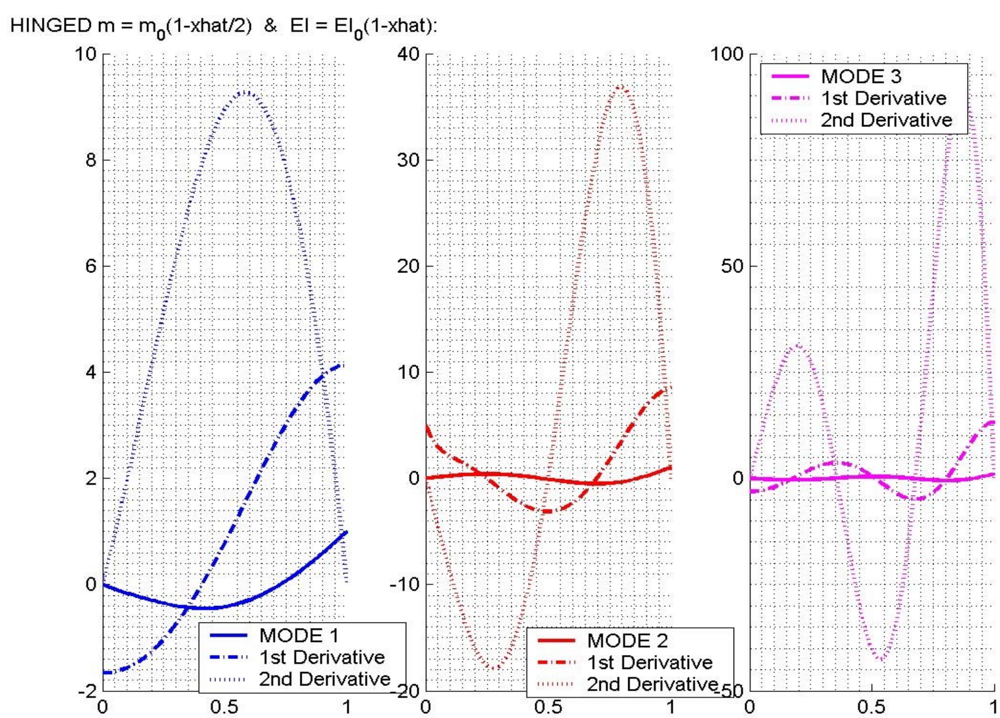



Figure 39. Nonrotating Hinged Beam Mode Shape. (After: Ref. [1])

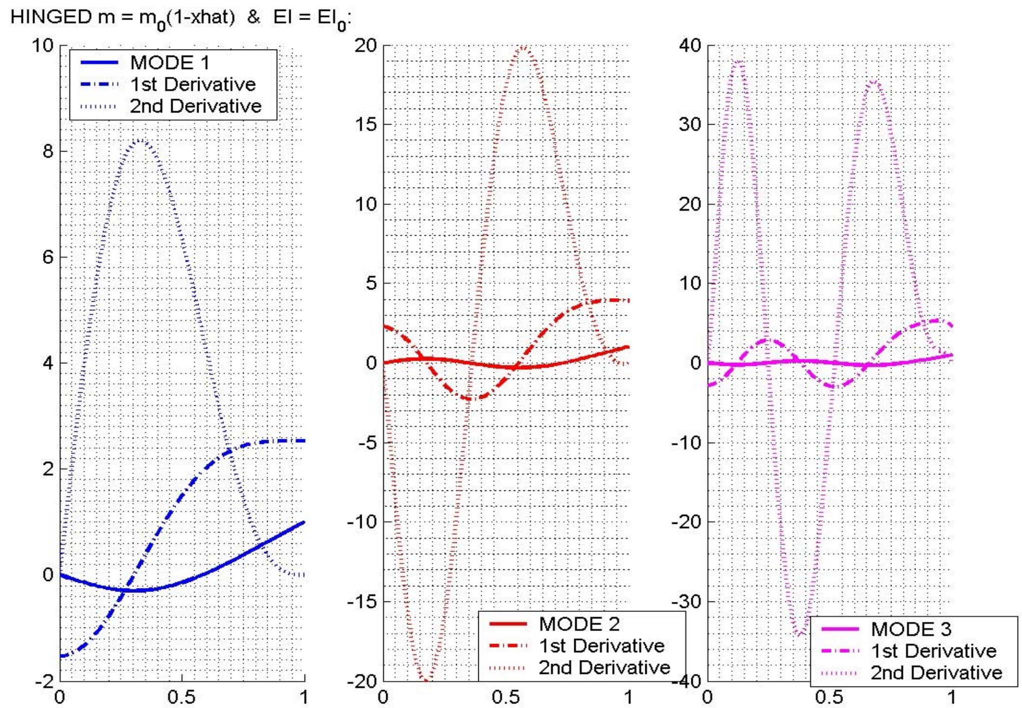



Figure 40. Nonrotating Hinged Beam Mode Shape. (After: Ref. [1])

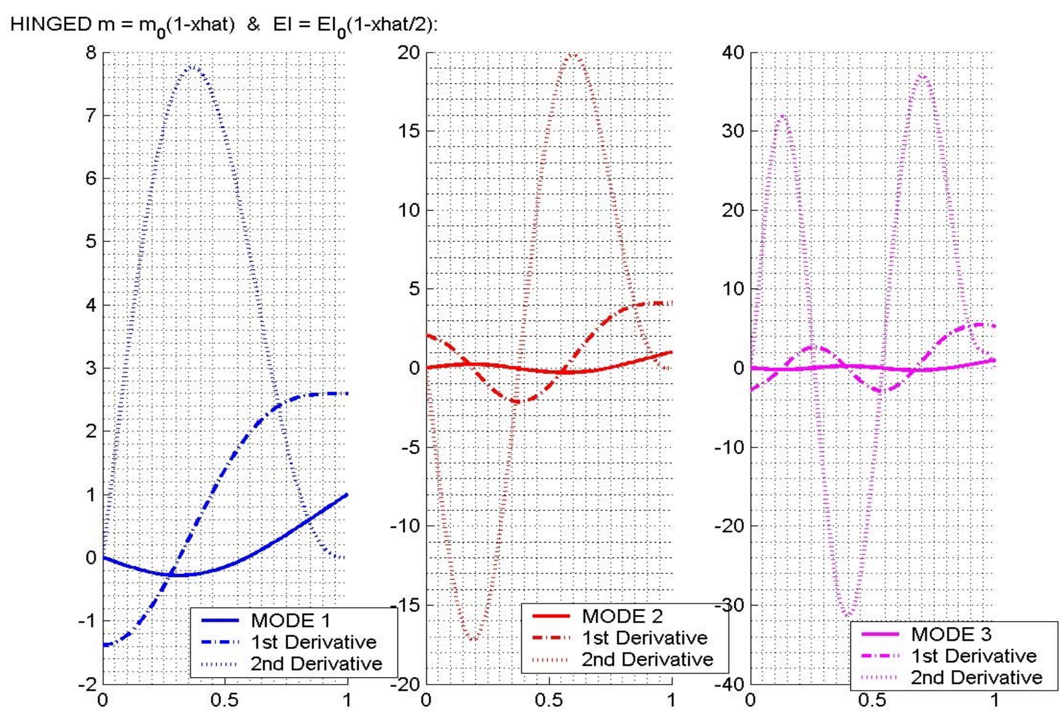



Figure 41. Nonrotating Hinged Beam Mode Shape. (After: Ref. [1])

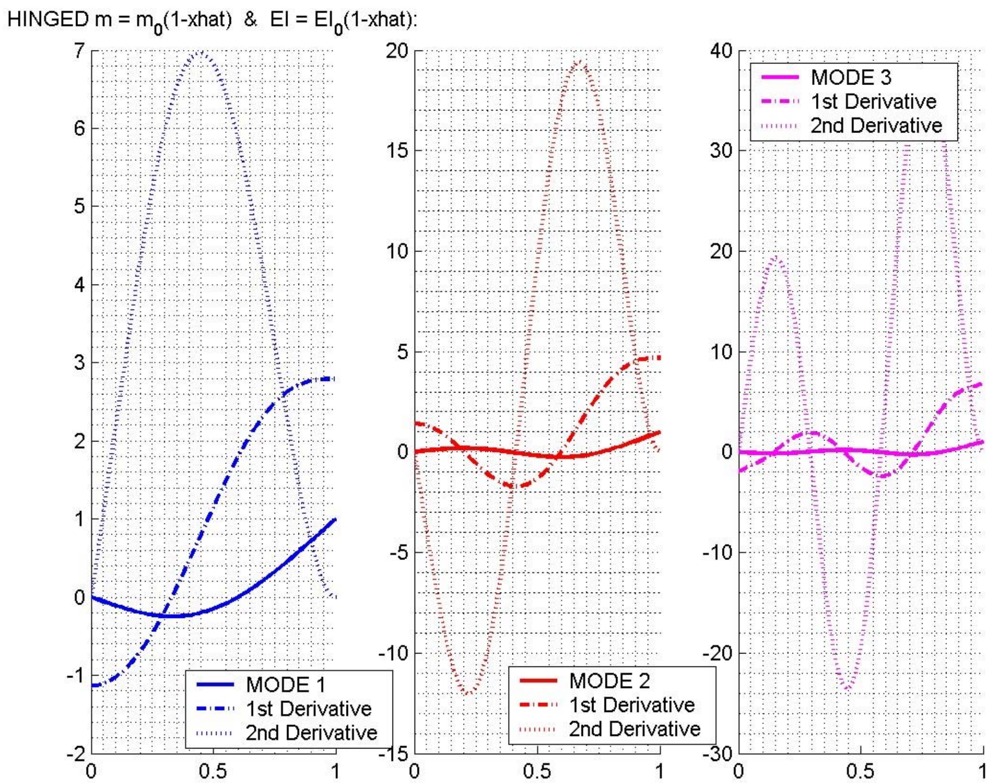



Figure 42. Nonrotating Hinged Beam Mode Shape. (After: Ref. [1])

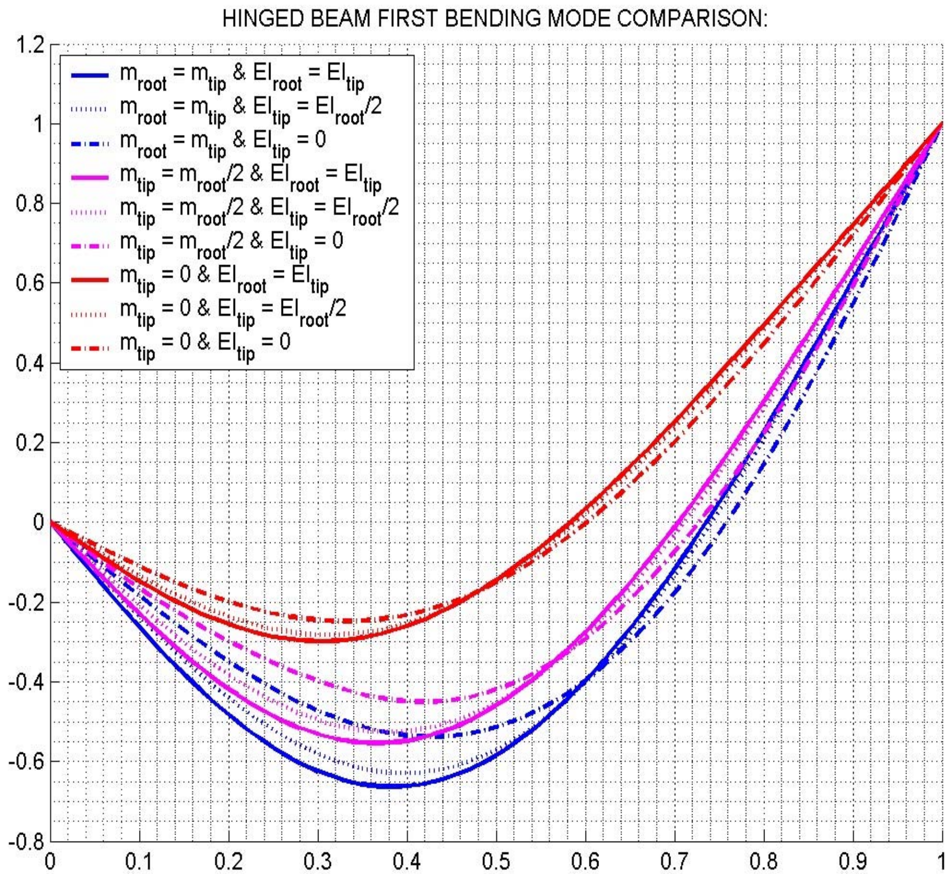



Figure 43. Bending Mode Shape Comparison for the First Mode of Hinged Beam

## 2. Verification And Validation Of The Function Files

### *a. Verification And Validation Of YNTEMA Function*

Because the size of the YNTEMA function file is very large, it is needed to verify the syntax of the function file in the beginning. Section 1 in Appendix E gives a MATLAB® code for making the YNTEMA function calls in a systematical way. This file has been executed successfully for a long time and for different input values, and therefore the YNTEMA function can be said to be free of syntax errors. Section 2 in Appendix E gives the code for checking if the YNTEMA function reads the polynomial coefficients and computes the bending frequency and Southwell coefficients correctly. This code has been executed and the plots have been compared with the original plots of the Yntema report for the bending frequency and Southwell coefficients. This comparison has demonstrated that the YNTEMA function perfectly regenerates the bending frequency and Southwell coefficients.

To demonstrate the accuracy of YNTEMA function, first let us consider a simple uniform cantilever beam. Bisplinghoff [Ref. 5] gives the bending frequency expressions of a uniform cantilever beam without root offset. Table 2 compares the results of Bisplinghoff with the results of YNTEMA function. As can be seen, the results are very accurate the differences being caused by slightly different frequency coefficients used by either method.

<table>
<tr><th colspan="4">COMMON INPUTS</th></tr>
<tr><td>Modulus of Elasticity</td><td>1x10<sup>7</sup> psi</td><td>Blade Length</td><td>200 inches</td></tr>
<tr><td>Moment of Inertia</td><td>2 in<sup>4</sup></td><td>Mass per Length</td><td>0.0022 lbm/in</td></tr>
<tr><th colspan="4">COMPUTATION</th></tr>
<tr><td>YNTEMA Function Call:</td><td colspan="3">yntema(1,`C`,0,200,1e7,0.0022,2)</td></tr>
<tr><td>Equations from Bisplinghoff:</td><td>1st mode: $(0.597)^2\left(\frac{\pi}{l}\right)^2\sqrt{\frac{EI}{m}}$</td><td>2nd mode: $(1.49)^2\left(\frac{\pi}{l}\right)^2\sqrt{\frac{EI}{m}}$</td><td>3rd mode: $\left(\frac{5}{2}\right)^2\left(\frac{\pi}{l}\right)^2\sqrt{\frac{EI}{m}}$</td></tr>
<tr><th colspan="4">BENDING FREQUENCY RESULTS</th></tr>
<tr><th>Modes</th><th colspan="2">YNTEMA Function</th><th>Bisplinghoff</th></tr>
<tr><td>1st</td><td colspan="2">4.0748 CPM = 0.4267 rad/sec</td><td>0.4267 rad/sec</td></tr>
<tr><td>2nd</td><td colspan="2">25.48 CPM = 2.6682 rad/sec</td><td>2.6581 rad/sec</td></tr>
<tr><td>3rd</td><td colspan="2">72.026 CPM = 7.5425 rad/sec</td><td>7.4831 rad/sec</td></tr>
</table>

Table 2. A Comparison of Results for a Uniform Cantilever Beam.

The YNTEMA function also returns the Southwell plot as well as a list of assumptions that has been made. Figure 44 gives the Southwell plot for this uniform cantilever beam and Table 3 gives the messages displayed by YNTEMA function.

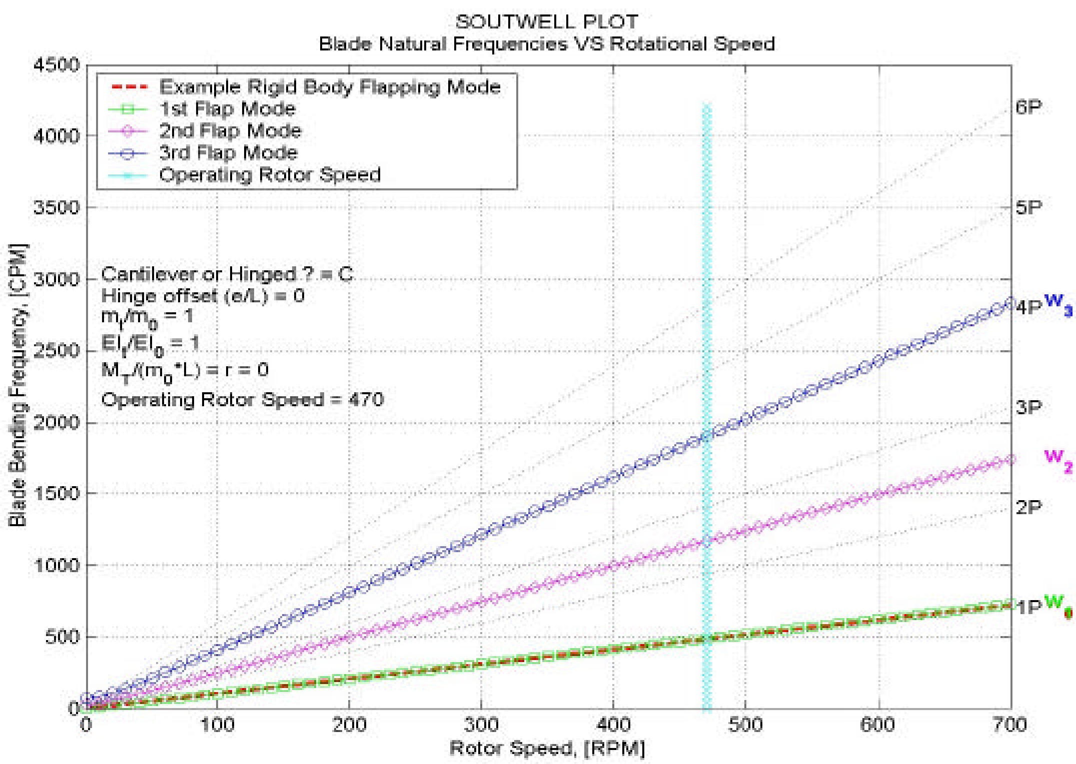



Figure 44. Southwell Plot for the Example Uniform Cantilever Beam.

```
>> [w1,w2,w3,w0,rotorSpeed]=yntema(1,'C',0,200,1e7,0.0022,2);
ATTENTION: You did not specify any extra input arguments.
           The beam is assumed to be uniform and without a concentrated mass at the tip.
WARNING: No operating rotor speed has been input as the last element of the rtrSpdRng
input argument... An arbitrary value has been assigned for the operating rotor speed.
WARNING: The rigid body bending mode cannot be calculated because the Southwell
coefficients are empty for the 0th-mode. The beam may be cantilever for which the 0th-
mode Southwell coefficients are empty... Check below. If the beam is Cantilever, there is
no problem:  THE BEAM IS CANTILEVER. THERE IS NO 0th MODE FOR CANTILEVER BEAMS.
0th-mode Bending Frequency can not be plotted... Its value is output below: (may be
empty!)   0   But it is assumed to be (1.03 * 1P) which is plotted for example.
Total Run Time is 1.271 sec.
Cantilever or Hinged ? = C
Root offset (e/L) = 0
m_t/m_0 = 1
EI_t/EI_0 = 1
M_T/(m_0*L) = r = 0
Operating Rotor Speed = 470
```

Table 3. The List of Messages Displayed by YNTEMA Function for the Example Uniform Cantilever Beam.

To examine the robustness of YNTEMA function, a cantilever beam but, this time, with a linear mass distribution and a nonlinear stiffness distribution is used as described in Bisplinghoff [Ref. 5]. Since the beam has a nonlinear stiffness distribution, YNTEMA function does not return accurate results for this beam. However, the stiffness distribution can be linearized with MAKELINE function, and then the YNTEMA function can be used for approximate bending frequency results. Figure 45 shows the beam with its actual and linearized stiffness distribution values. There were originally eleven stiffness values, but we can incorporate LINKSPOR function to get, e.g., a hundred stiffness values. Then we can linearize the distribution using MAKELINE function. Since the stiffness distribution seems to be very high near the root section, we can use a cutoff value of, say, 20 for the root. With the polynomial coefficients found from the MAKELINE function, the stiffness values at the root and the tip can be determined. The stiffness at the tip is negative but it can be assumed to be zero because its magnitude is very small. The YNTEMA function is then called to estimate approximate bending frequency results. Table 4 summarizes the procedure.

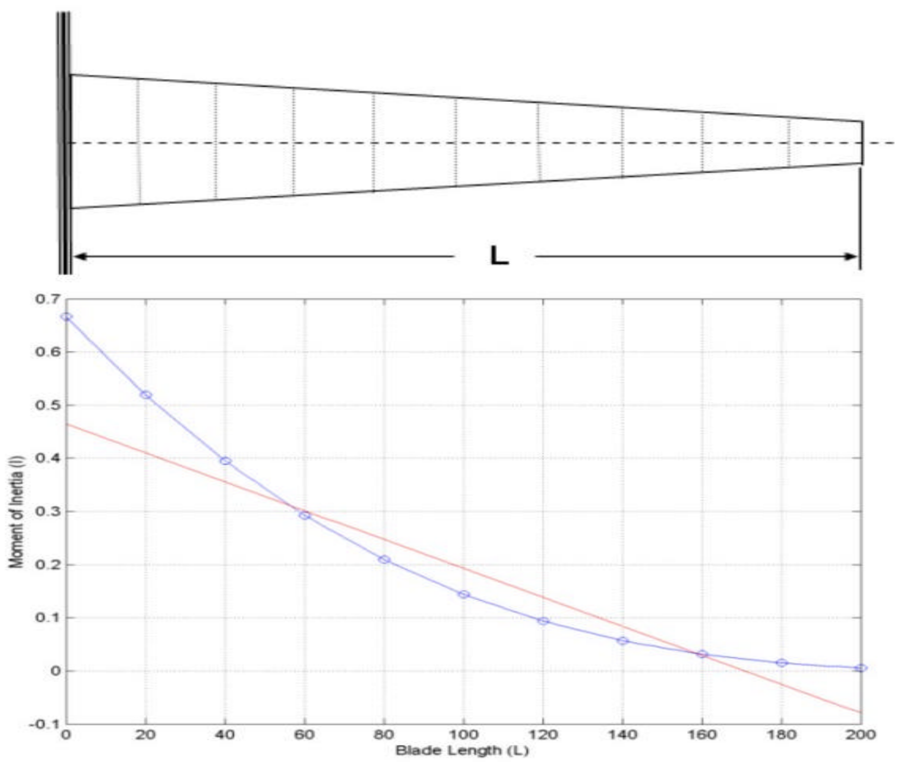



Figure 45. Example Cantilever Beam with Nonlinear Stiffness Distribution.

```
>> minert = (2/3) .* [1 .7787 .5927 .4389 .3144 .2160 .1406 .08518 .04665 .02195 .008];
>> xinert = [0:20:200];
>> [outvec,outord,outvecx]=linkspor(minert,200,100);
>> polyval(makeline(outvec,outvecx,20),0)
ans =    0.4648
>> polyval(makeline(outvec,outvecx,20),200)
ans =   -0.0792
>> I_root = polyval(makeline(outvec,outvecx,20),0)
I_root =    0.4648
>> I_tip = polyval(makeline(outvec,outvecx,20),200)
I_tip =   -0.0792
>> I_tip = 0;
>> [w1,w2,w3,w0,rotorSpeed,coeffs] = yntema(1,'C',0,200,1e7,0.001,I_root,'mtip',0.0002,
'Itip',I_tip);
```

Table 4. The Procedure for Nonlinear Cantilever Beam Computation.

The bending frequency results and the Southwell plot for the nonlinear cantilever beam are given in Table 5 and Figure 46 respectively. The YNTEMA result for the first mode is very accurate and for higher modes the error seems to be increasing. This case is valid for all methods as well as the Yntema method. Remember that the Yntema results are rapid approximations rather than exact solutions and a nonlinear cantilever beam is a potential worst case scenario. Errors of YNTEMA are well within the errors encountered by Bisplinghoff [Ref. 5].

| **COMMON INPUTS** | | | | |
|---|---|---|---|---|
| Modulus of Elasticity | $1x10^7$ psi | Blade Length | 200 inches | |
| Mass per Length at Root | 0.001 lbm/in | Mass per Length at Tip | 0.0002 lbm/in | |
| **BENDING FREQUENCY RESULTS** | | | | |
| **Modes** | **Exact [Ref. 5]** | **YNTEMA Function** | **Error Range by Bisplinghoff** | **Error in YNTEMA** |
| 1st | 0.45015 rad/sec | 4.2962 CPM = 0.4499 rad/sec | -0.04% ~ 2.38% | 0.06% |
| 2nd | 1.63592 rad/sec | 19.443 CPM = 2.0361 rad/sec | -0.96% ~ 42% | -19.6% |
| 3rd | 3.83161 rad/sec | 49.897 CPM = 5.2252 rad/sec | -11.2% ~ 66.2% | -26.7% |

Table 5. A Comparison of Results for a Nonlinear Cantilever Beam.

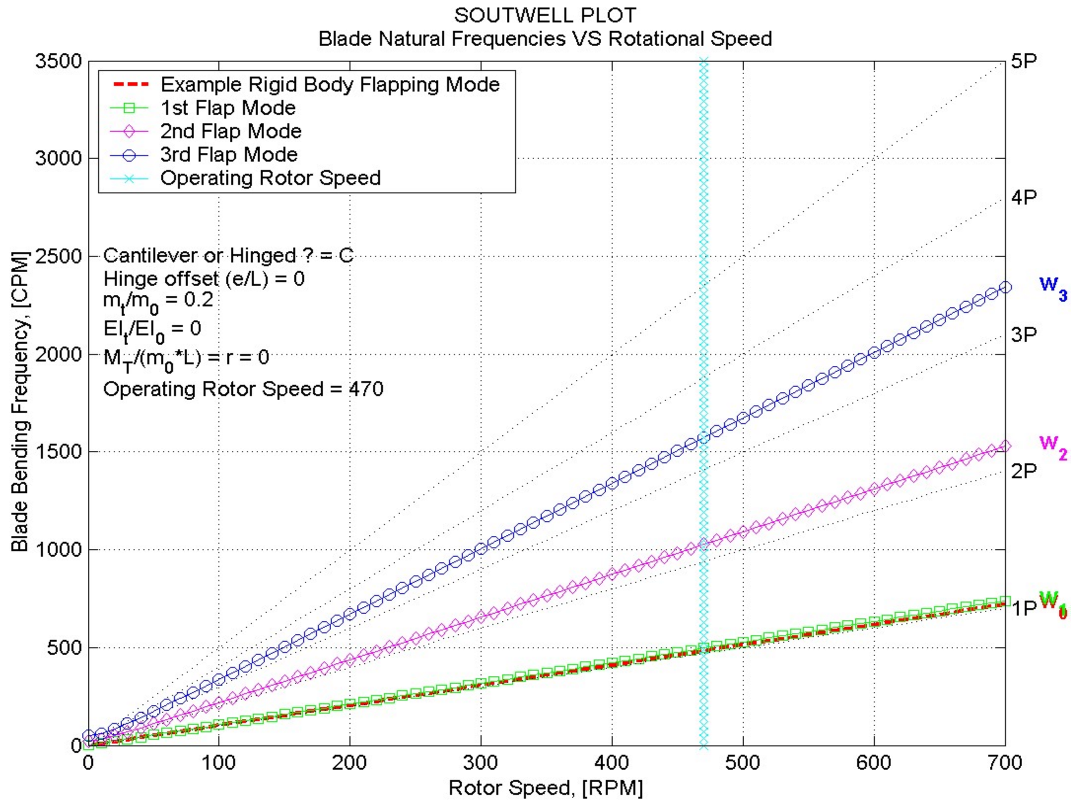



Figure 46. Southwell Plot for a Nonlinear Cantilever Beam.

### b. *Verification And Validation Of YNTERYGH Function*

As the basis for YNTERYGH function, Equation(1) has been adapted for numerical analysis. Equation(1) can be defined with appropriate parameters as follows:

$$\omega_{R_n}^2 = \frac{num_1}{denum} + \frac{(num_{20} + e \cdot num_{21})}{denum}\Omega^2 \tag{25}$$

For a beam with $n$ stations from the root to the tip, these parameters are given by

$$\left.\begin{aligned}
denum &= \int_0^L m y_n^2 dx = \sum_{i=1}^{n}\left[m(i)\cdot(y(i))^2\cdot\Delta x\right] \\
num_1 &= \int_0^L EI y_n''^2 dx = \sum_{i=1}^{n}\left[EI(i)\cdot(y''(i))^2\cdot\Delta x\right] \\
num_{20} &= \int_0^L y_n'^2\left[\int_x^L xm dx\right]dx = \sum_{i=1}^{n}\left[(y'(i))^2\cdot\left\{\sum_{j=i}^{n}\left[x(i)\cdot m(i)\cdot\Delta x\right]\right\}\cdot\Delta x\right] \\
num_{21} &= \int_0^L y_n'^2\left[\int_x^L m dx\right]dx = \sum_{i=1}^{n}\left[(y'(i))^2\cdot\left\{\sum_{j=i}^{n}\left[m(i)\cdot\Delta x\right]\right\}\cdot\Delta x\right]
\end{aligned}\right\} \tag{26}$$

where *(i)* shows the value of each parameter at corresponding station. The bending frequency and Southwell coefficients in terms of these parameters are given by

$$\left.\begin{aligned} a_n &= \sqrt{\frac{\left(\dfrac{num_1}{denum}\right)\cdot m(1)\cdot L^4}{EI(1)}} \\ K_{0_n} &= \frac{num_{20}}{denum} \\ K_{1_n} &= \frac{num_{21}}{denum} \end{aligned}\right\} \tag{27}$$

The nature of numerical computation requires the number of intervals, *n*, be as large as possible for more accurate results. In addition the distribution vectors of all variables must be in the same length. The length of the mass distribution vector can be increased by the LINKMASS function whereas the distribution vectors of deflection, derivatives, and stiffness can be increased in length by the LINKSPOR function.

In Appendix E, Section 3 lists a code for running YNTERYGH function in an effort to determine the bending frequency and Southwell coefficients given by Yntema. The results and error calculations are given in Figures 47 through 58 below. Even though the curves produced by YNTERYGH overestimate the results of Yntema, they predict the changes in mass and stiffness quite well. Except for the low and very high values of *m_tip/m_root,* the errors for the hinged beam are roughly ± 5% for the coefficients. For the cantilever beam, the error is higher, about an average of ± 10%.

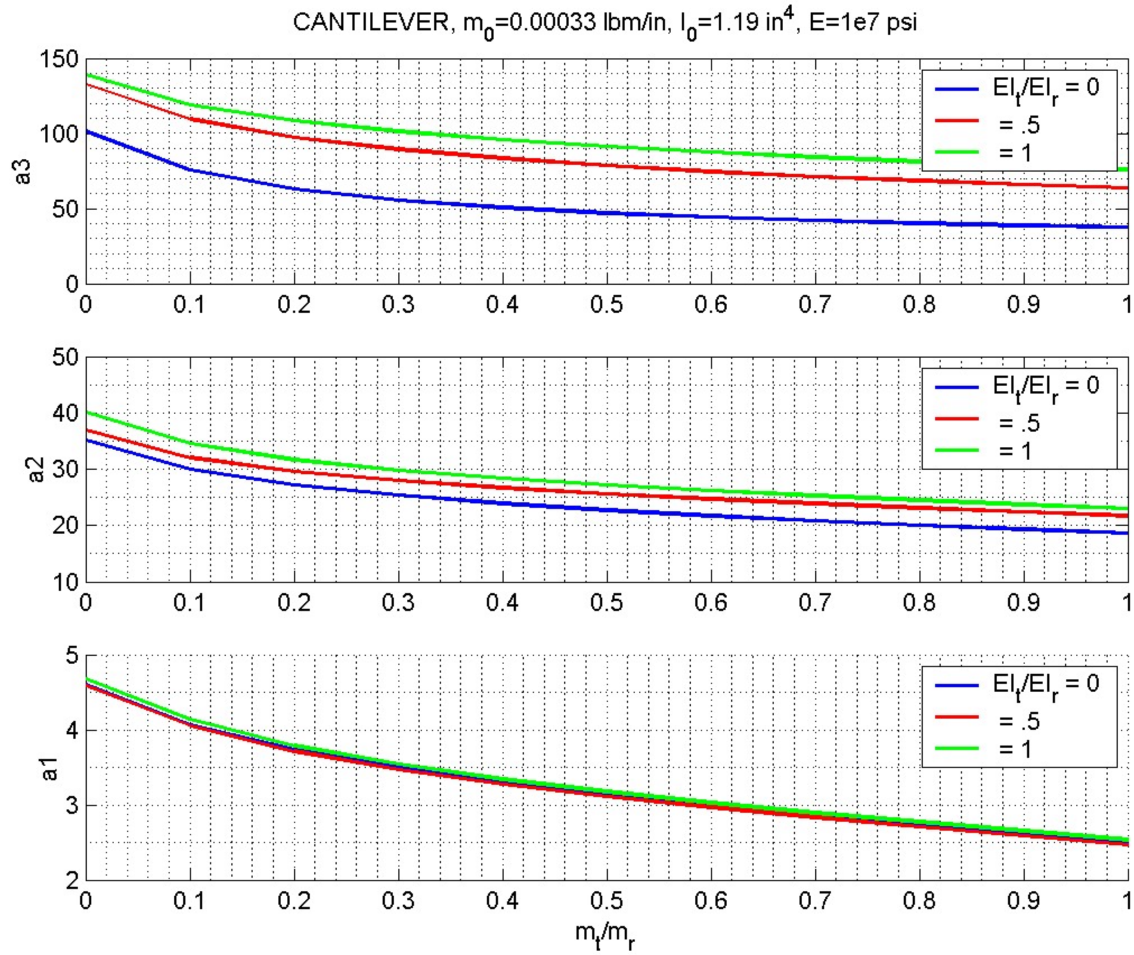



Figure 47. YNTERYGH Generated Bending Frequency Coefficients for Cantilever Beams with Linear Mass and Stiffness Distributions.

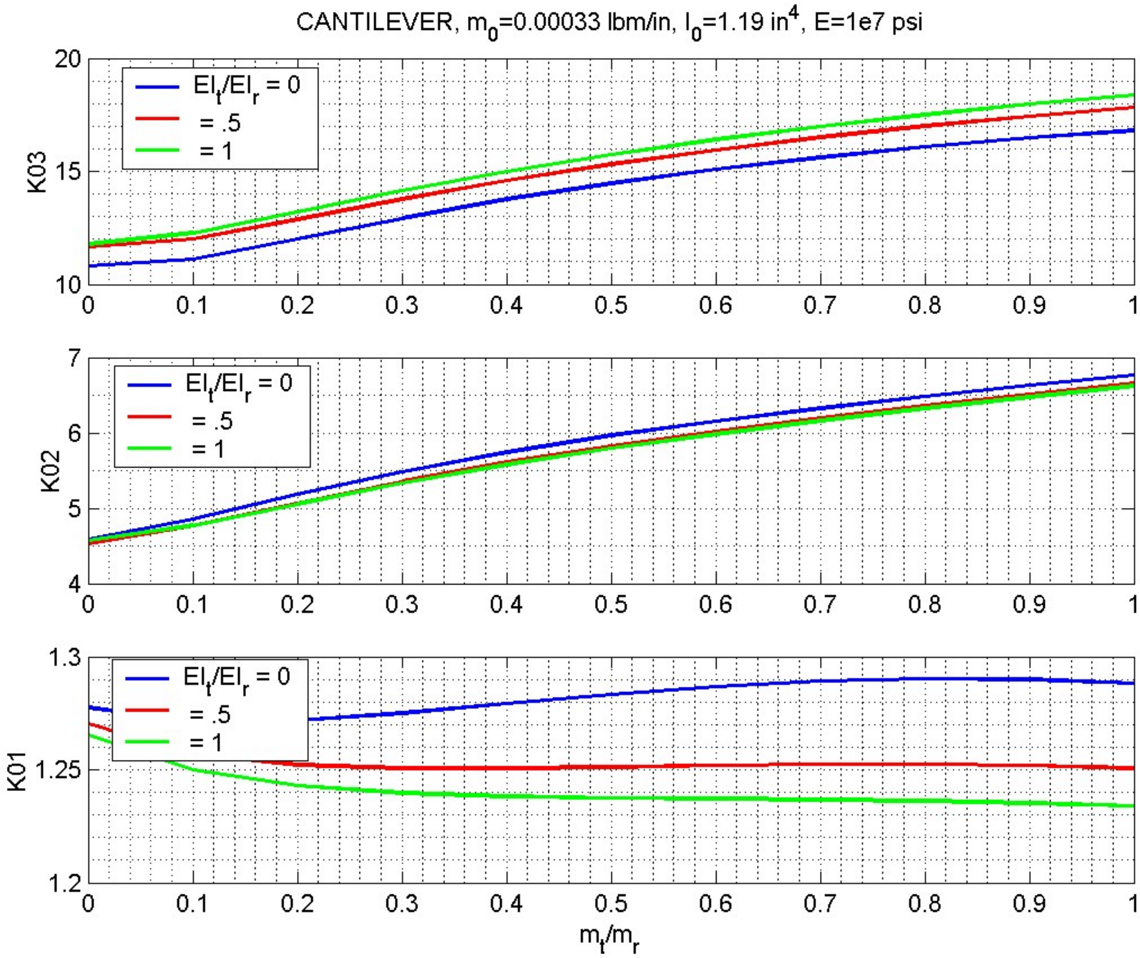


Figure 48. YNTERYGH Generated Zero-offset Southwell Coefficients for Cantilever Beams with Linear Mass and Stiffness Distributions.

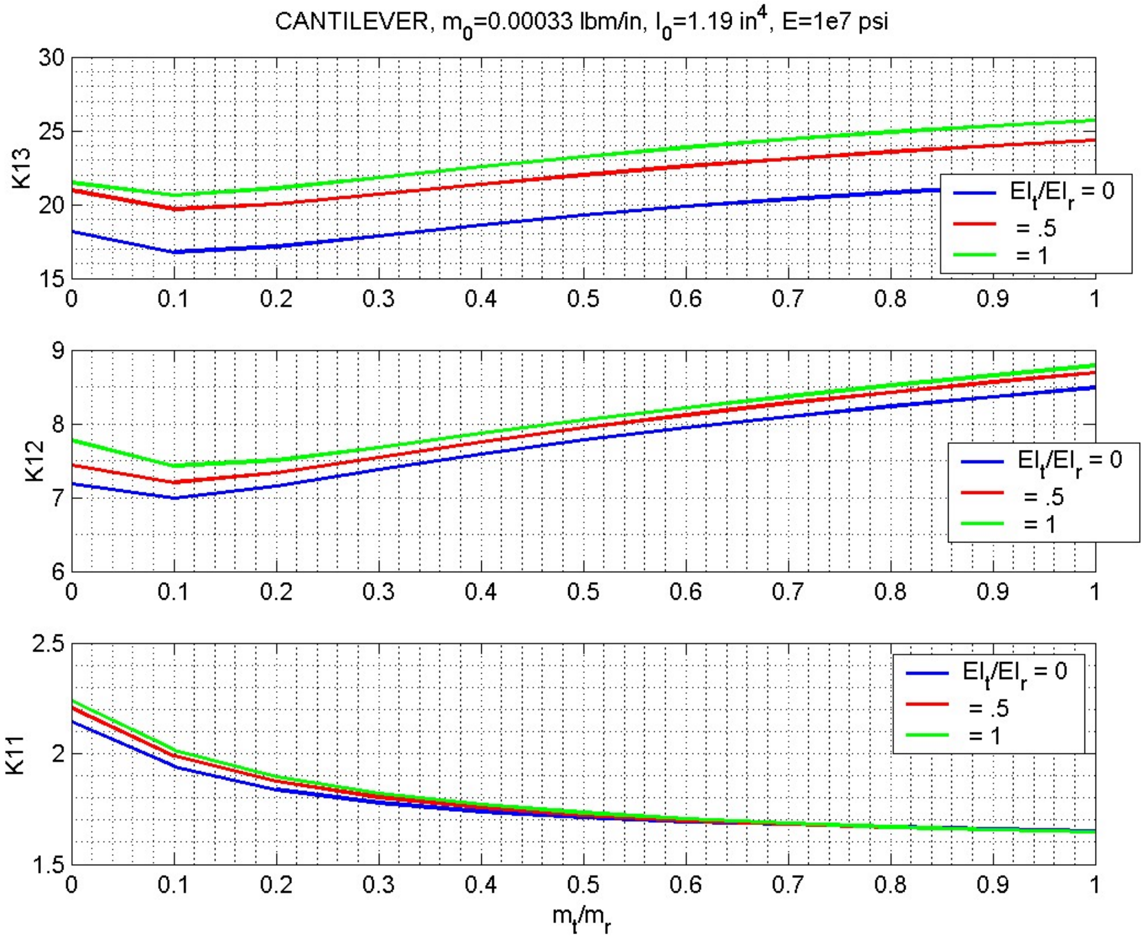



Figure 49. YNTERYGH Generated Offset-correction Factors for Southwell Coefficients for Cantilever Beams with Linear Mass and Stiffness Distributions.

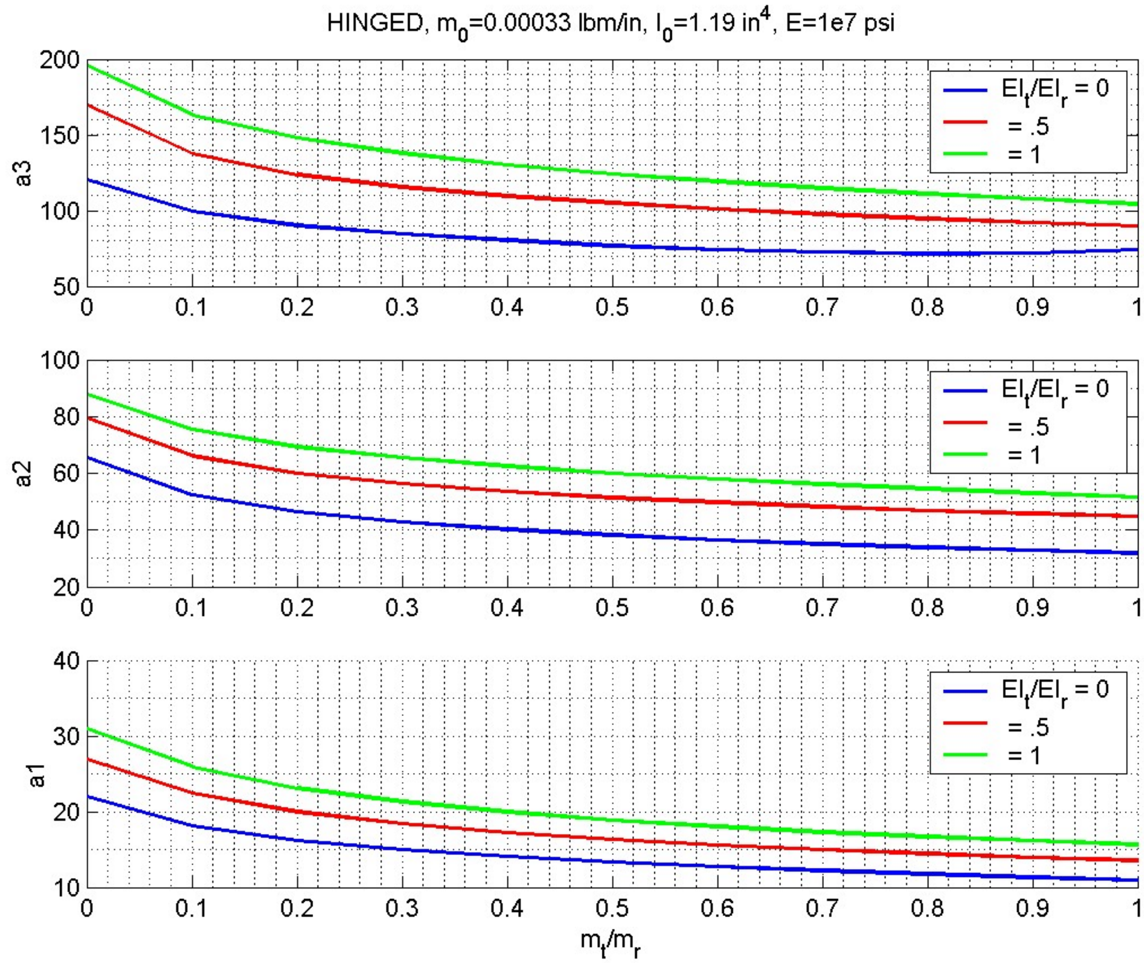



Figure 50. YNTERYGH Generated Bending Frequency Coefficients for Hinged Beams with Linear Mass and Stiffness Distributions.

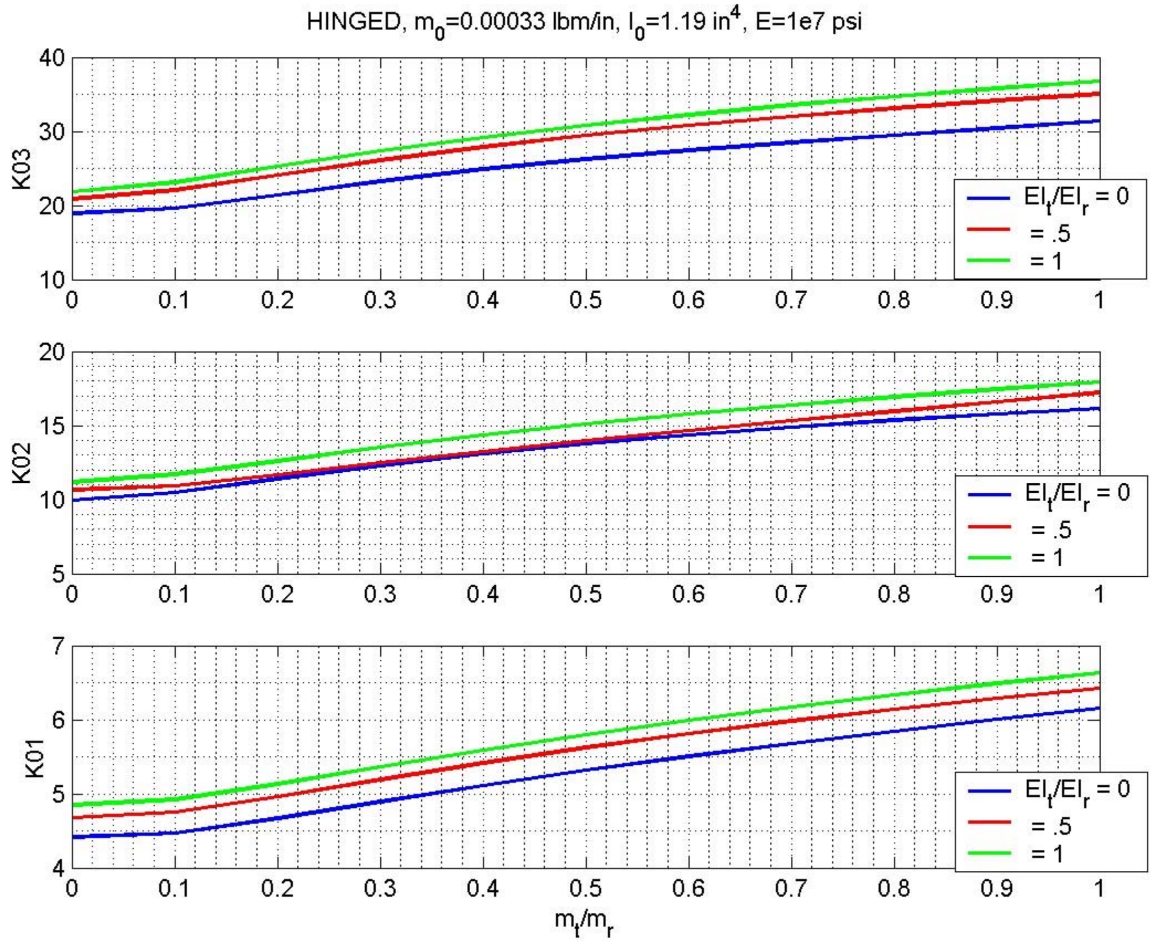



Figure 51. YNTERYGH Generated Zero-offset Southwell Coefficients for Hinged Beams with Linear Mass and Stiffness Distributions.

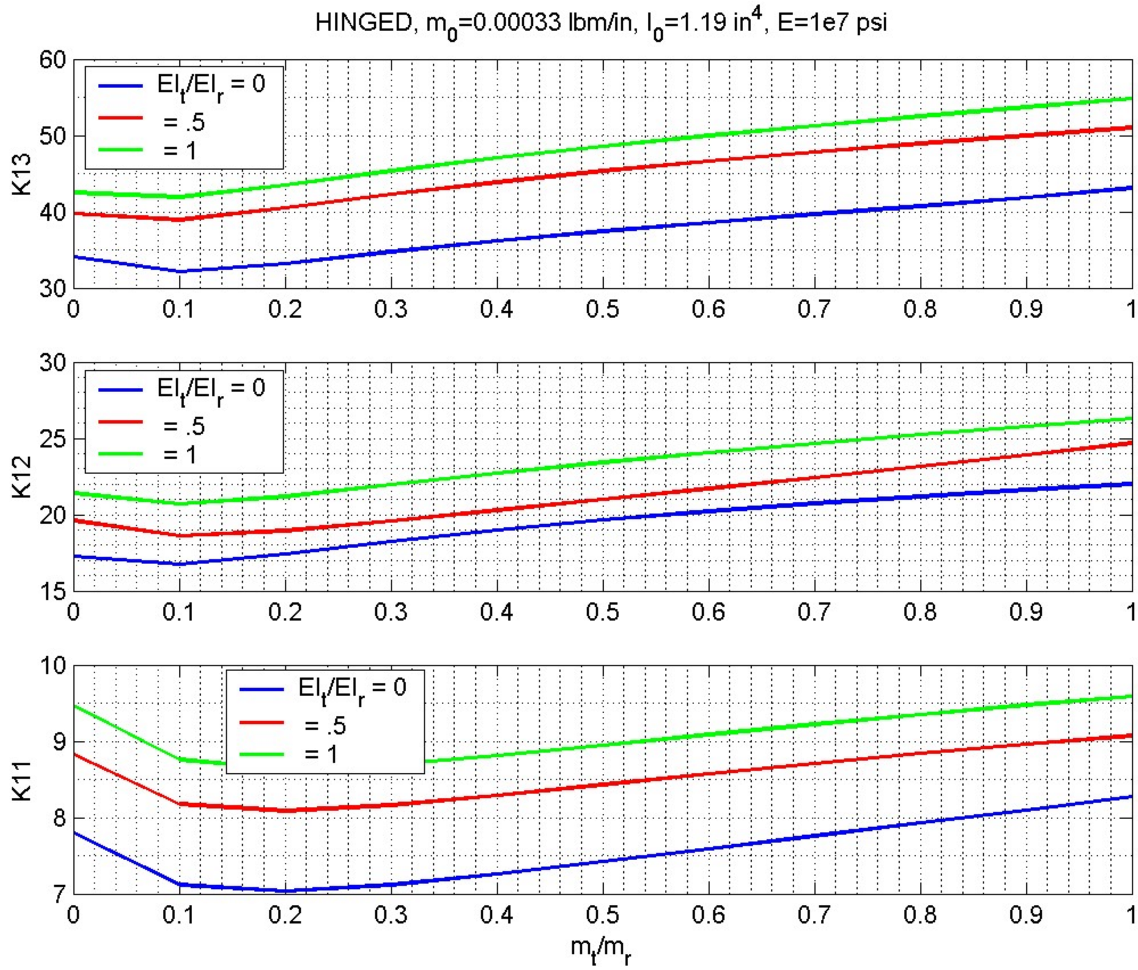


Figure 52. YNTERYGH Generated Offset-correction Factors for Southwell Coefficients for Hinged Beams with Linear Mass and Stiffness Distributions.

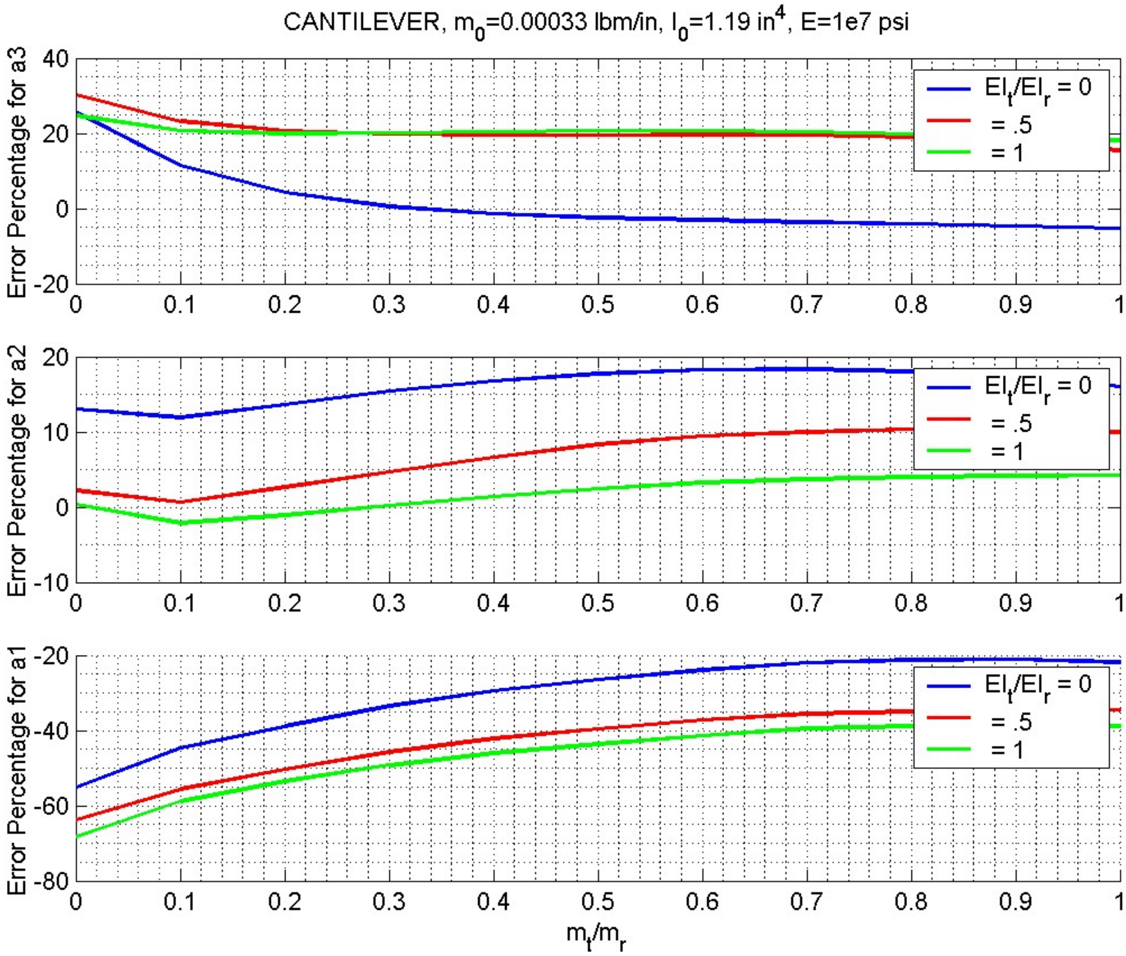



Figure 53. Error Percentages for YNTERYGH Generated Bending Frequency Coefficients for Cantilever Beams with Linear Mass and Stiffness Distributions.

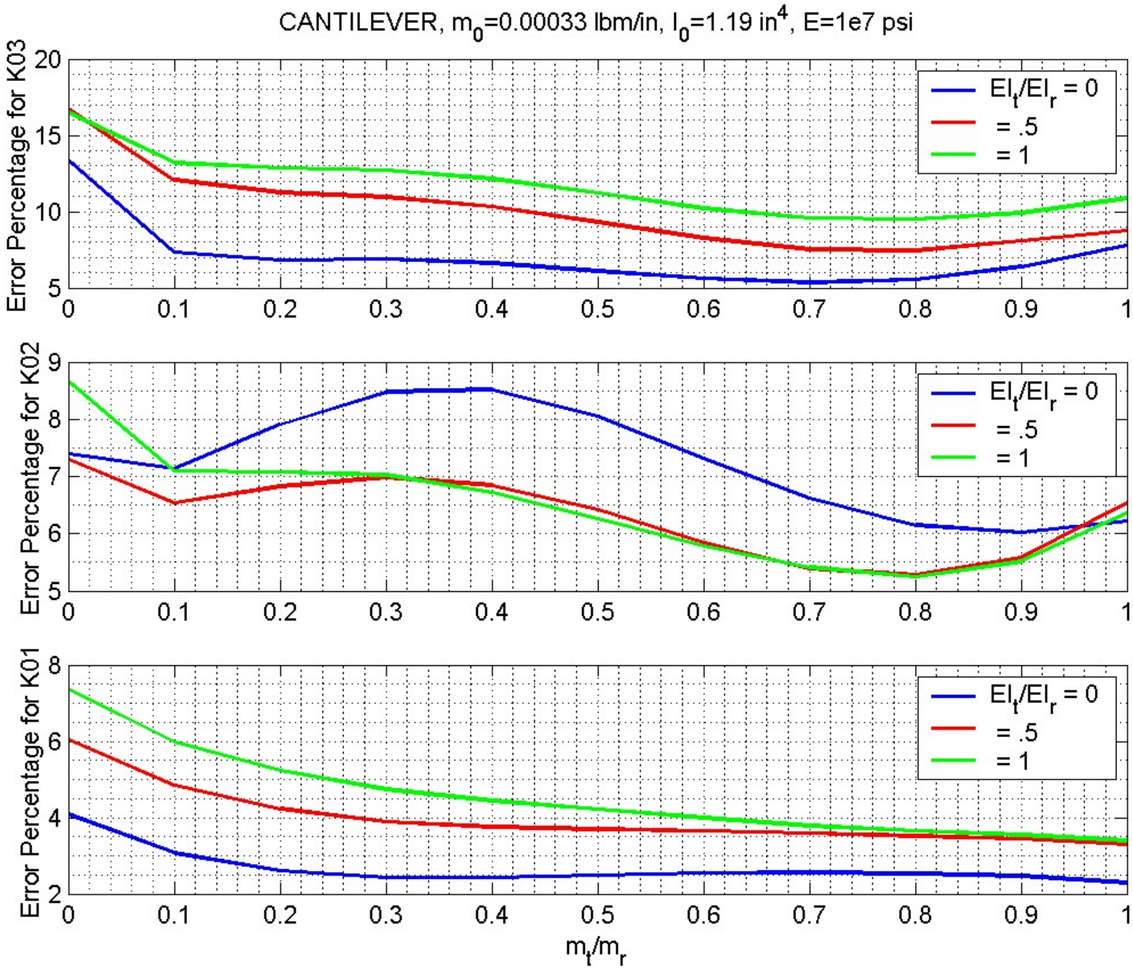



Figure 54. Error Percentages for YNTERYGH Generated Zero-offset Southwell Coefficients for Cantilever Beams with Linear Mass and Stiffness Distributions.

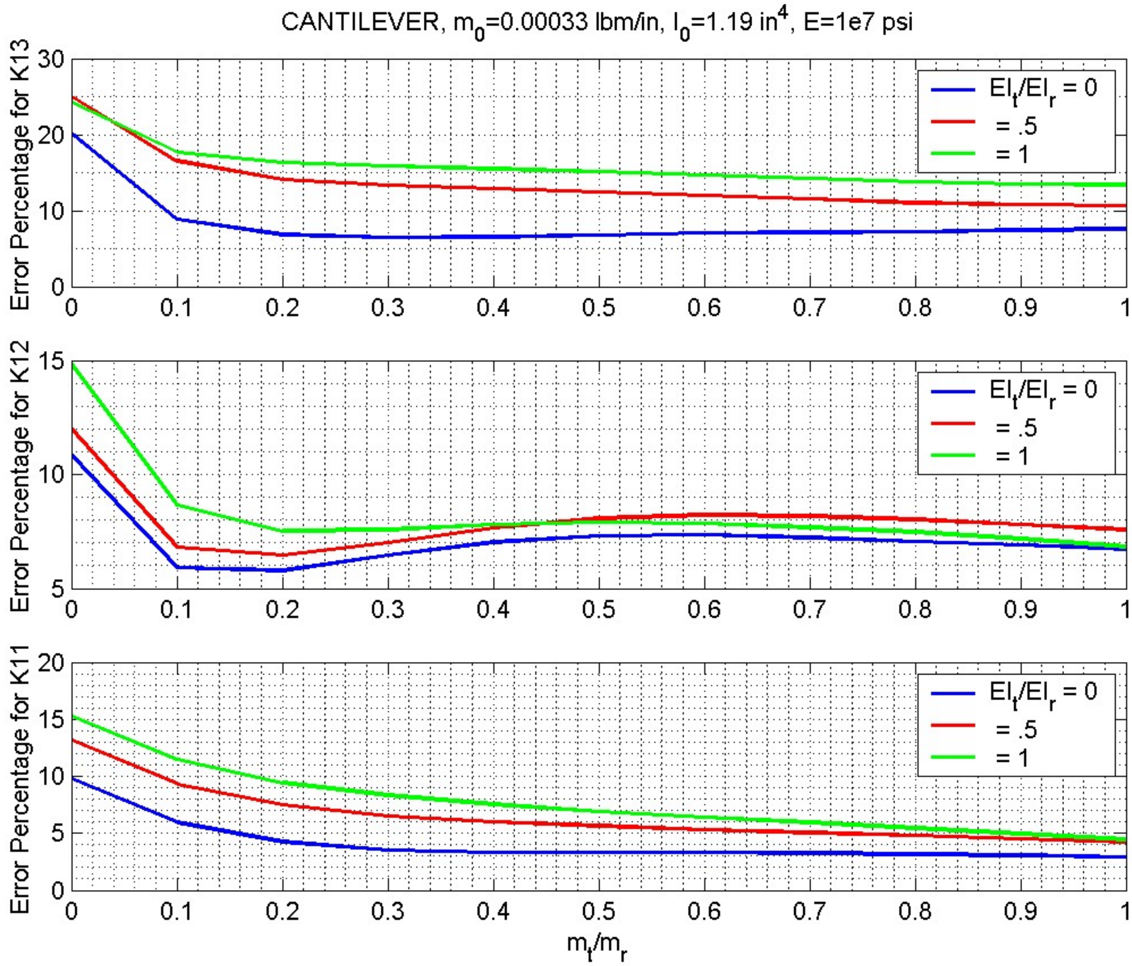



Figure 55. Error Percentages for YNTERYGH Generated Offset-correction Factors for Southwell Coefficients for Cantilever Beams with Linear Mass and Stiffness Distributions.

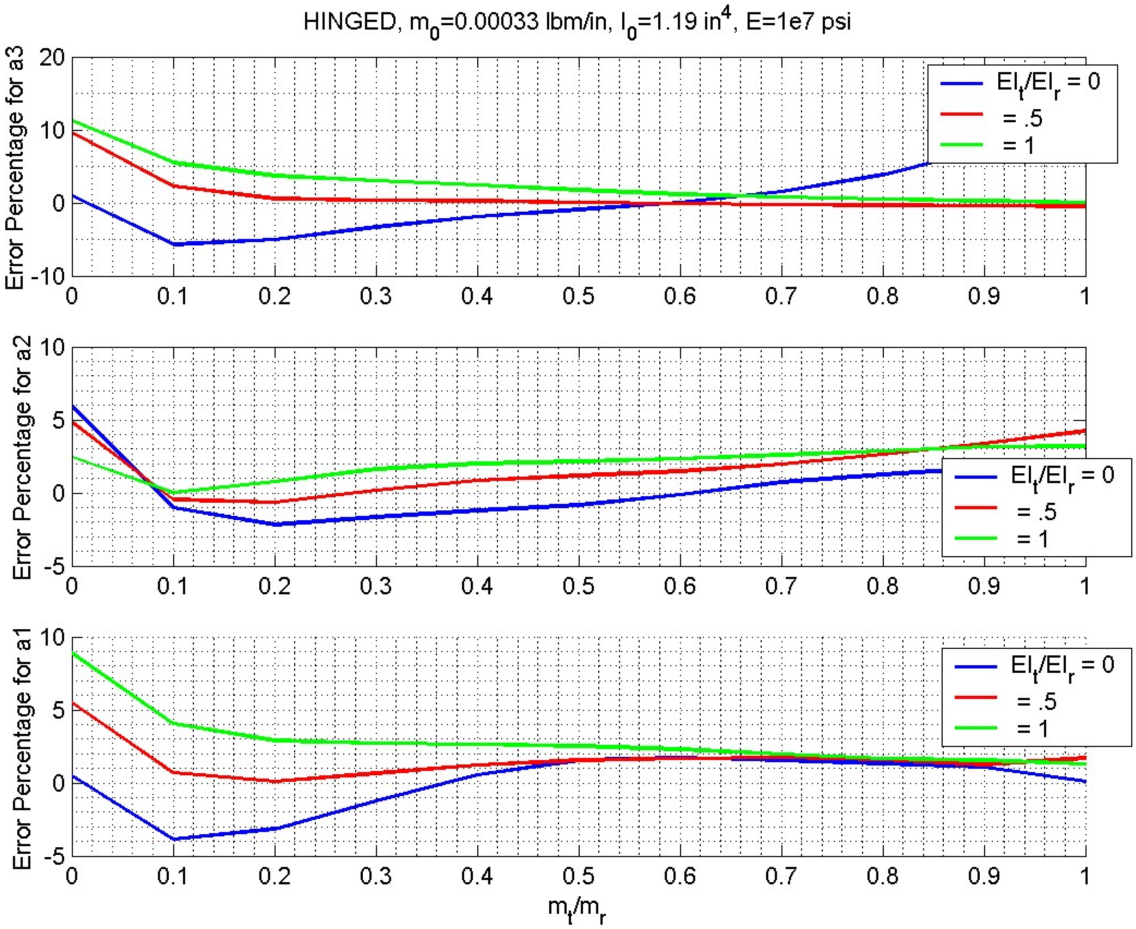



Figure 56. Error Percentages for YNTERYGH Generated Bending Frequency Coefficients for Hinged Beams with Linear Mass and Stiffness Distributions.

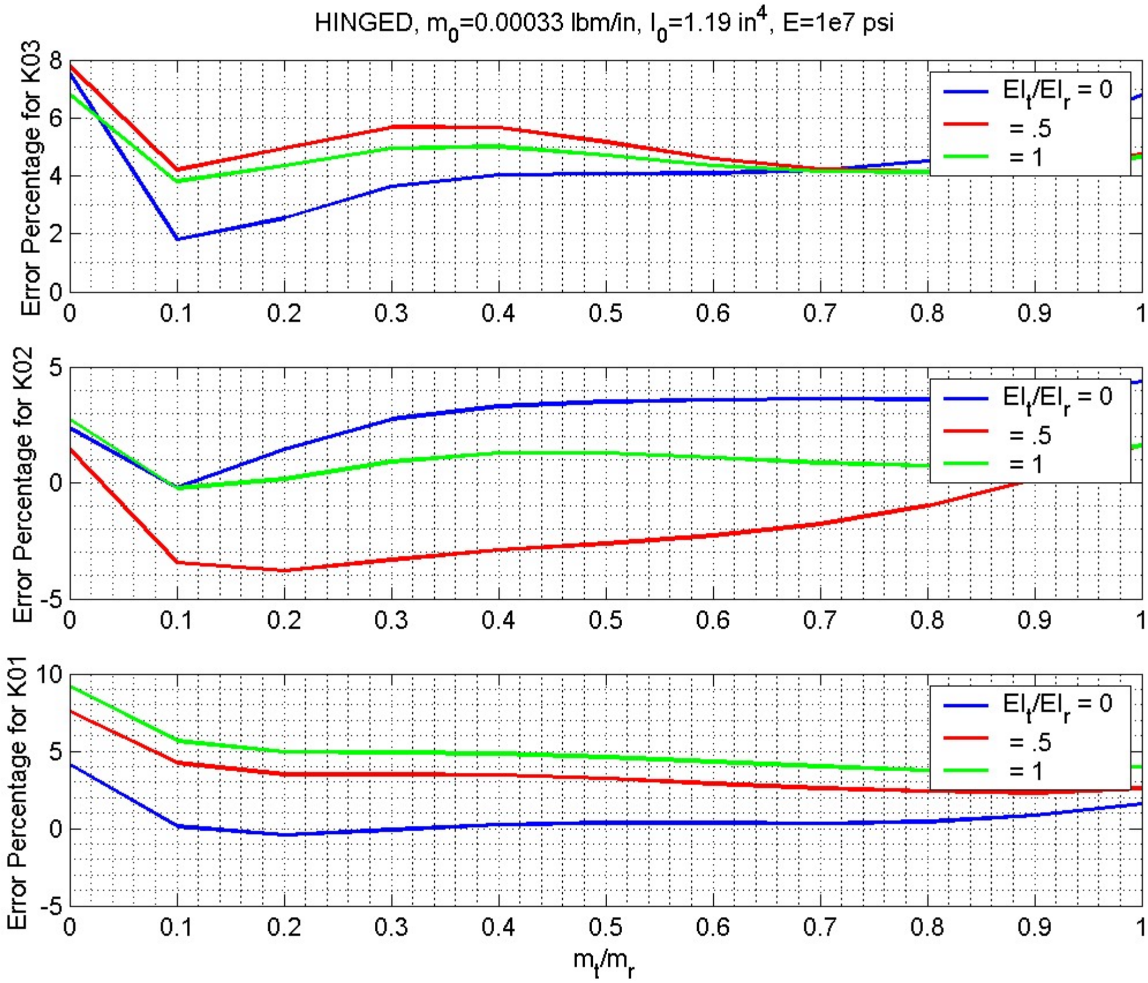



Figure 57. Error Percentages for YNTERYGH Generated Zero-offset Southwell Coefficients for Hinged Beams with Linear Mass and Stiffness Distributions.

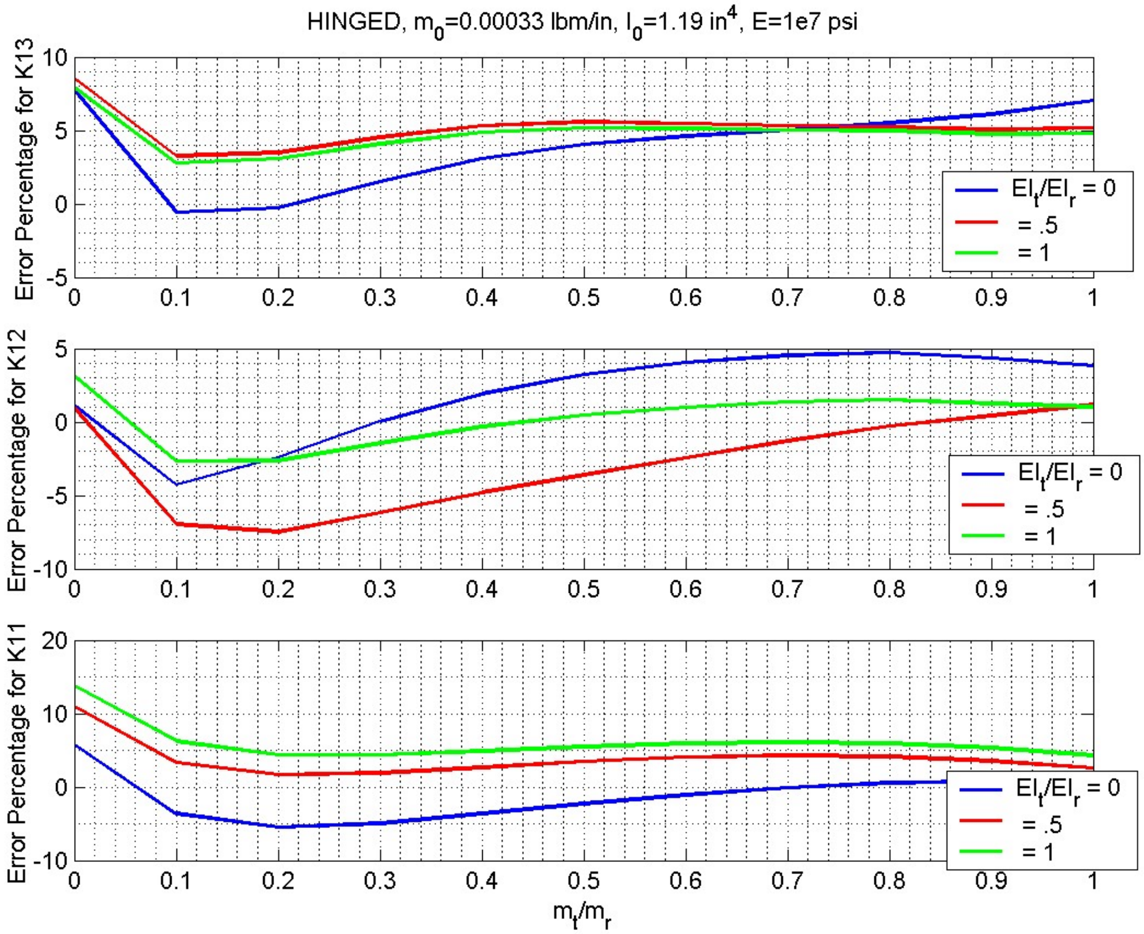



Figure 58. Error Percentages for YNTERYGH Generated Offset-correction Factors for Southwell Coefficients for Hinged Beams with Linear Mass and Stiffness Distributions.

It should be moted that the Yntema figures use the *nondimensionalized* Rayleigh equation, Equations(9)~(12), whereas the YNTERYGH function uses the *dimensional* Rayleigh equation, Equation(1). Although it has been tried to manually reproduce the original Yntema figures, the Yntema figures for the bending frequency coefficients are easy to verify but the Southwell coefficient figures are again overestimated by manual calculations.

#### c. Verification And Validation Of YNTEMOSHNR Function

Although a linear interpolation method has been incorporated for the mode shapes of nonrotating beams, YNTEMOSHNR function results may be accepted to be very accurate. Figures 59 and 60 demonstrate the accuracy of the YNTEMOSHNR function for a hinged and a cantilever beam respectively.

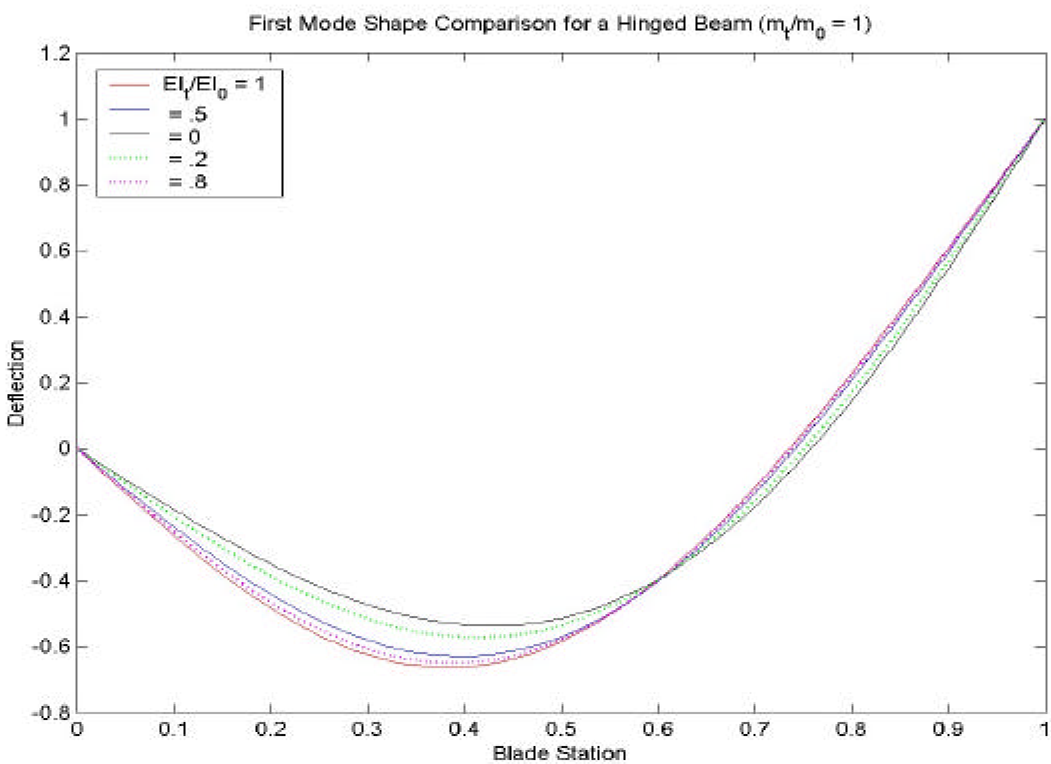


Figure 59. A Validation of YNTEMOSHNR Function for a Hinged Beam.

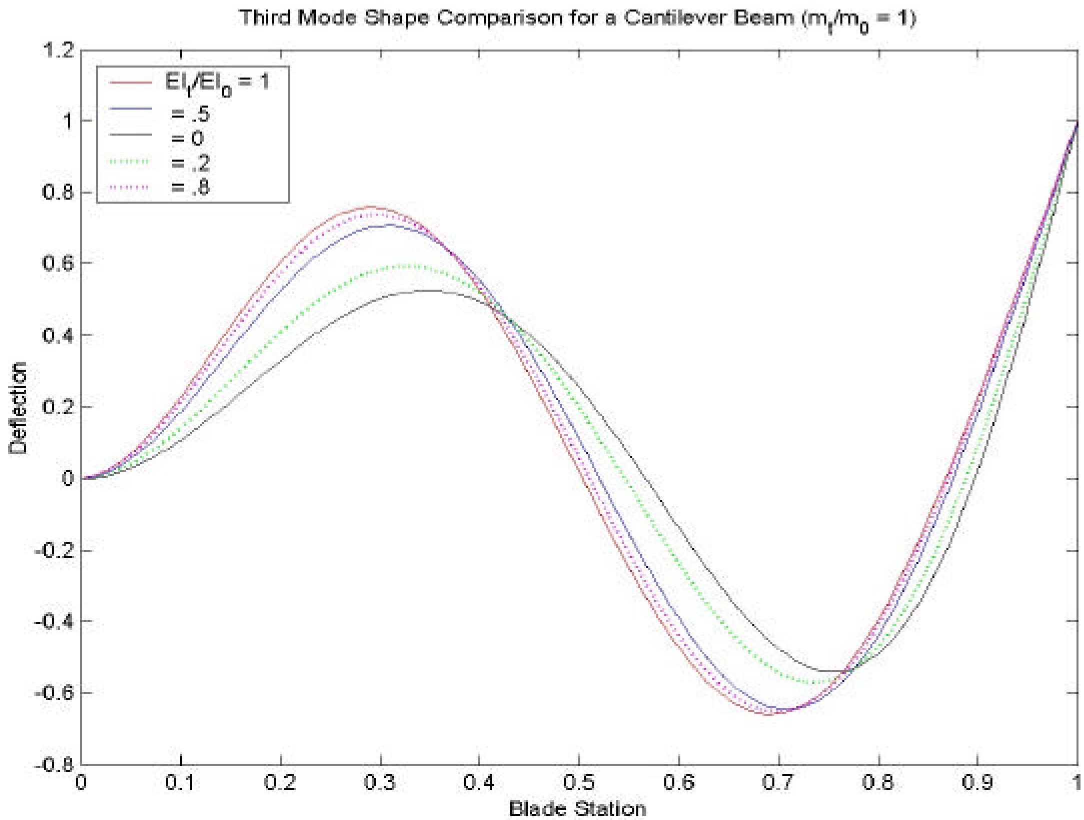



Figure 60. A Validation of YNTEMOSHNR Function for a Cantilever Beam.

Further effort could be spent on the accuracy of YNTEMOSHNR function which uses linear interpolation between {0 0.5} or {0.5 1} to get the results. A 2nd-degree curve could be fitted but this curve would not yield respectable results if $m_{tip}/m_{root}$ or $EI_{tip}/EI_{root}$ values are almost the same at intersection points. In this case, a 2nd-degree curve created by MATLAB® between three points extends beyond the logical limits. For example, a deflection value for $x$=0.7 can be greater than its values for $x$=0.5 and $x$=1. Figure 61 shows a comparison of linear and parabolic interpolations. The parabolic interpolation, which is expected to be more accurate than linear interpolation, can yield an inaccurate result especially in the shadowed area.

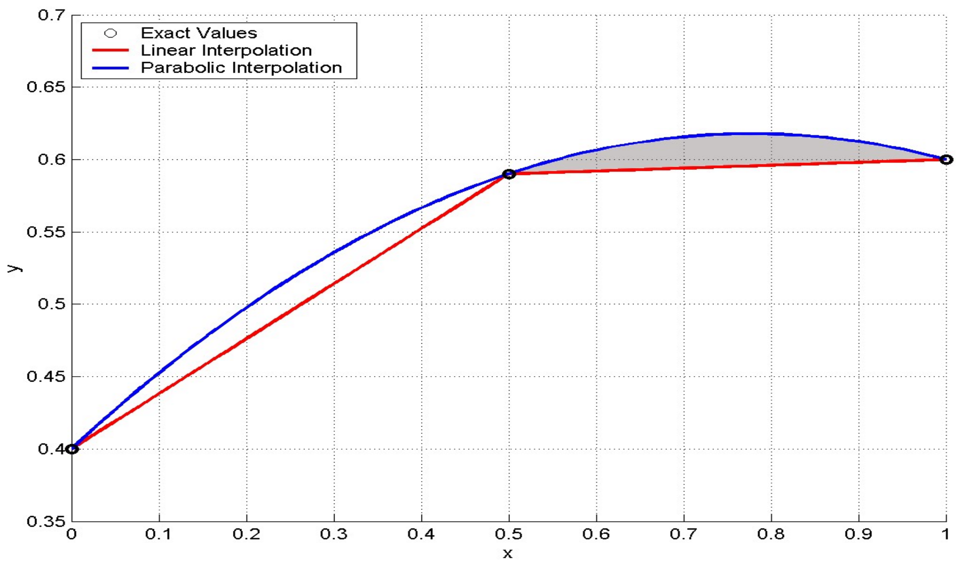



Figure 61. A Comparison of Linear and Parabolic Interpolation for the Figure 62. Nonrotating Beam Mode Shapes

### 3. Validation Of The Utility Files

#### *a. Validation Of MAKELINE Function*

The accuracy of MAKELINE function is directly proportional to the number of input data points, i.e., the length of the input vectors. Figure 62 demonstrates a validation of this function. Although depending on your intentions, this function yields very accurate results when cut-off values are not ignored and chosen carefully.

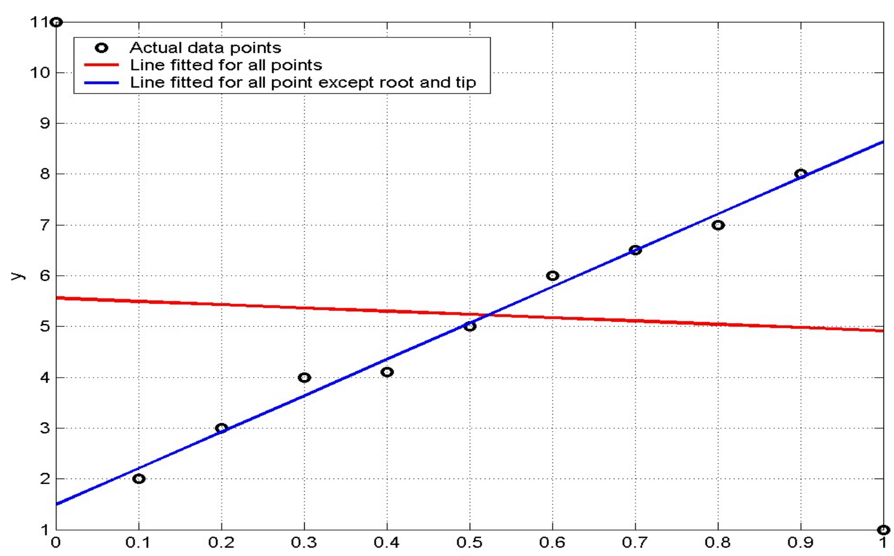



Figure 63. A Validation of the MAKELINE Function.

### *b. Validation Of LINKMASS Function*

Figure 63 validates the accuracy and usability of LINKMASS function. The data points, originally in a 1x11 vector, are redistributed into a 1x101 vector having their original abscissa values. In other words, zeros have been filled between two consecutive values in the original vector in order to rearrange the size of the vector. This demonstrates the functionality of LINKMASS function for concentrated mass distributions which need to have a different size but must not loose the original abscissa values.

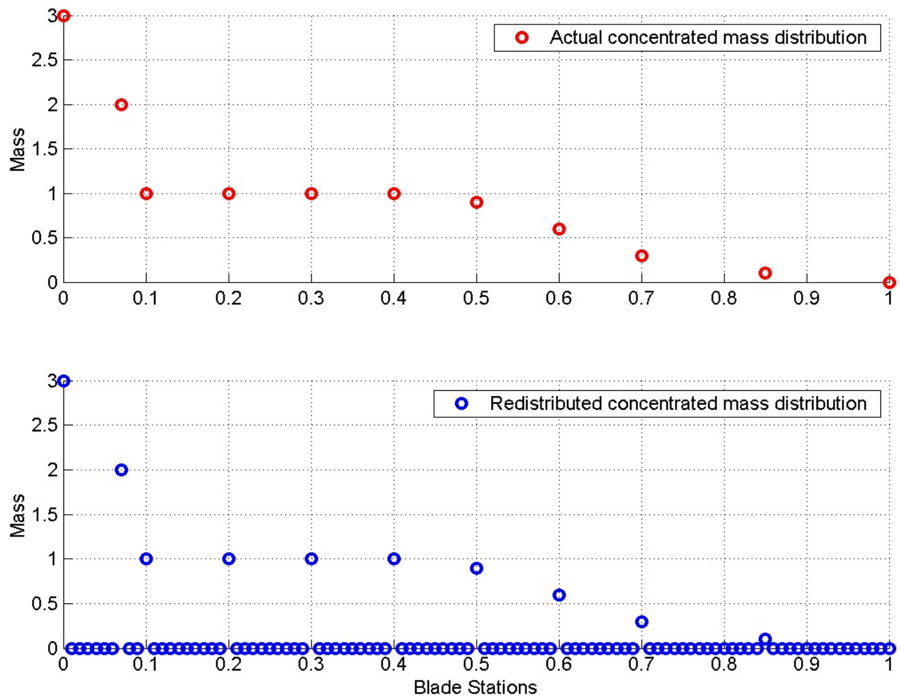



Figure 64. A Validation of the LINKMASS Function.

### *c. Validation Of LINKSPOR Function*

Figure 64 validates the accuracy and usability of LINKSPOR function. The data points, originally in a 1x11 vector, are redistributed into a 1x101 vector having their original abscissa values and the middle values are linearly added between two consecutive values in the original vector in order to rearrange the size of the vector. This demonstrates the functionality of LINKSPOR function for continuous distributions, such as stiffness distribution, which need to have a different size but must not loose the original abscissa values. Note that the values of the original vector must correspond to linear abscissa values.

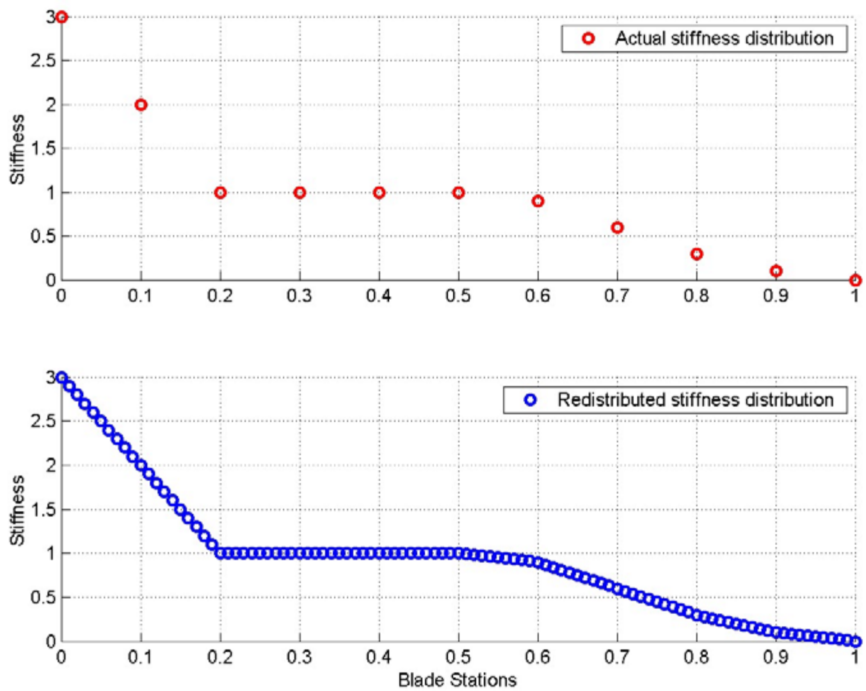



Figure 65. A Validation Of LINKSPOR Function.

### 4. Verification And Validation Of The Graphical User Interface (GUI)

The YNTEMAGUI.M file has been debugged to work properly. It has also been checked for syntax errors and no error could be found.

The GUI has been executed for several times and its results have been compared with the results acquired from the YNTEMA.M function. All results are found to be identical. Since the GUI file uses the YNTEMAG.M function file which is essentially identical to YNTEMA.M, the results will always be the same.

THIS PAGE INTENTIONALLY LEFT BLANK

# IV. CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE RESEARCH

## A. CONCLUDING REMARKS

A MATLAB® code, YNTEMA, which uses the method of Yntema [Ref. 1], has been created for a rapid estimation of bending frequencies of rotating and nonrotating blades. The Rayleigh energy approach of Yntema utilizes nonrotating beam mode shapes instead of rotating beam mode shapes to determine the bending frequencies. This method yields reasonably accurate results for rotating hinged or cantilever beams with arbitrary stiffness and mass distributions, and uniform hinged or cantilever beams with a concentrated mass at the tip at least within the rotational speed limits of helicopter blades. The Rayleigh energy approach of Yntema has the advantage of improved accuracy and wider applicability over other simplified approaches and the advantage of simplicity and flexibility over more exact approaches.

The bending frequency and Southwell coefficient plots given by Yntema have been digitized to be referred by several MATLAB® functions written for computations. Reproduction of the bending frequency and Southwell coefficient plots by the computer code show that these coefficients have been accurately transferred into a digital medium. The time required to calculate bending frequencies of rotating and nonrotating blades has been reduced significantly, which is needed for the preliminary design process. This software tries to fulfill the necessity of nearly five decades of transferring Yntema method into a computer medium where quick and reasonably accurate results can be achieved.

In case of nonlinear mass or stiffness distributions and lack of a mode shape result, several of utility functions have been created to linearize these distributions in the determination of approximate bending frequencies. The input and output arguments of these functions have been generalized to be used with any distribution. They should prove very useful especially in further researches on the topic of this study.

Another MATLAB® function, YNTERYGH, has been created to compute the exact Rayleigh energy equation for rotating beam bending frequencies provided that a mode shape is available. Approximate mode shapes give good results as well as the nonrotating beam bending mode shapes. This function is expected to yield results which are in error by less than three percent, except for the first mode which may be in error by as much as five percent. These errors can be corrected to give more accurate results. Nevertheless, care must be taken into account when selecting an approximate mode shape since an inappropriate mode shape can yield very rough results.

For the determination of the mode shape of a nonrotating hinged or cantilever beam without a tip mass and root offset, YNTEMOSHNR function has been created for linear mass and stiffness distributions. This function yields very accurate mode shapes which are also similar to rotating beam mode shapes.

Evaluation of the software has been made and it performs very well yielding very accurate results. The software can be used anywhere the Yntema method is used, especially for preliminary design purposes.

A detailed graphical user interface (GUI) has been produced for the YNTEMA function. The GUI provides extra ease of use and flexibility for a user. The GUI uses the same input arguments as the YNTEMA function but provides the advantages of being quicker and easier to use. The output results and plots are displayed on the same windows as the inputs and just changing one input value does not require to input all the other inputs again.

User's guides have been provided for all files and functions of the computer code as well as the graphical user interface. Detailed input and output argument forms have been tabulated and several sample runs of the software have been demonstrated.

## B. RECOMMENDATIONS FOR FUTURE RESEARCH

Although the software performs very well and generated reasonable results, it is bound by the flexibility of the original Yntema method. A further research topic could be expanding the application limits of Yntema report to incorporate more range for mass and stiffness distributions. By acquiring and utilizing more blade data from several blade manufacturing companies, the Rayleigh approach may be analyzed and developed to incorporate arbitrary concentrated mass distributions along the blade. Blade twist, which is absent in the Yntema report, can be researched about its effects on the computations and the results. Getting the mode shapes of rotating beams by Rayleigh approach can prove to be a useful task. Linearization of nonlinear distributions for optimum results can be investigated in more depth.

Also, this computer should be incorporated into the next upgrade to JANRAD (Joint Army Navy Research and Development Software for Rotorcraft) software for use with helicopter performance and stability analysis tools. For example, the YNTEMA software can be very useful in providing a good starting point for more exact iteration methods which may take up some time to yield a result for bending frequency values.

THIS PAGE INTENTIONALLY LEFT BLANK

# APPENDIX A. USER'S GUIDE FOR THE MATLAB® CODE AND GRAPHICAL USER INTERFACE

This Appendix provides a user's guide documentation for the MATLAB® codes and graphical user interface (GUI) generated for Yntema method. Appendices B through D list the MATLAB® codes for all of the files mentioned below.

## A. MATLAB® SCRIPT FILES FOR DEFAULT VALUES AND CURVE-FITTING

The MATLAB® functions require a default value file and several files containing polynomial coefficients that will be loaded onto the workspace when a function call is made. These files have the extension ".MAT". The files with the same name and with an extension ".M", reproduce these MAT files if changes are needed to default values.

### 1. YNTEDFLT.M And YNTEDFLT.MAT For Default Input Arguments

The default-value file, YNTEDFLT.MAT, provides the modulus of elasticity, the moment of inertia, the blade mass per length at root, the blade length, the operating rotor speed values and a range of rotor speed values which are loaded into workspace when YNTEMA function is called. Any of these values take the place of any corresponding input arguments missing or wrongfully specified in the function call statement.

All of the values are taken from the specifications of OH-6 helicopter. If these values need to be changed for practicality or convenience purposes, it is enough to change the values in the YNTEDFLT.M file. Then you have to run this file in a cleared workspace and save the workspace as YNTEDFLT.MAT. For example, if you have a fixed operating rotor speed value, you may want to change its corresponding variable in the default file and adjust the rotor speed range. Afterwards you will not need to input any operating rotor speed value and rotor speed range when you call the function because the default values will be used as corresponding inputs. The YNTEDFLT.MAT file must be in the same directory as the YNTEMA function file.

## 2. YNTECOEF.M And YNTECOEF.MAT For The Determination Of Frequency And Southwell Coefficients For A Given Beam

The file, YNTECOEF.M, fits a curve for each of the input values from the Yntema report curves and returns the eighth-degree polynomial coefficients for each curve. Run this file in a cleared workspace and then save the workspace as YNTECOEF.MAT which is to be referred by YNTEMA function.

Variables beginning with 'mf' are the measurements read with a 1/120-inch-ruler. Variables beginning with 'xmf' are the multipliers to convert 'mf' measurements into real measurements. Variables beginning with 'smf' are the offset values to add to real measurements so as to get the final values on the Yntema charts. Simply: Read a point on the graph with a 1/120-inch-ruler, (mf), then multiply it with an appropriate coefficient, (xmf), and add the offset value on the chart (smf). Then you have the readings for the curve. Variables beginning with 'a' and 'K' are the coefficients of the polynomials of the fitted curves. YNTECOEF.MAT file must be in the same directory as the YNTEMA function file. The MATLAB® code for YNTECOEF.M in Appendix B includes the plotting sections which are added for the verification of the file and can be disregarded totally if need be.

## 3. YNTEMOSHCOEF.M And YNTEMOSHCOEF.MAT For The Determination Of Mode Shapes Of Nonrotating Beams

The file, YNTEMOSHCOEF.M, generates the polynomial coefficients for the mode results of some characteristic nonrotating hinged/cantilever beams with linear mass and stiffness distributions and zero hinge offset. All results are taken from the Yntema Report Table III & IV. Run this file in a cleared workspace and save the workspace as YNTEMOSHCOEF.MAT which is to be referred by YNTEMOSHNR() function for the determination of mode shapes of corresponding rotating beams. YNTEMOSHCOEF.MAT file must be in the same directory as the YNTEMOSHNR function file. The MATLAB® code for YNTEMOSHCOEF.M in Appendix B includes the plotting sections which are added for the verification of the file and can be disregarded totally.

## B. MATLAB FUNCTION FILES

The MATLAB® functions below are the core of this study. They allow for the rapid calculation of frequencies. The scripts for all functions are given in Appendix C.

### 1. YNTEMA Function For Nonrotating And Rotating Beam Bending Frequencies

YNTEMA function rapidly gives the approximate bending frequencies of rotor blades by the Yntema method and can re-compute these frequencies in case the principal axis of the blade cross section is not parallel to but making an angle of attack with the plane of rotation. The Yntema method uses a Rayleigh energy approach utilizing the bending mode of nonrotating beam in the determination of the bending frequency of the rotating beam. Polynomial coefficients for the Yntema curves are read from the YNTECOEF.MAT file for the rapid estimation of the first three bending frequencies for rotating and nonrotating cantilever and hinged beams with a variable mass and stiffness distributions, as well as with root offsets from the axis of rotation. The case of rotating uniform beams with a tip mass is also included. General statement for the function call is

*[w1,w2,w3,w0,rtrSpd <,{a0,a1,a2,a3,K0,K1,K2,K3}<,{w1ksi,w2ksi,w3ksi,w0ksi}>> <,{w1ksi,w2ksi,w3ksi,w0ksi}>] = YNTEMA(flagSouth,cantHing,eoffset,blength,melast, maLen0,minert0<,'mtip',maLenT><,'Itip',minertT><,'Mt',massTip><,[rtrSpdRng <opRtrSpd>]><,'AoA',alfa>);*

or simply,

*[w1,w2,w3,w0,rtrSpd,varargout] = yntema(flagSouth,cantHing,eoffset,blength,melast, maLen0,minert0,varargin);*

where you have to input at least the first seven input arguments. You can omit all of the output arguments at once. VARARGIN may include any combination of optional inputs <,'mtip',maLenT> or <,'Itip',minertT> or <,'Mt',massTip> or <,[rtrSpdRng <opRtrSpd>]> or <,’AoA’,alfa>. You have to input **‘mtip’** *in single quotes* just before entering the *maLenT* value. The same procedure is required for ‘**Itip’**, **‘Mt’**, and **‘AoA’**. VARARGOUT may include <*yntema_coefficients*> or <*w1ksi,w2ksi,w3ksi,w0ksi*> or all five of them. *yntema_coefficients* is a cell array of only one cell containing {a0,a1,a2,a3,K0,K1,K2,K3} values. *<w1ksi,w2ksi,w3ksi,w0ksi>* returns corresponding {w1ksi, w2ksi, w3ksi, w0ksi} values. If VARARGIN is empty, the values are assumed to be equal to their root values or zero, or empty. If VARARGOUT is empty, only the original first five arguments are returned. Table 6 and 7 give the valid combinations for VARARGIN and VARARGOUT statements respectively. Note that, for safe results, the number of arguments in VARARGIN must be from zero to eight and the number of arguments in VARARGOUT must be zero, one, four, or five.

<table>
<tr><th colspan="2">VARARGIN</th></tr>
<tr><th>Valid Statement</th><th>Result</th></tr>
<tr><td>[empty]</td><td>The values are assumed to be equal to their presumed values.</td></tr>
<tr><td>'mtip',maLenT</td><td>Only maLenT value is assigned as the mass per unit length at the tip. The others are presumed.</td></tr>
<tr><td>'Itip',minertT</td><td>Only minertT value is assigned as the moment of inertia at the tip. The others are presumed.</td></tr>
<tr><td>'Mt',massTip</td><td>Only massTip value is assigned as the concentrated tip mass. The others are presumed.</td></tr>
<tr><td>[rtrSpdRng]</td><td>Only rtrSpdRng vector is assigned as the rotational speed range. The others are presumed.</td></tr>
<tr><td>[[rtrSpdRng] opRtrSpd]</td><td>Only rtrSpdRng vector and opRtrSpd values are assigned as the rotational speed range and the operating rotor speed. The others are presumed.</td></tr>
<tr><td>’AoA’,alfa</td><td>Only alfa value is assigned as the angle of attack of the blade. The others are presumed.</td></tr>
<tr><td>Any combination of above.</td><td>Only the specified values are assigned to their respective variables. The others are presumed.</td></tr>
</table>

Table 6. Valid Inputs for VARARGIN of the YNTEMA Function.

| VARARGOUT | |
|---|---|
| **Valid Statement** | **Result** |
| [*empty*] | [*empty*] |
| ,coeffs | {*a0,a1,a2,a3,K0,K1,K2,K3,K00,K01,K02,K03,Kh10,Kh11, Kh12,Kh13*} is returned in a cell array of one cell.<br>'coeffs(1)' returns '[*a0*]'.<br>'coeffs{1}' returns the value of *a0*. |
| ,w1k,w2k,w3k,w0k | {*w1ksi,w2ksi,w3ksi,w0ksi*} is returned in a cell array of four cells.<br>'w1k' returns the value of *w1ksi*. |
| ,coeffs,w1k,w2k,w3k,w0k | {{*a0,a1,a2,a3,K0,K1,K2,K3*},*w1ksi,w2ksi,w3ksi,w0ksi*} is returned in a cell array of five cells.<br>'coeffs{1}' returns the value of *a0*.<br>'w1k' returns the value of *w1ksi*. |

Table 7. Valid Inputs for VARARGOUT of the YNTEMA Function.

The descriptions and limitations of input arguments are given in Table 8. It is important to input these arguments according to their limitations and units. Otherwise, the function will probably result in an early termination or an infinite loop; at the very best, even if you get an non-terminated result, you should not count on this result. If you need to incorporate the angle-of-attack corrections, you should input a value for the angle of attack. If you have a nonlinear mass or stiffness distribution, *do not* use the YNTEMA function with the root and tip values you have. Instead, try one of the utility functions described in the next section and linearize the distributions. Then, you can use YNTEMA function with new root and tip values for approximate results.

The descriptions of output arguments are given in Table 9. The rotational speed range is in *revolutions per minute* (RPM) and the frequencies are in *cycles per minute* (CPM) as for the case of Southwell plots. Note that you can use three or four output arguments to get only the bending frequencies provided that you follow the sequence of output arguments. In this case, disregard the error message about the number of output arguments. As a reminder, you can use any variable name for an output argument.

| INPUT ARGUMENTS FOR *YNTEMA* | | | |
|---|---|---|---|
| **Name** | **Description** | **Limitations** | **Units** |
| **southFlag** | Flag for Southwell plot. | 0 = No plot.<br>1 = Southwell plot.<br>2 = Southwell plot together with w1ksi, w2ksi, w3ksi, and w0ksi.<br>3 = 3D plot for w1ksi, w2ksi, w3ksi, and w0ksi changes with respect to rotor speed and angle of attack.<br>Assumed 0 if different. | *none* |
| **cantHing** | Selection of cantilever or hinged beam.<br>**'h'** or **'H'** = hinged.<br>**'c'** or **'C'** = cantilever. | Must be in single quotes and begin with *c, C, h*, or *H*.<br>Assumed **'H'** if different. | *none* |
| **eoffset** | Root offset ratio of the blade, *e/L*.<br>If *eoffset* ≥ 1, then recomputed to be *eoffset/blength.* | 0 ≤ eoffset <blength<br>Must be scalar, numeric.<br>Assumed 0 if different. | **in**, or **in/in** |
| **blength** | Blade length. | Must be scalar, numeric, positive.<br>A default value is assigned if different. | **in** |
| **melast** | Modulus of elasticity.<br>(e.g., 1e7 for aluminum or 3e7 for steel) | Must be scalar, numeric, positive.<br>A default value is assigned if different. | **psi** |
| **maLen0** | The mass per unit length at the root. | Must be scalar, numeric, positive.<br>A default value is assigned if different. | **lbm/in** |
| **minert0** | Moment of inertia at the root. | Must be scalar, numeric, positive.<br>A default value is assigned if different. | **$in^4$** |
| **maLenT** | The mass per unit length at the tip.<br>Must follow **'mtip'**. | Must be scalar, numeric, positive, less than or equal to *maLen0.*<br>Assumed to be equal to *maLen0* if not specified. | **lbm/in** |
| **minertT** | Moment of inertia at the tip.<br>Must follow **'Itip'**. | Must be scalar, numeric, positive, less than or equal to *minert0.*<br>Assumed to be equal to *minert0* if not specified. | **$in^4$** |
| **massTip** | Concentrated tip mass.<br>Must follow **'Mt'**. | Must be scalar, numeric, positive, less than or equal to *2*maLen0*blength.*<br>Assumed 0 if not specified. | **lbm** |
| **rtrSpdRng** | Rotational Speed Range. | Must be numeric, vector (or scalar as the maximum value greater than zero).<br>Otherwise, a default value is assigned. | **RPM** |
| **opRtrSpd** | Operating Rotor Speed. | Must be input as the last element of *rtrSpdRng* and less than the previous element. Assigned a default value for the default *rtrSpdRng.*<br>Otherwise, assumed *max(rtSpdRng)*. | **RPM** |
| **alfa** | Angle of attack of the blade.<br>Must follow **'AoA'**. | Must be numeric, vector (middle value is chosen as the representative value) or scalar (range is from zero to 2*alfa).<br>Otherwise, assumed [] (empty). | **rad** |

Table 8. Input Argument Descriptions for the YNTEMA Function.

| **OUTPUT ARGUMENTS FOR *YNTEMA*** | | |
|---|---|---|
| **Name** | **Description** | **Units** |
| **w1** | $1^{st}$ mode bending frequency results. $1^{st}$ element is result for *opRtrSpd*. | **CPM** |
| **w2** | $2^{nd}$ mode bending frequency results.$1^{st}$ element is result for *opRtrSpd*. | **CPM** |
| **w3** | $3^{rd}$ mode bending frequency results. $1^{st}$ element is result for *opRtrSpd*. | **CPM** |
| **w0** | $0^{th}$ mode bending frequency results. $1^{st}$ element is result for *opRtrSpd*. | **CPM** |
| **rtrSpd** | Rotational speed range. $1^{st}$ element is *opRtrSpd*. | **RPM** |
| **a0, a1, a2, a3** | Nonrotating beam bending frequency coefficients for the zero, $1^{st}$, $2^{nd}$, and 3rd modes. | *none* |
| **K0, K1, K2, K3** | Southwell coefficients for the zero, $1^{st}$, $2^{nd}$, and 3rd modes. | *none* |
| **K00, K01, K02, K03** | Zero-offset Southwell coefficients for the zero, $1^{st}$, $2^{nd}$, and 3rd modes. | *none* |
| **Kh10, Kh11, Kh12, Kh13** | *Nondimensional* Offset-correction factors for Southwell coefficients for the zero, $1^{st}$, $2^{nd}$, and 3rd modes. | *none* |
| **w1ksi** | Bending frequency result for the first mode which incorporates angle of attack corrections. | **CPM** |
| **w2ksi** | Bending frequency result for the second mode which incorporates angle of attack corrections | **CPM** |
| **w3ksi** | Bending frequency result for the third mode which incorporates angle of attack corrections | **CPM** |
| **w0ksi** | Bending frequency result for the zero mode which incorporates angle of attack corrections | **CPM** |

Table 9. Output Argument Descriptions for the YNTEMA Function.

### 2. YNTMRYGH Function For Exact Bending Frequency Results

This function numerically calculates the bending frequencies of rotating beams from the knowledge of the mode shapes of rotating or nonrotating beams. It uses Equation(1) and numerically calculates the integrals. The method theoretically gives the exact result, but the precision of numerical computation is, as always, directly proportional to the size of input vector, *ynR*.

General statement for the function call is

*[wn<,an<,Kn><,K0n,K1n>>] = YNTERYGH(ynR<,'ninter',nInt><,'L',blength> <,'mx',mR><,'EIx',EIR><,'eofs',eoffset><,'omega',rtrSpdRng>)*

or simply,

*[wn,varargout] = ynterygh(ynR,varargin)*

The function returns one frequency output value for one mode shape input. Therefore, you need to call the function for each frequency result. This form of the function seems to be more convenient because there is only one input rather than multiple inputs. VARARGIN may be any combination of <,'ninter',nInt><,'L',blength><,'mx',mR> <,'EIx',EIR><,'eofs',eoffset><,'omega',rtrSpdRng> values. *nInt* value must be after *'ninter'* statement. Same procedure applies for all other optional inputs. VARARGOUT may be <,an> or <,an,Kn> or <,an,K0n,K1n> or <an,Kn,K0n,K1n>. Description of the input and output arguments are as follows:

- *ynR* is the mode shape to be used for calculating the bending frequency for that mode.
- *ninter* is the reqiured number of linear intervals between zero and blade length. The higher the *nInt*, the more precise the result and the longer the time elapsed. Default is 100.
- *L* is the blade length. Default is 1.
- *mx* is the mass per length distribution along the blade from the root to the tip. Default is a constant value, 0.001 lbm/in.
- *EIx* is stiffness distribution along the blade from the root to the tip. Default is a constant value, 1e7 psi-in$^4$.
- *eofs* is the root offset. Default is 0.
- *omega* is the rotational speed range. Default is [0:5:700] RPM.
- *wn* is the bending frequency result in cycles per minute (CPM).
- *an* is the nonrotating beam frequency coefficient.
- *Kn* is the Southwell coefficient.
- *K0n* is the zero-offset Southwell coefficient.
- *K1n* is the *dimensional* offset-correction factor for Southwell coefficient.

Table 10 gives the input argument forms that can be specified and their meanings.

<table>
<tr><th colspan="5">INPUT ARGUMENTS FOR YNTERYGH</th></tr>
<tr><th rowspan="3">Name</th><th rowspan="3">Limit</th><th colspan="3">Forms and Assumptions</th></tr>
<tr><th colspan="2">Form or</th><th rowspan="2">Assumed As</th></tr>
<tr><th>Row</th><th>Col.</th></tr>
<tr><td rowspan="13">ynR</td><td rowspan="9">Numeric</td><td>1</td><td>1</td><td>Constant deflection. Derivatives will be zero.</td></tr>
<tr><td>1</td><td>>1</td><td>Polynomial coefficients for deflection. Derivatives will be computed from this polynomial.</td></tr>
<tr><td>2</td><td>1</td><td>Constant deflection and 1st derivative. $2^{nd}$ derivative will be zero.</td></tr>
<tr><td>2</td><td>>1</td><td>Polynomial coefficients for deflection and $1^{st}$ derivative. $2^{nd}$ derivatives will be computed from $1^{st}$ derivative.</td></tr>
<tr><td>3</td><td>1</td><td>Constant deflection, $1^{st}$ and $2^{nd}$ derivatives.</td></tr>
<tr><td>3</td><td>>1</td><td>Polynomial coefficients for deflection, $1^{st}$ and $2^{nd}$ derivatives.</td></tr>
<tr><td>>3</td><td>1</td><td>Real deflection values. Derivatives will be computed from the curve fitted for real deflection values.</td></tr>
<tr><td>>3</td><td>2</td><td>Real deflection and $1^{st}$ derivative values. $2^{nd}$ derivative will be computed from the curve fitted for real $1^{st}$ derivative values.</td></tr>
<tr><td>>3</td><td>>= 3</td><td>Real deflection, $1^{st}$ and $2^{nd}$ derivative values. Fourth and higher numbered columns will be disregarded.</td></tr>
<tr><td rowspan="3">Cell array of strings</td><td colspan="2">Only 1 cell</td><td>The value in the cell is read as a MATLAB® expression for deflection, such as ‘sin(x)*x^2’. Derivatives are computed from this MATLAB® expression. The expression in the cell must have a valid MATLAB® form and use only one variable, such as x.</td></tr>
<tr><td colspan="2">Two cells</td><td>The values in the cells are read as MATLAB® expressions for deflection and $1^{st}$ derivative. $2^{nd}$ derivative is computed from the expression for $1^{st}$ derivative. The expressions in the cells must have a valid MATLAB® form and use only one variable, such as x.</td></tr>
<tr><td colspan="2">3 or more cells</td><td>The values in the first three cells are read as MATLAB® expressions for deflection, $1^{st}$ and $2^{nd}$ derivatives. The expressions in the cells must have a valid MATLAB® form and use only one variable, such as x. The fourth and higher numbered cells are disregarded.</td></tr>
<tr><td>Character array</td><td colspan="2">Any number of rows and columns</td><td>The character values are merged line-by-line into a cell array of one cell. This character is assumed as a MATLAB® expression for deflection. Derivatives are computed from this expression. The expression in the cell must have a valid MATLAB® form and use only one variable, such as x.</td></tr>
<tr><td rowspan="4">mx<br>and<br>EIx</td><td rowspan="3">Numeric</td><td colspan="2">scalar</td><td>Constant mass or stiffness distribution.</td></tr>
<tr><td>1</td><td>>1</td><td>Polynomial coefficients for mass-per-length or stiffness distribution.</td></tr>
<tr><td>>1</td><td>1</td><td>Actual values for mass-per-length of stiffness distribution.</td></tr>
<tr><td>Char. array</td><td></td><td></td><td>The character array assumed as a MATLAB® expression for mass-per-length or stiffness distribution. The expression in the array must have a valid MATLAB® form and use only one variable, such as x.</td></tr>
<tr><td rowspan="3"><b>eofs</b></td><td colspan="3">>=1 & <<i>L</i></td><td>Actual root offset value in inches.</td></tr>
<tr><td colspan="3"><1 & <<i>L</i></td><td>Root offset ratio. Will be multiplied with <i>L</i> to get actual root offset value.</td></tr>
<tr><td colspan="3">‘##%’</td><td>Percentage value in single quotes and have %. Actual value will be computed as eofs*L/100. Examples: ‘10%’, ‘5%’, ‘19%’, ‘1%9’, or ‘%19’</td></tr>
<tr><td>omega</td><td colspan="3">Numeric & >=0</td><td>May be scalar or vector.</td></tr>
<tr><td>L</td><td colspan="3">Numeric & >0</td><td>Must be scalar.</td></tr>
<tr><td>ninter</td><td colspan="3">Numeric & >0</td><td>Must be scalar.</td></tr>
</table>

Table 10. Input Argument Forms for the YNTERYGH Function.

### 3. YNTEMOSHNR Function For Nonrotating Beam Mode Shapes

This function gives the mode shapes of nonrotating hinged/cantilever beams by assuming the mode shapes of some characteristic nonrotating hinged or cantilever beams with linear mass and stiffness distributions. It uses the tables III&IV in the Yntema report to fit a 1st-degree curve between {zero & 0.5} or {0.5 & 1} for each given $m_{tip}/m_0$ and $EI_{tip}/EI_0$. General statement for the function call is

*[y1NR,y2NR,y3NR<,xout>] = YNTEMOSHNR(flagPlot,cantHing,mtm0,EItEI0<,xvar>)*

or simply,

*[y1NR,y2NR,y3NR,varargout]=YNTEMOSHNR(flagPlot,cantHing,mtm0,EItEI0,varargin)*

Input and output arguments are:

- *plotFlag* is 0 (zero) if no plot needed or 1 (one) if a plot is needed. Input 2 (two) or more if you want each mode shape on a different plot.
- *cantHing* is 'H' if the beam is hinged or 'C' if cantilever.
- *mtm0* is $m_{tip}/m_{root}$. It can be a scalar for linear mass distribution or it can be a vector which will be approximated by a line using the function MAKELINE() for $cutoff_{root}=cutoff_{tip}=0$.
- *EItEI0* is $EI_{tip}/EI_{root}$. It can be a scalar for linear stiffness distribution or it can be a vector which will be approximated by a line using the function MAKELINE() for $cutoff_{root}=cutoff_{tip}=0$. (You may want to see MAKELINE() function beforehand in order to get a better approximation for these values.)
- *xvar* is the range of x-values corresponding to the mass and stiffness distributions. Default is [0::1].
- *y1NR*, *y2NR*, *y3NR* are the mode shapes approximated from characteristic nonrotating beams.
- *xout* is the same as xvar.

## C. UTILITY FUNCTIONS

The functions below behave as the utility functions for the core functions of the previous section. Their scripts are provided in Appendix C.

### 1. MAKELINE Function

This function returns the polynomial coefficients of a line approximated for an input vector of any kind of values. It fits a curve of first order to a given vector of linear or nonlinear values. Polynomial coefficients are returned in a vector in comply with MATLAB® syntax and can be used with the *polyval* command of MATLAB® directly.

General statement for the function call is

*[polyCoef] = MAKELINE(inVec,inX <,cutoffr<,cutofft>>)*

or simply,

*[polyCoef] = MAKELINE(inVec,inX ,varargin)*

where the *varargin* is obvious.

Input and output arguments are described as below:

- *inVec* is a vector containing the values which are to be approximated by a first-order polynomial.
- *inX* is the corresponding x-values for *inVec* from zero to blade tip.
- *cutoffr* is the distance from the root which will be excluded from the approximation and it is optional. Default is zero. For example, you can input a *cutoffr* value if you want to exclude the root offset section of the beam from calculation because the distribution is irrelevant at that section.
- *cutofft* is the distance from the tip which will be excluded from the approximation and it is optional. Default is zero.
- *polyCoef* is a 1-by-2 vector which contains the polynomial coefficients for the approximate line.

## 2. LINKMASS Function

This function rearranges the mass values in a vector to their corresponding places in a longer vector. The middle values are assigned zero. The first and last values remain as the first and last values. For example, if you have a mass vector of length of 11 and if you want to rearrange it into a vector of length of 101, this function relocates the original mass values as the 1st, 11th, 21st,...,101st elements of new vector. Other elements are filled with zero. General statement for the function call is

*[massout <,xnew>] = LINKMASS(massinR <,xinR <,nint>>)*

or simply,

*[massout ,varargout] = LINKMASS(massinR,varargin)*

The input and output arguments are given below:

- *massinR* is a vector containing mass values at any station along the beam.
- *xinR* is a vector of actual/normalized x-values corresponding to each mass value. You can opt not to input this argument if they are linearly spaced. But you must input if they are nonlinear. Default is [0:1/(length(massinR)-1):1]. You must input zero or *zeros()* for *xinR* if you want to input a *nint* argument but x-values. Then *xinR* is defaulted and *nint* is accepted.
- *nint* is the number of intervals of the output vector which contains rearranged mass values. For the example above, *nint* would be 100. Default is 100. The lengths of *massinR* and *xinR* vectors must be equal.
- *massout* is the vector containing the rearranged mass values. If any two new x-values are the same distance from the original x-value, the first new x-value is used to relocate the mass value. For example, if the mass value is originally at x=2 and the massout vector has no x-value at 2 but at 1.9 and 2.1, the mass is relocated to x=1.9, assuming 1.9 comes before 2.1 in the *xnew* vector (type HELP MIN for more details).
- *xnew* is a vector of corresponding x-values newly arranged.

### 3. LINKSPOR Function

This function recalculates the values of the vector 'inVec' with the number of linear intervals 'nIntr'. Use this function when you have *n* number of values but need more than *n* number of values. Unlike the LINKMASS function which fills zeros between two consecutive values, LINKSPOR function linearly fills the interval between two consecutive values of 'inVec' with a number of values corresponding to 'nIntr'. For example, if you have 11 values (which correspond to a linear interval, x=0:bLen/10:bLen) but need 101 values (nIntr=100), then you will get a vector of length(outVec)=101 corresponding to an interval of x=0:bLen/100:bLen. The middle values of 'outVec' will correspond to a line fitted between two consecutive 'inVec' values. 'inVec' values are assumed to be 'y' values corresponding to a linear interval of x-values between 0 and 1. First value of 'inVec' corresponds to x=0, and its last value corresponds to x=bLen in the computations. General statement of the function call is

*[outVec <,outOrd <,outVecX>>] = LINKSPOR(inVec,nIntr)*

- *bLen* is the length of the blade.
- *nIntr* is the desired number of intervals for the range of x=[0:(bLen/nIntr):bLen]. It must not be less than *length(inVec)*. It might be equal to *length(inVec)*, but then the function would return the same values as of *inVec*. Use a higher *nIntr* if you need a better precision.
- *outVecX* is the range of *x* values corresponding to *outVec* values. It is optional.
- *outOrd* is the order of the polynomial which approximates the *outVec* values as precise as possible. It incorporates the number of turn-directions (CW-to-CCW or CCW-to-CW) changing along the curve.
- If only two output arguments are specified, only *outOrd* is output together with *outVec*.
- If no output arguments are specified, only the values of *outVec* are output.
- The *outVec* and *outVecX* vectors are the same size as *inVec*.

## D. GRAPHICAL USER INTERFACE (GUI)

Graphical user interface, a function file called YNTEMAGUI.M, provides the inputs to the YNTEMA function and returns the results on the same window. Its script is provided in Appendix D. The bending frequencies of rotating or nonrotating beams are displayed on the window together with their corresponding Southwell and nonrotating beam bending frequency coefficients. The plot is also created on the same window. When first run, the GUI window looks like Figure 65 below.

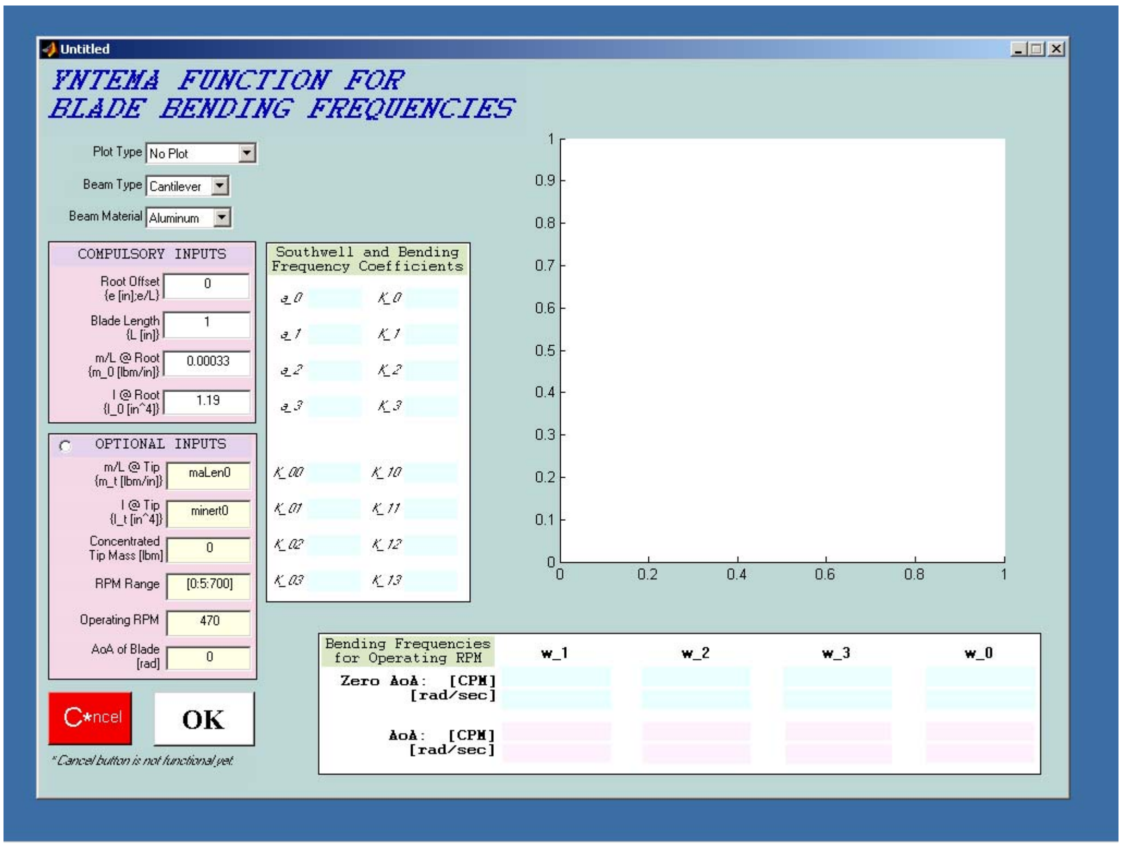



Figure 66. Graphical User Interface for YNTEMA Function at Opening.

The GUI uses the same type of input arguments as the YNTEMA function call. It actually utilizes the same MATLAB® file but with a different name, YNTEMAG.M. The only difference is that the YNTEMAG.M file does not have any diary statements. When the window is first opened, the default values for all input are displayed and the GUI maybe executed before entering any inputs but keeping the default values. There are seven compulsory inputs and six optional inputs. Three inputs are pop-up menus and the rest of inputs are edit boxes. You select one of the provided alternatives for the pop-up menus. You edit the value of the variable into the box provided for the edit boxes. One thing is you have to overwrite all of the displayed default values in the edit boxes if you want to change them at first time. Then you can change any of the inputs and keep the others untouched. To edit the optional input boxes, you need to click on the radio button first (ON) which decides whether or not to use the optional values. Clicking the radio button again (OFF) will use the default values for that run and keep your values on hold. Clicking it back ON will provide your inputs for execution. After entering the inputs, clicking on *OK* push-button will start the execution. If the *OK* button stays pushed-down for some time, it means the execution is in progress. If the *OK* button is up and no results are returned, see the workspace windows whether a problem has taken place.

The inputs for the GUI are selected or typed in just like the way they are input for the YNTEMA function.

- *Plot Type:* Select the type of plot you want to display. You can select not to display any plots, select to display Southwell plot, Southwell plot with angle of attack results, and 3D plot. 3D plot is created with the *mesh* command and shows the change in bending frequency with both rotational speed and angle of attack. Default is *No Plot*.
- *Beam Type:* Select the type of beam, cantilever or hinged. Default is *Cantilever.*
- *Beam Material:* Select the type of material the beam is made of, aluminum, composite, titanium, or steel. This selection will be used to decide which modulus of elasticity value [in *psi*] to be used. *Composite* has an approximate value of modulus of elasticity. Default is *Aluminum*.
- *Root Offset:* Enter the root (or hinge) offset value for the beam. Default is zero.
- *m/L @ Root:* Enter the mass per unit length at the root of the blade in pounds per inch-square. Default is 0.00033 lbm/in.
- *Blade Length:* Enter the length of the blade in inches. Default is 200 in.

- *I @ Root:* Enter the moment of inertia at the root of the blade. Default is 1.19 $in^4$.
- *m/L @ Tip:* Enter the mass per unit length at the tip of the blade in pounds per inch-square. Default is the value equal to *m/L @ Root.*
- *I @ Tip:* Enter the moment of inertia at the tip of the blade. Default is the value equal to *I @ Root*.
- *Concentrated Tip Mass:* Enter the concentrated mass at the tip of the blade in pounds-mass [lbm]. Default is zero or empty.
- *RPM Range:* Enter the range of rotational speed in revolutions per minute [RPM]. Default is [0:5:700] (or, shortly, 700) RPM.
- *Operating RPM:* Enter the operating RPM of the blade. This must be lower than the maximum of RPM Range input.
- *AoA:* Enter the angle of attack of the blade in radians. You need to provide an *AoA* input if you have selected to plot *Southwell AoA* or *3D AoA.* Default is zero.

The outputs of the GUI are displayed on the same window as the inputs. All of the original outputs of YNTEMA function, except the rotational speed range which is an input also, are displayed in the static text boxes provided for bending frequency and Southwell coefficients and bending frequency results. Figure 14 shows a sample page of the GUI that has been executed. If you experience problems with running the GUI for the first time, try entering all the inputs and try again.

- *Bending Frequencies for Operating RPM:* These boxes display the bending frequency results in cycles-per-minute (CPM) and radians-per-second (rad/sec) corresponding to the operating rotor speed input. If the bending frequency results are corrected for angle of attack inputs, these frequencies are also displayed below the original frequency values, also in CPM and rad/sec. *K_1n* values are results from the Yntema figures, i.e., $\bar{K}_{1n}$. Note that the bending frequency result for the zero mode can be returned as empty, zero, or equal to the operating rotor speed. If the beam is cantilever, know that there is no zero-mode for cantilever beams.

- *Southwell and Bending Frequency Coefficients:* These boxes provide you with the values of all of the bending frequency and Southwell coefficients for the beam.

- *Figure:* A figure is plotted with respect to your choice of plots. The input values accepted as to call the YNTEMAG function are also displayed on the plot for comparison to the values entered.

- *OK:* This push-button starts the execution of the program and returns the results.

- *CANCEL:* This button is included to stop the execution of the program at any time.

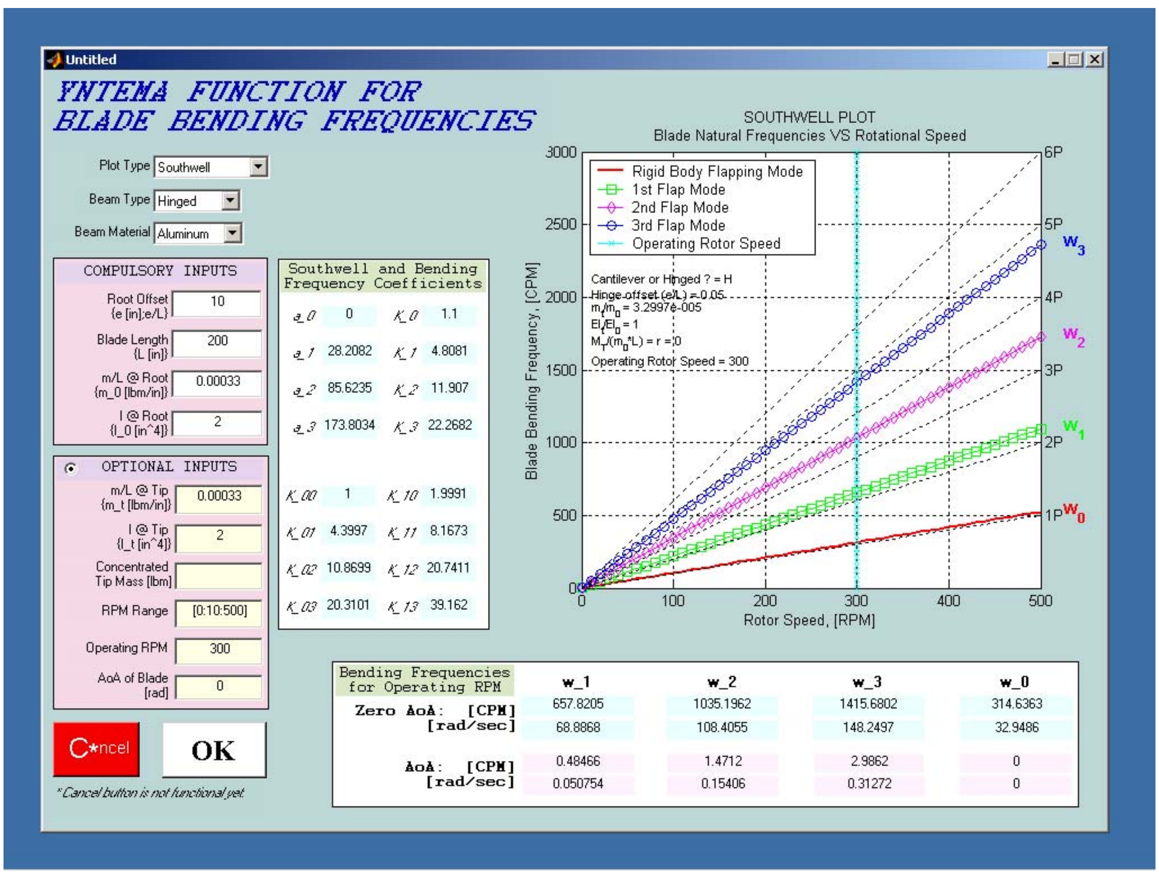



Figure 67. A Sample Window of the GUI Execution.

## E. RUN DIARY FILES

Each time YNTEMA or YNTEMAGUI is executed, a diary file of that execution is stored in a subfolder called *yntemarundiary* or *yntemaguirundiary,* respectively, in the current working directory. If this diary folder does not exist, it is created before the execution starts. If the current working directory is a medium that a subfolder can not be created directly, then the diary is waived. You can comment the lines which contain the diary statements if you need to get rid of keeping a diary file in case you encounter problems.

The current date and time are assigned to the name of the diary file. A diary file contains some input parameters, the warning messages displayed, and timing of each execution. If two consecutive executions of YNTEMA or YNTEMAGUI functions take less than a second, then the diary of second execution may be appended to the end of the diary of first execution.

## F. SAMPLE RUNS

This section provides some sample runs for the functions and GUI of the software. Additional function calls have also been listed for more detailed understanding of the input arguments.

### 1. Examples For Function Calls

#### *a. Sample Run For YNTEMA Function*

Let us assume we have a fictitious hinged beam with linear mass and stiffness distributions. The beam has a length of *250* inches and a root offset of ten percent (*25* inches or *0.10*). At the root, the mass per unit length is equal to *0.001* lbm/in and the stiffness is *2* $in^4$. At the tip, the mass per unit length is *0.0005* lbm/in and the stiffness is *1* $in^4$. The beam material is aluminum ($1 \times 10^7 \equiv$ *1e7* psi). We may select to have an angle of attack of 10 degrees (*10*pi/180* or *0.175* rad). The concentrated tip mass is assumed to be *zero* (If we selected a concentrated tip mass, the YNTEMA function would assume the beam as uniform, i.e., the mass and stiffness distributions are constant and equal to the values at the root, disregarding any inputs for the mass and stiffness distributions at the tip). The operating rotor speed is *312* RPM.

We need a Southwell plot with angle of attack corrections for a rotational speed range between *zero* and *400* RPM. We also need the bending frequencies for all the modes before and after the angle of attack corrections. We also want to see all the Southwell and nonrotating beam bending frequency coefficients.

Then the input arguments will be:

- flagSouth: 2
- cantHing: ‘H’
- eoffset: 25 or 0.10
- blength: 250
- melast: 1e7
- maLen0: 0.001
- minert0: 2
- maLenT: 0.0005
- minertT: 1
- massTip: 0 or [] or not specified at all.
- alfa: 0.175 or 10*pi/180
- [rtrSpdRng opRtrSpd]: [[0:10:400] 312]

And the corresponding function call in the workspace will be:

[w1,w2,w3,w0,coeffs,w1k,w2k,w3k,w0k]=yntema(2,’H’,25,250,1e7,0.001,2, ’mtip’,0.0005,'Itip',1,'AoA',0.175,[[0:10:400] 312]);

After the function is executed, the messages in the workspace are:

```
[w1,w2,w3,w0,rtrSpdRng,coeffs,w1k,w2k,w3k,w0k]=yntema(2,'H',25,250,1e7,0.001,2,
'mtip',0.0005,'Itip',1,'AoA',0.175,[[0:10:400] 312]);
================== START OF YNTEMA() FILE ======================
WARNING: Hinge offset input is greater than or equal to 1 (one). It is accepted
as (e), not (e/L).
         Hinge offset value is recomputed to be (e / blade_length = e/L) = 0.1
ATTENTION: You did not specify any input argument for the concentrated mass at
the tip.
           Assumed: M_t = 0.
Total Run Time is 1.062 sec.
Cantilever or Hinged ? = H
Blade Length = 250 in.
Root offset (e/L) = 0.1
m_t/m_0 = 0.5
EI_t/EI_0 = 0.5
M_T/(m_0*L) = r = 0
Angle of Attack = 0.175 rad.
Operating Rotor Speed = 312 RPM
Rigid Bending Frequency (at 312 RPM) = 336.0633 CPM
First Bending Frequency (at 312 RPM) = 780.4917 CPM
Second Bending Frequency (at 312 RPM) = 1268.8926 CPM
Third Bending Frequency (at 312 RPM) = 1773.7119 CPM
================== END OF YNTEMA() FILE ========================
```

The Southwell plot will be plotted in a new figure window as in Figure 67 below. The angle-of-attack corrections (dashed bold lines) are demonstrated along with the uncorrected results (solid lines with squares, diamonds, or circles). As can be seen the difference is very small and can be neglected for low angle of attack values.

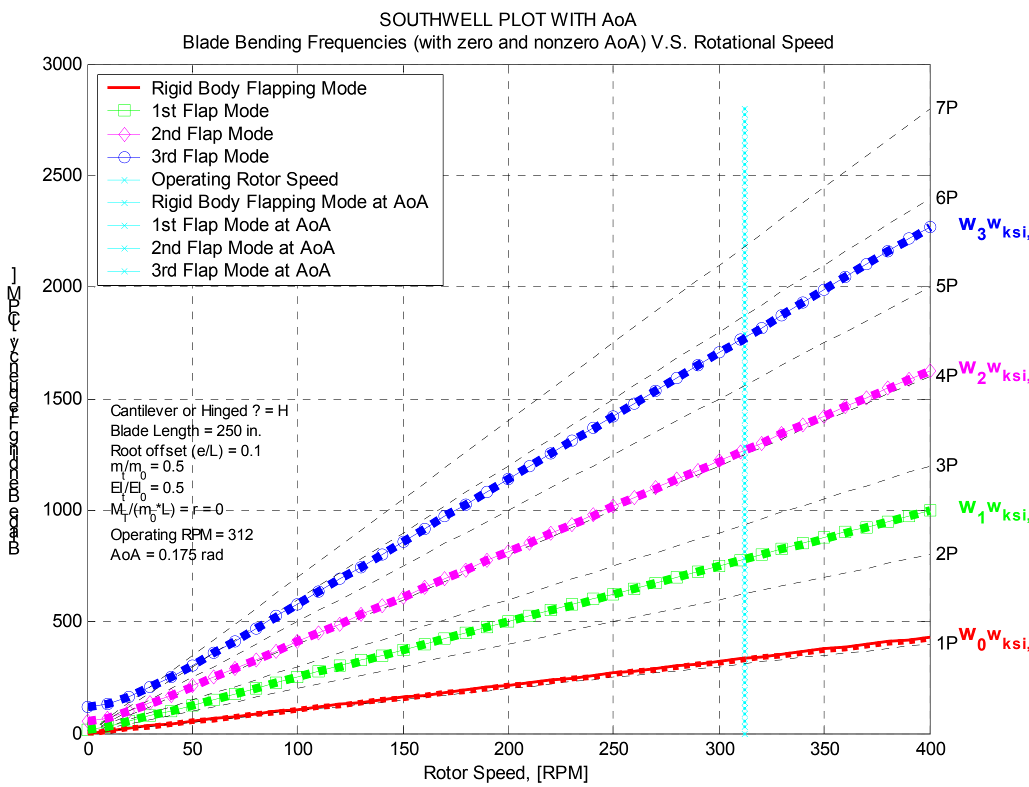



Figure 68. Southwell Plot with Angle-of-attack for the Sample Run of YNTEMA.

The bending frequencies and the coefficients will be loaded into the workspace as demonstrated in Figure 68. The first values in the bending frequency vectors will correspond to the operating rotor speed whereas the consecutive values will correspond to the rotational speed range specified in the function call. The bending frequencies corresponding to the operating rotor speed for the zero, 1$^{st}$, 2$^{nd}$, and 3$^{rd}$ modes can be read as w1(1), w2(1), w3(1), and w0(1) respectively. Likewise, the bending frequencies after angle-of-attack corrections and corresponding to the operating rotor speed for the zero, 1$^{st}$, 2$^{nd}$, and 3$^{rd}$ modes can be read as w1k(1), w2k(1), w3k(1), and w0k(1) respectively. The Southwell and nonrotating beam bending frequency coefficients are given in *coeffs* variable which is a cell array of { a0, a1, a2, a3, K0, K1, K2, K3, K00, K01, K02, K03, Kh10, Kh11, Kh12, Kh13}. For example, *coeffs{2}* (=a1) gives the value of the nonrotating beam bending frequency coefficient for the first mode and *coeffs{15}* (=Kh12) gives the value of the nondimensional offset-correction factor for the Southwell coefficient for the second mode. Note that the Southwell coefficients in the fifth through eighth elements, $K_n$, are computed from $K_{0n}$ and $\bar{K}_{1n}$ with the blade length and root offset values specified, i.e., $K_n = K_{0n} + (e/L) * \bar{K}_{1n}$.

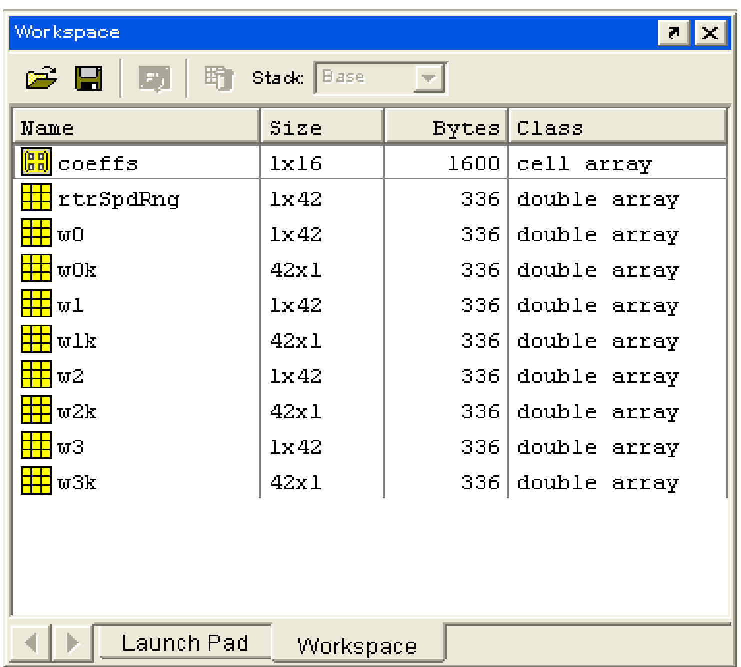



Figure 69. The Workspace Variables for the Sample Run of YNTEMA.

*rtrSpdRng* is the rotational rotor speed range identical to its corresponding input argument.

Some other valid function calls that may be experienced are listed below. Note that the first seven input arguments are mandatory and must be in the predetermined order. The eighth and subsequent input arguments are optional and can be in any order. Optional inputs are in couples, that is, a value follows its character expression (‘mtip’, ‘Itip’, ‘Mt’, ‘AoA’), except for the rotational speed range input which can be input as only a vector.

- yntema(1,'H',20.4,158.4,1e7,0.00033644,1.19);
- yntema(2,'C',20.4,158.4,1e7,0.00057000,1.5,’Mt’,0.06);
- yntema(1,'H',0.063,38.5*12,1e7,1,135,[0:5:600]);
- yntema(1,'H',0.11,15*12,1e7,0.00256,1,[[0:5:600] 432.9]);
- [w11 w21 w31 w01 omega1]=yntema(0,'H',0.10,200,3e7,0.003,1.3);
- [w13,w23,w33,w03,omega3]=yntema(1,'C',20.4,158.4,1e7,0.00033644,1.19,'mtip',0.00033);
- yntema(1,'C',20.4,158.4,1e7,0.00033644,1.19,'Itip',1);
- yntema(1,'C',20.4,158.4,1e7,0.00033644,1.19,'Mt',0.00033);
- yntema(1,'H',0,250,0.3e7,0.00005,1.19,'mtip',0,'Itip',0,'Mt',0);
- yntema(1,'H',0,250,0.3e7,0.00005,1.19,'mtip',0,'Itip',1.19,'Mt',0);
- yntema(1,'H',0,250,0.3e7,0.00005,1.19,'mtip',0,'Itip',1.19,'Mt',0.01);
- yntema(0,'C',0.5,121,1e7,0.0008,1.5,'mtip',0.0003,'Itip',0.2,'Mt',0.066,[[0:10:1000] 700]);
- [w110,w210,w310,w010,omega10]=yntema(1,'H',14,500,1e6,0.0001,1,'mtip',0.0001,'Itip',0.5,'Mt',0.01);
- [w110a,w210a,w310a,w010a,omega10a]=yntema(2,'H',14,500,1e6,0.0001,1,'mtip',0.0001,'Itip',0.5,'Mt',0.01,'AoA',0.75);
- [w110b,w210b,w310b,w010b,omega10b,coef,w1k,w2k,w3k,w0k]=yntema(3,'H',14,500,1e6,0.0001,1,'mtip',0.0001,'Itip',0.5,'AoA',0.75);

#### *b. Sample Run For YNTEMOSHNR Function*

Let us assume we have a hinged beam with linear mass and stiffness distributions. The mass ratio, $m_{tip}/m_{root}$, is *0.4* and the stiffness ratio, $EI_{tip}/EI_{root}$, is *0.7*. Since the distributions are linear, we do not need to input any abscissa values, *xvar*. And we want to see a plot for the nonrotating beam mode shape results. The function call is:

[y1NR,y2NR,y3NR]=yntemoshnr(1,'H',0.4,0.7);

*y1NR*, *y2NR*, and *y3NR* are returned as the workspace variables corresponding to the stations determined by the default value of *xvar* which is [0:0.01:1]. Figure 69 gives the mode shape results.

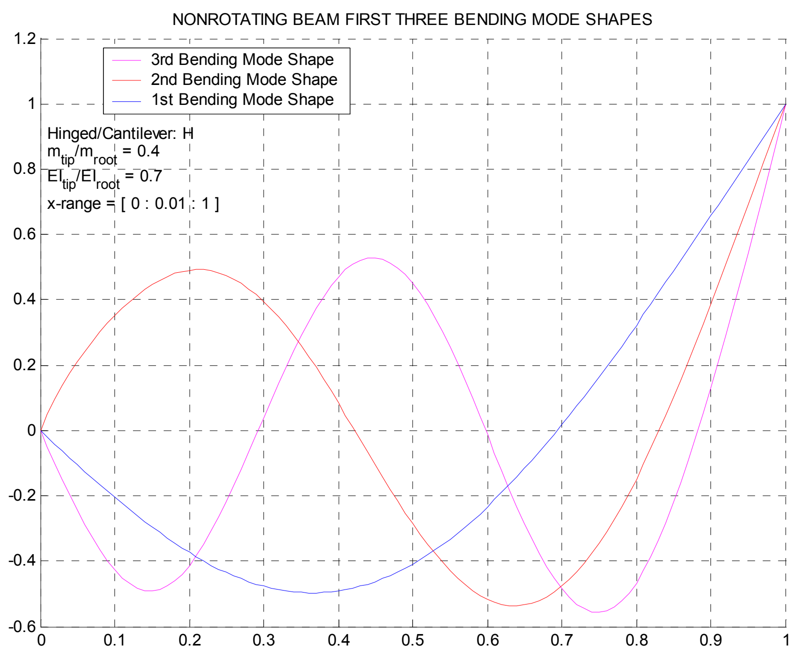



Figure 70. Mode Shape Results for the Sample Run of YNTEMOSHNR.

Some other valid function calls for YNTEMOSHNR() are listed below:

- [y1,y2,y3,xout] = yntemoshnr(0,’H’,0.2,0.8,[0:0.1:1]);
- [y1,y2,y3] = yntemoshnr(2,’C’,1,0.8,[0:0.05:1]);
- [y1] = yntemoshnr(1,’H’,0.2,0.8);
- [y1,y2] = yntemoshnr(3,’C’,0.2,0.8);

- [y1NR,y2NR,y3NR,xout]=yntemoshnr(3,'H',[0.001.*[1:-0.6/20:0.4]], [1.19e7.*[1:-0.3/20:0.7]]);
- [y1NR,y2NR,y3NR,xout]=yntemoshnr(1,'H',[0.001.*[1:-0.3/10:0.7]], [2e7.*[1:-0.5/10:0.5]],[0:0.1:1]);

### c. *Sample Run For YNTERYGH Function*

Let us assume we have the mode shape of a particular fictitious beam. The normalized deflection values of the mode shape are given in a matrix of one column. The blade has a length of *200* inches. The mass per unit length distribution is given by a row vector which corresponds to the polynomial coefficients of the distribution curve. The stiffness distribution is given by a character expression which determines the distribution curve. The blade has a hinge offset of *eight percent*. And the rotational speed range is between *zero* and *600* RPM. Hence the input arguments are:

- *y1NR* = [0.0000 -0.2248 -0.4169 -0.5455 -0.5856 -0.5226 -0.3545 -0.0929 0.2392 0.6125 1.0000]
- *massRatio* = [-0.0003 0.001]
- *stiffnessRatio* = [-1e7 2e7]
- *rotorSpeed* = [0:10:600]
- *rootOffset* = '8%'

Then the function call will be:

[w1,a1,K1,K01,K11] = YNTERYGH(y1NR.','ninter',100,'L',200,'mx',massRatio, 'EIx',stiffnessRatio,'eofs',rootOffset,'omega',rotorSpeed);

The messages displayed on the monitor are:

```
YNTERYGH() INPUT VALUES:
Blade Length = 200
Mass Distribution Polynomial = -5534023222112881/18446744073709551616*x+1/1000
Stiffness Distribution Polynomial = -10000000*x+1342177280000001/67108864
Root offset (e) = 16
Rotor speed range = [ 0 ... 600 ]
Number of intervals = 100
x-range = [ 0 : 2 : 200 ]
ATTENTION: There are more than three rows for the yn-input argument.
           Values have been assumed to be deflections!
ATTENTION: There is only one column...
           Derivatives will be computed from deflection values:
              yn' is computed from yn.
              yn'' is computed from yn'.
```

The output arguments are returned as workspace variables: $a_1$=17.49; $K_{01}$=8.39; $K_{11}$=0.052 (=> $\bar{K}_{11}$=0.052*200=10.4); and, $K_1$=9.23. The bending frequency results are given in Figure 70.

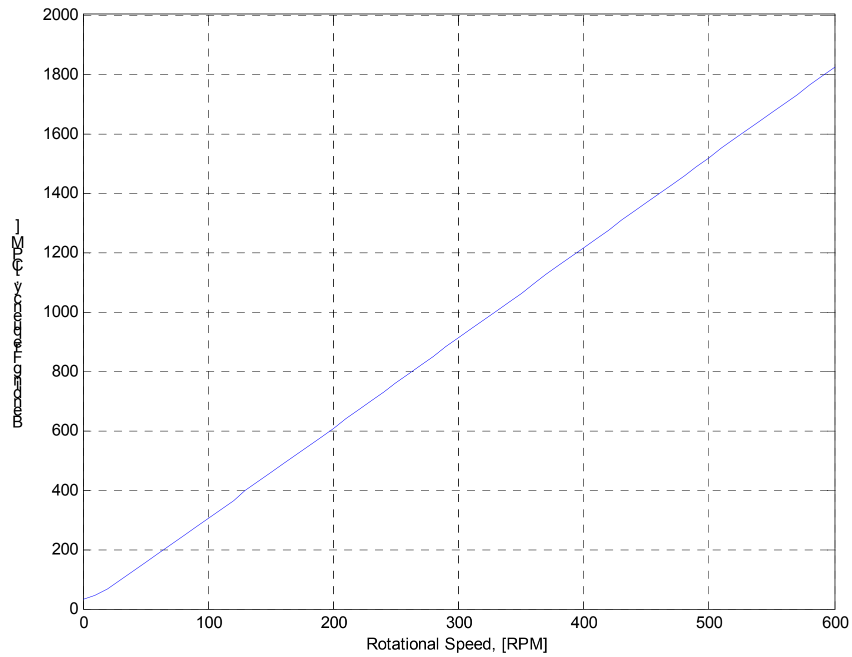



Figure 71. Frequency Results for the Sample Run of YNTERYGH.

For more information on the input arguments of YNTERYGH, please refer to the YNTERYGH section in Appendix A.

### *d. Sample Run For MAKELINE Utility Function*

Let us assume we have a nonlinear distribution along the blade and we need this distribution to be linear. The distribution has very large values for the first *ten percent* and the last *five percent* of the blade length from root to tip and these values need to be disregarded for a reasonable linearization. The blade length is *200* inches. The inputs may be summarized as follows:

- y_Nonlinear = [5.0000  1.0000  0.7071  0.5774  0.5000  0.4472  0.4082  0.3780  0.3536  0.3333  2.0000]
- x_Corresponding = [0  20  40  60  80  100  120  140  160  180  200]
- cutoffr = 200 * 10% = 20 inches
- cutofft = 200 * 5% = 10 inches

Then the function call will be as follows:

[y_LinCoef] = makeline(y_Nonlinear,x_Corresponding,20,10);

The *y_LinCoef* variable returned in the workspace gives the polynomial coefficients of the line as [-0.0025  0.7342]. Figure 71 demonstrates the results. The results for the function call without the cut-out values of 20 and 10, are also included for comparison.

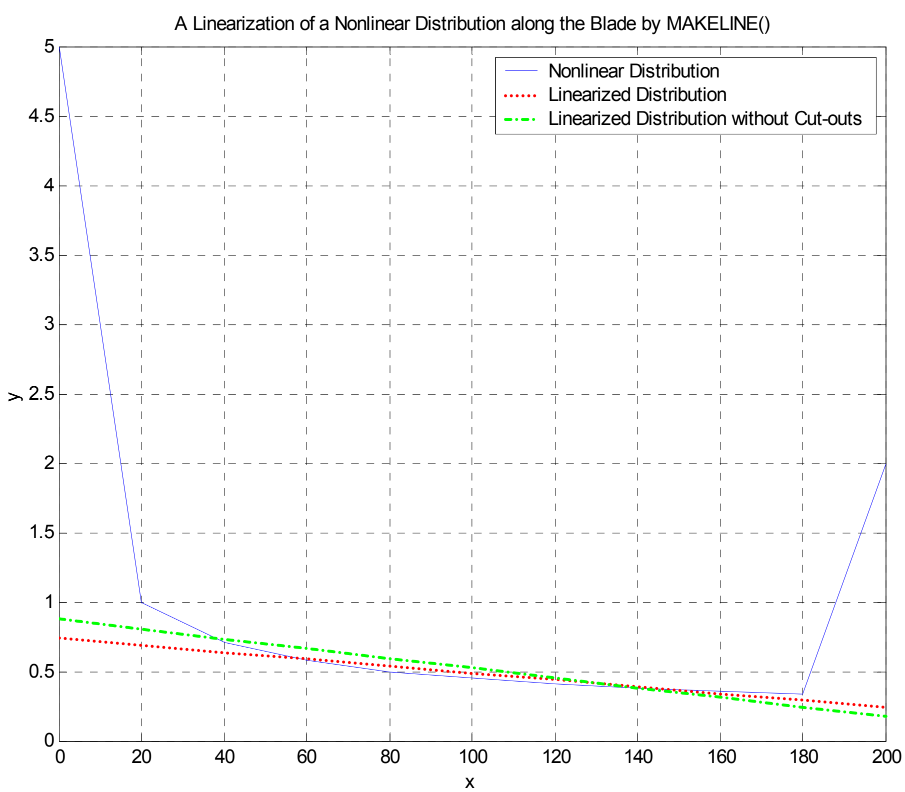



Figure 72. Sample Linearization of a Nonlinear Distribution by MAKELINE Function.

### *e. Sample Run For LINKMASS Utility Function*

Let us assume we have a mass distribution along the blade that has six mass values corresponding to six blade stations. This mass distribution therefore will have six elements in its vector form. But we need a vector of 26 elements for our purposes and we need to place these six values inside this vector of 26 elements. The other elements will be assigned zero. So the inputs arguments will be:

- massin = [0.0020 0.0010 0.0005 0.0005 0.0005 0.0010]
- xin = [0 0.2000 0.3000 0.5000 0.7000 1.0000]
- Number of intervals: 25

The function call will be:

```
[massout,xnew] = linkmass(massin,xin,25)
```

And the returned variables will have the values:

```
massout = [0.0020    0    0    0    0   0.0010    0   0.0005    0
              0      0    0   0.0005  0     0      0     0     0.0005
              0      0    0    0      0     0      0   0.0010]
xnew = [0   0.0400   0.0800   0.1200   0.1600   0.2000   0.2400   0.2800
        0.3200   0.3600   0.4000   0.4400   0.4800   0.5200   0.5600   0.6000
        0.6400   0.6800   0.7200   0.7600   0.8000   0.8400   0.8800
        0.9200   0.9600   1.0000]
```

### *f. Sample Run For LINKSPOR Utility Function*

Let us assume we have a distribution along the blade that has 11 elements in its vector for uniformly-spaced blade stations. But we need to have a distribution vector of *21* elements and the values between two consecutive points must be interpolated. The blade has a length of *200* inches. The input arguments will be as follows:

- y = [5.0000 1.0000 0.7071 0.5774 0.5000 0.4472 0.4082 0.3780 0.3536 0.3333 2.0000]
- Blade Length: 200
- Number of intervals: 20

The function call will be:

[outVec,outOrd,outVecX] = linkspor(y,200,20)

And the output variables will have the values:

- outVec =[5.0000 3.0000 1.0000 0.8536 0.7071 0.6422 0.5774 0.5387 0.5000 0.4736 0.4472 0.4277 0.4082 0.3931 0.3780 0.3658 0.3536 0.3434 0.3333 1.1667 2.0000
- outOrd = 3
- outVecX =[0 10 20 30 40 50 60 70 80 90 100 110 120 130 140 150 160 170 180 190 200]

The *outOrd* variable gives the order of the polynomial that the distribution curve can be approximated by. For example, if *outOrd* is 4, this means that you can fit a curve of $4^{th}$ degree to the distribution.

### 2. Sample Runs For GUI

Figures 73 through 83 show several consecutive executions of the GUI. Each figure has the input arguments and the corresponding outputs that have been returned after hitting the OK button. Note that everytime the *optional_inputs* button is unchecked the beam is assumed to be uniform, and checking the *optional_inputs* button again will use the last inputs. That is, everytime you change an input value, it is saved into the memory permanently. When you uncheck the *optional_inputs* button, temporary values are assigned for the optional inputs. And when you check the *optional_inputs* button again, the permanent values in the memory assigned to the optional inputs again.

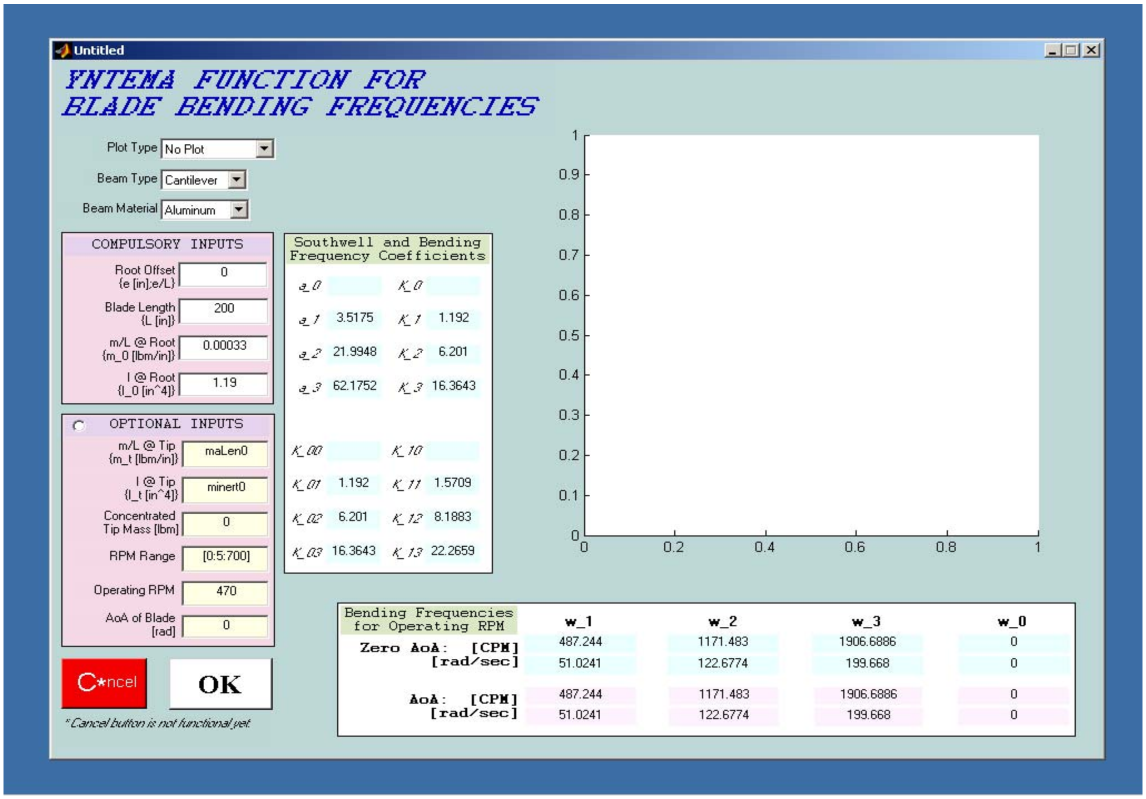


Figure 73. Sample Run 1 for GUI

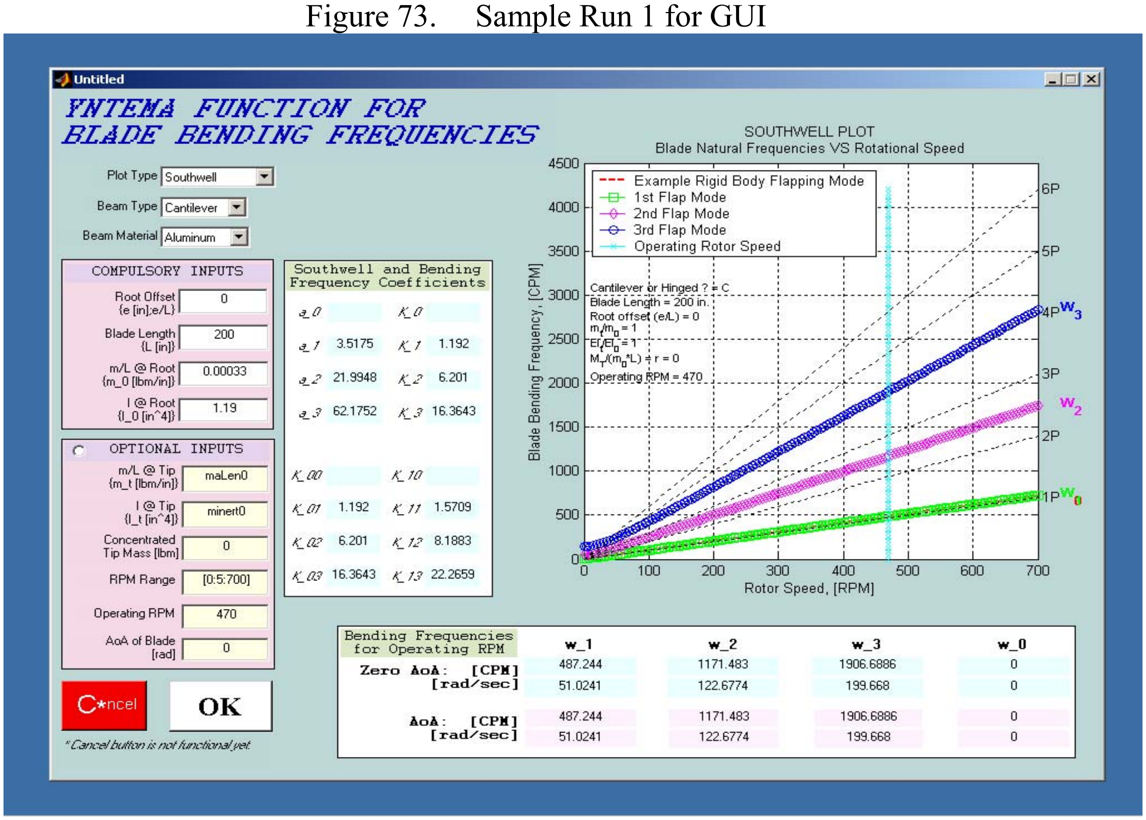


Figure 74. Sample Run 2 for GUI

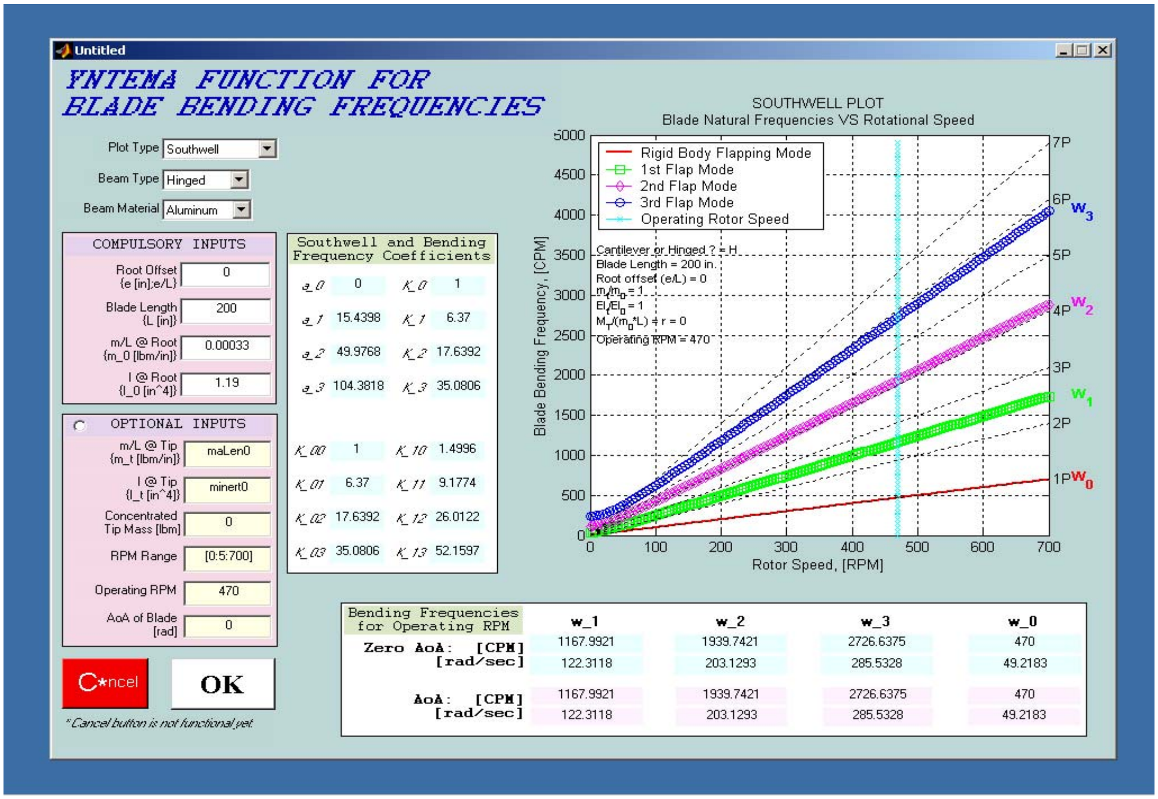


Figure 75. Sample Run 3 for GUI

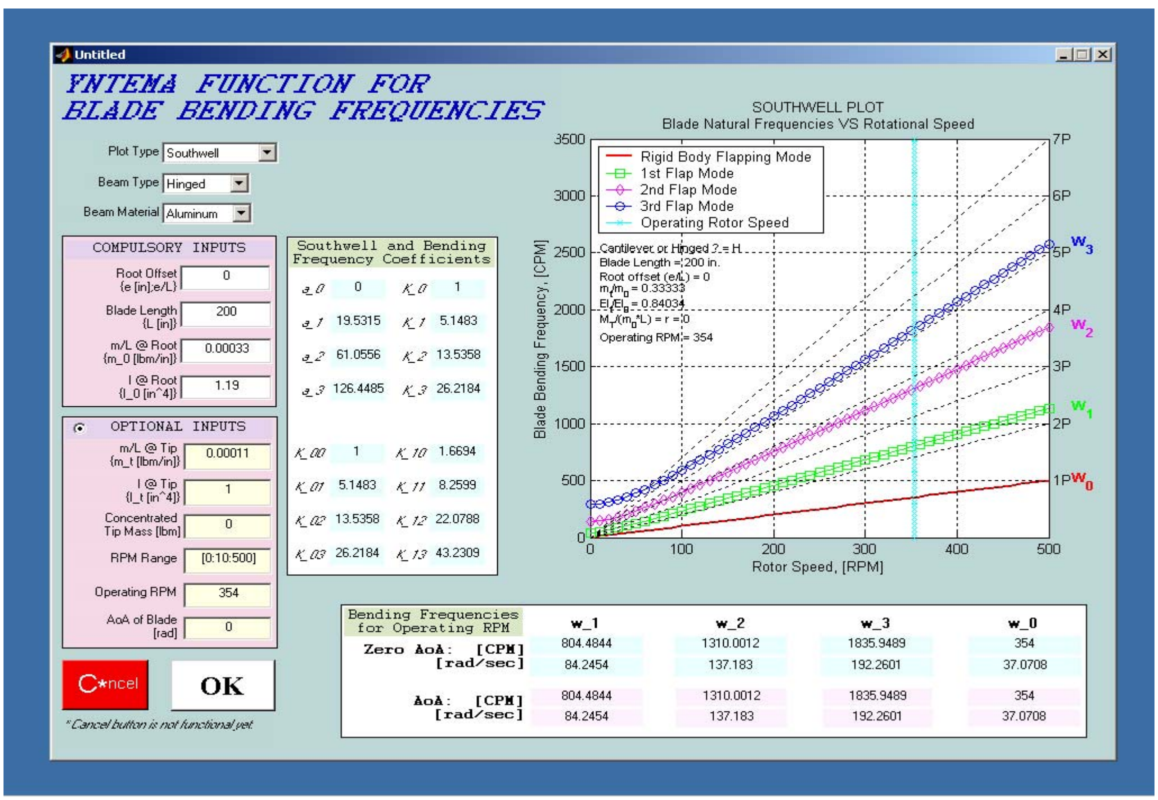


Figure 76. Sample Run 4 for GUI

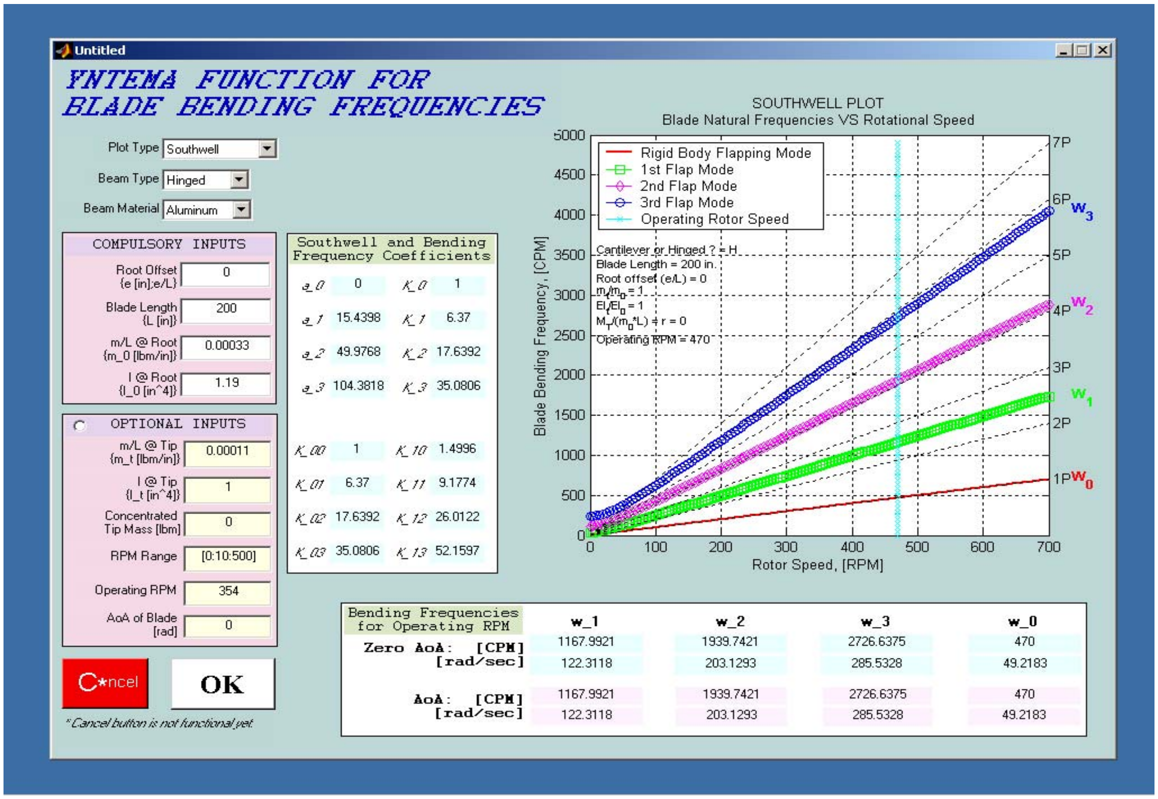



Figure 77. Sample Run 5 for GUI

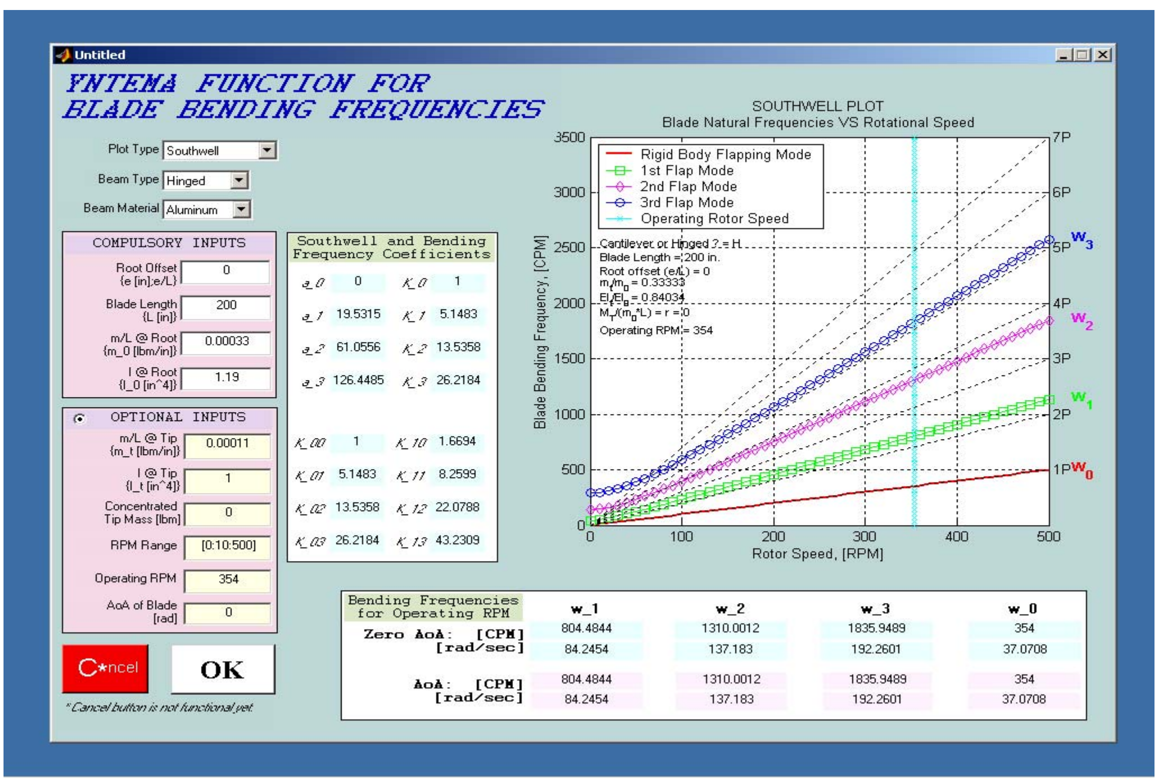



Figure 78. Sample Run 6 for GUI

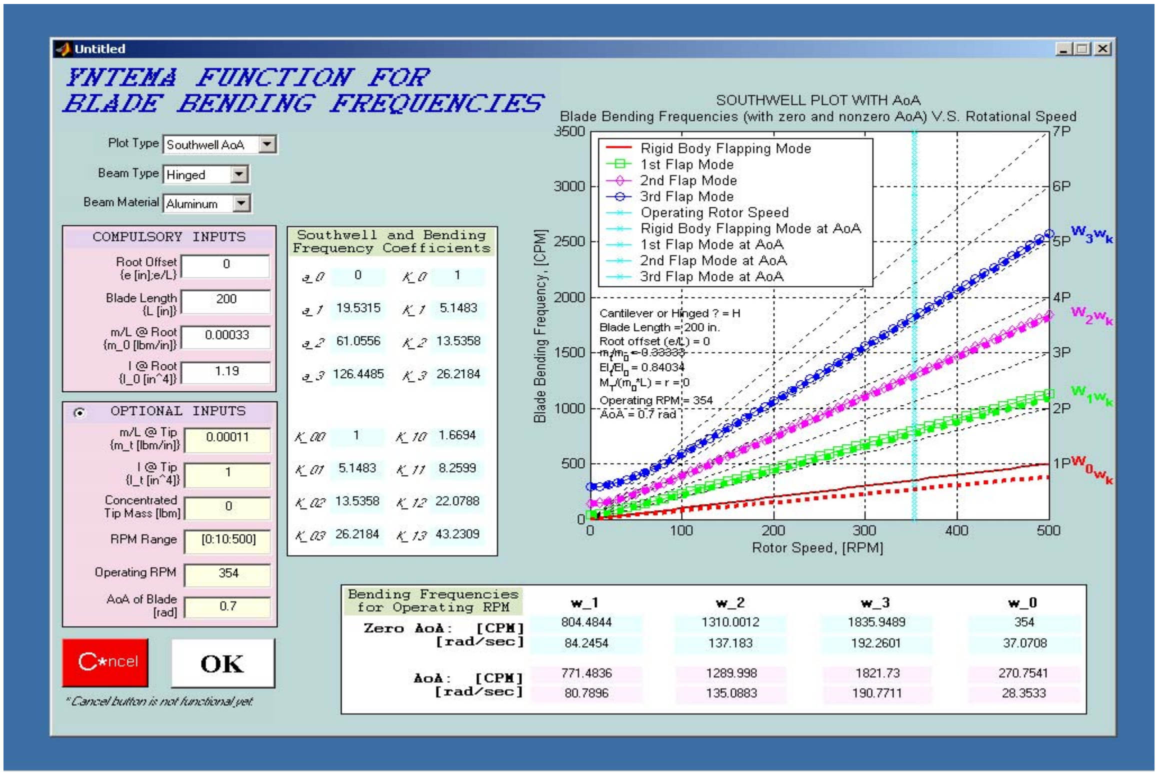



Figure 79. Sample Run 7 for GUI

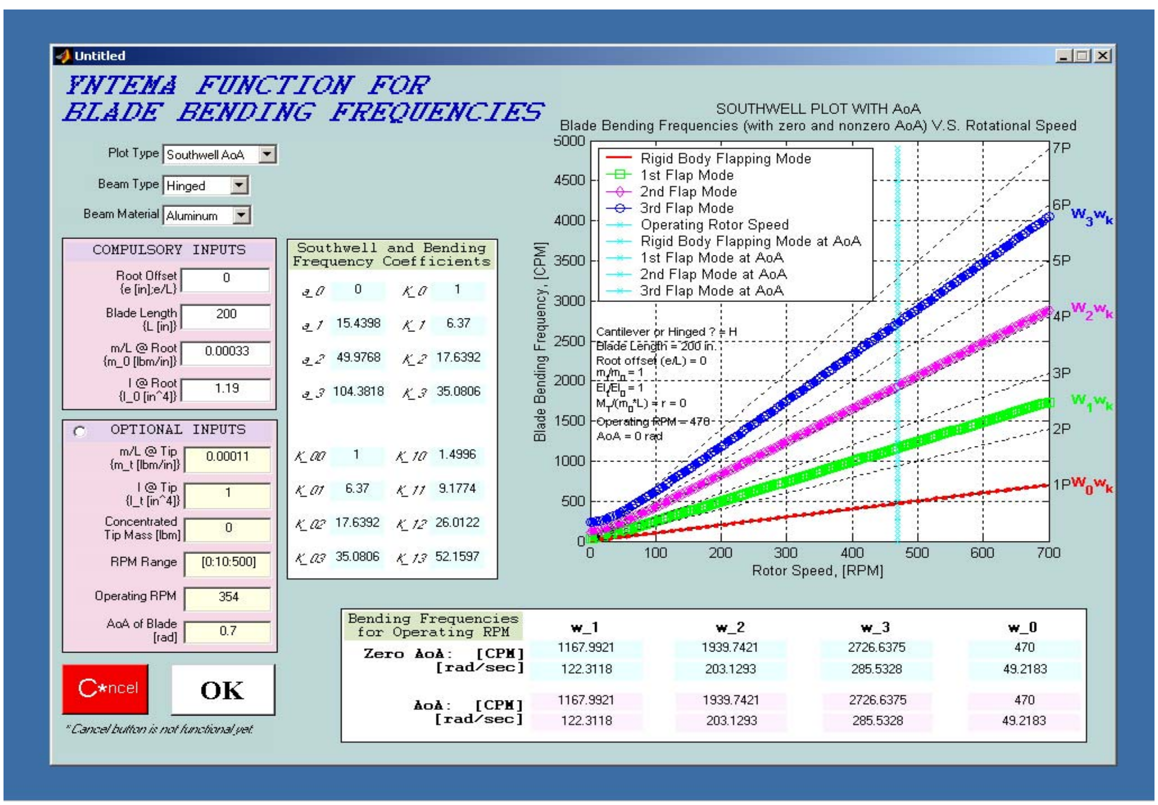



Figure 80. Sample Run 8 for GUI

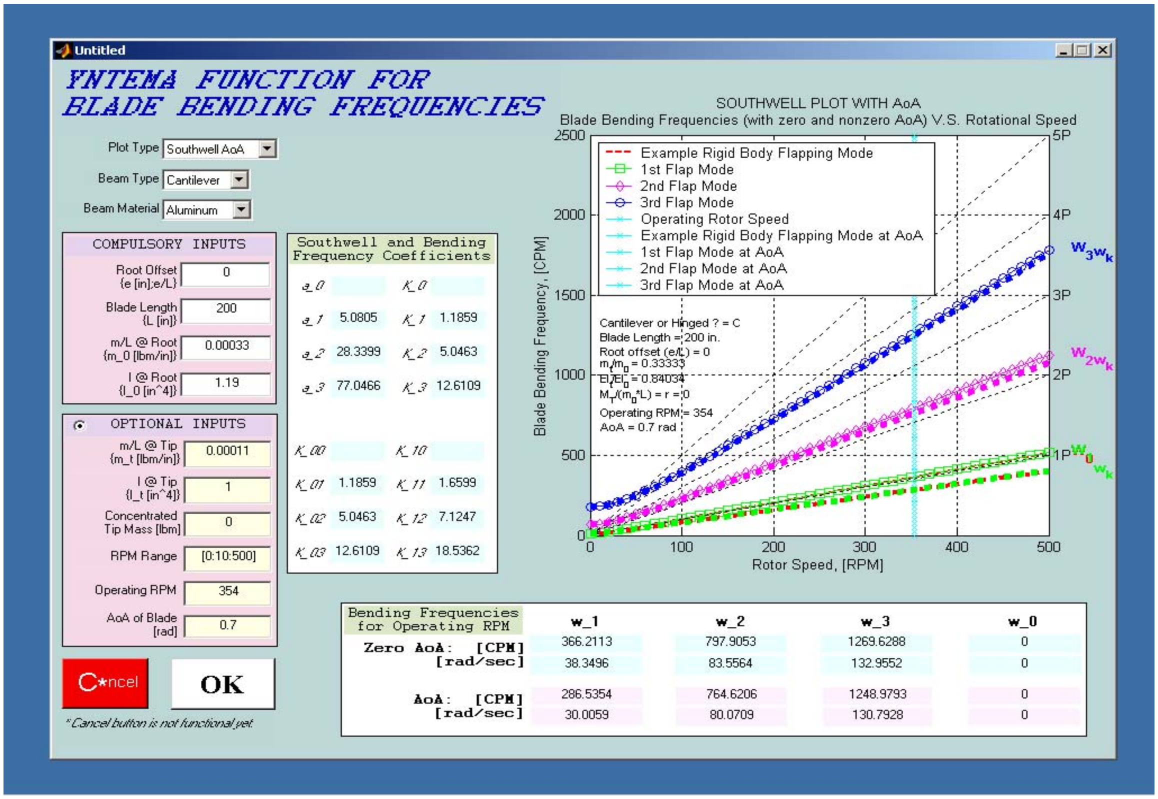



Figure 81. Sample Run 9 for GUI

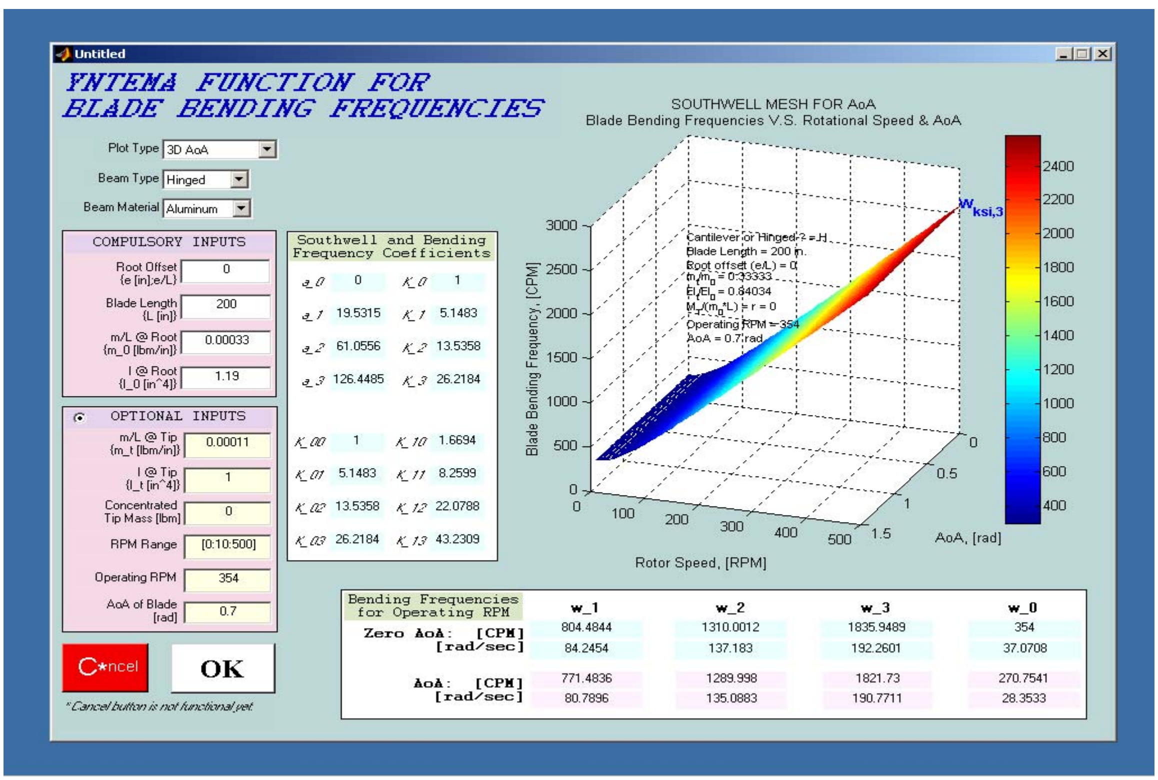



Figure 82. Sample Run 10 for GUI

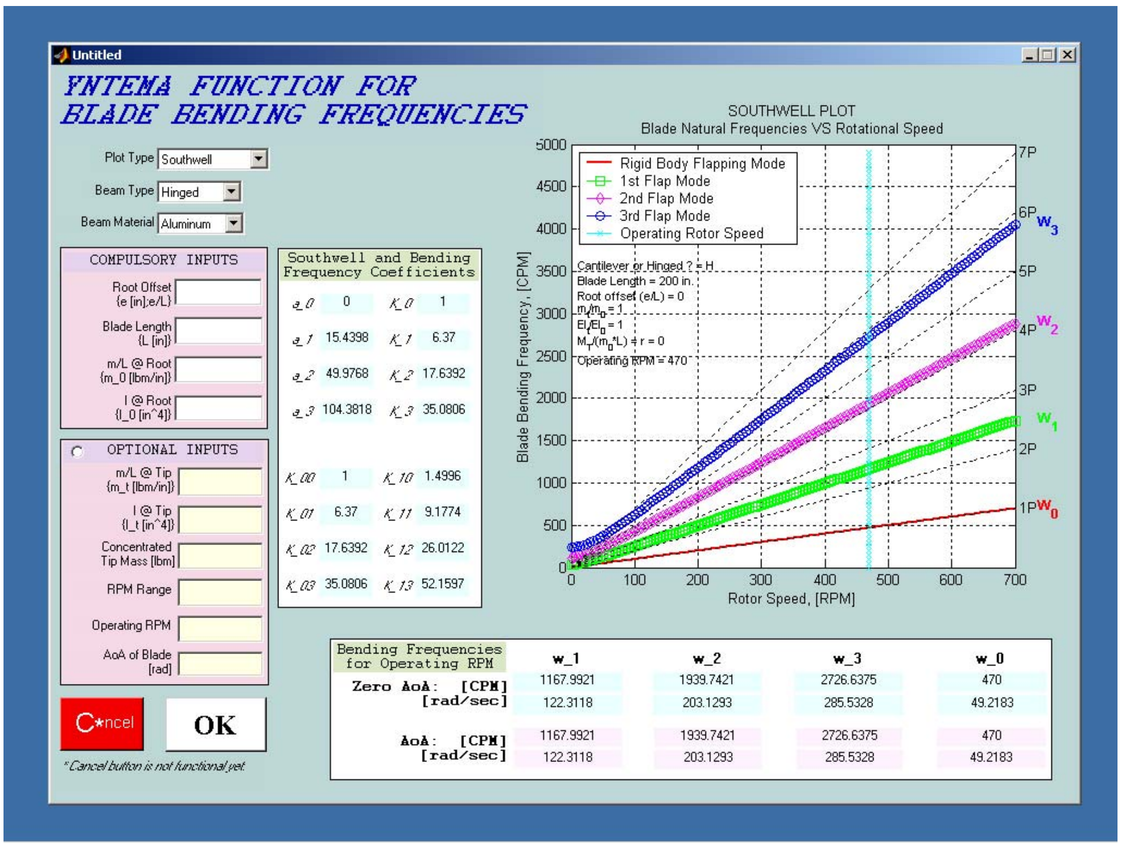



Figure 83. Sample Run 11 for GUI

# APPENDIX B. MATLAB® CODES FOR CURVE-FITTING AND DEFAULT FILES

## A. MATLAB® CODE FOR YNTEDFLT.M

```
% YNTEDFLT.M :  This file gives some default values for calculations in the Yntema
      method.
% This file is run in a cleared workspace and the workspace is then saved as
      yntedflt.mat.
% !LT>H>E>AKIN<TURKISHARMY<NPS<NAVY<MONTEREY<CA<SEP@))@
% DEFAULT VALUES: (are from OH-6 helicopter data)
elasAlDF = 1e7; %psi, modulus of elasticity for aluminum, default
minert0DF = 1.19; %in^4, moment of inertia at root, default
maLen0DF = 0.0003364; %lbm/in, blade mass per length at root
bLenDF = 158.4; %in, blade length from root offset to the tip.
rtrSpdRngDF = [0:10:700]; %RPM, (71x1), rotor speed
opRtrSpdDF = 470; %RPM, operating rotor speed
```

## B. MATLAB® CODE FOR YNTECOEF.M WITH PLOT STATEMENTS

```
% YNTECOEF.M: This file fits a curve for each of the input values from the Yntema report
      curves.
% The fourth-degree polynomial coefficients are intended for use in the 'yntema.m' file.
% !LT>H>E>AKIN<TURKISHARMY<NPS<NAVY<MONTEREY<CA<SEP@))@
% {Run this file in a cleared workspace and then save the workspace as 'yntecoef.mat'
      which is to be referred by 'yntema.m'}
%
% Variables beginning with 'mf' are the values read with a 1/120-inch-ruler
% Variables beginning with 'xmf' are the multipliers to convert 'mf' values into real
      read values
% Variables beginning with 'smf' are the offset values to add to real read values so as
      to get the final read values
% (Simply:  Read a point on the graph with a 1/120-inch-ruler, (mf)
%           Then multiply it with an appropriate coefficient, (xmf)
%           And add the offset value on the chart (smf): There you have the read values
      for the curve.
%           Simple when doing, just hard for ME to explain.)
% Variables beginning with 'a' and 'K' are the coefficients of the polynomial of the
      fitted curve.


% Range of (m_t/m_0) or (r = M_t/m_beam) ratios:
r0151 = [0.0:1/50:1.0].'; %(51x1)
r0126 = [0.0:1/25:1.0].'; %(26x1)
r0251 = [0.0:2/50:2.0].'; %(51x1)
r0226 = [0.0:2/25:2.0].'; %(26x1)

% HINGED BEAMS WITH LINEAR MASS AND STIFFNESS DISTRIBUTIONS:
      ======================================================================
% Figure-11
mf11_a1_E00 = [ 11.75   11.10   10.40   9.85    9.25    8.75    8.25    7.80    7.40
      7.00    6.60    ...
                        6.25    5.95    5.65    5.35    5.00    4.85    4.60    4.35
      4.15    3.95    ...
                        3.70    3.55    3.40    3.25    3.15    3.00    2.85    2.70
      2.60    2.50    ...
                        2.40    2.30    2.20    2.10    2.00    1.95    1.85    1.75
      1.65    1.55    ...
                        1.50    1.40    1.35    1.25    1.20    1.10    1.05    1.00
      0.95    0.90 ].';
mf11_a1_E05 = [ 15.40   14.65   13.95   13.35   12.80   12.20   11.65   11.20   10.70
      10.30   9.90    ...
                        9.50    9.15    8.80    8.50    8.15    7.90    7.65    7.45
      7.20    6.90    ...
                        6.75    6.55    6.40    6.15    5.95    5.85    5.65    5.55
      5.40    5.30    ...
                        5.10    5.00    4.90    4.80    4.65    4.55    4.40    4.30
      4.20    4.15    ...
                        4.05    3.95    3.90    3.80    3.75    3.65    3.55    3.45
      3.40    3.25 ].';
```

```
mf11_a1_E10 = [ 18.10   17.30   16.60   15.90   15.40   14.75   14.20   13.70   13.25
      12.80   12.40   ...
                        11.95   11.55   11.25   10.90   10.60   10.30   10.10   9.75
      9.60    9.40    ...
                        9.15    8.90    8.70    8.50    8.30    8.15    8.00    7.80
      7.65    7.50    ...
                        7.40    7.25    7.15    7.00    6.90    6.80    6.65    6.55
      6.45    6.35    ...
                        6.20    6.10    6.00    5.90    5.80    5.70    5.65    5.55
      5.50    5.35 ].';
mf11_a2_E00 = [ 15.70   14.60   13.60   12.85   12.05   11.40   10.80   10.20   9.60
      9.15    8.70    ...
                        8.20    7.75    7.45    7.00    6.70    6.40    6.15    5.85
      5.55    5.35    ...
                        5.05    4.80    4.60    4.45    4.25    4.00    3.85    3.65
      3.40    3.25    ...
                        3.05    2.85    2.70    2.60    2.45    2.25    2.10    1.95
      1.85    1.70    ...
                        1.55    1.45    1.35    1.20    1.10    1.00    0.85    0.70
      0.65    0.55 ].';
mf11_a2_E05 = [ 22.70   21.55   20.50   19.70   18.90   18.10   17.40   16.75   16.20
      15.60   15.10   ...
                        14.55   14.15   13.85   13.35   13.00   12.65   12.35   12.10
      11.80   11.50   ...
                        11.15   10.95   10.80   10.55   10.30   10.15   9.95    9.80
      9.60    9.40    ...
                        9.20    9.05    8.85    8.70    8.50    8.35    8.20    8.05
      7.90    7.75    ...
                        7.60    7.45    7.35    7.15    7.00    6.90    6.80    6.65
      6.55    6.40 ].';
mf11_a2_E10 = [ 27.70   26.40   25.30   24.25   23.40   22.60   21.80   21.10   20.40
      19.85   19.30   ...
                        18.70   18.25   17.85   17.40   17.10   16.75   16.40   16.05
      15.80   15.50   ...
                        15.20   14.95   14.70   14.50   14.25   14.05   13.80   13.60
      13.35   13.20   ...
                        13.00   12.80   12.60   12.40   12.20   12.05   11.85   11.65
      11.50   11.35   ...
                        11.20   11.05   10.90   10.70   10.55   10.40   10.25   10.15
      10.00   9.90 ].';
mf11_a3_E00 = [ 14.85   13.95   13.20   12.40   11.80   11.15   10.55   10.00   9.50
      9.05    8.60    ...
                        8.20    7.80    7.40    7.05    6.80    6.45    6.15    5.90
      5.65    5.40    ...
                        5.15    4.95    4.65    4.50    4.35    4.15    4.00    3.80
      3.65    3.50    ...
                        3.30    3.20    3.10    2.95    2.80    2.65    2.55    2.40
      2.25    2.15    ...
                        2.05    1.95    1.85    1.80    1.65    1.60    1.45    1.35
      1.25    1.20 ].';
mf11_a3_E05 = [ 23.15   21.90   21.00   20.10   19.20   18.50   17.80   17.25   16.65
      16.10   15.60   ...
                        15.20   14.75   14.40   14.00   13.70   13.35   13.00   12.75
      12.45   12.20   ...
                        11.95   11.75   11.50   11.30   11.10   10.90   10.70   10.50
      10.35   10.20   ...
                        10.00   9.80    9.60    9.45    9.35    9.20    9.05    8.90
      8.80    8.65    ...
                        8.45    8.30    8.20    8.10    8.00    7.90    7.75    7.60
      7.50    7.40 ].';
mf11_a3_E10 = [ 28.30   27.00   26.00   25.00   24.20   23.45   22.75   22.05   21.50
      20.90   20.35   ...
                        19.90   19.40   19.00   18.60   18.25   17.85   17.50   17.15
      16.85   16.60   ...
                        16.35   16.10   15.80   15.55   15.35   15.10   14.85   14.60
      14.40   14.25   ...
                        14.05   13.85   13.60   13.45   13.30   13.15   13.00   12.75
      12.55   12.40   ...
                        12.20   12.05   11.95   11.80   11.60   11.50   11.35   11.20
      11.05   10.90 ].';

a0HLMS =
      0.0;%OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputO
      utputOutputOutputOutputOutput

xmf11_a1 = 20 ./ [19.85:-0.05/50:19.80].'; %(51x1)
smf11_a1 = 10;
vf11_a1_E00 = mf11_a1_E00 .* xmf11_a1 + smf11_a1; %(51x1)
vf11_a1_E05 = mf11_a1_E05 .* xmf11_a1 + smf11_a1; %(51x1)
```

```
vf11_a1_E10 = mf11_a1_E10 .* xmf11_a1 + smf11_a1; %(51x1)
a100HLMS = polyfit(r0151,vf11_a1_E00,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput
a105HLMS = polyfit(r0151,vf11_a1_E05,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput
a110HLMS = polyfit(r0151,vf11_a1_E10,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput

xmf11_a2 = 60 ./ [29.85:-0.10/50:29.75].'; %(51x1)
smf11_a2 = 30;
vf11_a2_E00 = mf11_a2_E00 .* xmf11_a2 + smf11_a2; %(51x1)
vf11_a2_E05 = mf11_a2_E05 .* xmf11_a2 + smf11_a2; %(51x1)
vf11_a2_E10 = mf11_a2_E10 .* xmf11_a2 + smf11_a2; %(51x1)
a200HLMS = polyfit(r0151,vf11_a2_E00,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput
a205HLMS = polyfit(r0151,vf11_a2_E05,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput
a210HLMS = polyfit(r0151,vf11_a2_E10,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput

xmf11_a3 = 120 ./ [29.80:-0.30/50:29.50].'; %(51x1)
smf11_a3 = 60;
vf11_a3_E00 = mf11_a3_E00 .* xmf11_a3 + smf11_a3; %(51x1)
vf11_a3_E05 = mf11_a3_E05 .* xmf11_a3 + smf11_a3; %(51x1)
vf11_a3_E10 = mf11_a3_E10 .* xmf11_a3 + smf11_a3; %(51x1)
a300HLMS = polyfit(r0151,vf11_a3_E00,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput
a305HLMS = polyfit(r0151,vf11_a3_E05,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput
a310HLMS = polyfit(r0151,vf11_a3_E10,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput

figure;
subplot(3,1,3);
plot(r0151,polyval(a100HLMS,r0151),'-.'); hold on;
plot(r0151,polyval(a105HLMS,r0151),'--');
plot(r0151,polyval(a110HLMS,r0151),'-');
text(0.5,28,'1st mode');
ylabel('a_1'); xlabel('m_t/m_0'); grid minor;
subplot(3,1,2);
plot(r0151,polyval(a200HLMS,r0151),'-.'); hold on;
plot(r0151,polyval(a205HLMS,r0151),'--');
plot(r0151,polyval(a210HLMS,r0151),'-');
text(0.5,85,'2nd mode');
ylabel('a_2'); grid minor;
subplot(3,1,1);
plot(r0151,polyval(a300HLMS,r0151),'-.'); hold on;
plot(r0151,polyval(a305HLMS,r0151),'--');
plot(r0151,polyval(a310HLMS,r0151),'-');
text(0.5,170,'3rd mode');title('Figure-11');
ylabel('a_3'); grid minor;
legend('EI_t/EI_0=0','EI_t/EI_0=.5','EI_t/EI_0=1');
hold off;


clear mf11_a* xmf11_a* smf11_a* vf11_a*

% Figure-12
mf12_K01_E00 = [ 2.40   2.85    3.30    3.80    4.30    4.80    5.30    5.80    6.20
      6.65    7.10    ...
                        7.60    8.00    8.45    8.95    9.35    9.75    10.15   10.60
      11.00   11.40   ...
                        11.85   12.20   12.60   13.05   13.45   13.90   14.20   14.65
      15.05   15.40   ...
                        15.75   16.15   16.50   16.80   17.15   17.55   17.90   18.25
      18.55   18.85   ...
                        19.15   19.40   19.70   19.95   20.20   20.45   20.65   20.90
      21.10   21.30 ].';
mf12_K01_E05 = [ 3.30   3.75    4.20    4.70    5.20    5.70    6.20    6.75    7.20
      7.70    8.15    ...
                        8.70    9.15    9.60    10.10   10.50   10.95   11.40   11.85
      12.30   12.80   ...
                        13.20   13.65   14.10   14.50   14.95   15.40   15.80   16.25
      16.65   17.00   ...
                        17.40   17.80   18.20   18.55   18.95   19.30   19.60   20.00
      20.35   20.65   ...
                        21.00   21.30   21.60   21.90   22.20   22.45   22.75   23.00
      23.20   23.40 ].';
```

```
mf12_K01_E10 = [ 4.15   4.65    5.15    5.65    6.15    6.65    7.15    7.65    8.10
      8.65    9.10    ...
                        9.60    10.00   10.50   11.00   11.40   11.85   12.30   12.80
      13.25   13.70   ...
                        14.15   14.60   15.05   15.45   15.90   16.35   16.70   17.15
      17.60   18.00   ...
                        18.40   18.75   19.20   19.55   19.95   20.30   20.65   21.05
      21.40   21.70   ...
                        22.10   22.40   22.75   23.00   23.30   23.60   23.90   24.10
      24.35   24.55 ].';
mf12_K02_E00 = [ 4.45   4.85    5.25    5.65    6.05    6.45    6.90    7.30    7.60
      8.00    8.40    ...
                        8.80    9.15    9.50    9.90    10.30   10.65   11.00   11.45
      11.85   12.15   ...
                        12.50   12.80   13.10   13.45   13.85   14.10   14.35   14.65
      14.95   15.20   ...
                        15.45   15.70   16.00   16.30   16.55   16.75   17.00   17.20
      17.45   17.70   ...
                        17.90   18.05   18.25   18.45   18.65   18.80   18.95   19.10
      19.20   19.30 ].';
mf12_K02_E05 = [ 6.50   6.85    7.30    7.70    8.15    8.60    9.05    9.50    9.85
      10.25   10.70   ...
                        11.15   11.55   11.90   12.30   12.70   13.10   13.55   13.90
      14.30   14.65   ...
                        15.05   15.40   15.75   16.15   16.50   16.85   17.15   17.50
      17.80   18.10   ...
                        18.45   18.80   19.10   19.40   19.70   20.00   20.25   20.50
      20.75   21.05   ...
                        21.30   21.50   21.70   21.95   22.20   22.45   22.60   22.80
      23.00   23.15 ].';
mf12_K02_E10 = [ 7.50   7.90    8.35    8.80    9.25    9.70    10.20   10.70   11.10
      11.50   11.95   ...
                        12.35   12.80   13.20   13.65   14.05   14.50   14.90   15.25
      15.70   16.05   ...
                        16.45   16.80   17.20   17.55   18.00   18.35   18.70   19.05
      19.40   19.70   ...
                        20.10   20.45   20.80   21.10   21.40   21.70   22.00   22.30
      22.60   22.90   ...
                        23.20   23.40   23.65   23.90   24.15   24.35   24.50   24.70
      24.90   25.00 ].';
mf12_K03_E00 = [ 2.60   2.95    3.30    3.65    4.00    4.40    4.70    5.05    5.35
      5.70    6.05    ...
                        6.35    6.70    7.00    7.30    7.60    7.95    8.25    8.55
      8.90    9.15    ...
                        9.45    9.75    10.00   10.25   10.55   10.70   11.00   11.20
      11.45   11.65   ...
                        11.90   12.10   12.30   12.50   12.70   12.85   13.05   13.20
      13.35   13.50   ...
                        13.70   13.85   13.95   14.05   14.15   14.25   14.40   14.50
      14.55   14.70 ].';
mf12_K03_E05 = [ 4.40   4.75    5.15    5.55    5.95    6.35    6.70    7.05    7.45
      7.80    8.15    ...
                        8.55    8.90    9.20    9.60    9.95    10.30   10.65   11.00
      11.30   11.65   ...
                        12.00   12.35   12.65   12.95   13.30   13.65   13.95   14.25
      14.55   14.85   ...
                        15.10   15.35   15.60   15.85   16.15   16.35   16.60   16.85
      17.00   17.20   ...
                        17.45   17.65   17.80   18.00   18.20   18.35   18.50   18.60
      18.75   18.95 ].';
mf12_K03_E10 = [ 5.50   5.85    6.25    6.70    7.05    7.50    7.85    8.30    8.70
      9.05    9.45    ...
                        9.85    10.25   10.60   10.95   11.30   11.70   12.05   12.40
      12.80   13.10   ...
                        13.45   13.85   14.15   14.45   14.80   15.10   15.40   15.65
      16.00   16.30   ...
                        16.65   16.90   17.15   17.40   17.65   17.90   18.10   18.35
      18.60   18.85   ...
                        19.05   19.30   19.45   19.60   19.85   20.05   20.25   20.35
      20.50   20.70 ].';

K00HLMS =
      1.0;%OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputO
      utputOutputOutputOutputOutput

xmf12_K01 = 2.5 ./ [26.00:-0.10/50:25.90].'; %(51x1)
smf12_K01 = 4;
vf12_K01_E00 = mf12_K01_E00 .* xmf12_K01 + smf12_K01; %(51x1)
vf12_K01_E05 = mf12_K01_E05 .* xmf12_K01 + smf12_K01; %(51x1)
```

```
vf12_K01_E10 = mf12_K01_E10 .* xmf12_K01 + smf12_K01; %(51x1)
K0100HLMS = polyfit(r0151,vf12_K01_E00,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput
K0105HLMS = polyfit(r0151,vf12_K01_E05,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput
K0110HLMS = polyfit(r0151,vf12_K01_E10,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput

xmf12_K02 = 10 ./ [26.10:-0.15/50:25.95].'; %(51x1)
smf12_K02 = 8;
vf12_K02_E00 = mf12_K02_E00 .* xmf12_K02 + smf12_K02; %(51x1)
vf12_K02_E05 = mf12_K02_E05 .* xmf12_K02 + smf12_K02; %(51x1)
vf12_K02_E10 = mf12_K02_E10 .* xmf12_K02 + smf12_K02; %(51x1)
K0200HLMS = polyfit(r0151,vf12_K02_E00,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput
K0205HLMS = polyfit(r0151,vf12_K02_E05,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput
K0210HLMS = polyfit(r0151,vf12_K02_E10,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput

xmf12_K03 = 25 ./ [25.85:-0.10/50:25.75].'; %(51x1)
smf12_K03 = 15;
vf12_K03_E00 = mf12_K03_E00 .* xmf12_K03 + smf12_K03; %(51x1)
vf12_K03_E05 = mf12_K03_E05 .* xmf12_K03 + smf12_K03; %(51x1)
vf12_K03_E10 = mf12_K03_E10 .* xmf12_K03 + smf12_K03; %(51x1)
K0300HLMS = polyfit(r0151,vf12_K03_E00,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput
K0305HLMS = polyfit(r0151,vf12_K03_E05,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput
K0310HLMS = polyfit(r0151,vf12_K03_E10,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput

figure;
subplot(3,1,3);
plot(r0151,polyval(K0100HLMS,r0151),'-.'); hold on;
plot(r0151,polyval(K0105HLMS,r0151),'--');
plot(r0151,polyval(K0110HLMS,r0151),'-');
text(0.5,6.25,'1st mode');
ylabel('K_0_1'); xlabel('m_t/m_0'); grid minor;
subplot(3,1,2);
plot(r0151,polyval(K0200HLMS,r0151),'-.'); hold on;
plot(r0151,polyval(K0205HLMS,r0151),'--');
plot(r0151,polyval(K0210HLMS,r0151),'-');
text(0.5,17,'2nd mode');
ylabel('K_0_2'); grid minor;
subplot(3,1,1);
plot(r0151,polyval(K0300HLMS,r0151),'-.'); hold on;
plot(r0151,polyval(K0305HLMS,r0151),'--');
plot(r0151,polyval(K0310HLMS,r0151),'-');
text(0.5,37.5,'3rd mode');title('Figure-12');
ylabel('K_0_3'); grid minor;
legend('EI_t/EI_0=0','EI_t/EI_0=.5','EI_t/EI_0=1',2);
hold off;


clear mf12_K0* xmf12_K0* smf12_K0* vf12_K0*

% Figure-13
mf13_Kh10_EXX = [ 22.25 20.80   19.40   18.10   16.85   15.70   14.70   13.65   12.70
      11.80   11.10   ...
                        10.40   9.85    9.30    8.80    8.30    7.85    7.40    7.00
      6.60    6.25    ...
                        5.85    5.50    5.15    4.85    4.50    4.20    3.95    3.70
      3.50    3.20    ...
                        3.00    2.85    2.65    2.40    2.20    2.05    1.90    1.70
      1.50    1.35    ...
                        1.25    1.15    0.95    0.85    0.70    0.60    0.45    0.30
      0.15    0.00 ].';
mf13_Kh11_E00 = [ 3.05  3.10    3.20    3.25    3.30    3.35    3.45    3.50    3.60
      3.65    3.75    ...
                        3.80    3.90    4.00    4.10    4.15    4.25    4.35    4.45
      4.50    4.70    ...
                        4.80    4.90    5.05    5.15    5.25    5.40    5.55    5.70
      5.80    5.95    ...
                        6.15    6.30    6.50    6.65    6.80    7.00    7.15    7.45
      7.65    7.90    ...
                        8.10    8.35    8.60    8.90    9.20    9.50    9.80    10.10
      10.45   10.75 ].';
```

```
mf13_Kh11_E05 = [ 7.70  7.75    7.85    7.90    8.00    8.05    8.15    8.20    8.30
     8.40    8.45    ...
                      8.55    8.65    8.75    8.85    9.00    9.10    9.20    9.30
     9.40    9.50    ...
                      9.65    9.80    9.90    10.05   10.15   10.30   10.50   10.65
     10.80   10.95   ...
                      11.10   11.30   11.50   11.70   11.90   12.10   12.35   12.60
     12.85   13.10   ...
                      13.40   13.65   13.95   14.25   14.60   15.00   15.35   15.65
     16.05   16.40 ].';
mf13_Kh11_E10 = [ 10.45 10.50   10.60   10.65   10.75   10.80   10.90   10.95   11.05
     11.15   11.20   ...
                      11.30   11.40   11.50   11.60   11.70   11.85   11.95   12.10
     12.20   12.30   ...
                      12.45   12.60   12.75   12.85   13.00   13.15   13.30   13.45
     13.65   13.80   ...
                      14.00   14.20   14.40   14.60   14.80   15.00   15.25   15.50
     15.75   16.05   ...
                      16.35   16.60   16.90   17.20   17.50   17.90   18.30   18.65
     19.05   19.40 ].';
mf13_Kh12_E00 = [ 1.85  1.90    2.00    2.05    2.12    2.18    2.25    2.32    2.38
     2.45    2.52    ...
                      2.58    2.65    2.73    2.82    2.90    2.97    3.03    3.10
     3.17    3.23    ...
                      3.30    3.37    3.43    3.50    3.55    3.65    3.75    3.82
     3.88    3.95    ...
                      4.00    4.05    4.10    4.18    4.27    4.35    4.42    4.48
     4.55    4.63    ...
                      4.72    4.80    4.88    4.97    5.05    5.15    5.25    5.35
     5.40    5.50 ].';
mf13_Kh12_E05 = [ 3.95  4.05    4.10    4.20    4.28    4.37    4.45    4.53    4.62
     4.70    4.78    ...
                      4.87    4.95    5.03    5.12    5.20    5.27    5.33    5.40
     5.50    5.60    ...
                      5.70    5.78    5.87    5.95    6.00    6.10    6.25    6.33
     6.42    6.50    ...
                      6.57    6.63    6.70    6.78    6.87    6.95    7.05    7.15
     7.25    7.35    ...
                      7.45    7.55    7.65    7.75    7.85    7.97    8.08    8.20
     8.30    8.40 ].';
mf13_Kh12_E10 = [ 5.10  5.25    5.30    5.35    5.45    5.55    5.65    5.75    5.85
     5.95    6.03    ...
                      6.12    6.20    6.30    6.40    6.50    6.58    6.67    6.75
     6.85    6.95    ...
                      7.05    7.13    7.22    7.30    7.40    7.50    7.60    7.70
     7.80    7.90    ...
                      7.98    8.07    8.15    8.23    8.32    8.40    8.50    8.60
     8.70    8.80    ...
                      8.90    9.00    9.12    9.23    9.35    9.45    9.55    9.65
     9.75    9.85 ].';
mf13_Kh13_E00 = [ 0.65  0.70    0.75    0.85    0.95    1.05    1.13    1.21    1.29
     1.37    1.45    ...
                      1.53    1.61    1.69    1.77    1.85    1.93    2.01    2.09
     2.17    2.25    ...
                      2.325   2.40    2.475   2.55    2.60    2.70    2.775   2.85
     2.925   3.00    ...
                      3.08    3.16    3.24    3.32    3.40    3.47    3.54    3.61
     3.68    3.75    ...
                      3.82    3.89    3.96    4.03    4.10    4.18    4.25    4.35
     4.40    4.45 ].';
mf13_Kh13_E05 = [ 2.80  2.90    3.05    3.17    3.28    3.40    3.52    3.64    3.76
     3.88    4.00    ...
                      4.13    4.26    4.39    4.52    4.65    4.73    4.81    4.89
     4.97    5.05    ...
                      5.19    5.32    5.46    5.60    5.70    5.80    5.91    6.02
     6.13    6.25    ...
                      6.35    6.45    6.55    6.65    6.75    6.85    6.95    7.05
     7.15    7.25    ...
                      7.35    7.45    7.55    7.65    7.75    7.83    7.92    8.00
     8.05    8.15 ].';
mf13_Kh13_E10 = [ 4.10  4.20    4.35    4.50    4.65    4.80    4.92    5.04    5.16
     5.28    5.40    ...
                      5.52    5.64    5.76    5.88    6.00    6.11    6.22    6.33
     6.44    6.55    ...
                      6.66    6.77    6.88    7.00    7.15    7.25    7.36    7.47
     7.58    7.70    ...
                      7.82    7.94    8.06    8.18    8.30    8.40    8.50    8.60
     8.70    8.80    ...
```

```
                   8.91    9.02    9.13    9.24    9.35    9.47    9.58    9.70
      9.75    9.80 ].';

xmf13_Kh10 = 0.5 ./ [22.25 .* ones(1,51)].'; %(51x1)
smf13_Kh10 = 1.5;
vf13_Kh10_EXX = mf13_Kh10_EXX .* xmf13_Kh10 + smf13_Kh10; %(51x1)
Kh10XXHLMS = polyfit(r0151,vf13_Kh10_EXX,8); %
      OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput

xmf13_Kh11 = 2.5 ./ [22.35:-0.05/50:22.30].'; %(51x1)
smf13_Kh11 = 7;
vf13_Kh11_E00 = mf13_Kh11_E00 .* xmf13_Kh11 + smf13_Kh11; %(51x1)
vf13_Kh11_E05 = mf13_Kh11_E05 .* xmf13_Kh11 + smf13_Kh11; %(51x1)
vf13_Kh11_E10 = mf13_Kh11_E10 .* xmf13_Kh11 + smf13_Kh11; %(51x1)
Kh1100HLMS = polyfit(r0151,vf13_Kh11_E00,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput
Kh1105HLMS = polyfit(r0151,vf13_Kh11_E05,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput
Kh1110HLMS = polyfit(r0151,vf13_Kh11_E10,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput

xmf13_Kh12 = 15 ./ [13.40 .* ones(1,51)].'; %(51x1)
smf13_Kh12 = 15;
vf13_Kh12_E00 = mf13_Kh12_E00 .* xmf13_Kh12 + smf13_Kh12; %(51x1)
vf13_Kh12_E05 = mf13_Kh12_E05 .* xmf13_Kh12 + smf13_Kh12; %(51x1)
vf13_Kh12_E10 = mf13_Kh12_E10 .* xmf13_Kh12 + smf13_Kh12; %(51x1)
Kh1200HLMS = polyfit(r0151,vf13_Kh12_E00,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput
Kh1205HLMS = polyfit(r0151,vf13_Kh12_E05,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput
Kh1210HLMS = polyfit(r0151,vf13_Kh12_E10,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput

xmf13_Kh13 = 40 ./ [17.80:-0.10/50:17.70].'; %(51x1)
smf13_Kh13 = 30;
vf13_Kh13_E00 = mf13_Kh13_E00 .* xmf13_Kh13 + smf13_Kh13; %(51x1)
vf13_Kh13_E05 = mf13_Kh13_E05 .* xmf13_Kh13 + smf13_Kh13; %(51x1)
vf13_Kh13_E10 = mf13_Kh13_E10 .* xmf13_Kh13 + smf13_Kh13; %(51x1)
Kh1300HLMS = polyfit(r0151,vf13_Kh13_E00,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput
Kh1305HLMS = polyfit(r0151,vf13_Kh13_E05,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput
Kh1310HLMS = polyfit(r0151,vf13_Kh13_E10,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput

figure;
subplot(4,1,4);
plot(r0151,polyval(Kh10XXHLMS,r0151),'-','linewidth',2); hold on;
text(0.5,1.9,'Zero mode');
text(0.5,1.65,'FOR ALL EI_t/EI_0');
ylabel('Kh_1_0'); xlabel('m_t/m_0'); grid minor;
subplot(4,1,3);
plot(r0151,polyval(Kh1100HLMS,r0151),'-.'); hold on;
plot(r0151,polyval(Kh1105HLMS,r0151),'--');
plot(r0151,polyval(Kh1110HLMS,r0151),'-');
text(0.5,9.25,'1st mode');
ylabel('Kh_1_1'); grid minor;
subplot(4,1,2);
plot(r0151,polyval(Kh1200HLMS,r0151),'-.'); hold on;
plot(r0151,polyval(Kh1205HLMS,r0151),'--');
plot(r0151,polyval(Kh1210HLMS,r0151),'-');
text(0.5,28,'2nd mode');
ylabel('Kh_1_2'); grid minor;
subplot(4,1,1);
plot(r0151,polyval(Kh1300HLMS,r0151),'-.'); hold on;
plot(r0151,polyval(Kh1305HLMS,r0151),'--');
plot(r0151,polyval(Kh1310HLMS,r0151),'-');
text(0.5,55,'3rd mode');title('Figure-13');
ylabel('Kh_1_3'); grid minor;
legend('EI_t/EI_0=0','EI_t/EI_0=.5','EI_t/EI_0=1',2);
hold off;

clear mf13_Kh1* xmf13_Kh1* smf13_Kh1* vf13_Kh1*

% CANTILEVER BEAMS WITH LINEAR MASS AND STIFFNESS DISTRIBUTIONS:
      ===================================================================
% Figure-14
mf14_a1_E00 = [ 22.25   20.35   18.80   17.55   16.55   15.50   14.65   13.85   13.15
      12.35   11.75   ...
```

```
                  11.15   10.50   10.00   9.45    8.95    8.50    8.05    7.65
     7.30    6.95    ...
                  6.55    6.25    5.90    5.60    5.35    5.00    4.70    4.35
     4.15    3.85    ...
                  3.60    3.40    3.20    3.00    2.75    2.50    2.30    2.10
     1.95    1.75    ...
                  1.55    1.40    1.25    1.10    0.90    0.75    0.60    0.45
     0.30    0.20 ].';
mf14_a1_E05 = [ 24.30   22.60   21.10   19.85   18.80   17.80   16.90   16.05   15.30
     14.60   13.95   ...
                  13.30   12.70   12.10   11.45   10.95   10.55   10.15   9.75
     9.35    8.90    ...
                  8.50    8.20    7.90    7.55    7.15    6.90    6.60    6.35
     6.05    5.70    ...
                  5.50    5.25    4.95    4.70    4.50    4.35    4.15    3.95
     3.75    3.50    ...
                  3.30    3.15    2.95    2.75    2.50    2.35    2.20    2.00
     1.85    1.70 ].';
mf14_a1_E10 = [ 26.20   24.40   22.80   21.45   20.25   19.20   18.25   17.35   16.65
     15.80   15.20   ...
                  14.55   13.95   13.35   12.75   12.20   11.75   11.25   10.90
     10.50   10.15   ...
                  9.75    9.40    9.05    8.70    8.35    8.05    7.80    7.45
     7.15    6.90    ...
                  6.60    6.35    6.05    5.80    5.55    5.30    5.15    4.95
     4.80    4.55    ...
                  4.35    4.20    3.95    3.80    3.55    3.40    3.20    3.05
     2.90    2.75 ].';
mf14_a2_E00 = [ 16.85   15.80   14.85   13.90   12.95   12.25   11.55   10.85   10.30
     9.70    9.20    ...
                  8.65    8.15    7.70    7.30    6.95    6.60    6.20    5.90
     5.60    5.30    ...
                  5.00    4.75    4.50    4.25    3.95    3.75    3.55    3.35
     3.10    2.95    ...
                  2.80    2.60    2.45    2.35    2.15    2.05    1.85    1.75
     1.65    1.50    ...
                  1.40    1.30    1.20    1.10    1.00    0.90    0.85    0.80
     0.75    0.70 ].';
mf14_a2_E05 = [ 22.85   21.80   20.90   20.00   19.10   18.20   17.35   16.65   16.05
     15.45   14.95   ...
                  14.35   13.90   13.45   13.00   12.60   12.15   11.80   11.40
     11.05   10.65   ...
                  10.35   10.05   9.75    9.45    9.15    8.90    8.65    8.40
     8.15    7.95    ...
                  7.75    7.50    7.30    7.15    6.95    6.80    6.60    6.40
     6.25    6.10    ...
                  6.00    5.80    5.65    5.55    5.40    5.25    5.15    5.05
     4.95    4.85 ].';
mf14_a2_E10 = [ 26.95   25.85   24.70   23.70   22.75   21.95   21.10   20.30   19.65
     19.00   18.35   ...
                  17.75   17.25   16.80   16.35   15.90   15.50   15.10   14.70
     14.35   14.00   ...
                  13.65   13.35   13.00   12.75   12.40   12.15   11.90   11.65
     11.40   11.10   ...
                  10.85   10.65   10.45   10.25   10.05   9.85    9.65    9.45
     9.30    9.10    ...
                  8.95    8.75    8.60    8.45    8.25    8.10    7.95    7.85
     7.65    7.50 ].';
mf14_a3_E00 = [ 12.15   11.70   11.20   10.70   10.30   9.85    9.45    9.10    8.75
     8.45    8.10    ...
                  7.85    7.55    7.25    7.05    6.75    6.45    6.25    6.05
     5.85    5.70    ...
                  5.45    5.30    5.15    4.95    4.80    4.65    4.55    4.45
     4.30    4.20    ...
                  4.00    3.90    3.80    3.70    3.60    3.50    3.40    3.35
     3.25    3.20    ...
                  3.10    3.00    2.95    2.90    2.80    2.75    2.70    2.60
     2.55    2.50 ].';
mf14_a3_E05 = [ 16.80   16.35   15.80   15.40   14.95   14.50   14.10   13.70   13.35
     13.00   12.70   ...
                  12.35   12.05   11.75   11.45   11.20   10.95   10.70   10.35
     10.15   9.95    ...
                  9.70    9.45    9.25    9.10    8.90    8.75    8.55    8.35
     8.20    8.05    ...
                  7.85    7.75    7.55    7.45    7.30    7.15    7.00    6.90
     6.85    6.80    ...
                  6.75    6.70    6.65    6.60    6.60    6.55    6.50    6.45
     6.40    6.40 ].';
```

```
mf14_a3_E10 = [ 20.05   19.50   18.85   18.30   17.75   17.30   16.80   16.35   15.90
      15.60   15.25   ...
                        14.95   14.55   14.25   13.95   13.70   13.35   13.10   12.80
      12.60   12.40   ...
                        12.15   11.95   11.75   11.55   11.35   11.20   11.05   10.85
      10.65   10.50   ...
                        10.35   10.25   10.10   10.00   9.90    9.80    9.65    9.55
      9.45    9.40    ...
                        9.30    9.20    9.15    9.05    8.95    8.90    8.80    8.75
      8.70    8.60 ].';

a0CLMS =
      [];%OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOu
      tputOutputOutputOutputOutput

xmf14_a1 = 5 ./ [26.90:-0.10/50:26.80].'; %(51x1)
smf14_a1 = 3;
vf14_a1_E00 = mf14_a1_E00 .* xmf14_a1 + smf14_a1; %(51x1)
vf14_a1_E05 = mf14_a1_E05 .* xmf14_a1 + smf14_a1; %(51x1)
vf14_a1_E10 = mf14_a1_E10 .* xmf14_a1 + smf14_a1; %(51x1)
a100CLMS = polyfit(r0151,vf14_a1_E00,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput
a105CLMS = polyfit(r0151,vf14_a1_E05,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput
a110CLMS = polyfit(r0151,vf14_a1_E10,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput

xmf14_a2 = 30 ./ [[32.40 .* ones(1,25)] [32.40:-0.20/25:32.20]].'; %(51x1)
smf14_a2 = 15;
vf14_a2_E00 = mf14_a2_E00 .* xmf14_a2 + smf14_a2; %(51x1)
vf14_a2_E05 = mf14_a2_E05 .* xmf14_a2 + smf14_a2; %(51x1)
vf14_a2_E10 = mf14_a2_E10 .* xmf14_a2 + smf14_a2; %(51x1)
a200CLMS = polyfit(r0151,vf14_a2_E00,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput
a205CLMS = polyfit(r0151,vf14_a2_E05,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput
a210CLMS = polyfit(r0151,vf14_a2_E10,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput

xmf14_a3 = 80 ./ [21.50:-0.10/50:21.40].'; %(51x1)
smf14_a3 = 30;
vf14_a3_E00 = mf14_a3_E00 .* xmf14_a3 + smf14_a3; %(51x1)
vf14_a3_E05 = mf14_a3_E05 .* xmf14_a3 + smf14_a3; %(51x1)
vf14_a3_E10 = mf14_a3_E10 .* xmf14_a3 + smf14_a3; %(51x1)
a300CLMS = polyfit(r0151,vf14_a3_E00,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput
a305CLMS = polyfit(r0151,vf14_a3_E05,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput
a310CLMS = polyfit(r0151,vf14_a3_E10,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput

figure;
subplot(3,1,3);
plot(r0151,polyval(a100CLMS,r0151),'-.'); hold on;
plot(r0151,polyval(a105CLMS,r0151),'--');
plot(r0151,polyval(a110CLMS,r0151),'-');
text(0.5,7.5,'1st mode');
ylabel('a_1'); xlabel('m_t/m_0'); grid minor;
subplot(3,1,2);
plot(r0151,polyval(a200CLMS,r0151),'-.'); hold on;
plot(r0151,polyval(a205CLMS,r0151),'--');
plot(r0151,polyval(a210CLMS,r0151),'-');
text(0.5,35,'2nd mode');
ylabel('a_2'); grid minor;
subplot(3,1,1);
plot(r0151,polyval(a300CLMS,r0151),'-.'); hold on;
plot(r0151,polyval(a305CLMS,r0151),'--');
plot(r0151,polyval(a310CLMS,r0151),'-');
text(0.5,100,'3rd mode');title('Figure-14');
ylabel('a_3'); grid minor;
legend('EI_t/EI_0=0','EI_t/EI_0=.5','EI_t/EI_0=1');
hold off;

clear mf14_a* xmf14_a* smf14_a* vf14_a*

% Figure-15
mf15_K01_E00 = [ 13.00  13.10   13.25   13.40   13.50   13.70   13.85   14.00   14.10
      14.25   14.30   ...
```

```
                      14.45   14.55   14.70   14.80   14.90   15.00   15.10   15.20
      15.30   15.35   ...
                      15.40   15.45   15.50   15.55   15.65   15.70   15.75   15.80
      15.85   15.90   ...
                      16.00   16.00   16.05   16.10   16.15   16.20   16.20   16.20
      16.25   16.25   ...
                      16.30   16.30   16.30   16.30   16.35   16.35   16.35   16.35
      16.35   16.40 ].';
mf15_K01_E05 = [ 9.70  9.75    9.80    9.85    9.90    9.95    10.00   10.10   10.15
      10.25   10.25   ...
                      10.30   10.35   10.40   10.45   10.50   10.60   10.65   10.70
      10.70   10.75   ...
                      10.75   10.75   10.80   10.80   10.80   10.85   10.90   10.90
      10.95   10.95   ...
                      11.00   11.05   11.05   11.05   11.10   11.10   11.15   11.15
      11.15   11.15   ...
                      11.20   11.20   11.25   11.25   11.25   11.25   11.25   11.25
      11.30   11.30 ].';
mf15_K01_E10 = [ 7.50  7.55    7.60    7.65    7.70    7.75    7.85    7.90    7.95
      8.05    8.05    ...
                      8.10    8.15    8.25    8.30    8.35    8.40    8.50    8.55
      8.55    8.60    ...
                      8.65    8.70    8.75    8.80    8.80    8.85    8.90    8.95
      9.00    9.05    ...
                      9.15    9.20    9.25    9.25    9.30    9.35    9.35    9.35
      9.35    9.40    ...
                      9.40    9.45    9.45    9.45    9.45    9.45    9.45    9.50
      9.50    9.50 ].';
mf15_K02_E00 = [ 2.55  3.10    3.60    4.15    4.65    5.25    5.80    6.35    6.90
      7.50    8.00    ...
                      8.55    9.05    9.55    10.00   10.50   11.05   11.55   12.00
      12.50   12.95   ...
                      13.50   13.95   14.45   14.95   15.40   15.85   16.30   16.75
      17.15   17.60   ...
                      18.15   18.55   18.95   19.35   19.75   20.15   20.50   20.85
      21.20   21.50   ...
                      21.90   22.20   22.50   22.80   23.05   23.30   23.55   23.80
      24.05   24.25 ].';
mf15_K02_E05 = [ 2.05  2.55    3.10    3.65    4.15    4.75    5.30    5.85    6.40
      6.95    7.55    ...
                      8.10    8.60    9.10    9.65    10.20   10.75   11.20   11.70
      12.20   12.65   ...
                      13.15   13.65   14.15   14.60   15.10   15.55   15.95   16.40
      16.85   17.30   ...
                      17.75   18.15   18.50   18.90   19.30   19.65   19.95   20.30
      20.60   20.90   ...
                      21.20   21.45   21.75   22.00   22.20   22.40   22.55   22.70
      22.85   22.95 ].';
mf15_K02_E10 = [ 1.75  2.35    2.85    3.40    3.95    4.55    5.10    5.60    6.15
      6.70    7.25    ...
                      7.80    8.35    8.85    9.40    9.90    10.45   10.95   11.45
      11.95   12.40   ...
                      12.90   13.40   13.90   14.40   14.85   15.25   15.70   16.15
      16.55   16.95   ...
                      17.40   17.75   18.15   18.55   18.95   19.25   19.60   19.95
      20.30   20.60   ...
                      20.90   21.15   21.45   21.70   21.90   22.10   22.30   22.45
      22.60   22.75 ].';
mf15_K03_E00 = [ 3.50  4.00    4.50    4.95    5.45    5.90    6.40    6.80    7.30
      7.75    8.10    ...
                      8.65    9.05    9.55    10.00   10.45   10.85   11.20   11.65
      12.10   12.45   ...
                      12.90   13.25   13.70   14.05   14.35   14.75   15.10   15.40
      15.80   16.10   ...
                      16.40   16.65   16.95   17.20   17.45   17.65   17.90   18.10
      18.30   18.45   ...
                      18.65   18.80   18.90   19.00   19.10   19.15   19.15   19.20
      19.20   19.25 ].';
mf15_K03_E05 = [ 4.40  4.80    5.20    5.70    6.15    6.60    7.10    7.50    7.95
      8.35    8.85    ...
                      9.25    9.65    10.10   10.55   11.00   11.45   11.80   12.20
      12.75   13.10   ...
                      13.55   13.95   14.35   14.70   15.10   15.55   15.90   16.30
      16.70   17.05   ...
                      17.45   17.80   18.05   18.35   18.65   18.95   19.25   19.45
      19.70   19.90   ...
                      20.05   20.25   20.40   20.50   20.65   20.75   20.85   21.00
      21.10   21.20 ].';
```

```
mf15_K03_E10 = [ 4.75   5.15    5.55    6.00    6.40    6.90    7.30    7.70    8.15
      8.65    9.00    ...
                      9.45    9.90    10.35   10.80   11.20   11.65   12.05   12.45
      12.95   13.30   ...
                      13.75   14.20   14.55   14.95   15.35   15.85   16.20   16.60
      16.95   17.30   ...
                      17.65   18.05   18.35   18.65   18.95   19.20   19.45   19.70
      20.00   20.15   ...
                      20.40   20.60   20.75   20.90   21.05   21.15   21.30   21.40
      21.45   21.50 ].';

K00CLMS =
      [];%OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOu
      tputOutputOutputOutputOutput

xmf15_K01 = 0.20 ./ [20.70:-0.05/50:20.65].'; %(51x1)
smf15_K01 = 1.10;
vf15_K01_E00 = mf15_K01_E00 .* xmf15_K01 + smf15_K01; %(51x1)
vf15_K01_E05 = mf15_K01_E05 .* xmf15_K01 + smf15_K01; %(51x1)
vf15_K01_E10 = mf15_K01_E10 .* xmf15_K01 + smf15_K01; %(51x1)
K0100CLMS = polyfit(r0151,vf15_K01_E00,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput
K0105CLMS = polyfit(r0151,vf15_K01_E05,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput
K0110CLMS = polyfit(r0151,vf15_K01_E10,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput

xmf15_K02 = 3 ./ [[31.10 .* ones(1,25)] [31.10:-0.10/25:31.00]].'; %(51x1)
smf15_K02 = 4;
vf15_K02_E00 = mf15_K02_E00 .* xmf15_K02 + smf15_K02; %(51x1)
vf15_K02_E05 = mf15_K02_E05 .* xmf15_K02 + smf15_K02; %(51x1)
vf15_K02_E10 = mf15_K02_E10 .* xmf15_K02 + smf15_K02; %(51x1)
K0200CLMS = polyfit(r0151,vf15_K02_E00,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput
K0205CLMS = polyfit(r0151,vf15_K02_E05,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput
K0210CLMS = polyfit(r0151,vf15_K02_E10,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput

xmf15_K03 = 10 ./ [25.80:-0.10/50:25.70].'; %(51x1)
smf15_K03 = 8;
vf15_K03_E00 = mf15_K03_E00 .* xmf15_K03 + smf15_K03; %(51x1)
vf15_K03_E05 = mf15_K03_E05 .* xmf15_K03 + smf15_K03; %(51x1)
vf15_K03_E10 = mf15_K03_E10 .* xmf15_K03 + smf15_K03; %(51x1)
K0300CLMS = polyfit(r0151,vf15_K03_E00,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput
K0305CLMS = polyfit(r0151,vf15_K03_E05,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput
K0310CLMS = polyfit(r0151,vf15_K03_E10,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput

figure;
subplot(3,1,3);
plot(r0151,polyval(K0100CLMS,r0151),'-.'); hold on;
plot(r0151,polyval(K0105CLMS,r0151),'--');
plot(r0151,polyval(K0110CLMS,r0151),'-');
text(0.5,1.28,'1st mode');
ylabel('K_0_1'); xlabel('m_t/m_0'); grid minor;
subplot(3,1,2);
plot(r0151,polyval(K0200CLMS,r0151),'-.'); hold on;
plot(r0151,polyval(K0205CLMS,r0151),'--');
plot(r0151,polyval(K0210CLMS,r0151),'-');
text(0.5,6.25,'2nd mode');
ylabel('K_0_2'); grid minor;
subplot(3,1,1);
plot(r0151,polyval(K0300CLMS,r0151),'-.'); hold on;
plot(r0151,polyval(K0305CLMS,r0151),'--');
plot(r0151,polyval(K0310CLMS,r0151),'-');
text(0.5,17,'3rd mode');title('Figure-15');
ylabel('K_0_3'); grid minor;
legend('EI_t/EI_0=0','EI_t/EI_0=.5','EI_t/EI_0=1',2);
hold off;

clear mf15_K0* xmf15_K0* smf15_K0* vf15_K0*

% Figure-16
mf16_Kh11_E00 = [ 25.40 24.00   22.60   21.30   20.15   19.15   18.30   17.45   16.65
      15.95   15.30   ...
```

```
                    14.65   14.15   13.65   13.15   12.65   12.20   11.75   11.40
     11.05   10.75   ...
                    10.35   10.05   9.75    9.40    9.15    8.95    8.75    8.50
     8.30    8.15    ...
                    7.95    7.80    7.65    7.50    7.35    7.25    7.15    7.00
     6.95    6.80    ...
                    6.65    6.60    6.55    6.50    6.40    6.30    6.25    6.20
     6.10    6.05 ].';
mf16_Kh11_E05 = [ 24.35 22.80   21.45   20.10   18.95   17.90   16.90   16.00   15.20
     14.55   13.90   ...
                    13.25   12.55   11.95   11.50   11.00   10.45   10.00   9.65
     9.25    8.95    ...
                    8.60    8.25    8.00    7.70    7.40    7.20    7.00    6.80
     6.60    6.40    ...
                    6.20    6.05    5.90    5.80    5.65    5.55    5.40    5.30
     5.15    5.10    ...
                    4.95    4.90    4.85    4.80    4.75    4.65    4.60    4.60
     4.50    4.45 ].';
mf16_Kh11_E10 = [ 23.25 21.60   20.20   18.90   17.80   16.75   15.80   15.00   14.25
     13.55   12.80   ...
                    12.15   11.50   10.95   10.30   9.90    9.40    9.05    8.65
     8.35    8.05    ...
                    7.70    7.45    7.15    6.90    6.70    6.45    6.25    6.05
     5.85    5.70    ...
                    5.55    5.40    5.25    5.10    5.00    4.90    4.80    4.75
     4.65    4.60    ...
                    4.50    4.45    4.45    4.40    4.35    4.30    4.25    4.25
     4.20    4.15 ].';
mf16_Kh12_E00 = [ 4.75  5.15    5.55    5.95    6.35    6.85    7.15    7.55    7.90
     8.30    8.70    ...
                    9.05    9.40    9.85    10.30   10.65   10.95   11.35   11.70
     12.10   12.45   ...
                    12.80   13.15   13.55   13.95   14.30   14.65   15.05   15.40
     15.75   16.05   ...
                    16.40   16.80   17.10   17.45   17.80   18.20   18.55   18.85
     19.20   19.55   ...
                    19.85   20.10   20.40   20.75   21.00   21.35   21.65   21.95
     22.20   22.50 ].';
mf16_Kh12_E05 = [ 6.50  6.90    7.30    7.70    8.05    8.40    8.75    9.10    9.45
     9.80    10.15   ...
                    10.50   10.85   11.25   11.60   11.95   12.35   12.70   13.00
     13.35   13.70   ...
                    14.00   14.35   14.65   15.00   15.35   15.75   16.05   16.45
     16.80   17.15   ...
                    17.45   17.80   18.20   18.55   18.90   19.25   19.60   19.90
     20.30   20.65   ...
                    20.95   21.25   21.60   21.95   22.30   22.65   23.00   23.30
     23.60   23.90 ].';
mf16_Kh12_E10 = [ 7.35  7.75    8.20    8.60    8.90    9.30    9.70    10.00   10.40
     10.75   11.10   ...
                    11.45   11.85   12.20   12.60   12.95   13.35   13.70   14.05
     14.40   14.80   ...
                    15.15   15.55   15.95   16.30   16.65   17.05   17.40   17.80
     18.20   18.55   ...
                    18.90   19.30   19.65   20.05   20.45   20.80   21.10   21.45
     21.80   22.20   ...
                    22.55   22.90   23.25   23.60   23.95   24.30   24.65   25.00
     25.35   25.70 ].';
mf16_Kh13_E00 = [ 0.75  1.00    1.25    1.50    1.70    1.90    2.05    2.30    2.50
     2.70    2.95    ...
                    3.10    3.30    3.55    3.75    3.95    4.20    4.40    4.60
     4.80    5.00    ...
                    5.20    5.35    5.50    5.65    5.85    6.00    6.10    6.25
     6.40    6.50    ...
                    6.65    6.80    6.90    7.05    7.20    7.30    7.45    7.55
     7.65    7.75    ...
                    7.85    7.95    8.05    8.15    8.20    8.25    8.35    8.45
     8.50    8.55 ].';
mf16_Kh13_E05 = [ 2.50  2.70    2.90    3.15    3.35    3.55    3.80    4.05    4.25
     4.50    4.70    ...
                    4.90    5.10    5.35    5.60    5.75    5.90    6.20    6.35
     6.55    6.70    ...
                    6.95    7.15    7.35    7.55    7.75    7.90    8.05    8.20
     8.40    8.60    ...
                    8.75    8.95    9.05    9.25    9.40    9.60    9.75    9.90
     10.00   10.20   ...
                    10.30   10.45   10.55   10.70   10.80   10.90   11.00   11.10
     11.20   11.30 ].';
```

```
mf16_Kh13_E10 = [ 3.35  3.55    3.75    3.95    4.15    4.35    4.55    4.75    4.95
      5.15    5.35    ...
                        5.55    5.75    6.00    6.20    6.40    6.60    6.80    7.00
      7.25    7.40    ...
                        7.65    7.80    8.00    8.20    8.45    8.60    8.80    8.95
      9.15    9.30    ...
                        9.45    9.60    9.80    9.95    10.10   10.30   10.50   10.60
      10.75   10.90   ...
                        11.05   11.20   11.35   11.45   11.55   11.65   11.75   11.90
      11.95   12.05 ].';

Kh10CLMS =
      [];%OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOu
      tputOutputOutputOutputOutput

xmf16_Kh11 = 0.5 ./ [[29.40:0.10/10:29.50] [29.50 .* ones(1,29)] [29.50:-
      0.10/10:29.40]].'; %(51x1)
smf16_Kh11 = 1.5;
vf16_Kh11_E00 = mf16_Kh11_E00 .* xmf16_Kh11 + smf16_Kh11; %(51x1)
vf16_Kh11_E05 = mf16_Kh11_E05 .* xmf16_Kh11 + smf16_Kh11; %(51x1)
vf16_Kh11_E10 = mf16_Kh11_E10 .* xmf16_Kh11 + smf16_Kh11; %(51x1)
Kh1100CLMS = polyfit(r0151,vf16_Kh11_E00,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput
Kh1105CLMS = polyfit(r0151,vf16_Kh11_E05,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput
Kh1110CLMS = polyfit(r0151,vf16_Kh11_E10,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput

xmf16_Kh12 = 2.5 ./ [[29.50 .* ones(1,40)] [29.50:-0.15/10:29.35]].'; %(51x1)
smf16_Kh12 = 6;
vf16_Kh12_E00 = mf16_Kh12_E00 .* xmf16_Kh12 + smf16_Kh12; %(51x1)
vf16_Kh12_E05 = mf16_Kh12_E05 .* xmf16_Kh12 + smf16_Kh12; %(51x1)
vf16_Kh12_E10 = mf16_Kh12_E10 .* xmf16_Kh12 + smf16_Kh12; %(51x1)
Kh1200CLMS = polyfit(r0151,vf16_Kh12_E00,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput
Kh1205CLMS = polyfit(r0151,vf16_Kh12_E05,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput
Kh1210CLMS = polyfit(r0151,vf16_Kh12_E10,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput

xmf16_Kh13 = 12 ./ [17.65:-0.15/50:17.50].'; %(51x1)
smf16_Kh13 = 14;
vf16_Kh13_E00 = mf16_Kh13_E00 .* xmf16_Kh13 + smf16_Kh13; %(51x1)
vf16_Kh13_E05 = mf16_Kh13_E05 .* xmf16_Kh13 + smf16_Kh13; %(51x1)
vf16_Kh13_E10 = mf16_Kh13_E10 .* xmf16_Kh13 + smf16_Kh13; %(51x1)
Kh1300CLMS = polyfit(r0151,vf16_Kh13_E00,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput
Kh1305CLMS = polyfit(r0151,vf16_Kh13_E05,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput
Kh1310CLMS = polyfit(r0151,vf16_Kh13_E10,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput

figure;
subplot(3,1,3);
plot(r0151,polyval(Kh1100CLMS,r0151),'-.'); hold on;
plot(r0151,polyval(Kh1105CLMS,r0151),'--');
plot(r0151,polyval(Kh1110CLMS,r0151),'-');
text(0.5,1.95,'1st mode');
ylabel('Kh_1_1'); xlabel('m_t/m_0'); grid minor;
subplot(3,1,2);
plot(r0151,polyval(Kh1200CLMS,r0151),'-.'); hold on;
plot(r0151,polyval(Kh1205CLMS,r0151),'--');
plot(r0151,polyval(Kh1210CLMS,r0151),'-');
text(0.5,8,'2nd mode');
ylabel('Kh_1_2'); grid minor;
subplot(3,1,1);
plot(r0151,polyval(Kh1300CLMS,r0151),'-.'); hold on;
plot(r0151,polyval(Kh1305CLMS,r0151),'--');
plot(r0151,polyval(Kh1310CLMS,r0151),'-');
text(0.5,23,'3rd mode');title('Figure-16');
ylabel('Kh_1_3'); grid minor;
legend('EI_t/EI_0=0','EI_t/EI_0=.5','EI_t/EI_0=1',2);
hold off;

clear mf16_Kh* xmf16_Kh* smf16_Kh* vf16_Kh*

% NONROTATING UNIFORM CANTILEVER BEAMS WITH A MASS AT THE TIP:
      ==================================================================
% Figure-17
```

```
mf17_teta1sq = [ 21.80  20.60   19.45   18.35   17.40   16.55   15.80   15.10   14.45
      13.90   13.35   ...
                        12.90   12.60   12.25   11.95   11.70   11.40   11.10   10.90
      10.65   10.40   ...
                        10.20   10.00   9.75    9.55    9.30    9.10    8.95    8.80
      8.65    8.50    ...
                        8.40    8.25    8.15    8.05    7.95    7.85    7.80    7.70
      7.65    7.60    ...
                        7.55    7.45    7.40    7.40    7.35    7.30    7.30    7.25
      7.20    7.20 ].';
mf17_teta2sq = [ 17.45  14.75   12.50   10.60   9.20    8.05    7.20    6.65    6.25
      5.90    5.50    ...
                        5.20    4.90    4.70    4.50    4.30    4.10    3.95    3.85
      3.75    3.65    ...
                        3.55    3.45    3.35    3.25    3.20    3.10    3.00    2.95
      2.90    2.80    ...
                        2.75    2.70    2.65    2.60    2.55    2.50    2.45    2.40
      2.35    2.30    ...
                        2.25    2.20    2.20    2.20    2.15    2.15    2.10    2.10
      2.05    2.05 ].';
mf17_teta3sq = [ 14.35  11.80   9.40    7.40    5.85    4.70    3.90    3.40    3.05
      2.80    2.55    ...
                        2.40    2.20    2.05    1.95    1.85    1.75    1.70    1.65
      1.60    1.55    ...
                        1.50    1.45    1.40    1.30    1.25    1.20    1.15    1.15
      1.10    1.05    ...
                        1.00    1.00    0.95    0.95    0.90    0.90    0.90    0.90
      0.85    0.85    ...
                        0.85    0.85    0.85    0.85    0.80    0.80    0.80    0.80
      0.75    0.75 ].';

xmf17_teta1sq = 4 ./ [25.00:-0.15/50:24.85].'; %(51x1)
smf17_teta1sq = 0.0;
xmf17_teta2sq = 10 ./ [25.00:-0.10/50:24.90].'; %(51x1)
smf17_teta2sq = 15;
xmf17_teta3sq = 20 ./ [24.85:-0.10/50:24.75].'; %(51x1)
smf17_teta3sq = 50;
vf17_teta1sq = mf17_teta1sq .* xmf17_teta1sq + smf17_teta1sq; %(51x1)
vf17_teta2sq = mf17_teta2sq .* xmf17_teta2sq + smf17_teta2sq; %(51x1)
vf17_teta3sq = mf17_teta3sq .* xmf17_teta3sq + smf17_teta3sq; %(51x1)
teta0sqNUCMt = [];
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutpu
      tOutputOutputOut
teta1sqNUCMt = polyfit(r0251,vf17_teta1sq,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput
teta2sqNUCMt = polyfit(r0251,vf17_teta2sq,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput
teta3sqNUCMt = polyfit(r0251,vf17_teta3sq,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput

figure;
subplot(3,1,3);
plot(r0251,polyval(teta1sqNUCMt,r0251),'-'); hold on;
text(0.5,3.5,'1st mode');
ylabel('\theta_1^2'); xlabel('r'); grid minor;
subplot(3,1,2);
plot(r0251,polyval(teta2sqNUCMt,r0251),'-'); hold on;
text(0.5,23,'2nd mode');
ylabel('\theta_2^2'); grid minor;
subplot(3,1,1);
plot(r0251,polyval(teta3sqNUCMt,r0251),'-'); hold on;
text(0.5,63,'3rd mode');title('Figure-17');
ylabel('\theta_3^2'); grid minor;
hold off;

clear mf17_teta*sq xmf17_teta*sq smf17_teta*sq vf17_teta*sq

% Figure-18
mf18_K01 = [ 12.75  13.05   13.35   13.60   13.90   14.15   14.45   14.70   14.95   15.15
      15.40   ...
                13.60   15.80   16.05   16.25   16.45   16.55   16.75   16.85   16.95
      17.05   ...
                17.15   17.20   17.25   17.25   17.25 ].';%
mf18_K02 = [ 2.85   3.55    4.25    4.95    5.65    6.35    7.10    7.85    8.60    9.30
      10.10   ...
                10.85   11.65   12.40   13.20   14.00   14.85   15.65   16.55   17.40
      18.30   ...
                19.20   20.15   21.10   22.15   23.20 ].';%
```

```
mf18_K03 = [ 1.50   2.15    2.90    3.60    4.35    5.05    5.80    6.55    7.25    8.00
      8.70    ...
               9.45    10.15   10.85   11.60   12.35   13.05   13.75   14.45   15.15
      15.75   ...
               16.45   17.05   17.70   18.30   18.90 ].';%

xmf18_K01 = 0.02 ./ [19.80 .* ones(1,26)].'; %(26x1)
smf18_K01 = 1.18;
xmf18_K02 = 15 ./ [29.70:-0.05/25:29.65].'; %(26x1)
smf18_K02 = 5;
xmf18_K03 = 40 ./ [19.65 .* ones(1,26)].'; %(26x1)
smf18_K03 = 15;
vf18_K01 = mf18_K01 .* xmf18_K01 + smf18_K01; %(26x1)
vf18_K02 = mf18_K02 .* xmf18_K02 + smf18_K02; %(26x1)
vf18_K03 = mf18_K03 .* xmf18_K03 + smf18_K03; %(26x1)
K00NUCMt = [];
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutpu
      tOutputOutputOutpu
K01NUCMt = polyfit(r0126,vf18_K01,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput
K02NUCMt = polyfit(r0126,vf18_K02,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput
K03NUCMt = polyfit(r0126,vf18_K03,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput

figure;
subplot(3,1,3);
plot(r0151,polyval(K01NUCMt,r0151),'-'); hold on;
text(0.5,1.198,'1st mode');
ylabel('K_0_1'); xlabel('r'); grid minor;
subplot(3,1,2);
plot(r0151,polyval(K02NUCMt,r0151),'-'); hold on;
text(0.5,18,'2nd mode');
ylabel('K_0_2'); grid minor;
subplot(3,1,1);
plot(r0151,polyval(K03NUCMt,r0151),'-'); hold on;
text(0.5,50,'3rd mode');title('Figure-18');
ylabel('K_0_3'); grid minor;
hold off;

clear mf18_K0* xmf18_K0* smf18_K0* vf18_K0*

% NONROTATING UNIFORM HINGED BEAMS WITH A MASS AT THE TIP:
      ======================================================================
% Figure-19
mf19_teta1sq = [ 25.60  21.00   17.55   15.45   13.65   12.15   10.75   9.70    8.85
      8.15    7.60    ...
                   7.05    6.60    6.15    5.80    5.45    5.15    4.90    4.65
      4.40    4.25    ...
                   4.05    3.85    3.70    3.55    3.40    3.25    3.10    3.00
      2.90    2.80    ...
                   2.70    2.60    2.50    2.40    2.30    2.25    2.15    2.05
      2.00    1.95    ...
                   1.85    1.85    1.80    1.75    1.65    1.60    1.55    1.55
      1.50    1.45 ].';
mf19_teta2sq = [ 24.70  20.90   17.70   15.40   13.40   11.75   10.65   9.90    9.30
      8.90    8.60    ...
                   8.30    8.10    7.90    7.75    7.60    7.45    7.35    7.25
      7.10    7.00    ...
                   6.90    6.80    6.75    6.70    6.60    6.50    6.45    6.35
      6.30    6.25    ...
                   6.20    6.15    6.10    6.05    6.00    6.00    5.95    5.95
      5.90    5.90    ...
                   5.85    5.85    5.85    5.80    5.80    5.80    5.80    5.80
      5.80    5.80 ].';
mf19_teta3sq = [ 16.30  12.15   10.25   8.80    7.75    6.90    6.35    6.00    5.70
      5.45    5.20    ...
                   5.05    4.90    4.75    4.65    4.60    4.55    4.50    4.45
      4.40    4.35    ...
                   4.30    4.25    4.25    4.20    4.15    4.10    4.05    4.00
      3.95    3.95    ...
                   3.90    3.90    3.85    3.85    3.80    3.80    3.75    3.75
      3.75    3.70    ...
                   3.70    3.70    3.70    3.70    3.70    3.65    3.65    3.65
      3.65    3.65 ].';

xmf19_teta1sq = 6 ./ [[28.75:0.05/25:28.80] [28.798:-0.048/24:28.750]].'; %(51x1)
smf19_teta1sq = 10;
xmf19_teta2sq = 15 ./ [[28.80 .* ones(1,40)] [28.80:-0.05/10:28.75]].'; %(51x1)
```

```
smf19_teta2sq = 37;
xmf19_teta3sq = 20 ./ [17.20 .* ones(1,51)].'; %(51x1)
smf19_teta3sq = 85;
vf19_teta1sq = mf19_teta1sq .* xmf19_teta1sq + smf19_teta1sq; %(51x1)
vf19_teta2sq = mf19_teta2sq .* xmf19_teta2sq + smf19_teta2sq; %(51x1)
vf19_teta3sq = mf19_teta3sq .* xmf19_teta3sq + smf19_teta3sq; %(51x1)
teta0sqNUHMt = 0;
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutpu
      tOutputOutputOutputOu
teta1sqNUHMt = polyfit(r0251,vf19_teta1sq,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput
teta2sqNUHMt = polyfit(r0251,vf19_teta2sq,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput
teta3sqNUHMt = polyfit(r0251,vf19_teta3sq,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput

figure;
subplot(3,1,3);
plot(r0251,polyval(teta1sqNUHMt,r0251),'-'); hold on;
text(0.5,15,'1st mode');
ylabel('\theta_1^2'); xlabel('r'); grid minor;
subplot(3,1,2);
plot(r0251,polyval(teta2sqNUHMt,r0251),'-'); hold on;
text(0.5,48,'2nd mode');
ylabel('\theta_2^2'); grid minor;
subplot(3,1,1);
plot(r0251,polyval(teta3sqNUHMt,r0251),'-'); hold on;
text(0.5,103,'3rd mode');title('Figure-19');
ylabel('\theta_3^2'); grid minor;
hold off;

clear mf19_teta*sq xmf19_teta*sq smf19_teta*sq vf19_teta*sq

% Figure-20
mf20_K01 = [ 1.55   2.10    2.70    3.35    4.05    4.85    5.65    6.45    7.25    8.10
      8.95    ...
                 9.90    10.75   11.65   12.50   13.40   14.30   15.20   16.05   16.95
      17.80   ...
                 18.70   19.55   20.45   21.35   22.15 ].';%
mf20_K02 = [ 0.75   1.50    2.25    3.05    3.85    4.65    5.50    6.35    7.20    8.10
      8.95    ...
                 9.80    10.70   11.55   12.40   13.30   14.20   15.10   15.95   16.85
      17.70   ...
                 18.60   19.50   20.40   21.20   22.10 ].';%
mf20_K03 = [ 1.10   1.80    2.50    3.20    3.95    4.70    5.45    6.25    7.00    7.75
      8.50    ...
                 9.30    10.10   10.90   11.70   12.50   13.25   14.05   14.85   15.60
      16.35   ...
                 17.15   17.90   18.70   19.50   20.25 ].';%

xmf20_K01 = 20 ./ [22.45:-0.10/25:22.35].'; %(26x1)
smf20_K01 = 5;
xmf20_K02 = 80 ./ [22.45 .* ones(1,26)].'; %(26x1)
smf20_K02 = 15;
xmf20_K03 = 200 ./ [[22.300:-0.045/18:22.255] [22.252:-0.102/6:22.150]].'; %(26x1)
smf20_K03 = 25;
vf20_K01 = mf20_K01 .* xmf20_K01 + smf20_K01; %(26x1)
vf20_K02 = mf20_K02 .* xmf20_K02 + smf20_K02; %(26x1)
vf20_K03 = mf20_K03 .* xmf20_K03 + smf20_K03; %(26x1)
K00NUHMt = 1;
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutpu
      tOutputOutpu
K01NUHMt = polyfit(r0226,vf20_K01,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput
K02NUHMt = polyfit(r0226,vf20_K02,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput
K03NUHMt = polyfit(r0226,vf20_K03,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput

figure;
subplot(3,1,3);
plot(r0251,polyval(K01NUHMt,r0251),'-'); hold on;
text(0.5,23,'1st mode');
ylabel('K_0_1'); xlabel('r'); grid minor;
subplot(3,1,2);
plot(r0251,polyval(K02NUHMt,r0251),'-'); hold on;
text(0.5,85,'2nd mode');
ylabel('K_0_2'); grid minor;
subplot(3,1,1);
```

```
plot(r0251,polyval(K03NUHMt,r0251),'-'); hold on;
text(0.5,210,'3rd mode');title('Figure-20');
ylabel('K_0_3'); grid minor;
hold off;

clear mf20_K0* xmf20_K0* smf20_K0* vf20_K0*


% Figure-21
mf21_Kh10_r000 = [ 55.10    51.70   48.65   46.15   43.95   41.95   40.10   38.60   37.30
      36.25   35.25   ...
                            34.40   33.65   33.00   32.35   31.80   31.25   30.75   30.30
      29.90   29.50   ...
                            29.15   28.80   28.45   28.10   27.95 ].';%
mf21_Kh10_r002 = [ 44.40    42.25   40.40   38.85   37.50   36.30   35.15   34.25   33.35
      32.70   32.00   ...
                            31.40   30.85   30.30   29.80   29.40   28.90   28.50   28.15
      27.80   27.45   ...
                            27.15   26.80   26.55   26.30   26.15 ].';%
mf21_Kh10_r005 = [ 34.65    33.50   32.50   31.65   30.85   30.20   29.55   29.00   28.50
      28.05   27.65   ...
                            27.25   26.90   26.55   26.20   25.95   25.65   25.40   25.15
      25.00   24.80   ...
                            24.70   24.50   24.45   24.35   24.25 ].';%
mf21_Kh10_r010 = [ 25.30    24.90   24.55   24.25   24.05   23.80   23.55   23.40   23.20
      23.10   22.95   ...
                            22.80   22.65   22.55   22.35   22.25   22.10   21.95   21.85
      21.80   21.75   ...
                            21.70   21.55   21.55   21.55   21.50 ].';%
mf21_Kh10_r025 = [ 13.80    13.90   14.05   14.20   14.35   14.45   14.50   14.60   14.70
      14.80   14.90   ...
                            15.00   15.10   15.15   15.20   15.30   15.35   15.45   15.50
      15.55   15.65   ...
                            15.75   15.80   15.85   15.90   15.95 ].';%
mf21_Kh10_r050 = [ 7.95     8.05    8.20    8.35    8.50    8.65    8.80    8.95    9.10
      9.30    9.45    ...
                            9.55    9.70    9.80    9.90    10.00   10.10   10.25   10.35
      10.40   10.50   ...
                            10.65   10.70   10.85   10.95   11.05 ].';%
mf21_Kh10_r100 = [ 4.15     4.30    4.40    4.55    4.70    4.80    4.95    5.05    5.20
      5.35    5.50    ...
                            5.60    5.75    5.80    5.85    5.95    6.00    6.00    6.05
      6.15    6.25    ...
                            6.30    6.40    6.55    6.65    6.75 ].';%
mf21_Kh10_r150 = [ 2.65     2.75    2.90    3.00    3.15    3.25    3.35    3.45    3.50
      3.60    3.65    ...
                            3.75    3.85    3.90    4.00    4.10    4.15    4.25    4.35
      4.40    4.50    ...
                            4.55    4.60    4.70    4.80    4.85 ].';%

xmf21 = 1 ./ [54.40 .* ones(1,26)].'; %(26x1)
smf21 = 1;
vf21_Kh10_r000 = mf21_Kh10_r000 .* xmf21 + smf21; %(26x1)
vf21_Kh10_r002 = mf21_Kh10_r002 .* xmf21 + smf21; %(26x1)
vf21_Kh10_r005 = mf21_Kh10_r005 .* xmf21 + smf21; %(26x1)
vf21_Kh10_r010 = mf21_Kh10_r010 .* xmf21 + smf21; %(26x1)
vf21_Kh10_r025 = mf21_Kh10_r025 .* xmf21 + smf21; %(26x1)
vf21_Kh10_r050 = mf21_Kh10_r050 .* xmf21 + smf21; %(26x1)
vf21_Kh10_r100 = mf21_Kh10_r100 .* xmf21 + smf21; %(26x1)
vf21_Kh10_r150 = mf21_Kh10_r150 .* xmf21 + smf21; %(26x1)
Kh10r000 = polyfit(r0126, vf21_Kh10_r000,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput
Kh10r002 = polyfit(r0126, vf21_Kh10_r002,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput
Kh10r005 = polyfit(r0126, vf21_Kh10_r005,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput
Kh10r010 = polyfit(r0126, vf21_Kh10_r010,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput
Kh10r025 = polyfit(r0126, vf21_Kh10_r025,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput
Kh10r050 = polyfit(r0126, vf21_Kh10_r050,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput
Kh10r100 = polyfit(r0126, vf21_Kh10_r100,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput
Kh10r150 = polyfit(r0126, vf21_Kh10_r150,8);
      %OutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput

figure;
plot(r0126,polyval(Kh10r000,r0126),'-'); hold on;
```

```
plot(r0126,polyval(Kh10r002,r0126),'-'); hold on;
plot(r0126,polyval(Kh10r005,r0126),'-'); hold on;
plot(r0126,polyval(Kh10r010,r0126),'-'); hold on;
plot(r0126,polyval(Kh10r025,r0126),'-'); hold on;
plot(r0126,polyval(Kh10r050,r0126),'-'); hold on;
plot(r0126,polyval(Kh10r100,r0126),'-'); hold on;
plot(r0126,polyval(Kh10r150,r0126),'-'); hold on;
text(0.2,1.9,'r');
text(0.2,1.78,'0.0');
text(0.2,1.67,'.02');
text(0.2,1.56,'.05');
text(0.2,1.45,'.10');
text(0.2,1.28,'.25');
text(0.2,1.18,'.50');
text(0.2,1.11,'1.00');
text(0.2,1.03,'1.50');
ylabel('Kh_1_0'); xlabel('m_t/m_0'); grid minor;
title('Figure-21');
hold off;

clear mf21_Kh10_r*
clear vf21_Kh10_r*
clear xmf21 smf21

%
clear r01* r02*
```

## C. MATLAB® CODE FOR YNTEMOSHCOEF.M WITH PLOT STATEMENTS

```
%YNTEMOSHCOEF.M: This file generates the polynomial coefficients for the mode results of
some characteristic
% nonrotating hinged/cantilever beams with linear mass and stiffness distibutions and
zero hinge offset.
% All results are from the Yntema Report Table III & IV.
% Run this file in a cleared workspace and save the workspace as YNTEMOSHCOEF.MAT which
is to be used in
% YNTEMOSHNR.M function for the determination of mode shapes of corresponding rotating
beams.
%
% -- 1Lt.H.E.AKIN, Turkish Army, Naval Postgraduate School, Monterey, CA, USA
% -- Master of Science in Aeronautical Enginering (Avionics) Thesis, SEP2002
%=====================================================================================
==================
sttnC1 = [0.0:1:10].' ./ 10; %(11x1)
sttnC2 = [0.5:1:9.5].' ./ 10; %(10x1)
sttnH1 = [0.0:1:15].' ./ 15; %(16x1)
sttnH2 = [0.5:1:14.5].' ./ 15; %(15x1)
xx = [0:0.01:1];
%=====================================================================================
==================
% TABLE-III  CANTILEVER
%------------------------------------------------------
%c1 m = m_0  &  EI = EI_0:   (exact solution)
t3mEIy1 = [0.0000    0.0168  0.0639  0.1365  0.2299  0.3395  0.4611  0.5959  0.7255
0.8624  1.0000].';
t3mEIy1d = [0.0000   0.3274  0.6065  0.8378  1.0226  1.1631  1.2627  1.3266  1.3612
1.3745  1.3765].';
t3mEIy1dd = [3.5160  3.0332  2.5508  2.0775  1.6214  1.1938  0.8083  0.4799  0.2246
0.0590  0.0000].';
t3mEIy2 = [0.0000    -.0926  -.3011  -.5261  -.6835  -.7137  -.5895  -.3171  0.0700
0.5238  1.0000].';
t3mEIy2d = [0.0000   -1.6776 -2.3240 -2.0351 -1.0114 0.4531  2.0194  3.3709  4.2876
4.7095  4.7808].';
t3mEIy2dd = [-22.0345  -11.5406  -1.5432  6.9860  12.9888  15.7253  15.0599  11.5931
6.6336  2.0411  0.0000].';
t3mEIy3 = [0.0000    0.2281  0.6045  0.7562  0.5259  0.0197  -.4738  -.6574  -.3949
0.2285  1.0000].';
t3mEIy3d = [0.0000   3.7655  3.1181  -.3551  -4.0599 -5.5520 -3.7912 0.3568  4.7354
7.3385  7.8487].';
t3mEIy3dd = [61.6972  14.0984  -24.3627  -40.5613  -29.2300  1.2145  32.4481  46.6579
37.2963  14.0713  0.0000].';
CmEIy1 = polyfit(sttnC1,t3mEIy1,8);
CmEIy1d = polyfit(sttnC1,t3mEIy1d,8);
CmEIy1dd = polyfit(sttnC1,t3mEIy1dd,8);
```

```
CmEIy2 = polyfit(sttnC1,t3mEIy2,8);
CmEIy2d = polyfit(sttnC1,t3mEIy2d,8);
CmEIy2dd = polyfit(sttnC1,t3mEIy2dd,8);
CmEIy3 = polyfit(sttnC1,t3mEIy3,8);
CmEIy3d = polyfit(sttnC1,t3mEIy3d,8);
CmEIy3dd = polyfit(sttnC1,t3mEIy3dd,8);

figure; hold on;
subplot(1,3,1); hold on;
plot(xx,polyval(CmEIy1,xx),'b-','linewidth',2);
plot(xx,polyval(CmEIy1d,xx),'b-.','linewidth',2);
plot(xx,polyval(CmEIy1dd,xx),'b:','linewidth',2);
grid minor;
legend('MODE 1','1st Derivative','2nd Derivative',0);
title('CANTILEVER m = m_0  &  EI = EI_0:   (exact solution)');
subplot(1,3,2); hold on;
plot(xx,polyval(CmEIy2,xx),'r-','linewidth',2);
plot(xx,polyval(CmEIy2d,xx),'r-.','linewidth',2);
plot(xx,polyval(CmEIy2dd,xx),'r:','linewidth',2);
grid minor;
legend('MODE 2','1st Derivative','2nd Derivative',0);
subplot(1,3,3); hold on;
plot(xx,polyval(CmEIy3,xx),'m-','linewidth',2);
plot(xx,polyval(CmEIy3d,xx),'m-.','linewidth',2);
plot(xx,polyval(CmEIy3dd,xx),'m:','linewidth',2);
grid minor;
legend('MODE 3','1st Derivative','2nd Derivative',0);

clear t3mEIy*
%c2 m = m_0  &  EI = EI_0(1-xhat/2):
t3mEIx2y1 = [0.0000     0.0151  0.0582  0.1263  0.2159  0.3234  0.4450  0.5769  0.7154
0.8572  1.0000].';
t3mEIx2y1d = [0.1508    0.4312  0.6807  0.8962  1.0753  1.2163  1.3189  1.3847  1.4182
1.4278].';%
t3mEIx2y1dd = [3.0852   2.8044  2.4942  2.1551  1.7910  1.4102  1.0259  0.6580  0.3350
0.0967  0.0000].';
t3mEIx2y2 = [0.0000     -.0778  -.2612  -.4723  -.6361  -.6907  -.5971  -.3470  0.0355
0.5022  1.0000].';
t3mEIx2y2d = [-.7779    -1.8336 -2.1111 -1.6386 -.5456  .9362   2.5002  3.8252  4.6675
4.9778].';%
t3mEIx2y2dd = [-17.9825  -10.5955  -2.7533  4.8322  11.1252  15.0784  15.9132  13.4617
8.4741  2.8758  0.0000].';
t3mEIx2y3 = [0.0000     0.1871  0.5246  0.7053  0.5543  0.1097  -.3927  -.6432  -.4394
0.1846  1.0000].';
t3mEIx2y3d = [1.8715    3.3746  1.8048  -1.5083 -4.4456 -5.0240 -2.5050 2.0377  6.2401
8.1542].';%
t3mEIx2y3dd = [49.0088  14.8158  -16.8271  -34.9664  -31.1090  -6.4431  26.1221  47.5240
43.8272  18.6063  0.0000].';
CmEIx2y1 = polyfit(sttnC1,t3mEIx2y1,8);
CmEIx2y1d = polyfit(sttnC2,t3mEIx2y1d,8);
CmEIx2y1dd = polyfit(sttnC1,t3mEIx2y1dd,8);
CmEIx2y2 = polyfit(sttnC1,t3mEIx2y2,8);
CmEIx2y2d = polyfit(sttnC2,t3mEIx2y2d,8);
CmEIx2y2dd = polyfit(sttnC1,t3mEIx2y2dd,8);
CmEIx2y3 = polyfit(sttnC1,t3mEIx2y3,8);
CmEIx2y3d = polyfit(sttnC2,t3mEIx2y3d,8);
CmEIx2y3dd = polyfit(sttnC1,t3mEIx2y3dd,8);

figure; hold on;
subplot(1,3,1); hold on;
plot(xx,polyval(CmEIx2y1,xx),'b-','linewidth',2);
plot(xx,polyval(CmEIx2y1d,xx),'b-.','linewidth',2);
plot(xx,polyval(CmEIx2y1dd,xx),'b:','linewidth',2);
grid minor;
legend('MODE 1','1st Derivative','2nd Derivative',0);
title('CANTILEVER m = m_0  &  EI = EI_0(1-xhat/2):');
subplot(1,3,2); hold on;
plot(xx,polyval(CmEIx2y2,xx),'r-','linewidth',2);
plot(xx,polyval(CmEIx2y2d,xx),'r-.','linewidth',2);
plot(xx,polyval(CmEIx2y2dd,xx),'r:','linewidth',2);
grid minor;
legend('MODE 2','1st Derivative','2nd Derivative',0);
subplot(1,3,3); hold on;
plot(xx,polyval(CmEIx2y3,xx),'m-','linewidth',2);
plot(xx,polyval(CmEIx2y3d,xx),'m-.','linewidth',2);
plot(xx,polyval(CmEIx2y3dd,xx),'m:','linewidth',2);
grid minor;
legend('MODE 3','1st Derivative','2nd Derivative',0);
```

```
clear t3mEIx2y*
%c3 m = m_0  &  EI = EI_0(1-xhat):
t3mEIxy1 = [0.0000      0.0125  0.0491  0.1088  0.1900  0.2908  0.4093  0.5426  0.6879
0.8417  1.0000].';
t3mEIxy1d = [0.1247     0.3667  0.5966  0.8117  1.0088  1.1841  1.3337  1.4531  1.5378
1.5829].';%
t3mEIxy1dd = [2.5176    2.4201  2.2996  2.1512  1.9705  1.7531  1.4955  1.1943  0.8469
0.4513  0.0000].';
t3mEIxy2 = [0.0000      -.0518  -.1818  -.3471  -.4989  -.5863  -.5622  -.3908  -.0570
0.4244  1.0000].';
t3mEIxy2d = [-.5178     -1.3001 -1.6530 -1.5179 -.8741  0.2412  1.7135  3.3378  4.8142
5.7561].';%
t3mEIxy2dd = [-11.4951  -7.8970 -3.5774 1.3316  6.4680  11.2476 14.8933 16.4929 15.0877
9.7945  0.0000].';
t3mEIxy3 = [0.0000      0.1079  0.3289  0.5018  0.4938  0.2551  -.1374  -.4768  -.4899
0.0240  1.0000].';
t3mEIxy3d = [1.0789     2.2098  1.7288  -.0799  -2.3871 -3.9249 -3.3940 -.1303  5.1385
9.7602].';%
t3mEIxy3dd = [26.4729   11.4265 -5.0368 -18.7863    -24.1585    -16.5097    4.6998
33.2025 54.9056 49.9675 0.0000].';
CmEIxy1 = polyfit(sttnC1,t3mEIxy1,8);
CmEIxy1d = polyfit(sttnC2,t3mEIxy1d,8);
CmEIxy1dd = polyfit(sttnC1,t3mEIxy1dd,8);
CmEIxy2 = polyfit(sttnC1,t3mEIxy2,8);
CmEIxy2d = polyfit(sttnC2,t3mEIxy2d,8);
CmEIxy2dd = polyfit(sttnC1,t3mEIxy2dd,8);
CmEIxy3 = polyfit(sttnC1,t3mEIxy3,8);
CmEIxy3d = polyfit(sttnC2,t3mEIxy3d,8);
CmEIxy3dd = polyfit(sttnC1,t3mEIxy3dd,8);

figure; hold on;
subplot(1,3,1); hold on;
plot(xx,polyval(CmEIxy1,xx),'b-','linewidth',2);
plot(xx,polyval(CmEIxy1d,xx),'b-.','linewidth',2);
plot(xx,polyval(CmEIxy1dd,xx),'b:','linewidth',2);
grid minor;
legend('MODE 1','1st Derivative','2nd Derivative',0);
title('CANTILEVER m = m_0  &  EI = EI_0(1-xhat):');
subplot(1,3,2); hold on;
plot(xx,polyval(CmEIxy2,xx),'r-','linewidth',2);
plot(xx,polyval(CmEIxy2d,xx),'r-.','linewidth',2);
plot(xx,polyval(CmEIxy2dd,xx),'r:','linewidth',2);
grid minor;
legend('MODE 2','1st Derivative','2nd Derivative',0);
subplot(1,3,3); hold on;
plot(xx,polyval(CmEIxy3,xx),'m-','linewidth',2);
plot(xx,polyval(CmEIxy3d,xx),'m-.','linewidth',2);
plot(xx,polyval(CmEIxy3dd,xx),'m:','linewidth',2);
grid minor;
legend('MODE 3','1st Derivative','2nd Derivative',0);

clear t3mEIxy*
%c4 m = m_0(1-xhat/2)  &  EI = EI_0:
t3mx2EIy1 = [0.0000     0.0173  0.0654  0.1392  0.2336  0.3438  0.4654  0.5946  0.7283
0.8639  1.0000].';
t3mx2EIy1d = [0.1733    0.4812  0.7375  0.9437  1.1020  1.2164  1.2922  1.3362  1.3562
1.3613].';%
t3mx2EIy1dd = [3.5961   3.0782  2.5638  2.0614  1.5835  1.1442  0.7580  0.4391  0.1999
0.0509  0.0000].';
t3mx2EIy2 = [0.0000     -.0853  -.2738  -.4704  -.5968  -.6018  -.4665  -.2022  0.1574
0.5701  1.0000].';
t3mx2EIy2d = [-.8526    -1.8857 -1.9658 -1.2637 -.0501  1.3533  2.6423  3.5963  4.1269
4.2990].';%
t3mx2EIy2dd = [-20.5149 -10.2793    -.6581  7.2497  12.4090 14.2921 13.0765 9.6161
5.2513  1.5298  0.0000].';
t3mx2EIy3 = [0.0000     0.2031  0.5227  0.6145  0.3624  -.0993  -.4960  -.5848  -.2857
0.3018  1.0000].';
t3mx2EIy3d = [2.0307    3.1960  0.9179  -2.5205 -4.6174 -3.9663 -.8880  2.9901  5.8749
6.9825].';%
t3mx2EIy3dd = [56.8179  10.7772 -24.7575    -36.5657    -22.1859    6.8125  32.2249
40.3627 29.5511 10.3404 0.0000].';
Cmx2EIy1 = polyfit(sttnC1,t3mx2EIy1,8);
Cmx2EIy1d = polyfit(sttnC2,t3mx2EIy1d,8);
Cmx2EIy1dd = polyfit(sttnC1,t3mx2EIy1dd,8);
Cmx2EIy2 = polyfit(sttnC1,t3mx2EIy2,8);
Cmx2EIy2d = polyfit(sttnC2,t3mx2EIy2d,8);
Cmx2EIy2dd = polyfit(sttnC1,t3mx2EIy2dd,8);
Cmx2EIy3 = polyfit(sttnC1,t3mx2EIy3,8);
Cmx2EIy3d = polyfit(sttnC2,t3mx2EIy3d,8);
```

```
Cmx2EIy3dd = polyfit(sttnC1,t3mx2EIy3dd,8);

figure; hold on;
subplot(1,3,1); hold on;
plot(xx,polyval(Cmx2EIy1,xx),'b-','linewidth',2);
plot(xx,polyval(Cmx2EIy1d,xx),'b-.','linewidth',2);
plot(xx,polyval(Cmx2EIy1dd,xx),'b:','linewidth',2);
grid minor;
legend('MODE 1','1st Derivative','2nd Derivative',0);
title('CANTILEVER m = m_0(1-xhat/2)  &  EI = EI_0:');
subplot(1,3,2); hold on;
plot(xx,polyval(Cmx2EIy2,xx),'r-','linewidth',2);
plot(xx,polyval(Cmx2EIy2d,xx),'r-.','linewidth',2);
plot(xx,polyval(Cmx2EIy2dd,xx),'r:','linewidth',2);
grid minor;
legend('MODE 2','1st Derivative','2nd Derivative',0);
subplot(1,3,3); hold on;
plot(xx,polyval(Cmx2EIy3,xx),'m-','linewidth',2);
plot(xx,polyval(Cmx2EIy3d,xx),'m-.','linewidth',2);
plot(xx,polyval(Cmx2EIy3dd,xx),'m:','linewidth',2);
grid minor;
legend('MODE 3','1st Derivative','2nd Derivative',0);

clear t3mx2EIy*
%c5 m = m_0(1-xhat/2)  &  EI = EI_0(1-xhat/2):
t3mx2EIx2y1 = [0.0000   0.0155  0.0595  0.1287  0.2194  0.3276  0.4493  0.5808  0.7183
0.8587  1.0000].';
t3mx2EIx2y1d = [0.1546  0.4404  0.6921  0.9068  1.0823  1.2180  1.3144  1.3748  1.4046
1.4129].';%
t3mx2EIx2y1dd = [3.1688 2.8585  2.5168  2.1467  1.7557  1.3563  0.9647  0.6034  0.2979
0.0836  0.0000].';
t3mx2EIx2y2 = [0.0000   -.0722  -.2395  -.4259  -.5607  -.5889  -.4802  -.2337  0.1244
0.5502  1.0000].';
t3mx2EIx2y2d = [-.7222  -1.6728 -1.8638 -1.3480 -.2821  1.0866  2.4651  3.5808  4.2586
4.4978].';%
t3mx2EIx2y2dd = [-16.8561   -9.5343 -1.8680 5.2897  10.8679 13.9363 14.0175 11.3083
6.7837  2.1918  0.0000].';
t3mx2EIx2y3 = [0.0000   0.1686  0.4581  0.5817  0.4004  -.0162  -.4330  -.5851  -.3333
0.2605  1.0000].';
t3mx2EIx2y3d = [1.6861  2.8950  1.2361  -1.8134 -4.1661 -4.1669 -1.5214 2.5189  5.9375
7.3951].';%
t3mx2EIx2y3dd = [45.1241    11.7757 -17.8557    -32.3125    -24.9765    -.2658  27.5563
42.1923 35.4166 13.9636 0.0000].';%??????????????????????????1.0000].'; i corrected
Cmx2EIx2y1 = polyfit(sttnC1,t3mx2EIx2y1,8);
Cmx2EIx2y1d = polyfit(sttnC2,t3mx2EIx2y1d,8);
Cmx2EIx2y1dd = polyfit(sttnC1,t3mx2EIx2y1dd,8);
Cmx2EIx2y2 = polyfit(sttnC1,t3mx2EIx2y2,8);
Cmx2EIx2y2d = polyfit(sttnC2,t3mx2EIx2y2d,8);
Cmx2EIx2y2dd = polyfit(sttnC1,t3mx2EIx2y2dd,8);
Cmx2EIx2y3 = polyfit(sttnC1,t3mx2EIx2y3,8);
Cmx2EIx2y3d = polyfit(sttnC2,t3mx2EIx2y3d,8);
Cmx2EIx2y3dd = polyfit(sttnC1,t3mx2EIx2y3dd,8);

figure; hold on;
subplot(1,3,1); hold on;
plot(xx,polyval(Cmx2EIx2y1,xx),'b-','linewidth',2);
plot(xx,polyval(Cmx2EIx2y1d,xx),'b-.','linewidth',2);
plot(xx,polyval(Cmx2EIx2y1dd,xx),'b:','linewidth',2);
grid minor;
legend('MODE 1','1st Derivative','2nd Derivative',0);
title('CANTILEVER m = m_0(1-xhat/2)  &  EI = EI_0(1-xhat/2):');
subplot(1,3,2); hold on;
plot(xx,polyval(Cmx2EIx2y2,xx),'r-','linewidth',2);
plot(xx,polyval(Cmx2EIx2y2d,xx),'r-.','linewidth',2);
plot(xx,polyval(Cmx2EIx2y2dd,xx),'r:','linewidth',2);
grid minor;
legend('MODE 2','1st Derivative','2nd Derivative',0);
subplot(1,3,3); hold on;
plot(xx,polyval(Cmx2EIx2y3,xx),'m-','linewidth',2);
plot(xx,polyval(Cmx2EIx2y3d,xx),'m-.','linewidth',2);
plot(xx,polyval(Cmx2EIx2y3dd,xx),'m:','linewidth',2);
grid minor;
legend('MODE 3','1st Derivative','2nd Derivative',0);

clear t3mx2EIx2y*
%c6 m = m_0(1-xhat/2)  &  EI = EI_0(1-xhat):
t3mx2EIxy1 = [0.0000    0.0128  0.0505  0.1115  0.1940  0.2960  0.4150  0.5482  0.6924
0.8443  1.0000].';
```

```
t3mx2EIxy1d = [0.1284   0.3764  0.6098  0.8255  1.0200  1.1899  1.3318  1.4423  1.5184
1.5575].';%
t3mx2EIxy1dd = [2.5984  2.4797  2.3338  2.1566  1.9455  1.6994  1.4185  1.1048  0.7609
0.3911  0.0000].';
t3mx2EIxy2 = [0.0000    -.0491  -.1706  -.3206  -.4511  -.5143  -.4692  -.2887  0.0339
0.4796  1.0000].';
t3mx2EIxy2d = [-.4911   -1.2146 -1.5002 -1.3046 -.6320  0.4505  1.8050  3.2265  4.4562
5.2043].';%
t3mx2EIxy2dd = [-11.0011    -7.2927 -2.8931 1.9575  6.7820  10.9396 13.7170 14.4325
12.5408 7.7223  0.0000].';
t3mx2EIxy3 = [0.0000    0.1010  0.2992  0.4352  0.3901  0.1398  -.2167  -.4771  -.4135
0.1077  1.0000].';
t3mx2EIxy3d = [0 1.0099   1.9826  1.3592  -1.4506 -2.5029 -3.5647 -2.6041 0.6360  5.2112
8.9234].';%0 added by me!
t3mx2EIxy3dd = [25.2707     9.7964  -6.5750 -18.9042    -21.5881    -11.5213    9.3586
33.1815 47.5995 39.7770 0.0000].';
Cmx2EIxy1 = polyfit(sttnC1,t3mx2EIxy1,8);
Cmx2EIxy1d = polyfit(sttnC2,t3mx2EIxy1d,8);
Cmx2EIxy1dd = polyfit(sttnC1,t3mx2EIxy1dd,8);
Cmx2EIxy2 = polyfit(sttnC1,t3mx2EIxy2,8);
Cmx2EIxy2d = polyfit(sttnC2,t3mx2EIxy2d,8);
Cmx2EIxy2dd = polyfit(sttnC1,t3mx2EIxy2dd,8);
Cmx2EIxy3 = polyfit(sttnC1,t3mx2EIxy3,8);
Cmx2EIxy3d = polyfit([0;sttnC2],t3mx2EIxy3d,8);
Cmx2EIxy3dd = polyfit(sttnC1,t3mx2EIxy3dd,8);

figure; hold on;
subplot(1,3,1); hold on;
plot(xx,polyval(Cmx2EIxy1,xx),'b-','linewidth',2);
plot(xx,polyval(Cmx2EIxy1d,xx),'b-.','linewidth',2);
plot(xx,polyval(Cmx2EIxy1dd,xx),'b:','linewidth',2);
grid minor;
legend('MODE 1','1st Derivative','2nd Derivative',0);
title('CANTILEVER m = m_0(1-xhat/2)  &  EI = EI_0(1-xhat):');
subplot(1,3,2); hold on;
plot(xx,polyval(Cmx2EIxy2,xx),'r-','linewidth',2);
plot(xx,polyval(Cmx2EIxy2d,xx),'r-.','linewidth',2);
plot(xx,polyval(Cmx2EIxy2dd,xx),'r:','linewidth',2);
grid minor;
legend('MODE 2','1st Derivative','2nd Derivative',0);
subplot(1,3,3); hold on;
plot(xx,polyval(Cmx2EIxy3,xx),'m-','linewidth',2);
plot(xx,polyval(Cmx2EIxy3d,xx),'m-.','linewidth',2);
plot(xx,polyval(Cmx2EIxy3dd,xx),'m:','linewidth',2);
grid minor;
legend('MODE 3','1st Derivative','2nd Derivative',0);

clear t3mx2EIxy*
%c7 m = m_0(1-xhat)  &  EI = EI_0:
t3mxEIy1 = [0.0000      0.0194  0.0722  0.1515  0.2506  0.3636  0.4853  0.6120  0.7408
0.8704  1.0000].';
t3mxEIy1d = [0.1939     0.5284  0.7929  0.9911  1.1294  1.2175  1.2665  1.2887  1.2955
1.2962].';%
t3mxEIy1dd = [4.0558    3.3450  2.6451  1.9813  1.3838  0.8803  0.4904  0.2218  0.0680
0.0068  0.0000].';
t3mxEIy2 = [0.0000      -.0607  -.1842  -.2908  -.3237  -.2570  -.0946  0.1394  0.4144
0.7056  1.0000].';
t3mxEIy2d = [-.6069     -1.2348 -1.0659 -.3292  0.6672  1.6238  2.3397  2.7503  2.9121
2.9437].';%
t3mxEIy2dd = [-15.2162  -6.2023 1.8823  7.6453  10.2183 9.7358  7.2110  4.0543  1.5122
0.2282  0.0000].';
t3mxEIy3 = [0.0000      0.1174  0.2666  0.2375  0.0253  -.2135  -.3018  -.1633  0.1591
0.5699  1.0000].';
t3mxEIy3d = [1.1740     1.4916  -.2901  -2.1229 -2.3870 -.8834  1.3851  3.2238  4.1081
4.3010].';%
t3mxEIy3dd = [35.5410   2.2191  -19.6272    -19.7496    -2.7888 15.9480 23.7445 18.8323
8.5833  1.4574  0.0000].';
CmxEIy1 = polyfit(sttnC1,t3mxEIy1,8);
CmxEIy1d = polyfit(sttnC2,t3mxEIy1d,8);
CmxEIy1dd = polyfit(sttnC1,t3mxEIy1dd,8);
CmxEIy2 = polyfit(sttnC1,t3mxEIy2,8);
CmxEIy2d = polyfit(sttnC2,t3mxEIy2d,8);
CmxEIy2dd = polyfit(sttnC1,t3mxEIy2dd,8);
CmxEIy3 = polyfit(sttnC1,t3mxEIy3,8);
CmxEIy3d = polyfit(sttnC2,t3mxEIy3d,8);
CmxEIy3dd = polyfit(sttnC1,t3mxEIy3dd,8);

figure; hold on;
subplot(1,3,1); hold on;
```

```
plot(xx,polyval(CmxEIy1,xx),'b-','linewidth',2);
plot(xx,polyval(CmxEIy1d,xx),'b-.','linewidth',2);
plot(xx,polyval(CmxEIy1dd,xx),'b:','linewidth',2);
grid minor;
legend('MODE 1','1st Derivative','2nd Derivative',0);
title('CANTILEVER m = m_0(1-xhat)  &  EI = EI_0:');
subplot(1,3,2); hold on;
plot(xx,polyval(CmxEIy2,xx),'r-','linewidth',2);
plot(xx,polyval(CmxEIy2d,xx),'r-.','linewidth',2);
plot(xx,polyval(CmxEIy2dd,xx),'r:','linewidth',2);
grid minor;
legend('MODE 2','1st Derivative','2nd Derivative',0);
subplot(1,3,3); hold on;
plot(xx,polyval(CmxEIy3,xx),'m-','linewidth',2);
plot(xx,polyval(CmxEIy3d,xx),'m-.','linewidth',2);
plot(xx,polyval(CmxEIy3dd,xx),'m:','linewidth',2);
grid minor;
legend('MODE 3','1st Derivative','2nd Derivative',0);

clear t3mxEIy*
%c8 m = m_0(1-xhat)  &  EI = EI_0(1-xhat/2):
t3mxEIx2y1 = [0.0000    0.0175  0.0666  0.1419  0.2383  0.3502  0.4728  0.6017  0.7338
0.8668  1.0000].';
t3mxEIx2y1d = [0.1752   0.4903  0.7539  0.9635  1.1195  1.2257  1.2893  1.3203  1.3307
1.3318].';%
t3mxEIx2y1dd = [3.6228  3.1505  2.6356  2.0961  1.5602  1.0620  0.6358  0.3106  0.1034
0.0113  0.0000].';
t3mxEIx2y2 = [0.0000    -.0521  -.1636  -.2680  -.3109  -.2604  -.1118  0.1170  0.3955
0.6955  1.0000].';
t3mxEIx2y2d = [-.5212   -1.1148 -1.0442 -.4288  .5049  1.4867  2.2874  2.7856  2.9997
3.0448].';%
t3mxEIx2y2dd = [-12.6640    -5.9369 0.8058  6.3475  9.5678  10.0140 8.1088  4.9844
2.0367  0.3379  0.0000].';
t3mxEIx2y3 = [0.0000    0.0990  0.2389  0.2357  0.0566  -.1781  -.2941  -.1865  0.1270
0.5504  1.0000].';
t3mxEIx2y3d = [0.9900   1.3986  -.0314  -1.7917 -2.3466 -1.1598 1.0759  3.1346  4.2345
4.4958].';%
t3mxEIx2y3dd = [28.6145 3.6038  -15.6241    -18.9598    -6.0124 12.4980 23.4526 21.2750
10.8683 2.0489  0.0000].';
CmxEIx2y1 = polyfit(sttnC1,t3mxEIx2y1,8);
CmxEIx2y1d = polyfit(sttnC2,t3mxEIx2y1d,8);
CmxEIx2y1dd = polyfit(sttnC1,t3mxEIx2y1dd,8);
CmxEIx2y2 = polyfit(sttnC1,t3mxEIx2y2,8);
CmxEIx2y2d = polyfit(sttnC2,t3mxEIx2y2d,8);
CmxEIx2y2dd = polyfit(sttnC1,t3mxEIx2y2dd,8);
CmxEIx2y3 = polyfit(sttnC1,t3mxEIx2y3,8);
CmxEIx2y3d = polyfit(sttnC2,t3mxEIx2y3d,8);
CmxEIx2y3dd = polyfit(sttnC1,t3mxEIx2y3dd,8);

figure; hold on;
subplot(1,3,1); hold on;
plot(xx,polyval(CmxEIx2y1,xx),'b-','linewidth',2);
plot(xx,polyval(CmxEIx2y1d,xx),'b-.','linewidth',2);
plot(xx,polyval(CmxEIx2y1dd,xx),'b:','linewidth',2);
grid minor;
legend('MODE 1','1st Derivative','2nd Derivative',0);
title('CANTILEVER m = m_0(1-xhat)  &  EI = EI_0(1-xhat/2):');
subplot(1,3,2); hold on;
plot(xx,polyval(CmxEIx2y2,xx),'r-','linewidth',2);
plot(xx,polyval(CmxEIx2y2d,xx),'r-.','linewidth',2);
plot(xx,polyval(CmxEIx2y2dd,xx),'r:','linewidth',2);
grid minor;
legend('MODE 2','1st Derivative','2nd Derivative',0);
subplot(1,3,3); hold on;
plot(xx,polyval(CmxEIx2y3,xx),'m-','linewidth',2);
plot(xx,polyval(CmxEIx2y3d,xx),'m-.','linewidth',2);
plot(xx,polyval(CmxEIx2y3dd,xx),'m:','linewidth',2);
grid minor;
legend('MODE 3','1st Derivative','2nd Derivative',0);

clear t3mxEIx2y*
%c9 m = m_0(1-xhat)  &  EI = EI_0(1-xhat):
t3mxEIxy1 = [0.0000     0.0151  0.0584  0.1271  0.2176  0.3261  0.4484  0.5804  0.7182
0.8589  1.0000].';
t3mxEIxy1d = [0.1507    0.4337  0.6869  0.9052  1.0846  1.2227  1.3198  1.3788  1.4061
1.4116].';%
t3mxEIxy1dd = [3.0750   2.8299  2.5321  2.1832  1.7937  1.3813  0.9705  0.5903  0.2735
0.0547  0.0000].';%????????????????????1.0000].'; i corrected
```

```
t3mxEIxy2 = [0.0000     -.0385  -.1266  -.2190  -.2720  -.2507  -.1368  0.0684  0.3458
0.6657  1.0000].';
t3mxEIxy2d = [-.3854  -.8801  -.9247  -.5297  .2128  1.1393  2.0523  2.7735  3.1987
3.3434].';%
t3mxEIxy2dd = [-8.9963  -4.9931 -.4387  4.0280  7.5634  9.4334  9.2835  7.3013  4.2407
1.3128  0.0000].';
t3mxEIxy3 = [0.0000     0.0671  0.1769  0.2049  0.0969  -.0965  -.2458  -.2175  0.0451
0.4861  1.0000].';
t3mxEIxy3d = [0.6710    1.0985  0.2793  -1.0798 -1.9343 -1.4924 0.2725  2.6264  4.4100
5.1389].';%
t3mxEIxy3dd = [18.1439  4.1930  -8.8006 -14.5066    -9.2434 4.3850  18.4222 24.3923
18.9564 6.8571  0.0000].';
CmxEIxy1 = polyfit(sttnC1,t3mxEIxy1,8);
CmxEIxy1d = polyfit(sttnC2,t3mxEIxy1d,8);
CmxEIxy1dd = polyfit(sttnC1,t3mxEIxy1dd,8);
CmxEIxy2 = polyfit(sttnC1,t3mxEIxy2,8);
CmxEIxy2d = polyfit(sttnC2,t3mxEIxy2d,8);
CmxEIxy2dd = polyfit(sttnC1,t3mxEIxy2dd,8);
CmxEIxy3 = polyfit(sttnC1,t3mxEIxy3,8);
CmxEIxy3d = polyfit(sttnC2,t3mxEIxy3d,8);
CmxEIxy3dd = polyfit(sttnC1,t3mxEIxy3dd,8);

figure; hold on;
subplot(1,3,1); hold on;
plot(xx,polyval(CmxEIxy1,xx),'b-','linewidth',2);
plot(xx,polyval(CmxEIxy1d,xx),'b-.','linewidth',2);
plot(xx,polyval(CmxEIxy1dd,xx),'b:','linewidth',2);
grid minor;
legend('MODE 1','1st Derivative','2nd Derivative',0);
title('CANTILEVER m = m_0(1-xhat)  &  EI = EI_0(1-xhat):');
subplot(1,3,2); hold on;
plot(xx,polyval(CmxEIxy2,xx),'r-','linewidth',2);
plot(xx,polyval(CmxEIxy2d,xx),'r-.','linewidth',2);
plot(xx,polyval(CmxEIxy2dd,xx),'r:','linewidth',2);
grid minor;
legend('MODE 2','1st Derivative','2nd Derivative',0);
subplot(1,3,3); hold on;
plot(xx,polyval(CmxEIxy3,xx),'m-','linewidth',2);
plot(xx,polyval(CmxEIxy3d,xx),'m-.','linewidth',2);
plot(xx,polyval(CmxEIxy3dd,xx),'m:','linewidth',2);
grid minor;
legend('MODE 3','1st Derivative','2nd Derivative',0);

clear t3mxEIxy*
%
figure; hold on;
plot(xx,polyval(CmEIy1,xx),'b-','linewidth',2);
plot(xx,polyval(CmEIx2y1,xx),'b:','linewidth',2);
plot(xx,polyval(CmEIxy1,xx),'b-.','linewidth',2);
plot(xx,polyval(Cmx2EIy1,xx),'m-','linewidth',2);
plot(xx,polyval(Cmx2EIx2y1,xx),'m:','linewidth',2);
plot(xx,polyval(Cmx2EIxy1,xx),'m-.','linewidth',2);
plot(xx,polyval(CmxEIy1,xx),'r-','linewidth',2);
plot(xx,polyval(CmxEIx2y1,xx),'r:','linewidth',2);
plot(xx,polyval(CmxEIxy1,xx),'r-.','linewidth',2); grid on;
title('CANTILEVER BEAM FIRST BENDING MODE COMPARISON:');
grid minor;
legend('m_r_o_o_t = m_t_i_p & EI_r_o_o_t = EI_t_i_p','m_r_o_o_t = m_t_i_p & EI_t_i_p =
EI_r_o_o_t/2', ...
    'm_r_o_o_t = m_t_i_p & EI_t_i_p = 0','m_t_i_p = m_r_o_o_t/2 & EI_r_o_o_t = EI_t_i_p',
...
    'm_t_i_p = m_r_o_o_t/2 & EI_t_i_p = EI_r_o_o_t/2','m_t_i_p = m_r_o_o_t/2 & EI_t_i_p =
0', ...
    'm_t_i_p = 0 & EI_r_o_o_t = EI_t_i_p','m_t_i_p = 0 & EI_t_i_p =
EI_r_o_o_t/2','m_t_i_p = 0 & EI_t_i_p = 0',2);
%=========================================================================================
==================
% TABLE-IV  NR HINGED
%-------------------------------------------------------
%h1 m = m_0  &  EI = EI_0:
t4mEIy1 = [0.0000       -.1778  -.3428  -.4830  -.5879  -.6494  -.6620  -.6233  -.5339  -
.3973  -.2195  ...
                        -.0083  0.2274  0.4788  0.7382  1.0000].';
t4mEIy1d = [-2.6675     -2.4751 -2.1026 -1.5737 -.9222  -.1890  0.5809  1.3413  2.0490
2.6669  3.1679  ...
                        3.5357  3.7713  3.8904  3.9269].';%
t4mEIy1dd = [0.0000     2.9017  5.6183  7.9748  9.8240  11.0559 11.6059 11.4610 10.6619
9.3030  7.5279  ...
                        5.5239  3.5133  1.7444  0.4821  0.0000].';
```

```
t4mEIy2 = [0.0000       0.3217  0.5731  0.6997  0.6745  0.5034  0.2246  -.1000  -.3977  -
.6006  -.6595  ...
                        -.5667  -.2935  0.0847  0.5309  1.0000].';
t4mEIy2d = [4.8253      3.7708  1.9000  -.3784  -2.5660 -4.1825 -4.8692 -4.4659 -3.0431 -
.8832  1.5878  ...
                        3.9028  5.6720  6.6930  7.0367].';%
t4mEIy2dd = [0.0000     -16.1228    -28.5821    -34.8100    -33.4256    -24.7037    -
10.5043    6.1398  ...
                        21.7025 32.9430 37.6597 35.2256 26.7997 15.2316 4.6639
0.0000].';
t4mEIy3 = [0.0000       -.4342  -.6814  -.6298  -.3017  0.1561  0.5389  0.6758  0.5070
0.1106  -.3311  ...
                        -.6112  -.5878  -.2400  0.3365  1.0000].';
t4mEIy3d = [-6.5137     -3.7078 0.7747  4.9212  6.8673  5.7417  2.0541  -2.5328 -5.9464 -
6.6245 -4.2021 ...
                        0.3507  5.2169  8.6477  9.9528].';%
t4mEIy3dd = [0.0000     43.4748 69.7829 64.4864 30.1462 -17.7598    -57.7175    -71.7578
-53.4956    ...
                        -10.8931    37.3827 70.4947 75.1702 52.3442 18.2343 0.0000].';
HmEIy1 = polyfit(sttnH1,t4mEIy1,8);
HmEIy1d = polyfit(sttnH2,t4mEIy1d,8);
HmEIy1dd = polyfit(sttnH1,t4mEIy1dd,8);
HmEIy2 = polyfit(sttnH1,t4mEIy2,8);
HmEIy2d = polyfit(sttnH2,t4mEIy2d,8);
HmEIy2dd = polyfit(sttnH1,t4mEIy2dd,8);
HmEIy3 = polyfit(sttnH1,t4mEIy3,8);
HmEIy3d = polyfit(sttnH2,t4mEIy3d,8);
HmEIy3dd = polyfit(sttnH1,t4mEIy3dd,8);

figure; hold on;
subplot(1,3,1); hold on;
plot(xx,polyval(HmEIy1,xx),'b-','linewidth',2);
plot(xx,polyval(HmEIy1d,xx),'b-.','linewidth',2);
plot(xx,polyval(HmEIy1dd,xx),'b:','linewidth',2);
grid minor;
legend('MODE 1','1st Derivative','2nd Derivative',0);
title('HINGED m = m_0  &  EI = EI_0:');
subplot(1,3,2); hold on;
plot(xx,polyval(HmEIy2,xx),'r-','linewidth',2);
plot(xx,polyval(HmEIy2d,xx),'r-.','linewidth',2);
plot(xx,polyval(HmEIy2dd,xx),'r:','linewidth',2);
grid minor;
legend('MODE 2','1st Derivative','2nd Derivative',0);
subplot(1,3,3); hold on;
plot(xx,polyval(HmEIy3,xx),'m-','linewidth',2);
plot(xx,polyval(HmEIy3d,xx),'m-.','linewidth',2);
plot(xx,polyval(HmEIy3dd,xx),'m:','linewidth',2);
grid minor;
legend('MODE 3','1st Derivative','2nd Derivative',0);

clear t4mEIy*
%h2 m = m_0  &  EI = EI_0(1-xhat/2):
t4mEIx2y1 = [0.0000     -.1601  -.3104  -.4411  -.5430  -.6079  -.6293  -.6030  -.5272  -
.4031  -.2347  ...
                        -.0283  0.2076  0.4635  0.7300  1.0000].';
t4mEIx2y1d = [-2.4011   -2.2545 -1.9610 -1.5283 -.9732  -.3211  0.3949  1.1363  1.8610
2.5269  3.0959  ...
                        3.5382  3.8377  3.9981  4.0501].';%
t4mEIx2y1dd = [0.0000   2.1968  4.4128  6.5121  8.3585  9.8221  10.7894 11.1731 10.9231
10.0375 8.5728  ...
                        6.6547  4.4883  2.3693  0.6982  0.0000].';
t4mEIx2y2 = [0.0000     0.2823  0.5112  0.6413  0.6441  0.5151  0.2771  -.0227  -.3201  -
.5466  -.6433  ...
                        -.5747  -.3369  0.0406  0.5047  1.0000].';
t4mEIx2y2d = [4.2349    3.4337  1.9512  0.0414  -1.9342 -3.5700 -4.4964 -4.4618 -3.3977 -
1.4512 1.0303  ...
                        3.5665  5.6618  6.9620  7.4296].';%
t4mEIx2y2dd = [0.0000   -12.1633    -22.5509    -29.0968    -30.1375    -24.9921    -
14.2022    0.4449  ...
                        16.1636 29.6539 37.8479 38.6874 31.8978 19.5720 6.4608
0.0000].';
t4mEIx2y3 = [0.0000     -.3806  -.6187  -.6118  -.3523  0.0612  0.4585  0.6662  0.5843
0.2385  -.2199  ...
                        -.5723  -.6286  -.3172  0.2820  1.0000].';
t4mEIx2y3d = [-5.7084   -3.5718 0.0330  3.8918  6.2024  5.9598  3.1149  -1.2129 -5.1621 -
6.8767 -5.2857 ...
                        -.8435  4.6710  8.9872  10.7702].';%
t4mEIx2y3dd = [0.0000   32.8485 57.1675 59.0998 36.3014 -3.3443 -43.9333    -67.5091    -
61.8610    -26.7887    ...
```

```
                        24.2333 68.7554 85.5398 66.4744 25.3995 0.0000].';
HmEIx2y1 = polyfit(sttnH1,t4mEIx2y1,8);
HmEIx2y1d = polyfit(sttnH2,t4mEIx2y1d,8);
HmEIx2y1dd = polyfit(sttnH1,t4mEIx2y1dd,8);
HmEIx2y2 = polyfit(sttnH1,t4mEIx2y2,8);
HmEIx2y2d = polyfit(sttnH2,t4mEIx2y2d,8);
HmEIx2y2dd = polyfit(sttnH1,t4mEIx2y2dd,8);
HmEIx2y3 = polyfit(sttnH1,t4mEIx2y3,8);
HmEIx2y3d = polyfit(sttnH2,t4mEIx2y3d,8);
HmEIx2y3dd = polyfit(sttnH1,t4mEIx2y3dd,8);

figure; hold on;
subplot(1,3,1); hold on;
plot(xx,polyval(HmEIx2y1,xx),'b-','linewidth',2);
plot(xx,polyval(HmEIx2y1d,xx),'b-.','linewidth',2);
plot(xx,polyval(HmEIx2y1dd,xx),'b:','linewidth',2);
grid minor;
legend('MODE 1','1st Derivative','2nd Derivative',0);
title('HINGED m = m_0  &  EI = EI_0(1-xhat/2):');
subplot(1,3,2); hold on;
plot(xx,polyval(HmEIx2y2,xx),'r-','linewidth',2);
plot(xx,polyval(HmEIx2y2d,xx),'r-.','linewidth',2);
plot(xx,polyval(HmEIx2y2dd,xx),'r:','linewidth',2);
grid minor;
legend('MODE 2','1st Derivative','2nd Derivative',0);
subplot(1,3,3); hold on;
plot(xx,polyval(HmEIx2y3,xx),'m-','linewidth',2);
plot(xx,polyval(HmEIx2y3d,xx),'m-.','linewidth',2);
plot(xx,polyval(HmEIx2y3dd,xx),'m:','linewidth',2);
grid minor;
legend('MODE 3','1st Derivative','2nd Derivative',0);

clear t4mEIx2y*
%h3 m = m_0  &  EI = EI_0(1-xhat):
t4mEIxy1 = [0.0000      -.1241  -.2425  -.3492  -.4377  -.5015  -.5344  -.5308  -.4860  -
.3966  -.2609  ...
                        -.0795  0.1447  0.4059  0.6951  1.0000].';
t4mEIxy1d = [-1.8612    -1.7765 -1.6000 -1.3272 -.9572  -.4940  0.0539  0.6722  1.3411
2.0346  2.7212  ...
                        3.3634  3.9182  4.3385  4.5730].';%
t4mEIxy1dd = [0.0000    1.2599  2.6412  4.0919  5.5538  6.9600  8.2356  9.2994  10.0650
10.4435 10.3457 ...
                        9.6854  8.3825  6.3673  3.5845  0.0000].';
t4mEIxy2 = [0.0000      0.1893  0.3518  0.4622  0.4997  0.4526  0.3215  0.1222  -.1135  -
.3402  -.5027  ...
                        -.5451  -.4204  -.1047  0.3896  1.0000].';
t4mEIxy2d = [2.8391     2.4377  1.6557  0.5632  -.7071  -1.9666 -2.9887 -3.5362 -3.3992 -
2.4389 -.6351  ...
                        1.8695  4.7359  7.4140  9.1568].';%
t4mEIxy2dd = [0.0000    -6.0403 -11.8108    -16.5435    -19.2804    -19.1626    -15.6121
-8.4580 1.9007  ...
                        14.3995 27.2556 38.0134 43.7041 41.1267 27.2670 0.0000].';
t4mEIxy3 = [0.0000      -.2299  -.3984  -.4489  -.3552  -.1335  0.1449  0.3882  0.4934
0.3964  0.1086  ...
                        -.2615  -.5209  -.4517  -.0822  1.0000].';
t4mEIxy3d = [-3.4482    -2.5277 -.7583  1.4065  3.2949  4.2065  3.6496  1.5779  -1.4555 -
4.3171 -5.5508 ...
                        -3.8912 1.0372  8.0094  13.7667].';%
t4mEIxy3dd = [0.0000    13.9649 27.3446 33.6311 29.5841 14.6758 -7.9710 -31.5748    -
46.9233    -44.9846    ...
                        -20.5659    23.7570 74.7164 108.0221    92.3208 0.0000].';
HmEIxy1 = polyfit(sttnH1,t4mEIxy1,8);
HmEIxy1d = polyfit(sttnH2,t4mEIxy1d,8);
HmEIxy1dd = polyfit(sttnH1,t4mEIxy1dd,8);
HmEIxy2 = polyfit(sttnH1,t4mEIxy2,8);
HmEIxy2d = polyfit(sttnH2,t4mEIxy2d,8);
HmEIxy2dd = polyfit(sttnH1,t4mEIxy2dd,8);
HmEIxy3 = polyfit(sttnH1,t4mEIxy3,8);
HmEIxy3d = polyfit(sttnH2,t4mEIxy3d,8);
HmEIxy3dd = polyfit(sttnH1,t4mEIxy3dd,8);

figure; hold on;
subplot(1,3,1); hold on;
plot(xx,polyval(HmEIxy1,xx),'b-','linewidth',2);
plot(xx,polyval(HmEIxy1d,xx),'b-.','linewidth',2);
plot(xx,polyval(HmEIxy1dd,xx),'b:','linewidth',2);
grid minor;
legend('MODE 1','1st Derivative','2nd Derivative',0);
title('HINGED m = m_0  &  EI = EI_0(1-xhat):');
```

```
subplot(1,3,2); hold on;
plot(xx,polyval(HmEIxy2,xx),'r-','linewidth',2);
plot(xx,polyval(HmEIxy2d,xx),'r-.','linewidth',2);
plot(xx,polyval(HmEIxy2dd,xx),'r:','linewidth',2);
grid minor;
legend('MODE 2','1st Derivative','2nd Derivative',0);
subplot(1,3,3); hold on;
plot(xx,polyval(HmEIxy3,xx),'m-','linewidth',2);
plot(xx,polyval(HmEIxy3d,xx),'m-.','linewidth',2);
plot(xx,polyval(HmEIxy3dd,xx),'m:','linewidth',2);
grid minor;
legend('MODE 3','1st Derivative','2nd Derivative',0);

clear t4mEIxy*
%h4 m = m_0(1-xhat/2)  &  EI = EI_0:
t4mx2EIy1 = [0.0000     -.1558  -.2990  -.4182  -.5038  -.5484  -.5478  -.5005  -.4083  -
.2751  -.1071  ...
                        0.0884  0.3037  0.5312  0.7647  1.0000].';
t4mx2EIy1d = [-2.3363   -2.1488 -1.7886 -1.2830 -.6693  0.0096  0.7083  1.3836  1.9975
2.5203  2.9330  ...
                        3.2285  3.4124  3.5029  3.5298].';%
t4mx2EIy1dd = [0.0000   2.8306  5.4383  7.6302  9.2606  10.2394 10.5356 10.1773 9.2456
7.8663  6.1992  ...
                        4.4254  2.7358  1.3192  0.3538  0.0000].';
t4mx2EIy2 = [0.0000     0.2727  0.4785  0.5674  0.5186  0.3447  0.0872  -.1932  -.4314  -
.5716  -.5787  ...
                        -.4431  -.1812  0.1730  0.5782  1.0000].';
t4mx2EIy2d = [4.0912    3.0857  1.3335  -.7308  -2.6093 -3.8621 -4.2068 -3.5727 -2.1027 -
.1044  2.0318  ...
                        3.9288  5.3124  6.0784  6.3271].';%
t4mx2EIy2dd = [0.0000   -15.4147    -26.8339    -31.5958    -28.7323    -19.1453    -
5.2516 9.6994  22.4495 ...
                        30.4780 32.5262 28.8072 20.8910 11.3644 3.3416  0.0000].';
t4mx2EIy3 = [0.0000     -.3730  -.5607  -.4680  -.1402  0.2634  0.5497  0.5846  0.3524  -
.0405  -.4169  ...
                        -.6068  -.5150  -.1510  0.3921  1.0000].';
t4mx2EIy3d = [-5.5947   -2.8156 1.3899  4.9171  6.0537  4.2957  0.5225  -3.4830 -5.8934 -
5.6463 -2.8473 ...
                        1.3765  5.4594  8.1471  9.1186].';%
t4mx2EIy3dd = [0.0000   43.3868 65.7475 55.0307 17.6072 -27.5745    -58.9163    -62.3944
-37.4169    ...
                        3.9909  43.5972 65.5174 62.9664 40.7963 13.4256 0.0000].';
Hmx2EIy1 = polyfit(sttnH1,t4mx2EIy1,8);
Hmx2EIy1d = polyfit(sttnH2,t4mx2EIy1d,8);
Hmx2EIy1dd = polyfit(sttnH1,t4mx2EIy1dd,8);
Hmx2EIy2 = polyfit(sttnH1,t4mx2EIy2,8);
Hmx2EIy2d = polyfit(sttnH2,t4mx2EIy2d,8);
Hmx2EIy2dd = polyfit(sttnH1,t4mx2EIy2dd,8);
Hmx2EIy3 = polyfit(sttnH1,t4mx2EIy3,8);
Hmx2EIy3d = polyfit(sttnH2,t4mx2EIy3d,8);
Hmx2EIy3dd = polyfit(sttnH1,t4mx2EIy3dd,8);

figure; hold on;
subplot(1,3,1); hold on;
plot(xx,polyval(Hmx2EIy1,xx),'b-','linewidth',2);
plot(xx,polyval(Hmx2EIy1d,xx),'b-.','linewidth',2);
plot(xx,polyval(Hmx2EIy1dd,xx),'b:','linewidth',2);
grid minor;
legend('MODE 1','1st Derivative','2nd Derivative',0);
title('HINGED m = m_0(1-xhat/2)  &  EI = EI_0:');
subplot(1,3,2); hold on;
plot(xx,polyval(Hmx2EIy2,xx),'r-','linewidth',2);
plot(xx,polyval(Hmx2EIy2d,xx),'r-.','linewidth',2);
plot(xx,polyval(Hmx2EIy2dd,xx),'r:','linewidth',2);
grid minor;
legend('MODE 2','1st Derivative','2nd Derivative',0);
subplot(1,3,3); hold on;
plot(xx,polyval(Hmx2EIy3,xx),'m-','linewidth',2);
plot(xx,polyval(Hmx2EIy3d,xx),'m-.','linewidth',2);
plot(xx,polyval(Hmx2EIy3dd,xx),'m:','linewidth',2);
grid minor;
legend('MODE 3','1st Derivative','2nd Derivative',0);

clear t4mx2EIy*
%h5 m = m_0(1-xhat/2)  &  EI = EI_0(1-xhat/2):
t4mx2EIx2y1 = [0.0000       -.1405  -.2714  -.3831  -.4670  -.5157  -.5236  -.4877  -
.4074  -.2847  -.1240  ...
                            0.0684  0.2847  0.5168  0.7571  1.0000].';
```

```
t4mx2EIx2y1d = [-2.1075     -1.9632 -1.6764 -1.2583 -.7296  -.1191  0.5384  1.2046
1.8407  2.4110  2.8856  ...
                            3.2447  3.4812  3.6044  3.6433].';%
t4mx2EIx2y1dd = [0.0000     2.1650  4.3162  6.2980  7.9670  9.2014  9.9102  10.0412
9.5875  8.5912  7.1455  ...
                            5.3954  3.5362  1.8125  0.5182  0.0000].';
t4mx2EIx2y2 = [0.0000       0.2937  0.4281  0.5231  0.5013  0.3642  0.1393  -.1251  -
.3694  -.5347  -.5760  ...
                            -.4711  -.2251  0.1318  0.5545  1.0000].';
t4mx2EIx2y2d = [3.5959      2.8262  1.4247  -.3274  -2.0558 -3.3735 -3.9664 -3.6638 -
2.4809 -.6194  1.5746  ...
                            3.6896  5.3528  6.3407  6.6828].';%
t4mx2EIx2y2dd = [0.0000     -11.7139    -21.3762    -26.7516    -26.4102    -20.1587    -
9.1019 4.5670  ...
                            18.0105 28.3717 33.4411 32.2076 25.2411 14.7929 4.6798
0.0000].';
t4mx2EIx2y3 = [0.0000       -.3245  -.5079  -.4640  -.2030  0.1630  0.4700  0.5746
0.4204  0.0681  -.3262  ...
                            -.5765  -.5468  -.2092  0.3517  1.0000].';
t4mx2EIx2y3d = [-4.8668     -2.7515 0.6587  3.9138  5.4909  4.6057  1.5683  -2.3134 -
5.2843 -5.9149 -3.7545 ...
                            0.4454  5.0645  8.4144  9.7239].';%
t4mx2EIx2y3dd = [0.0000     32.6840 53.5102 50.6902 24.6040 -13.7307    -47.2762    -
60.4825    -46.3612    ...
                            -9.9582 33.4867 65.2075 71.5806 51.3484 18.4557 0.0000].';
Hmx2EIx2y1 = polyfit(sttnH1,t4mx2EIx2y1,8);
Hmx2EIx2y1d = polyfit(sttnH2,t4mx2EIx2y1d,8);
Hmx2EIx2y1dd = polyfit(sttnH1,t4mx2EIx2y1dd,8);
Hmx2EIx2y2 = polyfit(sttnH1,t4mx2EIx2y2,8);
Hmx2EIx2y2d = polyfit(sttnH2,t4mx2EIx2y2d,8);
Hmx2EIx2y2dd = polyfit(sttnH1,t4mx2EIx2y2dd,8);
Hmx2EIx2y3 = polyfit(sttnH1,t4mx2EIx2y3,8);
Hmx2EIx2y3d = polyfit(sttnH2,t4mx2EIx2y3d,8);
Hmx2EIx2y3dd = polyfit(sttnH1,t4mx2EIx2y3dd,8);

figure; hold on;
subplot(1,3,1); hold on;
plot(xx,polyval(Hmx2EIx2y1,xx),'b-','linewidth',2);
plot(xx,polyval(Hmx2EIx2y1d,xx),'b-.','linewidth',2);
plot(xx,polyval(Hmx2EIx2y1dd,xx),'b:','linewidth',2);
grid minor;
legend('MODE 1','1st Derivative','2nd Derivative',0);
title('HINGED m = m_0(1-xhat/2)  &  EI = EI_0(1-xhat/2):');
subplot(1,3,2); hold on;
plot(xx,polyval(Hmx2EIx2y2,xx),'r-','linewidth',2);
plot(xx,polyval(Hmx2EIx2y2d,xx),'r-.','linewidth',2);
plot(xx,polyval(Hmx2EIx2y2dd,xx),'r:','linewidth',2);
grid minor;
legend('MODE 2','1st Derivative','2nd Derivative',0);
subplot(1,3,3); hold on;
plot(xx,polyval(Hmx2EIx2y3,xx),'m-','linewidth',2);
plot(xx,polyval(Hmx2EIx2y3d,xx),'m-.','linewidth',2);
plot(xx,polyval(Hmx2EIx2y3dd,xx),'m:','linewidth',2);
grid minor;
legend('MODE 3','1st Derivative','2nd Derivative',0);

clear t4mx2EIx2y*
%h6 m = m_0(1-xhat/2)  &  EI = EI_0(1-xhat):
t4mx2EIxy1 = [0.0000    -.1103  -.2148  -.3075  -.3500  -.4323  -.4526  -.4383  -.3856  -
.2924  -.1583  ...
                        0.0153  0.2248  0.4645  0.7262  1.0000].';
t4mx2EIxy1d = [-1.6543  -1.5682 -1.3902 -1.1178 -.7536  -.3051  0.2153  0.7902  1.3977
2.0120  2.6041  ...
                        3.1425  3.5944  3.9265  4.1066].';%
t4mx2EIxy1dd = [0.0000  1.2820  2.6679  4.0886  5.4721  6.7435  7.8273  8.6503  9.1450
9.2516  8.9210  ...
                        8.1173  6.8194  5.0225  2.7389  0.0000].';
t4mx2EIxy2 = [0.0000    0.1643  0.3020  0.3882  0.4044  0.3423  0.2070  0.0184  -.1901  -
.3744  -.4862  ...
                        -.4806  -.3248  -.0078  0.4518  1.0000].';
t4mx2EIxy2d = [3.4651   2.0642  1.2932  0.2431  -.9314  -2.0293 -2.8297 -3.1264 -2.7646 -
1.6775 0.0846  ...
                        2.3363  4.7546  6.8945  8.2232].';%
t4mx2EIxy2dd = [0.0000  -6.0465 -11.6724    -15.9371    -17.8592    -16.7340    -12.2504
-4.6272 ...
                        5.3591  16.3886 26.6827 34.1999 36.8503 32.7806 20.6465
0.0000].';
t4mx2EIxy3 = [0.0000    -.2003  -.3380  -.3597  -.2482  -.0333  0.2116  0.3916  0.4231
0.2712  -.0238  ...
```

```
                        -.3423  -.5123  -.3684  0.1667  1.0000].';
t4mx2EIxy3d = [-3.0051  -2.0645 -.3264  1.6735  3.2227  3.6744  2.6991  0.4730  -2.2779 -
4.4252 -4.7776 ...
                        -2.5501 2.1574  8.0186  12.4991].';%
t4mx2EIxy3dd = [0.0000  14.3350 26.9253 31.1310 24.3216 7.4502  -14.6474    -34.2032    -
42.7143    -33.8828    ...
                        -6.5779 33.1451 71.8610 90.9456 71.1259 0.0000].';
Hmx2EIxy1 = polyfit(sttnH1,t4mx2EIxy1,8);
Hmx2EIxy1d = polyfit(sttnH2,t4mx2EIxy1d,8);
Hmx2EIxy1dd = polyfit(sttnH1,t4mx2EIxy1dd,8);
Hmx2EIxy2 = polyfit(sttnH1,t4mx2EIxy2,8);
Hmx2EIxy2d = polyfit(sttnH2,t4mx2EIxy2d,8);
Hmx2EIxy2dd = polyfit(sttnH1,t4mx2EIxy2dd,8);
Hmx2EIxy3 = polyfit(sttnH1,t4mx2EIxy3,8);
Hmx2EIxy3d = polyfit(sttnH2,t4mx2EIxy3d,8);
Hmx2EIxy3dd = polyfit(sttnH1,t4mx2EIxy3dd,8);

figure; hold on;
subplot(1,3,1); hold on;
plot(xx,polyval(Hmx2EIxy1,xx),'b-','linewidth',2);
plot(xx,polyval(Hmx2EIxy1d,xx),'b-.','linewidth',2);
plot(xx,polyval(Hmx2EIxy1dd,xx),'b:','linewidth',2);
grid minor;
legend('MODE 1','1st Derivative','2nd Derivative',0);
title('HINGED m = m_0(1-xhat/2)  &  EI = EI_0(1-xhat):');
subplot(1,3,2); hold on;
plot(xx,polyval(Hmx2EIxy2,xx),'r-','linewidth',2);
plot(xx,polyval(Hmx2EIxy2d,xx),'r-.','linewidth',2);
plot(xx,polyval(Hmx2EIxy2dd,xx),'r:','linewidth',2);
grid minor;
legend('MODE 2','1st Derivative','2nd Derivative',0);
subplot(1,3,3); hold on;
plot(xx,polyval(Hmx2EIxy3,xx),'m-','linewidth',2);
plot(xx,polyval(Hmx2EIxy3d,xx),'m-.','linewidth',2);
plot(xx,polyval(Hmx2EIxy3dd,xx),'m:','linewidth',2);
grid minor;
legend('MODE 3','1st Derivative','2nd Derivative',0);

clear t4mx2EIxy*
%h7 m = m_0(1-xhat)  &  EI = EI_0:
t4mxEIy1 = [0.0000      -.1006  -.1894  -.2561  -.2928  -.2947  -.2604  -.1916  -.0920
0.0325  0.1759  ...
                        0.3320  0.4956  0.6628  0.8313  1.0000].';
t4mxEIy1d = [-1.5091    -1.3324 -1.0003 -.5501  -.0288  0.5141  1.0334  1.4928  1.8686
2.1505  2.3411  ...
                        2.4537  2.5083  2.5274  2.5308].';%
t4mxEIy1dd = [0.0000    2.6772  5.0299  6.8123  7.8813  8.2013  7.8349  6.9215  5.6493
4.2248  2.8430  ...
                        1.6634  0.7904  0.2606  0.0359  0.0000].';
t4mxEIy2 = [0.0000      0.1427  0.2364  0.2496  0.1783  0.0457  -.1069  -.2339  -.2980  -
.2787  -.1744  ...
                        0.0008  0.2251  0.4760  0.7370  1.0000].';
t4mxEIy2d = [2.1411     1.4053  0.1972  -1.0696 -1.9884 -2.2893 -1.9048 -.9611  0.2897
1.5634  2.6292  ...
                        3.3640  3.7635  3.9154  3.9444].';%
t4mxEIy2dd = [0.0000    -11.3779    -18.6533    -19.5232    -14.1242    -4.5930 5.9348
14.4825 19.1285 ...
                        19.3997 16.1429 11.0251 5.8786  2.1146  0.3109  0.0000].';
t4mxEIy3 = [0.0000      -.1705  -.2249  -.1258  0.0617  0.2187  0.2493  0.1352  -.0409  -
.2358  -.3002  ...
                        -.2148  0.0047  0.3098  0.6512  1.0000].';
t4mxEIy3d = [-2.5582    -.8148  1.4862  2.8127  2.3549  0.4591  -1.7115 -2.9417 -2.6244 -
.9652  1.2810  ...
                        3.2917  4.5778  5.1208  5.2315].';%
t4mxEIy3dd = [0.0000    27.6738 36.4609 20.8593 -7.4162 -30.0183    -34.1493    -19.1967
5.1045  25.9492 ...
                        34.8202 30.8241 19.3206 7.7297  1.2214  0.0000].';
HmxEIy1 = polyfit(sttnH1,t4mxEIy1,8);
HmxEIy1d = polyfit(sttnH2,t4mxEIy1d,8);
HmxEIy1dd = polyfit(sttnH1,t4mxEIy1dd,8);
HmxEIy2 = polyfit(sttnH1,t4mxEIy2,8);
HmxEIy2d = polyfit(sttnH2,t4mxEIy2d,8);
HmxEIy2dd = polyfit(sttnH1,t4mxEIy2dd,8);
HmxEIy3 = polyfit(sttnH1,t4mxEIy3,8);
HmxEIy3d = polyfit(sttnH2,t4mxEIy3d,8);
HmxEIy3dd = polyfit(sttnH1,t4mxEIy3dd,8);

figure; hold on;
subplot(1,3,1); hold on;
```

```
plot(xx,polyval(HmxEIy1,xx),'b-','linewidth',2);
plot(xx,polyval(HmxEIy1d,xx),'b-.','linewidth',2);
plot(xx,polyval(HmxEIy1dd,xx),'b:','linewidth',2);
grid minor;
legend('MODE 1','1st Derivative','2nd Derivative',0);
title('HINGED m = m_0(1-xhat)  &  EI = EI_0:');
subplot(1,3,2); hold on;
plot(xx,polyval(HmxEIy2,xx),'r-','linewidth',2);
plot(xx,polyval(HmxEIy2d,xx),'r-.','linewidth',2);
plot(xx,polyval(HmxEIy2dd,xx),'r:','linewidth',2);
grid minor;
legend('MODE 2','1st Derivative','2nd Derivative',0);
subplot(1,3,3); hold on;
plot(xx,polyval(HmxEIy3,xx),'m-','linewidth',2);
plot(xx,polyval(HmxEIy3d,xx),'m-.','linewidth',2);
plot(xx,polyval(HmxEIy3dd,xx),'m:','linewidth',2);
grid minor;
legend('MODE 3','1st Derivative','2nd Derivative',0);

clear t4mxEIy*
%h8 m = m_0(1-xhat)  &  EI = EI_0(1-xhat/2):
t4mxEIx2y1 = [0.0000    -.0912  -.1731  -.2366  -.2743  -.2808  -.2535  -.1922  -.0995
0.0203  0.1615  ...
                        0.3179  0.4838  0.6545  0.8271  1.0000].';
t4mxEIx2y1d = [-1.3685  -1.2273 -.9526  -.5655  -.0984  0.4100  0.9190  1.3913  1.7972
2.1176  2.3460  ...
                        2.4884  2.5613  2.5883  2.5934].';%
t4mxEIx2y1dd = [0.0000  2.1268  4.1481  5.8470  7.0565  7.6765  7.6819  7.1221  6.1118
4.8141  3.4186  ...
                        2.1148  1.0648  0.3729  0.0547  0.0000].';
t4mxEIx2y2 = [0.0000    0.1254  0.2125  0.2332  0.1796  0.0658  -.0759  -.2045  -.2813  -
.2802  -.1924  ...
                        -.0264  0.1975  0.4550  0.7261  1.0000].';
t4mxEIx2y2d = [1.8817   1.3053  0.3843  -.8044  -1.7075 -2.1257 -1.9280 -1.1527 0.0165
1.3170  2.4906  ...
                        3.3583  3.8625  4.0667  4.1079].';%
t4mxEIx2y2dd = [0.0000  -8.8479 -15.2839    -17.1710    -13.8954    -6.4412 3.0165
11.8656 17.8750 19.8367 ...
                        17.8332 13.0866 7.4790  2.8817  0.4541  0.0000].';
t4mxEIx2y3 = [0.0000    -.1505  -.2086  -.1360  0.0265  0.1836  0.2419  0.1637  -.0134  -
.1976  -.2913  ...
                        -.2359  -.0314  0.2773  0.6335  1.0000].';
t4mxEIx2y3d = [-2.2572  -.8715  1.0885  2.4377  2.3568  0.8750  -1.1742 -2.6558 -2.7629 -
1.4052 0.8298  ...
                        3.0677  4.6312  5.3430  5.4972].';%
t4mxEIx2y3dd = [0.0000  21.7677 30.8281 21.1573 -1.3903 -23.3985    -32.2246    -23.1981
-1.5876 21.2899 ...
                        34.8049 34.5490 23.7284 10.2935 1.7510  0.0000].';
HmxEIx2y1 = polyfit(sttnH1,t4mxEIx2y1,8);
HmxEIx2y1d = polyfit(sttnH2,t4mxEIx2y1d,8);
HmxEIx2y1dd = polyfit(sttnH1,t4mxEIx2y1dd,8);
HmxEIx2y2 = polyfit(sttnH1,t4mxEIx2y2,8);
HmxEIx2y2d = polyfit(sttnH2,t4mxEIx2y2d,8);
HmxEIx2y2dd = polyfit(sttnH1,t4mxEIx2y2dd,8);
HmxEIx2y3 = polyfit(sttnH1,t4mxEIx2y3,8);
HmxEIx2y3d = polyfit(sttnH2,t4mxEIx2y3d,8);
HmxEIx2y3dd = polyfit(sttnH1,t4mxEIx2y3dd,8);

figure; hold on;
subplot(1,3,1); hold on;
plot(xx,polyval(HmxEIx2y1,xx),'b-','linewidth',2);
plot(xx,polyval(HmxEIx2y1d,xx),'b-.','linewidth',2);
plot(xx,polyval(HmxEIx2y1dd,xx),'b:','linewidth',2);
grid minor;
legend('MODE 1','1st Derivative','2nd Derivative',0);
title('HINGED m = m_0(1-xhat)  &  EI = EI_0(1-xhat/2):');
subplot(1,3,2); hold on;
plot(xx,polyval(HmxEIx2y2,xx),'r-','linewidth',2);
plot(xx,polyval(HmxEIx2y2d,xx),'r-.','linewidth',2);
plot(xx,polyval(HmxEIx2y2dd,xx),'r:','linewidth',2);
grid minor;
legend('MODE 2','1st Derivative','2nd Derivative',0);
subplot(1,3,3); hold on;
plot(xx,polyval(HmxEIx2y3,xx),'m-','linewidth',2);
plot(xx,polyval(HmxEIx2y3d,xx),'m-.','linewidth',2);
plot(xx,polyval(HmxEIx2y3dd,xx),'m:','linewidth',2);
grid minor;
legend('MODE 3','1st Derivative','2nd Derivative',0);
```

```
clear t4mxEIx2y*
%h9 m = m_0(1-xhat)  &  EI = EI_0(1-xhat):
t4mxEIxy1 = [0.0000     -.0748  -.1433  -.1989  -.2356  -.2480  -.2323  -.1862  -.1095  -
.0036  0.1282  ...
                        0.2813  0.4502  0.6275  0.8139  1.0000].';
t4mxEIxy1d = [-1.1222   -1.0267 -.8343  -.5499  -.1864  0.2356  0.6908  1.1511  1.5883
1.9768  2.2964  ...
                        2.5345  2.6884  2.7669  2.7912].';%
t4mxEIxy1dd = [0.0000   1.4218  2.8943  4.2872  5.4778  6.3621  6.8635  6.9403  6.5903
5.8531  4.8088  ...
                        3.5747  2.2986  1.1517  0.3206  0.0000].';
t4mxEIxy2 = [0.0000     0.0915  0.1602  0.1878  0.1620  0.0879  -.0193  -.1331  -.2216  -
.2553  -.2124  ...
                        -.0846  0.1218  0.3878  0.6882  1.0000].';
t4mxEIxy2d = [1.3728    1.0305  0.4020  -.3755  -1.1116 -1.6078 -1.7061 -1.3297 -.5045
0.6438  1.9177  ...
                        3.0957  3.9904  4.5050  4.6775].';%
t4mxEIxy2dd = [0.0000   -5.2011 -9.6016 -11.9049    -11.2941    -7.6441 -1.5725 5.6793
12.5389 17.4785 ...
                        19.3913 17.9102 13.5472 7.6745  2.3373  0.0000].';
t4mxEIxy3 = [0.0000     -.1037  -.1562  -.1277  -.0274  0.0975  0.1824  0.1774  0.0741  -
.0852  -.2212  ...
                        -.2503  -.1201  0.1683  0.5645  1.0000].';
t4mxEIxy3d = [-1.5560   -.7877  0.4284  1.5050  1.8722  1.2737  -.0748  -1.5500 -2.3886 -
2.0398 -.4364  ...
                        1.9527  4.3263  5.9419  6.5395].';%
t4mxEIxy3dd = [0.0000   11.8732 18.9034 16.7673 5.7258  -9.3345 -21.0490    -23.0546    -
13.1568    5.3345  ...
                        24.8731 37.0753 36.7275 24.6765 8.3232  0.0000].';
HmxEIxy1 = polyfit(sttnH1,t4mxEIxy1,8);
HmxEIxy1d = polyfit(sttnH2,t4mxEIxy1d,8);
HmxEIxy1dd = polyfit(sttnH1,t4mxEIxy1dd,8);
HmxEIxy2 = polyfit(sttnH1,t4mxEIxy2,8);
HmxEIxy2d = polyfit(sttnH2,t4mxEIxy2d,8);
HmxEIxy2dd = polyfit(sttnH1,t4mxEIxy2dd,8);
HmxEIxy3 = polyfit(sttnH1,t4mxEIxy3,8);
HmxEIxy3d = polyfit(sttnH2,t4mxEIxy3d,8);
HmxEIxy3dd = polyfit(sttnH1,t4mxEIxy3dd,8);

figure; hold on;
subplot(1,3,1); hold on;
plot(xx,polyval(HmxEIxy1,xx),'b-','linewidth',2);
plot(xx,polyval(HmxEIxy1d,xx),'b-.','linewidth',2);
plot(xx,polyval(HmxEIxy1dd,xx),'b:','linewidth',2);
grid minor;
legend('MODE 1','1st Derivative','2nd Derivative',0);
title('HINGED m = m_0(1-xhat)  &  EI = EI_0(1-xhat):');
subplot(1,3,2); hold on;
plot(xx,polyval(HmxEIxy2,xx),'r-','linewidth',2);
plot(xx,polyval(HmxEIxy2d,xx),'r-.','linewidth',2);
plot(xx,polyval(HmxEIxy2dd,xx),'r:','linewidth',2);
grid minor;
legend('MODE 2','1st Derivative','2nd Derivative',0);
subplot(1,3,3); hold on;
plot(xx,polyval(HmxEIxy3,xx),'m-','linewidth',2);
plot(xx,polyval(HmxEIxy3d,xx),'m-.','linewidth',2);
plot(xx,polyval(HmxEIxy3dd,xx),'m:','linewidth',2);
grid minor;
legend('MODE 3','1st Derivative','2nd Derivative',0);

clear t4mxEIxy*
%
figure; hold on;
plot(xx,polyval(HmEIy1,xx),'b-','linewidth',2);
plot(xx,polyval(HmEIx2y1,xx),'b:','linewidth',2);
plot(xx,polyval(HmEIxy1,xx),'b-.','linewidth',2);
plot(xx,polyval(Hmx2EIy1,xx),'m-','linewidth',2);
plot(xx,polyval(Hmx2EIx2y1,xx),'m:','linewidth',2);
plot(xx,polyval(Hmx2EIxy1,xx),'m-.','linewidth',2);
plot(xx,polyval(HmxEIy1,xx),'r-','linewidth',2);
plot(xx,polyval(HmxEIx2y1,xx),'r:','linewidth',2);
plot(xx,polyval(HmxEIxy1,xx),'r-.','linewidth',2); grid on;
title('HINGED BEAM FIRST BENDING MODE COMPARISON:');
grid minor;
legend('m_r_o_o_t = m_t_i_p & EI_r_o_o_t = EI_t_i_p','m_r_o_o_t = m_t_i_p & EI_t_i_p =
EI_r_o_o_t/2', ...
    'm_r_o_o_t = m_t_i_p & EI_t_i_p = 0','m_t_i_p = m_r_o_o_t/2 & EI_r_o_o_t = EI_t_i_p',
...
```

```
    'm_t_i_p = m_r_o_o_t/2 & EI_t_i_p = EI_r_o_o_t/2','m_t_i_p = m_r_o_o_t/2 & EI_t_i_p =
0', ...
    'm_t_i_p = 0 & EI_r_o_o_t = EI_t_i_p','m_t_i_p = 0 & EI_t_i_p =
EI_r_o_o_t/2','m_t_i_p = 0 & EI_t_i_p = 0',2);
%===============================================================================
clear sttn*

%===============================================================================
```

# APPENDIX C. MATLAB® CODES FOR FUNCTION FILES

## A. MATLAB® CODE FOR *YNTEMA* FUNCTION

```
function [w1ynt,w2ynt,w3ynt,w0ynt,omegaRng,varargout] =
      yntema(flagSouth,cantHing,eoffset,blength,melast,maLen0,minert0,varargin);

%[w1,w2,w3,w0,rtrSpd <,{a0,a1,a2,a3,K0,K1,K2,K3,K00,K01,K02,K03,Kh10,Kh11,Kh12,Kh13}
      <,w1ksi,w2ksi,w3ksi,w0ksi>> <,w1ksi,w2ksi,w3ksi,w0ksi>] =
      YNTEMA(flagSouth,cantHing,eoffset,blength,melast,maLen0,minert0 <,'mtip',maLenT>
      <,'Itip',minertT><,'Mt',massTip> <,'AoA',alfa> <,[rtrSpdRng opRtrSpd]>);
%
%<<<<<<< A COMPUTER CODE FOR RAPID CALCULATION OF BENDING FREQUENCIES OF ROTOR
      BLADES>>>>>>>>>>>>>>>>>>>>>>
%<<<<<<<<<<<<<< -- (Advisor: Prof.E.R.WOOD; Co-Advisor:
      CDR.M.COUCH)>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%<<<<<<<<(C)1Lt.H.E.AKIN, Turkish Army, Naval Postgraduate School, Monterey, CA,
      USA>>>>>>>>>>>>>>>>>>>>>>>
%<<<<<<<<<<<Master of Science in Aeronautical Enginering (Avionics) Thesis,
      SEP2002>>>>>>>>>>>>>>>>>>>>>>>>
%<<<<<<<<<<<<<<<<<<<<<<[MATLAB 6.1.0.450 Release 12.1, May
      18,2001]>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%
% This code rapidly gives the approximate bending frequencies of rotor blades
% by the Yntema method. The Yntema method uses a Rayleigh energy approach utilizing
% the bending mode of nonrotating beam in the determination of the bending frequency
% of the rotating beam.
% Charts are read for the rapid estimation of the first three bending frequencies
% for rotating and nonrotating cantilever and hinged beams with a variable mass and
% stiffness distributions, as well as with root offsets from the axis of rotation.
% The case of rotating beams with a tip mass is also included.
%-------------------------------------------------------------------------------
% 'VARARGIN' looks for ('mtip',maLenT,'Itip',minertT,'Mt',massTip,rtrSpdRng,alfa) values;
% if VARARGIN is empty, the values are assumed to be equal to their root values or zero,
      or empty.
% 'VARARGOUT' may be <yntema_coefficients> or <w1ksi,w2ksi,w3ksi,w0ksi> or all five of
      them;
% it is a cell array containing one, four, or five cells.
%===============================================================================
runtimestart = cputime;
disp('=================== START OF YNTEMA() FILE =======================');
%MAKING A DIARY FILE INSIDE THE 'yntemarundiary' SUBDIRECTORY OF THE CURRENT WORKSPACE:
%(Execute this file on hard disk to avoid warning messages caused by diary statements
      here.)
dirynte = dir(pwd);
flagdirec = 0;
for ddd = 1:size(dirynte,1)
    chkvar111 = struct2cell(dirynte(ddd));
    if ((size(chkvar111{1},2) == 14) & ((chkvar111{1} == 'yntemarundiary') &
      (isdir('yntemarundiary'))))
        flagdirec = 1;
    end
end

if flagdirec == 0
    eval('mkdir yntemarundiary',...
        'disp(''The diary folder for this run of YNTEMA() function could not be
      created!'')');
end
simdi = datestr(now);
simdi = [simdi(1:14) simdi(16:17) simdi(19:20)];
eval('diary([pwd ''\yntemarundiary\'' simdi ''.txt''])',...
    'disp(''Diary will not be saved for this run of YNTEMA() function!'')');
%==============================================================================
% CHECKING INPUT ARGUMENTS AND MAKING NECESSARY CORRECTIONS:
load yntedflt.mat; %default values for E, I_0, m_0, L, a rotor speed range. For use if
      not input properly.

%Check for number of input arguments:
if nargin < 7
    errordlg({'The number of input arguments can not be less than 7 (seven)... ' 'PLEASE
      TRY AGAIN!'}, ...
        'WARNING','modal');
    error('Early Termination: PLEASE TRY AGAIN WITH PROPER INPUT ARGUMENTS!');
    disp('WARNING: The number of input arguments can not be less than 7 (seven).');
```

```
    disp('         Early Termination: PLEASE TRY AGAIN WITH PROPER INPUT ARGUMENTS!');
    disp(' ');
end
if nargin > 16
    errordlg({'The number of input arguments can not be more than 16 (sixteen)... ' ...
            ' The excessive input arguments will be truncated.'},'WARNING','modal');
    disp('WARNING: The number of input arguments can not be more than 16 (sixteen).');
    disp('         The excessive input arguments will be truncated.');
    disp(' ');
end

%Check for compulsory input arguments:
if ( isempty(flagSouth) | ((flagSouth ~= 0 & flagSouth ~= 1) & (flagSouth ~= 2 &
      flagSouth ~= 3)) )
    flagSouth = 0;
    disp('WARNING: The Southwell Plot Flag input is not 0, 1, 2, or 3...');
    disp('         The Southwell Plot Flag is assumed to be 0 (plot not needed)!');
    disp(' ');
end
if ( isempty(cantHing) | ((upper(cantHing(1)) ~= 'C') & (upper(cantHing(1)) ~= 'H')) )
    cantHing = 'H';
    disp('WARNING: The selection is not specified as cantilever or hinged in the second
      input argument.');
    disp('         The beam is assumed to be HINGED!');
    disp(' ');
end
if ( isempty(blength) | (((blength <= 0) | (size(blength) ~= [1 1])) |
      (~isnumeric(blength))) )
    blength = bLenDF;
    disp('WARNING: Blade length input is zero or less or not a scalar.');
    disp(['         Blade length is assumed to be ',num2str(blength),' ft!\n']);
    disp(' ');
end
if ( isempty(eoffset) | (((eoffset < 0) | (eoffset >= blength)) | ((size(eoffset) ~= [1
      1]) | (~isnumeric(eoffset)))) )
    eoffset = 0;
    disp('WARNING: Hinge offset input (e/L or e) is less than zero or not a scalar or it
      is greater than blade_length!');
    disp('         Hinge offset is assumed to be ZERO!');
    disp(' ');
elseif ( ~isempty(eoffset) & (eoffset >= 1) )
    eoffset = eoffset./blength;
    disp('WARNING: Hinge offset input is greater than or equal to 1 (one). It is accepted
      as (e), not (e/L).');
    disp(['         Hinge offset value is recomputed to be (e / blade_length = e/L) = '
      num2str(eoffset)]);
    disp(' ');
end
if ( isempty(melast) | (((melast <= 0) | (size(melast) ~= [1 1])) | (~isnumeric(melast)))
      )
    melast = elasAlDF;
    disp('WARNING: Modulus of Elasticity input is less than or equal to zero or it is not
      a scalar.');
    disp(['         Modulus of Elasticity is assumed to be ' num2str(elasAlDF) ' psi
      which is for aluminum!\n']);
    disp(' ');
end
melast = melast / (32.174 * 12); %converting from psi to lbm/in-sec^2:
      [psi/(32.174*12)=lbm/in-sec^2]
if ( isempty(maLen0) | (((maLen0 <= 0) | (size(maLen0) ~= [1 1])) | (~isnumeric(maLen0)))
      )
    maLen0 = maLen0DF;
    disp('WARNING: Blade Mass per Length at Root input is less than or equal to zero or
      it is not a scalar.');
    disp(['         Blade Mass per Length at Root is assumed to be ' num2str(maLen0DF) '
      lb-sec^2!\n']);
    disp(' ');
end
if ( isempty(minert0) | (((minert0 <= 0) | (size(minert0) ~= [1 1])) |
      (~isnumeric(minert0))) )
    minert0 = minert0DF;
    disp('WARNING: Moment of Inertia at Root input is less than zero or it is not a
      scalar.');
    disp(['         Moment of Inertia at Root is assumed to be ' num2str(minert0DF) '
      in^4!\n']);
    disp(' ');
end

% Check for optional VARARGIN arguments:
```

```
maLenT = [];
minertT = [];
massTip = [];
rtrSpdRng = [];
alfa = []; alfaMid = []; alfaR = [];

if isempty(varargin)
    maLenT = maLen0;
    minertT = minert0;
    massTip = 0;
    rtrSpdRng = rtrSpdRngDF;
    disp('ATTENTION: You did not specify any extra input arguments.');
    disp('           The beam is assumed to be uniform and without a concentrated mass at
      the tip.');
    disp(' ');
else
    if (size(varargin,2) < 9)
        for vii = (size(varargin,2)+1:1:9)
            varargin{1,vii} = '...';
        end
    end
    viii = 1;
    while (viii <= 9)
        if (prod(size(varargin{1,viii}) == [1 4]) & (varargin{1,viii} == 'mtip'))
            if ( ((~isempty(varargin{1,viii+1})) & (isnumeric(varargin{1,viii+1}))) & ...
                    ((varargin{1,viii+1} >= 0) & (size(varargin{1,viii+1}) == [1 1])) )
                maLenT = varargin{1,viii+1};
                viii = viii + 2;
                if maLenT > maLen0
                    maLenT = maLen0;
                    disp('WARNING: m_tip can not be greater than m_0. Assumed: m_tip =
      m_0.');
                    disp(' ');
                end
            else
                maLenT = maLen0;
                viii = viii + 1;
                disp('WARNING: m_tip value is empty/matrix/less than zero/not numeric/not
      after ''mtip'' expression.');
                disp('         m_tip is assumed to be equal to m_0 (constant mass
      distribution).');
                disp(' ');
            end %if4
        elseif (prod(size(varargin{1,viii}) == [1 3]) & (varargin{1,viii} == 'AoA'))
            if ((~isempty(varargin{1,viii+1})) & (isnumeric(varargin{1,viii+1})))
                alfa = varargin{1,viii+1};
                alfa = alfa(:);
                alfaR = alfa;
                alfaMid = mean(alfa);
                if (size(varargin{1,viii+1}) == [1 1])
                    alfaR = [0:alfa/50:2*alfa].';
                end
                viii = viii + 2;
            else
                alfa = 0;
                alfaMid = 0;
                alfaR = [];
                viii = viii + 1;
                disp('WARNING: AoA value is empty or not numeric or not after ''AoA''
      expression.');
                disp('         AoA is assumed to be zero.');
                disp(' ');
            end %if4
        elseif (prod(size(varargin{1,viii}) == [1 4]) & (varargin{1,viii} == 'Itip'))
            if ( ((~isempty(varargin{1,viii+1})) & (isnumeric(varargin{1,viii+1}))) & ...
                    ((varargin{1,viii+1} >= 0) & (size(varargin{1,viii+1}) == [1 1])) )
                minertT = varargin{1,viii+1};
                viii = viii + 2;
                if minertT > minert0
                    minertT = minert0;
                    disp('WARNING: I_tip cannot be greater than I_0. Assumed: I_tip =
      I_0.');
                    disp(' ');
                end
            else
                minertT = minert0;
                viii = viii + 1;
                disp('WARNING: I_tip value is empty/matrix/less than zero/not numeric/not
      after ''Itip'' expression.');
```

```
            disp('         I_tip is assumed to be equal to I_0 (constant stiffness
distribution).');
            disp(' ');
          end %if4
        elseif (prod(size(varargin{1,viii}) == [1 2]) & (varargin{1,viii} == 'Mt'))
          if ( ((~isempty(varargin{1,viii+1})) & (isnumeric(varargin{1,viii+1}))) & ...
                ((varargin{1,viii+1} > 0) & (size(varargin{1,viii+1}) == [1 1])) )
            massTip = varargin{1,viii+1};
            if (((isempty(maLenT~=maLen0)) | ((maLenT~=maLen0)==1)) |
((isempty(minertT~=minert0)) | ((minertT~=minert0)==1)))
                maLenT = maLen0;
                minertT = minert0;
                disp('WARNING: The beam must be uniform to have a mass at the tip.');
                disp('         Assumed: m_tip = m_0  and  I_tip = I_0.');
                disp(' ');
            end
            viii = viii + 2;
          elseif varargin{1,viii+1} == 0
            massTip = 0;
            viii = viii + 2;
          else
            massTip = 0;
            viii = viii + 1;
            disp('WARNING: M_t value is empty/matrix/less than zero/not numeric/not
after ''Mt'' expression.');
            disp('         M_t is assumed to be equal to zero (no tip mass).');
            disp(' ');
          end %if4
        elseif ( (length(varargin{1,viii}) >= 1) & (isnumeric(varargin{1,viii})) )
          rtrSpdR = varargin{1,viii};
          rtrSpdRng = [rtrSpdR(:)].';
          if (length(rtrSpdRng) == 1) & (rtrSpdRng ~= 0)
            rtrSpdRng = [0:rtrSpdRng/50:2*rtrSpdRng];
          elseif (rtrSpdRng == 0)
            rtrSpdRng = rtrSpdRngDF;
          end
          viii = viii + 1;
        elseif ((length(varargin{1,viii}) == 3) & (varargin{1,viii} == '...'))
          viii = 999;
        elseif isempty(varargin{1,viii})
          viii = viii + 1;
        else
          viii = viii + 1;
          errordlg({'An input argument could not be attributed to any variable and was
disregarded... ' ...
                'The other inputs may be attributed, but do not count on the results
you get!' ...
            'ALTHOUGH THE PROGRAMS MAY RUN, PLEASE CHECK YOUR INPUT ARGUMENTS AND TRY
AGAIN!'},'WARNING','modal');
          disp('WARNING: An input argument could not be attributed to any variable and
was disregarded.');
          disp('         The other inputs may be attributed. But do not count on the
results you get!');
          disp('         ALTHOUGH THE PROGRAM MAY RUN, PLEASE CHECK YOUR INPUT
ARGUMENTS AND TRY AGAIN!');
          disp(' ');
        end %if3
    end %while2
end %if1
if isempty(alfaR) | (alfaR == 0)
    alfaR = [0:0.1:6.3].';
end

%Check for empty VARARGIN arguments:
if isempty(maLenT)
    maLenT = maLen0;
    disp('ATTENTION: You did not specify any input argument for mass per length at the
  tip.');
    disp('           Assumed: m_tip = m_0.');
    disp(' ');
end
if isempty(minertT)
    minertT = minert0;
    disp('ATTENTION: You did not specify any input argument for moment of inertia at the
  tip.');
    disp('           Assumed: I_tip = I_0.');
    disp(' ');
end
if isempty(massTip)
```

```
    massTip = 0;
    rr = 0;
    disp('ATTENTION: You did not specify any input argument for the concentrated mass at
      the tip.');
    disp('           Assumed: M_t = 0.');
    disp(' ');
end
if isempty(rtrSpdRng)
    rtrSpdRng = rtrSpdRngDF;
    disp('ATTENTION: You did not specify any input argument for the rotor speed range.');
    disp('           A default rotor speed range was assumed.');
    disp(' ');
end

%Check for operating rotor speed: (it must be less than the maximum rotor speed
      specified)
if ((rtrSpdRng(length(rtrSpdRng))) <= (rtrSpdRng(length(rtrSpdRng)-1)))
    opRtrSpd = rtrSpdRng(max(size(rtrSpdRng)));
    rtrSpdRng = rtrSpdRng(1:max(size(rtrSpdRng))-1);
else
    if ( (size(rtrSpdRng) == size(rtrSpdRngDF)) & (prod(rtrSpdRng == rtrSpdRngDF) == 1) )
        opRtrSpd = opRtrSpdDF;
    else
        opRtrSpd = mean(rtrSpdRng);
    end
    disp('WARNING: No operating rotor speed has been input as the last element of the
      rtrSpdRng input argument...');
    disp('         An arbitrary value has been assigned for the operating rotor speed.');
    disp(' ');
end

%================================================================================
% READING THE a_n, K_0_n, Kh_1_n COEFFICIENT VALUES FROM THE YNTEMA CHARTS:
mtm0 = maLenT/maLen0;
EItEI0 = minertT/minert0;
rr = massTip/(maLen0*blength);

switch upper(cantHing(1))
case 'C'
    if (massTip == 0)
        % NR bending coefficients
        load yntecoef.mat *CLMS;
        a0 = a0CLMS;
        a100 = polyval(a100CLMS,mtm0);
        a105 = polyval(a105CLMS,mtm0);
        a110 = polyval(a110CLMS,mtm0);
        if (EItEI0 == 1)
            a1 = a110;
        elseif (EItEI0 == 0.5)
            a1 = a105;
        elseif (EItEI0 == 0)
            a1 = a100;
        else
            a1 = polyval(polyfit([0.0; 0.5; 1.0],[a100; a105; a110],2),EItEI0);
        end
        a200 = polyval(a200CLMS,mtm0);
        a205 = polyval(a205CLMS,mtm0);
        a210 = polyval(a210CLMS,mtm0);
        if (EItEI0 == 1)
            a2 = a210;
        elseif (EItEI0 == 0.5)
            a2 = a205;
        elseif (EItEI0 == 0)
            a2 = a200;
        else
            a2 = polyval(polyfit([0.0; 0.5; 1.0],[a200; a205; a210],2),EItEI0);
        end
        a300 = polyval(a300CLMS,mtm0);
        a305 = polyval(a305CLMS,mtm0);
        a310 = polyval(a310CLMS,mtm0);
        if (EItEI0 == 1)
            a3 = a310;
        elseif (EItEI0 == 0.5)
            a3 = a305;
        elseif (EItEI0 == 0)
            a3 = a300;
        else
            a3 = polyval(polyfit([0.0; 0.5; 1.0],[a300; a305; a310],2),EItEI0);
        end
```

```
    % zero-offset coefficients
    K00 =  K00CLMS;
    K0100 = polyval(K0100CLMS,mtm0);
    K0105 = polyval(K0105CLMS,mtm0);
    K0110 = polyval(K0110CLMS,mtm0);
    if (EItEI0 == 0)
        K01 = K0100;
    elseif (EItEI0 == 0.5)
        K01 = K0105;
    elseif (EItEI0 == 1)
        K01 = K0110;
    else
        K01 = polyval(polyfit([0.0; 0.5; 1.0],[K0100; K0105; K0110],2),EItEI0);
    end
    K0200 = polyval(K0200CLMS,mtm0);
    K0205 = polyval(K0205CLMS,mtm0);
    K0210 = polyval(K0210CLMS,mtm0);
    if (EItEI0 == 0)
        K02 = K0200;
    elseif (EItEI0 == 0.5)
        K02 = K0205;
    elseif (EItEI0 ==1)
        K02 = K0210;
    else
        K02 = polyval(polyfit([0.0; 0.5; 1.0],[K0200; K0205; K0210],2),EItEI0);
    end
    K0300 = polyval(K0300CLMS,mtm0);
    K0305 = polyval(K0305CLMS,mtm0);
    K0310 = polyval(K0310CLMS,mtm0);
    if (EItEI0 == 0)
        K03 = K0300;
    elseif (EItEI0 == 0.5)
        K03 = K0305;
    elseif (EItEI0 ==1)
        K03 = K0310;
    else
        K03 = polyval(polyfit([0.0; 0.5; 1.0],[K0300; K0305; K0310],2),EItEI0);
    end
    % offset-correction factors
    Kh10 = Kh10CLMS;
    Kh1100 = polyval(Kh1100CLMS,mtm0);
    Kh1105 = polyval(Kh1105CLMS,mtm0);
    Kh1110 = polyval(Kh1110CLMS,mtm0);
    if (EItEI0 == 0)
        Kh11 = Kh1100;
    elseif (EItEI0 == 0.5)
        Kh11 = Kh1105;
    elseif (EItEI0 ==1)
        Kh11 = Kh1110;
    else
        Kh11 = polyval(polyfit([0.0; 0.5; 1.0],[Kh1100; Kh1105; Kh1110],2),EItEI0);
    end
    Kh1200 = polyval(Kh1200CLMS,mtm0);
    Kh1205 = polyval(Kh1205CLMS,mtm0);
    Kh1210 = polyval(Kh1210CLMS,mtm0);
    if (EItEI0 == 0)
        Kh12 = Kh1200;
    elseif (EItEI0 == 0.5)
        Kh12 = Kh1205;
    elseif (EItEI0 ==1)
        Kh12 = Kh1210;
    else
        Kh12 = polyval(polyfit([0.0; 0.5; 1.0],[Kh1200; Kh1205; Kh1210],2),EItEI0);
    end
    Kh1300 = polyval(Kh1300CLMS,mtm0);
    Kh1305 = polyval(Kh1305CLMS,mtm0);
    Kh1310 = polyval(Kh1310CLMS,mtm0);
    if (EItEI0 == 0)
        Kh13 = Kh1300;
    elseif (EItEI0 == 0.5)
        Kh13 = Kh1305;
    elseif (EItEI0 ==1)
        Kh13 = Kh1310;
    else
        Kh13 = polyval(polyfit([0.0; 0.5; 1.0],[Kh1300; Kh1305; Kh1310],2),EItEI0);
    end
elseif (~isempty(massTip))
    load yntecoef.mat *NUCMt;
    if isempty(teta0sqNUCMt)
```

```
            a0 = [];
        else
            a0 = polyval(teta0sqNUCMt,rr); %when teta0sqNUCMt is empty, POLYVAL results
      to zero for every rr.
        end
        a1 = polyval(teta1sqNUCMt,rr);
        a2 = polyval(teta2sqNUCMt,rr);
        a3 = polyval(teta3sqNUCMt,rr);
        if isempty(K00NUCMt)
            K00 = [];
        else
            K00 = polyval(K00NUCMt,rr); %when K00NUHMt is empty, POLYVAL results to zero
      for every rr.
        end
        K01 = polyval(K01NUCMt,rr);
        K02 = polyval(K02NUCMt,rr);
        K03 = polyval(K03NUCMt,rr);
        Kh10 = []; Kh11 = []; Kh12 = []; Kh13 = [];
    else
        disp('THERE IS SOMETHING WRONG IN SWITCH CASE ''C'' ');
        disp(' ');
    end %if2
%HINGED BEAMS
case 'H'
    if (massTip == 0)
        % NR bending coefficients
        load yntecoef.mat *HLMS;
        a0 = a0HLMS;
        a100 = polyval(a100HLMS,mtm0);
        a105 = polyval(a105HLMS,mtm0);
        a110 = polyval(a110HLMS,mtm0);
        if (EItEI0 == 1)
            a1 = a110;
        elseif (EItEI0 == 0.5)
            a1 = a105;
        elseif (EItEI0 == 0)
            a1 = a100;
        else
            a1 = polyval(polyfit([0.0; 0.5; 1.0],[a100; a105; a110],2),EItEI0);
        end
        a200 = polyval(a200HLMS,mtm0);
        a205 = polyval(a205HLMS,mtm0);
        a210 = polyval(a210HLMS,mtm0);
        if (EItEI0 == 1)
            a2 = a210;
        elseif (EItEI0 == 0.5)
            a2 = a205;
        elseif (EItEI0 == 0)
            a2 = a200;
        else
            a2 = polyval(polyfit([0.0; 0.5; 1.0],[a200; a205; a210],2),EItEI0);
        end
        a300 = polyval(a300HLMS,mtm0);
        a305 = polyval(a305HLMS,mtm0);
        a310 = polyval(a310HLMS,mtm0);
        if (EItEI0 == 1)
            a3 = a310;
        elseif (EItEI0 == 0.5)
            a3 = a305;
        elseif (EItEI0 == 0)
            a3 = a300;
        else
            a3 = polyval(polyfit([0.0; 0.5; 1.0],[a300; a305; a310],2),EItEI0);
        end
        % zero-offset coefficients
        K00 = K00HLMS;
        K0100 = polyval(K0100HLMS,mtm0);
        K0105 = polyval(K0105HLMS,mtm0);
        K0110 = polyval(K0110HLMS,mtm0);
        if (EItEI0 == 0)
            K01 = K0100;
        elseif (EItEI0 == 0.5)
            K01 = K0105;
        elseif (EItEI0 ==1)
            K01 = K0110;
        else
            K01 = polyval(polyfit([0.0; 0.5; 1.0],[K0100; K0105; K0110],2),EItEI0);
        end
        K0200 = polyval(K0200HLMS,mtm0);
```

```
    K0205 = polyval(K0205HLMS,mtm0);
    K0210 = polyval(K0210HLMS,mtm0);
    if (EItEI0 == 0)
        K02 = K0200;
    elseif (EItEI0 == 0.5)
        K02 = K0205;
    elseif (EItEI0 ==1)
        K02 = K0210;
    else
        K02 = polyval(polyfit([0.0; 0.5; 1.0],[K0200; K0205; K0210],2),EItEI0);
    end
    K0300 = polyval(K0300HLMS,mtm0);
    K0305 = polyval(K0305HLMS,mtm0);
    K0310 = polyval(K0310HLMS,mtm0);
    if (EItEI0 == 0)
        K03 = K0300;
    elseif (EItEI0 == 0.5)
        K03 = K0305;
    elseif (EItEI0 ==1)
        K03 = K0310;
    else
        K03 = polyval(polyfit([0.0; 0.5; 1.0],[K0300; K0305; K0310],2),EItEI0);
    end
    % offset-correction factors
    Kh10 = polyval(Kh10XXHLMS,mtm0);%%
    Kh1100 = polyval(Kh1100HLMS,mtm0);
    Kh1105 = polyval(Kh1105HLMS,mtm0);
    Kh1110 = polyval(Kh1110HLMS,mtm0);
    if (EItEI0 == 0)
        Kh11 = Kh1100;
    elseif (EItEI0 == 0.5)
        Kh11 = Kh1105;
    elseif (EItEI0 ==1)
        Kh11 = Kh1110;
    else
        Kh11 = polyval(polyfit([0.0; 0.5; 1.0],[Kh1100; Kh1105; Kh1110],2),EItEI0);
    end
    Kh1200 = polyval(Kh1200HLMS,mtm0);
    Kh1205 = polyval(Kh1205HLMS,mtm0);
    Kh1210 = polyval(Kh1210HLMS,mtm0);
    if (EItEI0 == 0)
        Kh12 = Kh1200;
    elseif (EItEI0 == 0.5)
        Kh12 = Kh1205;
    elseif (EItEI0 ==1)
        Kh12 = Kh1210;
    else
        Kh12 = polyval(polyfit([0.0; 0.5; 1.0],[Kh1200; Kh1205; Kh1210],2),EItEI0);
    end
    Kh1300 = polyval(Kh1300HLMS,mtm0);
    Kh1305 = polyval(Kh1305HLMS,mtm0);
    Kh1310 = polyval(Kh1310HLMS,mtm0);
    if (EItEI0 == 0)
        Kh13 = Kh1300;
    elseif (EItEI0 == 0.5)
        Kh13 = Kh1305;
    elseif (EItEI0 ==1)
        Kh13 = Kh1310;
    else
        Kh13 = polyval(polyfit([0.0; 0.5; 1.0],[Kh1300; Kh1305; Kh1310],2),EItEI0);
    end
elseif (~isempty(massTip))
    load yntecoef.mat *NUHMt;
    if isempty(teta0sqNUHMt)
        a0 = [];
    else
        a0 = polyval(teta0sqNUHMt,rr); %when teta0sqNUHMt is empty, POLYVAL results
  to zero for every rr.
    end
    a1 = polyval(teta1sqNUHMt,rr);
    a2 = polyval(teta2sqNUHMt,rr);
    a3 = polyval(teta3sqNUHMt,rr);
    if isempty(K00NUHMt)
        K00 = [];
    else
        K00 = polyval(K00NUHMt,rr); %when K00NUHMt is empty, POLYVAL results to zero
  for every rr.
    end
    K01 = polyval(K01NUHMt,rr);
```

```
        K02 = polyval(K02NUHMt,rr);
        K03 = polyval(K03NUHMt,rr);
        Kh10 = []; Kh11 = []; Kh12 = []; Kh13 = [];
    else
        disp('THERE IS SOMETHING WRONG IN SWITCH CASE ''H'' ');
        disp(' ');
    end %if2
otherwise
    disp('HOW CAN upper(cantHing(1)) BE DIFFERENT FROM ''C'' or ''H''? ');
    disp(' ');
end %switch1
%================================================================================
% MAKING THE CALCULATIONS FOR ROTATING BEAM FREQUENCIES:
if (~isempty(a0))
    w0NRsq = (a0^2*(melast*minert0/maLen0/blength^4)).*(30/pi)^2; %[cycles/minute]^2
    if massTip == 0
        K0 = K00 + (Kh10 * eoffset);
    else
        K0 = K00;
    end
    w0sq = w0NRsq + (K0 .* rtrSpdRng.^2);
else
    K0 = [];
    w0sq = [];
    disp('WARNING: The rigid body bending mode cannot be calculated because the Southwell
      coefficients are');
    disp('         empty for the 0th-mode. The beam may be cantilever for which the 0th-
      mode Southwell ');
    disp('         coefficients are empty...');
    disp('         Check below. If the beam is Cantilever, there is no problem:');
    if upper(cantHing(1)) == 'C'
        disp('         THE BEAM IS CANTILEVER. THERE IS NO 0th MODE FOR CANTILEVER
      BEAMS.');
    elseif upper(cantHing(1)) == 'H'
        disp('         THE BEAM IS HINGED. THERE MUST BE A 0th MODE!');
    else
        disp('         HEY, WHAT''S THE PROBLEM? ISN''T THE BEAM CANTILEVER OR HINGED?');
    end
    disp(' ');
end

w1NRsq = (a1^2*(melast*minert0/maLen0/blength^4)).*(30/pi)^2; %[cycles/minute]^2
if massTip == 0
    K1 = K01 + (Kh11 * eoffset);
else
    K1 = K01;
end
w1sq = w1NRsq + (K1 .* rtrSpdRng.^2);

w2NRsq = (a2^2*(melast*minert0/maLen0/blength^4)).*(30/pi)^2; %[cycles/minute]^2
if massTip == 0
    K2 = K02 + (Kh12 * eoffset);
else
    K2 = K02;
end
w2sq = w2NRsq + (K2 .* rtrSpdRng.^2);

w3NRsq = (a3^2*(melast*minert0/maLen0/blength^4)).*(30/pi)^2; %[cycles/minute]^2
if massTip == 0
    K3 = K03 + (Kh13 * eoffset);
else
    K3 = K03;
end
w3sq = w3NRsq + (K3 .* rtrSpdRng.^2);

%================================================================================
% INCORPORATING NECESSARY CORRECTIONS:

%Figure-2: (uniform hinged beam without a tip mass)
%(3% error correction for all rot-spd-prtr values)
if ( (((upper(cantHing(1)) == 'H') & (massTip == 0)) & (eoffset <= 0.10)) & ...
        (((mtm0 > 0.995) & (mtm0 <= 1)) & ((EItEI0 > 0.99) & (EItEI0 <= 1))) ) %almost
      linear
    ccc1 = (0.03/1.03/2) * (1 + 2 * K1);
    ccc2 = (0.03/1.03/2) * ((w2NRsq/w1NRsq) + 2 * K2);
    ccc3 = (0.03/1.03/2) * ((w3NRsq/w1NRsq) + 2 * K3);
    w1sq = w1sq - (ccc1 .* (rtrSpdRng.^2));
    w2sq = w2sq - (ccc2 .* (rtrSpdRng.^2));
    w3sq = w3sq - (ccc3 .* (rtrSpdRng.^2));
```

```
end

%Figure-5: (uniform cantilever beams with (0%~10%] offset)
%(respectively 9.7%~8.3% error correction for high [>2] rotational-speed parameter
      values)
if ( ((upper(cantHing(1)) == 'C') & (massTip == 0)) & ...
        (((eoffset > 0.001) & (eoffset <= 0.10)) & (((mtm0 > 0.995) & (mtm0 <= 1)) & ...
        ((EItEI0 > 0.99) & (EItEI0 <= 1)))) )
    hhh = 2; %rayleigh results assumed accurate for rot-spd-prtr < 2.
    ccc1 = ((0.097 - 0.14 * eoffset)/(1.097 - 0.14 * eoffset)) * (1/(35.000001-hhh)) * (1
      + 35 * K1);
    ccc11 = (hhh * w1NRsq) .* ones(size(rtrSpdRng));
    %
    gii = max(size(rtrSpdRng)) + 1;
    tempRtrSpd = rtrSpdRng; %creating a rotor speed range to be used in error correction
    while (gii <= (max(size(rtrSpdRng))+1))
        gii = gii - 1;
        if ((rtrSpdRng(gii)^2 / w1NRsq) < hhh)
            for hii = 1:gii,
                tempRtrSpd(hii) = 0;
                ccc11(hii) = 0;
            end
            gii = max(size(rtrSpdRng)) + 10; %getting out of WHILE loop
        end %if3
    end %while2
    w1sq = w1sq - (ccc1 .* (tempRtrSpd.^2 - ccc11));
    tempRtrSpd = [];
end %if1

%Figure-6: ("linear" cantilever beams with (0%~10%] offset)
%(respectively 9.7%~8.3% error correction for high [>2] rotational-speed parameter
      values)
if ( ((upper(cantHing(1)) == 'C') & (massTip == 0)) & ...
        (((eoffset > 0.001) & (eoffset <= 0.10)) & (((mtm0 < 0.005) & (mtm0 >= 0)) & ...
        ((EItEI0 < 0.01) & (EItEI0 >= 0)))) )
    hhh = 4; %rayleigh results assumed accurate for rot-spd-prtr < 4.
    ccc1 = ((0.11 - 0.16 * eoffset)/(1.11 - 0.16 * eoffset)) * (1/(35.000001-hhh)) * (1 +
      35 * K1);
    ccc11 = (hhh * w1NRsq) .* ones(size(rtrSpdRng));
    %
    gii = max(size(rtrSpdRng)) + 1;
    tempRtrSpd = rtrSpdRng; %creating a rotor speed range to be used in error correction
    while (gii <= (max(size(rtrSpdRng))+1))
        gii = gii - 1;
        if ((rtrSpdRng(gii)^2 / w1NRsq) < hhh)
            for hii = 1:gii,
                tempRtrSpd(hii) = 0;
                ccc11(hii) = 0;
            end
            gii = max(size(rtrSpdRng)) + 10; %getting out of WHILE loop
        end %if3
    end %while2
    w1sq = w1sq - (ccc1 .* (tempRtrSpd.^2 - ccc11));
    tempRtrSpd = [];
end %if1

%Figure-7: (uniform and "linear" cantilever beams with zero offset)
%(10% error correction for high [>2] rotational-speed parameter values)
if ( ((upper(cantHing(1)) == 'C') & (massTip == 0)) & ...
        ((eoffset <= 0.001) & (((mtm0 >= 0) & (mtm0 <= 1)) & ...
        ((EItEI0 >= 0) & (EItEI0 <= 1)))) )
    hhh = 2; %rayleigh results assumed accurate for rot-spd-prtr < 2.
    ccc1 = (0.1/1.1) * (1/(35.000001-hhh)) * (1 + 35 * K1);
    ccc11 = (hhh * w1NRsq) .* ones(size(rtrSpdRng));
    %
    gii = max(size(rtrSpdRng)) + 1;
    tempRtrSpd = rtrSpdRng; %creating a rotor speed range to be used in error correction
    while (gii <= (max(size(rtrSpdRng))+1))
        gii = gii - 1;
        if ((rtrSpdRng(gii)^2 / w1NRsq) < hhh)
            for hii = 1:gii,
                tempRtrSpd(hii) = 0;
                ccc11(hii) = 0;
            end
            gii = max(size(rtrSpdRng)) + 10; %getting out of WHILE loop
        end %if3
    end %while2
    w1sq = w1sq - (ccc1 .* (tempRtrSpd.^2 - ccc11));
    tempRtrSpd = [];
```

```
end %if1

%Figure-8: (uniform cantilever beam w/ zero offset and w/ tip mass)
%(%13 error correction for high [>5] rotational-speed parameter and for all r values)
if ( ((upper(cantHing(1)) == 'C') & (massTip > 0)) & (eoffset <= 0.001 & ... %zero or
      very small offset
        (((mtm0 > 0.995) & (mtm0 <= 1)) & ((EItEI0 > 0.99) & (EItEI0 <= 1)))) ) %linear
      or almost linear
    hhh = 5; %for rot-spd-prtr <= 5, the result is assumed to be accurate.
    ccc11 = ((0.13/1.13) * (1 + 200 * K1)) / (200.00001 - hhh);
    ccc12 = (hhh * w1NRsq) .* ones(size(rtrSpdRng));
    %
    gii = max(size(rtrSpdRng)) + 1;
    tempRtrSpd = rtrSpdRng; %creating a rotor speed range to be used in error correction
    while (gii <= (max(size(rtrSpdRng))+1))
        gii = gii - 1;
        if ((rtrSpdRng(gii)^2 / w1NRsq) <= hhh)
            for hii = 1:gii,
                tempRtrSpd(hii) = 0;
                ccc12(hii) = 0;
            end
            gii = max(size(rtrSpdRng)) + 10; %getting out of WHILE loop
        end %if3
    end %while2
    w1sq = w1sq - (ccc11 .* (tempRtrSpd.^2 - ccc12));
    tempRtrSpd = [];
end %if1

%==================================================================================
% ANGLE OF ATTACK COMPUTATIONS:
if (~isempty(alfa))
    if (length(alfa) == 1)
        tempKsi = (sin(alfa) .* rtrSpdRng.').^2;
        tempKsiR = (rtrSpdRng.' * sin(alfaR.')).^2; %'...R' is for plotting purposes
      only.
        if (~isempty(w0sq))
            w0ksi = w0sq.' - tempKsi;
            w0ksiR = (w0sq.' * ones(size(alfaR.'))) - tempKsiR;
        else
            w0ksi = [];
            w0ksiR = [];
        end
        w1ksi = w1sq.' - tempKsi;
        w2ksi = w2sq.' - tempKsi;
        w3ksi = w3sq.' - tempKsi;
        w1ksiR = (w1sq.' * ones(size(alfaR.'))) - tempKsiR;
        w2ksiR = (w2sq.' * ones(size(alfaR.'))) - tempKsiR;
        w3ksiR = (w3sq.' * ones(size(alfaR.'))) - tempKsiR;
        tempKsiMid = (rtrSpdRng.' .* sin(alfaMid)).^2; %'...Mid' is for plotting purposes
      only.
    else
        tempKsi = (rtrSpdRng.' * sin(alfa.')).^2;
        tempKsiMid = (rtrSpdRng.' .* sin(alfaMid)).^2; %'...Mid' is for plotting purposes
      only.
        if (~isempty(w0sq))
            w0ksi = (w0sq.' * ones(1,length(alfa))) - tempKsi;
            w0ksiMid = w0sq.' - tempKsiMid;
        else
            w0ksi = [];
            w0ksiMid = [];
        end
        w1ksi = (w1sq.' * ones(1,length(alfa))) - tempKsi;
        w1ksiMid = w1sq.' - tempKsiMid;
        w2ksi = (w2sq.' * ones(1,length(alfa))) - tempKsi;
        w2ksiMid = w2sq.' - tempKsiMid;
        w3ksi = (w3sq.' * ones(1,length(alfa))) - tempKsi;
        w3ksiMid = w3sq.' - tempKsiMid;
    end
end
%================================================================================
% OUTPUT ARGUMENTS:
w1ynte = w1sq.^0.5;
w2ynte = w2sq.^0.5;
w3ynte = w3sq.^0.5;
w0ynte = w0sq.^0.5;
if w0ynte == []
    w0ynte = 0;
end
for ss = 2:length(rtrSpdRng),
```

```
    if (rtrSpdRng(ss) >= opRtrSpd) & (rtrSpdRng(ss-1) < opRtrSpd)
        rlim = ss - 1;
    end
end
w1op = w1ynte(rlim)+((w1ynte(rlim+1)-w1ynte(rlim))*(opRtrSpd-
      rtrSpdRng(rlim))/(rtrSpdRng(rlim+1)-rtrSpdRng(rlim)));
w2op = w2ynte(rlim)+((w2ynte(rlim+1)-w2ynte(rlim))*(opRtrSpd-
      rtrSpdRng(rlim))/(rtrSpdRng(rlim+1)-rtrSpdRng(rlim)));
w3op = w3ynte(rlim)+((w3ynte(rlim+1)-w3ynte(rlim))*(opRtrSpd-
      rtrSpdRng(rlim))/(rtrSpdRng(rlim+1)-rtrSpdRng(rlim)));
if length(rtrSpdRng) == length(w0ynte)
    w0op = w0ynte(rlim)+((w0ynte(rlim+1)-w0ynte(rlim))*(opRtrSpd-
      rtrSpdRng(rlim))/(rtrSpdRng(rlim+1)-rtrSpdRng(rlim)));
else
    w0op = 0;
end
w1ynt = [w1op w1ynte];
w2ynt = [w2op w2ynte];
w3ynt = [w3op w3ynte];
w0ynt = [w0op w0ynte];


if (~isempty(alfa))
    if length(alfa) == 1
        w1kt = w1ksi;
        w2kt = w2ksi;
        w3kt = w3ksi;
        if (~isempty(w0sq))
            w0kt = w0ksi;
        else
            w0kt = [];
        end
    else
        w1kt = w1ksiMid;
        w2kt = w2ksiMid;
        w3kt = w3ksiMid;
        if (~isempty(w0sq))
            w0kt = w0ksiMid;
        else
            w0kt = [];
        end
    end
    w1ksisqrt = w1kt.^0.5;
    w2ksisqrt = w2kt.^0.5;
    w3ksisqrt = w3kt.^0.5;
    w0ksisqrt = w0kt.^0.5;
    % finding frequencies for opRtrSpd:
    for ss = 2:length(rtrSpdRng),
        if (rtrSpdRng(ss) >= opRtrSpd) & (rtrSpdRng(ss-1) < opRtrSpd)
            rlim = ss - 1;
        end
    end
    w1kop = w1ksisqrt(rlim)+((w1ksisqrt(rlim+1)-w1ksisqrt(rlim))*(opRtrSpd-
      rtrSpdRng(rlim))/(rtrSpdRng(rlim+1)-rtrSpdRng(rlim)));
    w2kop = w2ksisqrt(rlim)+((w2ksisqrt(rlim+1)-w2ksisqrt(rlim))*(opRtrSpd-
      rtrSpdRng(rlim))/(rtrSpdRng(rlim+1)-rtrSpdRng(rlim)));
    w3kop = w3ksisqrt(rlim)+((w3ksisqrt(rlim+1)-w3ksisqrt(rlim))*(opRtrSpd-
      rtrSpdRng(rlim))/(rtrSpdRng(rlim+1)-rtrSpdRng(rlim)));
    if length(rtrSpdRng) == size(w0ksisqrt,1)
        w0kop = w0ksisqrt(rlim)+((w0ksisqrt(rlim+1)-w0ksisqrt(rlim))*(opRtrSpd-
      rtrSpdRng(rlim))/(rtrSpdRng(rlim+1)-rtrSpdRng(rlim)));
    else
        w0kop = 0;
    end
    w1ksisqrt = [w1kop ; w1ksisqrt];
    w2ksisqrt = [w2kop ; w2ksisqrt];
    w3ksisqrt = [w3kop ; w3ksisqrt];
    w0ksisqrt = [w0kop ; w0ksisqrt];
else
    w1ksisqrt = w1ynt.';
    w2ksisqrt = w2ynt.';
    w3ksisqrt = w3ynt.';
    w0ksisqrt = w0ynt.';
end

omegaRng = [opRtrSpd rtrSpdRng];
if (nargout == 5)
    varargout = {}; %suppressing error output display.
elseif (nargout == 6)
```

```
    varargout = {{a0,a1,a2,a3,K0,K1,K2,K3,K00,K01,K02,K03,Kh10,Kh11,Kh12,Kh13}}; %output
      is a cell array!
elseif ( (nargout == 9) & (~isempty(alfa)) )
    varargout = {w1ksisqrt,w2ksisqrt,w3ksisqrt,w0ksisqrt}; %output is a cell array!
elseif ( (nargout == 10) & (~isempty(alfa)) )
    varargout = {{a0,a1,a2,a3,K0,K1,K2,K3,K00,K01,K02,K03,Kh10,Kh11,Kh12,Kh13}, ...
            w1ksisqrt,w2ksisqrt,w3ksisqrt,w0ksisqrt}; %output is a cell array!
elseif (nargout == 0)
else
    disp('ATTENTION: The number of output arguments is not 5, 6, 9, 10;');
    disp('           or it is 9 or 10 but AoA expression is empty!...');
    disp('           The output arguments may be improper.');
    disp(' ');
end
%=================================================================================
% SOUTHWELL PLOT:
if ( (flagSouth == 1) | (flagSouth == 2) )
    bii = 0;
    while (bii >= 0)
        bii = bii + 1;
        if ((bii * max(rtrSpdRng)) > max([w0ynte w1ynte w2ynte w3ynte]))
            nnn = bii + 1; %maximum n value of n-per-rev on the plot
            bii = -9; %getting out of WHILE loop
        end
    end
    figure;
    if ((~isempty(w0ynte)) & (size(w0ynte) == size(rtrSpdRng)))
        plot(rtrSpdRng,w0ynte,'r-','linewidth',2); hold on;
        text(max(rtrSpdRng),max(w0ynte),'
      w_0','color','r','fontsize',12,'fontweight','bold');
        egz = '';
    else
        disp(['0th-mode Bending Frequency can not be plotted... Its value is (may be
      empty!): ' num2str(w0ynte)]);
        disp('But it is assumed to be (1.03 * 1P) which is plotted for example.');
        disp(' ');
        plot(rtrSpdRng,1.03.*rtrSpdRng,'r--','linewidth',2); hold on;
        text(max(rtrSpdRng),max(1.03.*rtrSpdRng),'
      w_0','color','r','fontsize',12,'fontweight','bold');
        egz = 'Example ';
    end
    plot(rtrSpdRng,w1ynte,'gs-'); hold on;
    text(max(rtrSpdRng),max(w1ynte),'
      w_1','color','g','fontsize',12,'fontweight','bold');
    plot(rtrSpdRng,w2ynte,'md-'); hold on;
    text(max(rtrSpdRng),max(w2ynte),'
      w_2','color','m','fontsize',12,'fontweight','bold');
    plot(rtrSpdRng,w3ynte,'bo-'); hold on;
    text(max(rtrSpdRng),max(w3ynte),'
      w_3','color','b','fontsize',12,'fontweight','bold');
    plot(opRtrSpd,[0:max(rtrSpdRng)*nnn/100:max(rtrSpdRng)*nnn],'cx-'); hold on;
    for iii = 1:nnn,
        plot(rtrSpdRng,rtrSpdRng.*iii,'k:');
        text(max(rtrSpdRng),max(rtrSpdRng.*iii),{[' ',num2str(iii),'P']});
        hold on;
    end
    if (flagSouth == 1)
        title({'SOUTHWELL PLOT' 'Blade Natural Frequencies VS Rotational Speed'});
        xlabel('Rotor Speed, [RPM]'); ylabel('Blade Bending Frequency, [CPM]'); grid on;
      hold on;
        legend([egz 'Rigid Body Flapping Mode'],'1st Flap Mode','2nd Flap Mode','3rd Flap
      Mode','Operating Rotor Speed',2);
        %%info text on the plot:
        format short;
        text(min(rtrSpdRng)+11.33,max(rtrSpdRng)*nnn*0.615,{['Cantilever or Hinged ? = '
      cantHing]; ...
                ['Blade Length = ' num2str(blength) ' in.'];['Root offset (e/L) = '
      num2str(eoffset)]; ...
                ['m_t/m_0 = ' num2str(mtm0)];['EI_t/EI_0 = '
      num2str(EItEI0)];['M_T/(m_0*L) = r = ' num2str(rr)]; ...
                ['Operating RPM = ' num2str(opRtrSpd)]},'fontsize',8);
        hold on;
    end
end
%
if ( (flagSouth == 2) & (~isempty(alfa)) )
    if (~isempty(w0ksi))
        if (size(w0ksi,2) == 1)
            plot(rtrSpdRng,w0ksi.^0.5,'r:','linewidth',3); hold on;
```

```
            text(max(rtrSpdRng),max(w0ksi.^0.5),'
      w_k_s_i_,_0','color','r','fontsize',10,'fontweight','bold');
        else
            plot(rtrSpdRng,w0ksiMid.^0.5,'r:','linewidth',3); hold on;
            text(max(rtrSpdRng),max(w0ksiMid.^0.5),'          w_k_s_i_,_0_,_a_v_e',...
                'color','r','fontsize',10,'fontweight','bold');
        end
        egz2 = '';
    else
        disp(['AoA 0th-mode Bending Frequency can not be plotted... Its value is (may be
      empty!): ' num2str(w0ksi.^0.5)]);
        disp('But it is assumed to be (1.03*1P - omega^2*sin(AoA)^2) which is plotted for
      example.');
        disp(' ');
        egz2 = 'Example ';
        if (size(w0ksi,2) == 1)
            plot(rtrSpdRng,((1.03 .* rtrSpdRng).^2 - tempKsi.').^0.5,'r:','linewidth',3);
      hold on;
            text(max(rtrSpdRng),max(((1.03 .* rtrSpdRng).^2 - tempKsi.').^0.5),'
      w_k_s_i_,_0',...
                'color','r','fontsize',10,'fontweight','bold');
        else
            plot(rtrSpdRng,((1.03 .* rtrSpdRng).^2 -
      tempKsiMid.').^0.5,'r:','linewidth',3); hold on;
            text(max(rtrSpdRng),max(((1.03 .* rtrSpdRng).^2 - tempKsiMid.').^0.5),'
      w_k_s_i_,_0_,_a_v_e',...
                'color','r','fontsize',10,'fontweight','bold');
        end
    end
    if (size(w1ksi,2) == 1)
        plot(rtrSpdRng,w1ksi.^0.5,'g:','linewidth',5); hold on;
        text(max(rtrSpdRng),max(w1ksi.^0.5),'
      w_k_s_i_,_1','color','g','fontsize',10,'fontweight','bold');
    else
        plot(rtrSpdRng,w1ksiMid.^0.5,'g:','linewidth',5); hold on;
        text(max(rtrSpdRng),max(w1ksiMid.^0.5),'
      w_k_s_i_,_1_,_a_v_e','color','g','fontsize',10,'fontweight','bold');
    end
    if (size(w2ksi,2) == 1)
        plot(rtrSpdRng,w2ksi.^0.5,'m:','linewidth',5); hold on;
        text(max(rtrSpdRng),max(w2ksi.^0.5),'
      w_k_s_i_,_2','color','m','fontsize',10,'fontweight','bold');
    else
        plot(rtrSpdRng,w2ksiMid.^0.5,'m:','linewidth',5); hold on;
        text(max(rtrSpdRng),max(w2ksiMid.^0.5),'
      w_k_s_i_,_2_,_a_v_e','color','m','fontsize',10,'fontweight','bold');
    end
    if (size(w3ksi,2) == 1)
        plot(rtrSpdRng,w3ksi.^0.5,'b:','linewidth',5); hold on;
        text(max(rtrSpdRng),max(w3ksi.^0.5),'
      w_k_s_i_,_3','color','b','fontsize',10,'fontweight','bold');
    else
        plot(rtrSpdRng,w3ksiMid.^0.5,'b:','linewidth',5); hold on;
        text(max(rtrSpdRng),max(w3ksiMid.^0.5),'
      w_k_s_i_,_3_,_a_v_e','color','b','fontsize',10,'fontweight','bold');
    end
    title({'SOUTHWELL PLOT WITH AoA' 'Blade Bending Frequencies (with zero and nonzero
      AoA) V.S. Rotational Speed'});
    xlabel('Rotor Speed, [RPM]'); ylabel('Blade Bending Frequency, [CPM]'); grid on; hold
      on;
    legend([egz 'Rigid Body Flapping Mode'],'1st Flap Mode','2nd Flap Mode','3rd Flap
      Mode','Operating Rotor Speed', ...
           [egz2 'Rigid Body Flapping Mode at AoA'],'1st Flap Mode at AoA','2nd Flap Mode
      at AoA','3rd Flap Mode at AoA',2);
    %%info text on the plot:
    format short;
    text(min(rtrSpdRng)+11.33,max(rtrSpdRng)*nnn*0.4,{['Cantilever or Hinged ? = '
      cantHing]; ...
            ['Blade Length = ' num2str(blength) ' in.'];['Root offset (e/L) = '
      num2str(eoffset)]; ...
            ['m_t/m_0 = ' num2str(mtm0)];['EI_t/EI_0 = ' num2str(EItEI0)];['M_T/(m_0*L) =
      r = ' num2str(rr)]; ...
            ['Operating RPM = ' num2str(opRtrSpd)];['AoA = ' num2str(alfaMid) '
      rad']},'fontsize',8);
end
%
if ( (flagSouth == 3) & (~isempty(alfa)) )
    figure; hold on;
    if (length(alfa) == 1)
```

```
        if (~isempty(w0ksi))
            mesh(alfaR,rtrSpdRng,w0ksiR.^0.5);

      text(min(alfaR),max(rtrSpdRng),max(max(w0ksiR.^0.5)),'w_k_s_i_,_0','color','r','font
      size',12,'fontweight','bold');
        else
            disp('ATTENTION: w_0_ksi could not be plotted by MESH() because it is
      empty...');
            disp(' ');
        end
        mesh(alfaR,rtrSpdRng,w1ksiR.^0.5);

      text(min(alfaR),max(rtrSpdRng),max(max(w1ksiR.^0.5)),'w_k_s_i_,_1','color','g','font
      size',12,'fontweight','bold');
        mesh(alfaR,rtrSpdRng,w2ksiR.^0.5);

      text(min(alfaR),max(rtrSpdRng),max(max(w2ksiR.^0.5)),'w_k_s_i_,_2','color','m','font
      size',12,'fontweight','bold');
        mesh(alfaR,rtrSpdRng,w3ksiR.^0.5);

      text(min(alfaR),max(rtrSpdRng),max(max(w3ksiR.^0.5)),'w_k_s_i_,_3','color','b','font
      size',12,'fontweight','bold');
    else
        if (~isempty(w0ksi))
            mesh(alfaR,rtrSpdRng,w0ksi.^0.5);

      text(min(alfaR),max(rtrSpdRng),max(max(w0ksi.^0.5)),'w_k_s_i_,_0','color','r','fonts
      ize',12,'fontweight','bold');
        else
            disp('ATTENTION: w_0_ksi could not be plotted by MESH() because it is
      empty...');
            disp(' ');
        end
        mesh(alfaR,rtrSpdRng,w1ksi.^0.5);

      text(min(alfaR),max(rtrSpdRng),max(max(w1ksi.^0.5)),'w_k_s_i_,_1','color','g','fonts
      ize',12,'fontweight','bold');
        mesh(alfaR,rtrSpdRng,w2ksi.^0.5);

      text(min(alfaR),max(rtrSpdRng),max(max(w2ksi.^0.5)),'w_k_s_i_,_2','color','m','fonts
      ize',12,'fontweight','bold');
        mesh(alfaR,rtrSpdRng,w3ksi.^0.5);

      text(min(alfaR),max(rtrSpdRng),max(max(w3ksi.^0.5)),'w_k_s_i_,_3','color','b','fonts
      ize',12,'fontweight','bold');
    end
    colorbar; grid on; view(110,20);
    title({'SOUTHWELL MESH FOR AoA' 'Blade Bending Frequencies V.S. Rotational Speed &
      AoA'});
    xlabel('AoA, [rad]'); ylabel('Rotor Speed, [RPM]'); zlabel('Blade Bending Frequency,
      [CPM]');
    %%info text on the plot:
    format short;
    text(mean(alfaR),mean(rtrSpdRng)/3,max(max(w3ksi.^0.5))/1.4,{['Cantilever or Hinged ?
      = ' cantHing]; ...
            ['Blade Length = ' num2str(blength) ' in.'];['Root offset (e/L) = '
      num2str(eoffset)]; ...
            ['m_t/m_0 = ' num2str(mtm0)];['EI_t/EI_0 = ' num2str(EItEI0)];['M_T/(m_0*L) =
      r = ' num2str(rr)]; ...
            ['Operating RPM = ' num2str(opRtrSpd)];['AoA = ' num2str(alfaMid) '
      rad']},'fontsize',8);
end
%hold off;
%===============================================================================
totalruntime = cputime - runtimestart;
disp(['Total Run Time is ' num2str(totalruntime) ' sec.']);
% Turning Diary off will save a diary text file of this run of Yntema.m in the
      \yntemarundiary\ directory:
% (The numbers after the date in the filename are the numbers created by NOW command of
      MATLAB)
%===============================================================================
%Output for diary file:
disp(['Cantilever or Hinged ? = ' cantHing(1)]);
disp(['Blade Length = ' num2str(blength) ' in.']);
disp(['Root offset (e/L) = ' num2str(eoffset)]);
disp(['m_t/m_0 = ' num2str(mtm0)]);
disp(['EI_t/EI_0 = ' num2str(EItEI0)]);
disp(['M_T/(m_0*L) = r = ' num2str(rr)]);
if ((~isempty(alfa)) & (length(alfa) == 1))
```

```
    disp(['Angle of Attack = ' num2str(alfa) ' rad.']);
elseif ((~isempty(alfa)) & (length(alfa) > 1))
    disp(['Angle of Attack (average)= ' num2str(alfaMid) ' rad.']);
end
disp(['Operating Rotor Speed = ' num2str(opRtrSpd) ' RPM']);
disp(['Rigid Bending Frequency (at ',num2str(opRtrSpd),' RPM) = ', num2str(w0op), '
      CPM']);
disp(['First Bending Frequency (at ',num2str(opRtrSpd),' RPM) = ', num2str(w1op), '
      CPM']);
disp(['Second Bending Frequency (at ',num2str(opRtrSpd),' RPM) = ', num2str(w2op), '
      CPM']);
disp(['Third Bending Frequency (at ',num2str(opRtrSpd),' RPM) = ', num2str(w3op), '
      CPM']);
disp('==================== END OF YNTEMA() FILE ==========================');
%==============================================================================

diary off;
```

## B. MATLAB® CODE FOR *YNTERYGH* FUNCTION

```
function [wn,varargout] = ynterygh(ynR,varargin)

%[wn <,an <,Kn> <,K0n,K1n>>] = YNTERYGH(ynR <,'ninter',nInt> <,'L',blength> <,'mx',mR>
      <,'EIx',EIR>
%                                       <,'eofs',eoffset> <,'omega',rtrSpdRng>)
%This function numerically calculates the bending frequencies of rotating beams from the
      knowledge of
%the mode shapes of rotating beams. It is the method developed by Yntema who applied
      Rayleigh approach
%to a rotating beam. The method theoretically gives the exact result, but the precision
      of numerical
%computation here is, as always, directly proportional to the size of ynR vector.
%Inputs:
%       ynR is the deflection of any mode shape. It can be input as the actual/normalized
      deflection
%           values taken from linearly-spaced stations, only as a matrix with one, two,
      or three columns
%           and more than four rows. (Second column corresponds to first derivative
      values, whereas
%           third column to second derivative values. If only one column, derivatives are
      computed from
%           the deflection values.) It can also be input as the coefficients of the
      polynomial for
%           deflection values, only as a matrix with one, two, or three rows (and any
      number of columns)
%           corresponding to deflection, 1st derivative, 2nd derivative values
      respectively. You can
%           input it as a character or cell array for the distribution expression, e.g.
      'sin(x)*x^2'
%           in MATLAB® notation.
%       ninter is the required number of linear intervals between zero and blade length.
      The higher the
%           nInt, the more precise the result and the longer the time elapsed. Default is
      100.
%       L is the blade length. Default is 1 without any units. But you may input the real
      value in inches.
%       mx is the mass distribution along the blade from the root to the tip. You can
      input it as a
%           constant scalar, an actual-value column vector corresponding to linearly-
      spaced stations,
%           a row vector of polynomial coefficients, or a character array of MATLAB®
      expression.
%           Default is a constant value, 0.001 lbm/in.
%       EIx is stiffness distribution along the blade from the root to the tip. You can
      input it as a
%           constant scalar, an actual-value column vector corresponding to linearly-
      spaced stations,
%           a row vector of polynomial coefficients, or a character array of MATLAB®
      expression.
%           Default is a constant value, 1e7 psi-in^4.
%       eofs is the hinge offset. You can input as an actual value in inches (when eofs
      >= 1), as an
%           e/L ratio, or a percentage value character ('10%' or '%10' or '1%0'). Default
      is 0.
%       omega is the rotor speed range in RPM. Default is [0:5:700].
%Outputs:
```

```
%       wn is the rotating beam bending frequency range corresponding to omega in cycles
      per minute.
%       an is the non-rotating beam bending frequency coefficient.
%       Kn is the Southwell coefficient.(=K0n+e*K1n)
%       K0n is the zero-offset Southwell coefficient.
%       K1n is the offset-correction factor for the Southwell coefficient.
%
%<<<<<<<<<<<<<<1st.Lt.H.E.AKIN, Master of Science in Aeronautical
      Engineering,>>>>>>>>>>>>>>>>>>>>>>>>>>
%<<<<<<<<<<<<<<<<<<<<<Naval Postgraduate School, CA, USA, September
      2002.>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%=====================================================================================
home;
wnR = [];
%READING VARARGIN:===================================================================
if nargin == 0
    plotme;
    disp('PLEASE INPUT AT LEAST ONE INPUT ARGUMENT TO YNTERYGH() FUNCTION!');
elseif nargin < 1
    disp('ERROR: The number of input arguments can not be less than 1 (one).');
    disp('Early Termination: PLEASE TRY AGAIN WITH PROPER NUMBER OF INPUT ARGUMENTS!');
    disp(' ');
elseif nargin > 13
    disp('ERROR: The number of input arguments can not be more than 13 (thirteen).');
    disp('Early Termination: PLEASE TRY AGAIN WITH PROPER NUMBER OF INPUT ARGUMENTS!');
    disp(' ');
end

%DEFAULTING:
nInt = 100; %[]
blength = 1; %[] or [in]
mR = 0.001; %[lbm/in]
EIR = 1e7; %[psi-in^4]
eoffset = 0; %e/L or e(>1) or 'e/L%'
rtrSpdRng = [0:5:700]; %[RPM]

nain = nargin-1;
if (nain >= 1)
    nn = 1;
    while (nn <= nain),
        if (prod(size(varargin{nn}) == [1 6]) & (varargin{nn} == 'ninter')) %look for
      number of intervals
            if ( ((~isempty(varargin{nn+1})) & (isnumeric(varargin{nn+1}))) & ...
                    ((varargin{nn+1} > 0) & (size(varargin{nn+1}) == [1 1])) )
                nInt = varargin{nn+1};
                nn = nn + 2;
            else
                nn = nn + 1;
                disp('WARNING: ninter input value is not an appropriate scalar value...
      ');
                disp('         ninter is defaulted to 100!');
                disp(' ');
            end %if4
        elseif (prod(size(varargin{nn}) == [1 1]) & (varargin{nn} == 'L')) %look for
      blade length input
            if ( ((~isempty(varargin{nn+1})) & (isnumeric(varargin{nn+1}))) & ...
                    ((varargin{nn+1} > 0) & (size(varargin{nn+1}) == [1 1])) )
                blength = varargin{nn+1};
                nn = nn + 2;
            else
                nn = nn + 1;
                disp('WARNING: L input value is not an appropriate scalar value... ');
                disp('         L is defaulted to 1 (one)!');
                disp(' ');
            end %if4
        elseif (prod(size(varargin{nn}) == [1 2]) & (varargin{nn} == 'mx')) %look for
      mass distribution input
            if (~isempty(varargin{nn+1}))
                mR = varargin{nn+1};
                nn = nn + 2;
            else
                nn = nn + 1;
                disp('WARNING: mx input value is empty or zero or negative... ');
                disp('         mx is defaulted to 0.001!');
                disp(' ');
            end %if4
        elseif (prod(size(varargin{nn}) == [1 3]) & (varargin{nn} == 'EIx')) %look for
      stiffness distribution input
            if (~isempty(varargin{nn+1}))
```

```
                EIR = varargin{nn+1};
                nn = nn + 2;
            else
                nn = nn + 1;
                disp('WARNING: EIx input value is empty or zero or negative... ');
                disp('         EIx is defaulted to 1e7!');
                disp(' ');
            end %if4
        elseif (prod(size(varargin{nn}) == [1 4]) & (varargin{nn} == 'eofs')) %look for
hinge offset input
            if ( ((~isempty(varargin{nn+1})) & ((isnumeric(varargin{nn+1})) |
(ischar(varargin{nn+1})))) )
                eoffset = varargin{nn+1};
                nn = nn + 2;
            else
                nn = nn + 1;
                disp('WARNING: Hinge offset (eofs) input value is not an appropriate
scalar value... ');
                disp('         Hinge offset (eofs) is defaulted to 0 (zero)!');
                disp(' ');
            end %if4
        elseif (prod(size(varargin{nn}) == [1 5]) & (varargin{nn} == 'omega')) %look for
rotor speed range input
            if ( ((~isempty(varargin{nn+1})) & (isnumeric(varargin{nn+1}))) & ...
                    (prod(varargin{nn+1} >= 0)) )
                rtrSpdRng = varargin{nn+1};
                rtrSpdRng = rtrSpdRng(:);
                nn = nn + 2;
            else
                nn = nn + 1;
                disp('WARNING: Rotor Speed (omega) input value is not an appropriate
numeric value... ');
                disp('         Rotor Speed (omega) is defaulted to [0:5:700] RPM!');
                disp(' ');
            end %if4
        else
            nn = nn + 1;
            disp('ERROR: An input argument could not be attributed to any variable and
was disregarded.');
            disp('       The other inputs may be attributed. But do not count on the
results you get!');
            disp('       ALTHOUGH THE PROGRAM MAY RUN, PLEASE CHECK YOUR INPUT ARGUMENTS
AND TRY AGAIN!');
            disp(' ');
        end %if3
    end %while2
end %if1
xr = [0:(blength/nInt):blength];

%Rearranging varargin
      inputs:==================================================================
disp('YNTERYGH() INPUT VALUES: ');
disp('======================= ');
disp(['Blade Length = ' num2str(blength)]);
if (size(mR) == [1 1])
    disp(['Mass Distribution = constant = ' num2str(mR)]);
    mR = mR.*ones(size(xr));
elseif (size(mR,1) == 1) %assume as polynomial coefficients
    disp(['Mass Distribution Polynomial = ' char(poly2sym(mR)) ]);
    mR = polyval(mR,xr);
elseif (size(mR,2) == 1) %assume as actual mass values
    disp(['Mass Distribution Actual Values = [root: ' num2str(mR(1)) ' ... ' ...
            num2str(mR(length(mR))) ' :tip]']);
    mR = linkspor(mR,blength,nInt);
    mR = mR';
elseif ischar(mR)
    disp(['Mass Distribution Expression = ' mR]);
    tmR = mR';
    tmR = mR(:)';
    mRi = inline(tmR);
    mR = mRi(xr);
else
    disp('ERROR: mx value could not be recognized. PLEASE TRY AGAIN!');
    disp(' ');
end %if1
if (size(EIR) == [1 1])
    disp(['Stiffness Distribution = constant = ' num2str(EIR) ]);
    EIR = EIR.*ones(size(xr));
elseif (size(EIR,1) == 1) %assume as polynomial coefficients
```

```
    disp(['Stiffness Distribution Polynomial = ' char(poly2sym(EIR)) ]);
    EIR = polyval(EIR,xr);
elseif (size(EIR,2) == 1) %assume as actual stiffness values
    disp(['Stiffness Distribution Actual Values = [root: ' num2str(EIR(1))  ' ... ' ...
            num2str(EIR(length(EIR))) ' :tip]']);
    EIR = linkspor(EIR,blength,nInt);
    EIR = EIR';
elseif ischar(EIR)
    disp(['Stiffness Distribution Expression = ' EIR]);
    tEIR = EIR';
    tEIR = tEIR(:)';
    EIRi = inline(EIR);
    EIR = EIRi(xr);
else
    disp('ERROR: EIx value could not be recognized. PLEASE TRY AGAIN!');
    disp(' ');
end %if1
if ischar(eoffset) %must use the % mark anywhere in the '...', e.g. '10%' or '%10' or
      '1%0'
    tefs = [];
    for jj = 1:length(eoffset),
        if (eoffset(jj) ~= '%')
            tefs = [tefs eoffset(jj)];
        end %if3
    end %for2
    if (str2num(tefs) >= 100)
        eoffset = 0;
        disp('WARNING: Hinge offset (eofs) percentage value is more than or equal to
      100%...');
        disp('         It is assumed to be zero!');
        disp(' ');
    end
    eoffset = str2num(tefs) * blength / 100;
elseif (eoffset == 0)
elseif ((eoffset < 1) & (eoffset < blength)) %e/L
    eoffset = eoffset * blength; %e
    disp('WARNING: Hinge offset (eofs) value is less than 1 (one) and less than blade
      length...');
    disp('         It is recomputed to be eofs*L!');
    disp(' ');
elseif (eoffset >= blength)
    disp('ERROR: Hinge offset (eofs) value can not be larger than blade length. PLEASE
      TRY AGAIN!');
    disp(' ');
else
    disp('ERROR: Hinge offset (eofs) value could not be recognized. PLEASE TRY AGAIN!');
    disp(' ');
end %if1
disp(['Root offset (e) = ' num2str(eoffset)]);
disp(['Rotor speed range = [ ' num2str(rtrSpdRng(1)) ' ... '
      num2str(rtrSpdRng(length(rtrSpdRng))) ' ]']);
disp(['Number of intervals = ' num2str(nInt)]);
disp(['x-range = [ 0 : ' num2str(blength/nInt) ' : ' num2str(blength) ' ]']);
disp(' ');

%Unit conversions:
EIR = EIR ./ (32.174 * 12); %converting psi to lbm/in-sec^2: [psi/(32.174*12)=lbm/in-
      sec^2].
rtrSpdRng = rtrSpdRng .* (pi/30); %converting RPM to rad/sec.

%READING y_n:=====================================================================
if (isempty(ynR) | (ynR == 0))
    wnR = 0;
elseif ( (~isempty(ynR)) & (isnumeric(ynR)) )
    if (size(ynR,1) == 1) %only one row => assume its elements are polynomial
      coefficients...
        disp('ATTENTION: There is only one row for the yn-input argument.');
        disp('           Values have been assumed to be elements of polynomial
      coefficients!');
        switch size(ynR,2)
        case 1 %one column
            disp('ATTENTION: The number of columns is one.');
            disp('           Value has been assumed to be scalar and its derivatives
      zero!');
            disp(' ');
            yn = ynR.*xr; %deflection.
            ydn = 0.*xr; %first derivative.
            yddn = 0.*xr; %second derivative.
        otherwise %more than one column
```

```
        disp('ATTENTION: The number of columns is more than one.');
        disp('           Derivatives will be computed from deflection values:');
        ynpoly = ynR;
        yn = polyval(ynpoly,xr);
        %first derivative:
        ynsym = poly2sym(ynpoly);
        ydnsym = diff(ynsym);
        ydnpoly = sym2poly(ydnsym);
        ydn = polyval(ydnpoly,xr);
        disp('              yn'' is computed from yn.');
        %second derivative:
        yddnsym = diff(ydnsym);
        yddnpoly = sym2poly(yddnsym);
        yddn = polyval(yddnpoly,xr);
        disp('              yn'''' is computed from yn''.');
        disp(' ');
    end %switch3
elseif (size(ynR,1) == 2) %only two rows => assume its elements are polynomial
  coefficients...
    disp('ATTENTION: There is only two rows for the yn-input argument...');
    disp('           The values have been assumed to be elements of polynomial
  coefficients!');
    switch size(ynR,2)
    case 1 %one column
        disp('ATTENTION: The number of columns is one...');
        disp('           Deflection and first derivative have been assumed to be
  scalars!');
        disp('           Second derivative is assumed to be zero!');
        disp(' ');
        ynpoly = ynR(1,:); %deflection.
        yn = polyval(ynpoly,xr);
        ydnpoly = ynR(2,:); %first derivative.
        ydn = polyval(ydnpoly,xr);
        yddn = 0.*xr; %second derivative.
    otherwise %more than one column
        disp('ATTENTION: The number of columns is more than one...');
        disp('           Deflection and first derivative have been assumed to be
  vectors!');
        disp('           Second derivative will be computed from first derivative:');
        ynpoly = ynR(1,:); %deflection.
        yn = polyval(ynpoly,xr);
        ydnpoly = ynR(2,:); %first derivative.
        ydn = polyval(ydnpoly,xr);
        %second derivative:
        ydnsym = poly2sym(ydnpoly);
        yddnsym = diff(ydnsym);
        yddnpoly = sym2poly(yddnsym);
        yddn = polyval(yddnpoly,xr);
        disp('              yn'''' is computed from yn''.');
        disp(' ');
    end %switch3
elseif (size(ynR,1) == 3) %only three rows => assume its elements are polynomial
  coefficients...
    disp('ATTENTION: There is only three rows for the yn-input argument.');
    disp('           Values have been assumed to be elements of polynomial
  coefficients!');
    switch size(ynR,2)
    case 1 %one column
        disp('ATTENTION: The number of columns is one...');
        disp('           Deflection and derivatives have been assumed to be
  scalars!');
        disp(' ');
    otherwise
        disp('ATTENTION: The number of columns is more than one...');
        disp('           Deflection and derivatives have been assumed to be
  vectors!');
        disp(' ');
    end %switch3
    ynpoly = ynR(1,:); %deflection.
    yn = polyval(ynpoly,xr);
    ydnpoly = ynR(2,:); %first derivative.
    ydn = polyval(ydnpoly,xr);
    yddnpoly = ynR(3,:); %second derivative.
    yddn = polyval(yddnpoly,xr);
elseif (size(ynR,1) > 3) %more than 3 rows => assume its elements are deflections
  normalized wrt tip...
    disp('ATTENTION: There is more than three rows for the yn-input argument.');
    disp('           Values have been assumed to be deflections!');
    switch size(ynR,2)
```

```
        case 1 %only one column
            disp('ATTENTION: There is only one column...');
            disp('           Derivatives will be computed from deflection values:');
            [yn,ynord,ynxr] = linkspor(ynR(:,1).',blength,nInt); %deflections.
            ynpoly = polyfit(ynxr,yn,ynord);
            %first derivative:
            ynsym = poly2sym(ynpoly);
            ydnsym = diff(ynsym);
            ydnpoly = sym2poly(ydnsym);
            ydn = polyval(ydnpoly,xr);
            disp('              yn'' is computed from yn.')
            %second derivative:
            yddnsym = diff(ydnsym);
            yddnpoly = sym2poly(yddnsym);
            yddn = polyval(yddnpoly,xr);
            disp('              yn'''' is computed from yn''.')
            disp(' ');
        case 2 %only two columns
            disp('ATTENTION: There is only two columns...');
            disp('           Second derivative will be computed from first derivative:');
            [yn,ynord,ynxr]= linkspor(ynR(:,1)',blength,nInt); %deflections.
            [ydn,ydnord,ydnxr]= linkspor(ynR(:,2)',blength,nInt); %first derivative.
            %second derivative:
            ynpoly = polyfit(ynxr,yn,ynord);
            ydnpoly = polyfit(ydnxr,ydn,ydnord);
            ydnsym = poly2sym(ydnpoly);
            yddnsym = diff(ydnsym);
            yddnpoly = sym2poly(yddnsym);
            yddn = polyval(yddnpoly,xr);
            disp('              yn'''' is computed from yn''.');
            disp(' ');
        otherwise %more than two columns
            disp('ATTENTION: There are more than three columns for the yn-input
      argument.');
            disp('           When there are more than three columns,');
            disp('           only first three columns will be used as inputs!');
            disp(' ');
            yn = linkspor(ynR(:,1)',blength,nInt); %deflections.
            ydn = linkspor(ynR(:,2)',blength,nInt); %first derivative.
            yddn = linkspor(ynR(:,3)',blength,nInt); %second derivative.
        end %switch3
    else
        yn = 0.*xr;
        ydn = 0.*xr;
        yddn = 0.*xr;
        disp('WARNING: The input argument for [yn;yn'';yn''''] must have:');
        disp('         -- 1~3 rows AND any number of columns, or');
        disp('         -- More than 3 rows AND 1~3 columns (there may be more columns but
      they are disregarded).');
        disp('(So,whatistheproblemhere?)');
        disp('As a precaution, deflection and derivatives have been assumed to be
      zero!');
        disp(' ');
    end %if2
elseif ( (~isempty(ynR)) & ((ischar(ynR)) | iscellstr(ynR)) )
    disp('ATTENTION: The yn-input argument is of character or cell array of strings
      type...');
    disp(' ');
    if iscellstr(ynR)
        for tt = size(ynR,1),
            ynRstr = [];
            for ss = 1:size(ynR,2),
                ynRstr = [ynRstr char(ynR(tt,ss))];
            end %for3
            ynR{tt} = ynRstr;
        end %for3
        ynR = ynR(:,1); %cell array of one or more rows and one column
    end %if2
    if ischar(ynR)
        if size(ynR,1) > 1
            ynRchar = [];
            for aa = 1:size(ynR,1),
                ynRchar = [ynRchar ynR(aa,:)];
            end %for4
            ynR = ynRchar;
        end %if3
        ynR = {ynR}; %cell array of one row and one column
    end
    if length(ynR) == 1 %only one element
```

```
        disp('ATTENTION: There is only one expression in the yn-input argument...');
        disp('           It is assumed to be an expression for the deflection!');
        disp('           Derivatives will be derived from this expression:');
        ynchar = char(ynR);
        ydnchar = char(diff(ynchar));
        yddnchar = char(diff(ydnchar));
        yni = inline(ynchar);
        ydni = inline(ydnchar);
        yddni = inline(yddnchar);
        yn = yni(xr);
        ydn = ydni(xr);
        yddn = yddni(xr);
        disp('              1st and 2nd derivatives have been derived from deflection
      expression!');
        disp(' ');
    elseif length(ynR) == 2 %only two elements
        disp('ATTENTION: There is only two expressions in the yn-input argument...');
        disp('           They are assumed to be expressions for deflection and first
      derivative!');
        disp('           Second derivative will be derived from the first derivative
      expression:');
        ynchar = char(ynR{1});
        ydnchar = char(ynR{2});
        yddnchar = char(diff(ydnchar));
        yni = inline(ynchar);
        ydni = inline(ydnchar);
        yddni = inline(yddnchar);
        yn = yni(xr);
        ydn = ydni(xr);
        yddn = yddni(xr);
        disp('              2nd derivative has been derived from the first derivative!');
        disp(' ');
    elseif length(ynR) >= 3 %three or more elements
        disp('ATTENTION: There is more than two expressions in the yn-input
      argument...');
        disp('           They are assumed to be expressions for deflection and
      derivatives!');
        disp('           If there are more than three expressions, ');
        disp('           the fourth and subsequent expressions will be disregarded!');
        disp(' ');
        ynchar = char(ynR{1});
        ydnchar = char(ynR{2});
        yddnchar = char(ynR{3});
        yni = inline(ynchar);
        ydni = inline(ydnchar);
        yddni = inline(yddnchar);
        yn = yni(xr);
        ydn = ydni(xr);
        yddn = yddni(xr);
    else
        disp(' ');
        disp('ERROR: Thereisanerrorhereatthecell/charinputsbutwhat?');
        disp(' ');
        disp(' ');
    end %if2
else
    disp('WARNING: There is something wrong with the yn-input argument.');
    disp('         It is assumed to be zero!');
    disp(' ');
    yn = 0.*xr;
    ydn = 0.*xr;
    yddn = 0.*xr;
end %if1

%================================================================================
%COMPUTATIONS:
disp(' ');
disp('YNTERYGH COMPUTATION WARNINGS: ');
disp('============================== ');

if ((wnR ~= []) & (wnR == 0))
    an = 0;
    K0n = 0;
    K1n = 0;
    Kn = 0;
    disp('WARNING: wn, an, Kn, K0n, K1n values are all zero or empty!');
    disp(' ');
else
    deltax = blength/nInt;
```

```
    xrmR = xr .* mR;
    num1 = deltax .* sum(EIR .* yddn.^2);
    denum = deltax .* sum(mR .* yn.^2);
    an = ( (num1 ./ denum) .* mR(1) .* blength.^4 ./ (EIR(1)+eps) ).^0.5;
    varA = [];
    for ii = 1:length(xrmR),
        varA(ii) = deltax .* sum(xrmR(ii:length(xrmR)));
    end %for2ii
    varB = [];
    for jj = 1:length(mR),
        varB(jj) = deltax .* sum(mR(jj:length(mR)));
    end %for2jj
    num20 = deltax .* sum(varA .* ydn.^2);
    num21 = deltax .* sum(varB .* ydn.^2);
    K0n = num20 ./ denum;
    K1n = num21 ./ denum;
    Kn = K0n + (eoffset .* K1n);
    wnR = (num1 ./ denum) + (Kn .* rtrSpdRng.^2);
end %if1wnR

wn = wnR .* (30/pi); %converting from rad/sec to CPM
nout = nargout-1;
if (nout == 1)
    varargout = {an};
elseif (nout == 2)
    varargout = {an Kn};
elseif (nout == 3)
    varargout = {an K0n K1n};
elseif (nout == 4)
    varargout = {an Kn K0n K1n};
end

%end of file - YNTERYGH()...

%================================================================================
%Subfunction:
function [goster] = plotme()

teta = [0.001:0.1:13*pi];
xxx = teta.^0.5 ./ csc(teta);
yyy = teta.^0.5 ./ sec(teta);
teta2 = [[2*pi+0.001-0.1:-0.1:0.001] [teta + 2*pi]];
xxx2 = teta2.^0.5 ./ csc(teta2);
yyy2 = teta2.^0.5 ./ sec(teta2);
xxx2 = (2.*[zeros(size([2*pi+0.001-0.1:-0.1:0.001])) xxx] + xxx2) ./ 3;
yyy2 = (2.*[zeros(size([2*pi+0.001-0.1:-0.1:0.001])) yyy] + yyy2) ./ 3;
xxx3 = ([zeros(size([2*pi+0.001-0.1:-0.1:0.001])) xxx] + 2.*xxx2) ./ 3;
yyy3 = ([zeros(size([2*pi+0.001-0.1:-0.1:0.001])) yyy] + 2.*yyy2) ./ 3;
figure; whitebg(gcf,'black');
axis([-7 7 -7 7]);
title('PLEASE INPUT AT LEAST ONE INPUT ARGUMENT TO YNTERYGH() FUNCTION!');
hold on; pause(0.1);
renk = 'krkbkgkmkcky'; sekil = 'hoovvddss**p';
for ff = 1:length(renk),
    tem = [renk(ff) sekil(ff)];
    plot(xxx,yyy,tem); hold on;
    pause(0.001);
    whitebg(gcf,renk(13-ff));
end %for1
plot(xxx,yyy,tem); hold on;
plot(xxx3,yyy3,'rp'); hold on;
plot(xxx2,yyy2,'bp'); hold on;
polar(teta,((xxx+0.7).^2+(yyy-1.1).^2)./160,'w-');
goster = 'PLEASE INPUT AT LEAST ONE INPUT ARGUMENT TO YNTERYGH() FUNCTION!';
return
```

## C. MATLAB® CODE FOR *YNTEMOSHNR* FUNCTION

```
function [y1NR,y2NR,y3NR,varargout]=yntemoshnr(plotFlag,cantHing,mtm0,EItEI0,varargin)

%[y1NR,y2NR,y3NR <,xout>]=YNTEMOSHNR(flagPlot,cantHing,mtm0,EItEI0 <,xvar>)
% This file gives the mode shapes of nonrotating hinged/cantilever beams by assuming the
mode shapes of
```

```
% some characteristic nonrotating hinged or cantilever beams with linear mass and
stiffness distributions.
% It uses the tables III&IV given in Yntema report to fit a 1st-degree curve between zero
& 0.5 or 0.5 & 1
% for each given m_tip/m_0 and EI_tip/EI_0.
%Inputs:
%       plotFlag is 0 (zero) if no plot needed or 1 (one) if a plot is needed. Input 2
(two) or more if you want
%           each mode shape on a different plot.
%       cantHing is 'H' if the beam is hinged or 'C' if cantilever.
%       mtm0 is m_tip/m_root. It can be a scalar for linear mass distribution or it can
be a vector which
%           will be approximated by a line using the function MAKELINE() for
cutoffr=cutofft=0.
%       EItEI0 is EI_tip/EI_root. It can be a scalar for linear stiffness distribution or
it can be a vector which
%           will be approximated by a line using the function MAKELINE() for
cutoffr=cutofft=0.
%          (See HELP MAKELINE() function beforehand in order to get a better
approximation for these values.)
%       xvar is the range of x-values corresponding to the mass and stiffness
distributions. Default is [0::1].
%Output:
%       y1NR, y2NR, y3NR are the mode shapes approximated from characteristic nonrotating
beams.
%       xout is the same as xvar.
%
%<<<<<<<<<<<<<<1st.Lt.H.E.AKIN, Master of Science in Aeronautical
Engineering,>>>>>>>>>>>>>>>>>>>>>>>>>
%<<<<<<<<<<<<<<<<<<<<Naval Postgraduate School, CA, USA, September
2002.>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%==========================================================================
load yntemoshcoef.mat;
%Reading inputs:===================================================================
if (nargin < 4)
    plotme;
    error('PLEASE INPUT AT LEAST FOUR INPUT ARGUMENTS TO YNTEMOSHNR() FUNCTION!');
end
mtm = mtm0;
EItEI = EItEI0;
if ~prod(size(mtm) == [1 1])
    mtm = mtm(:);
end
if ~prod(size(EItEI) == [1 1])
    EItEI = EItEI(:);
end
if ((nargin > 4) & (length(varargin{1}) == length(mtm)))
    xvar = varargin{1};
elseif (length(mtm) > 1)
    xvar = [0:(1/(length(mtm)-1)):1].';
else
    xvar = [0:0.01:1].';
end
if ~prod(size(mtm) == [1 1])
    mtmcoef = makeline(mtm,xvar);
    mtm = polyval(mtmcoef,xvar);
    mtmr = mtm(length(mtm)) / mtm(1);
else
    mtmr = mtm;
end
if ~prod(size(EItEI) == [1 1])
    EItEIcoef = makeline(EItEI,xvar);
    EItEI = polyval(EItEIcoef,xvar);
    EItEIr = EItEI(length(EItEI)) / EItEI(1);
else
    EItEIr = EItEI;
end

%Computations:====================================================================
y1 = [];
y2 = [];
y3 = [];
xx = [1 0.5 0];
xx1 = [1 0.5];
xx2 = [0.5 0];
switch cantHing(1)
case {'H','h'}
%1ST MODE: (fit a curve for m_tip/m_root and then a curve for EI_tip/IE_root)
    %EIt/EI0 = 1
```

```
    y1m1e1 = polyval(HmEIy1,xvar);
    y1m2e1 = polyval(Hmx2EIy1,xvar);
    y1m3e1 = polyval(HmxEIy1,xvar);
    y1me1 = [];
    for ii=1:length(xvar),
        yy11 = [y1m1e1(ii) y1m2e1(ii) y1m3e1(ii)];
        y1me1 = [y1me1 (polyval(polyfit(xx,yy11,2),mtmr))];
    end %for2
    %EIt/EI0 = 0.5
    y1m1e2 = polyval(HmEIx2y1,xvar);
    y1m2e2 = polyval(Hmx2EIx2y1,xvar);
    y1m3e2 = polyval(HmxEIx2y1,xvar);
    y1me2 = [];
    for ii=1:length(xvar),
        yy12 = [y1m1e2(ii) y1m2e2(ii) y1m3e2(ii)];
        y1me2 = [y1me2 (polyval(polyfit(xx,yy12,2),mtmr))];
    end %for2
    %EIt/EI0 = 0
    y1m1e3 = polyval(HmEIxy1,xvar);
    y1m2e3 = polyval(Hmx2EIxy1,xvar);
    y1m3e3 = polyval(HmxEIxy1,xvar);
    y1me3 = [];
    for ii=1:length(xvar),
        yy13 = [y1m1e3(ii) y1m2e3(ii) y1m3e3(ii)];
        y1me3 = [y1me3 (polyval(polyfit(xx,yy13,2),mtmr))];
    end %for2
    %mt/m0
    y1me = [];
    for ii=1:length(xvar),
        yy11 = [y1me1(ii) y1me2(ii)];
        yy12 = [y1me2(ii) y1me3(ii)];
        if ( (EItEIr <= 1) & (EItEIr > 0.5) )
            yy1 = polyval(polyfit(xx1,yy11,1),EItEIr);
        elseif ( (EItEIr <= 0.5) & (EItEIr >= 0) )
            yy1 = polyval(polyfit(xx2,yy12,1),EItEIr);
        end
        y1me = [y1me yy1];
    end %for2
%2ND MODE: (fit a curve for m_tip/m_root and then a curve for EI_tip/IE_root)
    %EIt/EI0 = 1
    y2m1e1 = polyval(HmEIy2,xvar);
    y2m2e1 = polyval(Hmx2EIy2,xvar);
    y2m3e1 = polyval(HmxEIy2,xvar);
    y2me1 = [];
    for ii=1:length(xvar),
        yy21 = [y2m1e1(ii) y2m2e1(ii) y2m3e1(ii)];
        y2me1 = [y2me1 (polyval(polyfit(xx,yy21,2),mtmr))];
    end %for2
    %EIt/EI0 = 0.5
    y2m1e2 = polyval(HmEIx2y2,xvar);
    y2m2e2 = polyval(Hmx2EIx2y2,xvar);
    y2m3e2 = polyval(HmxEIx2y2,xvar);
    y2me2 = [];
    for ii=1:length(xvar),
        yy22 = [y2m1e2(ii) y2m2e2(ii) y2m3e2(ii)];
        y2me2 = [y2me2 (polyval(polyfit(xx,yy22,2),mtmr))];
    end %for2
    %EIt/EI0 = 0
    y2m1e3 = polyval(HmEIxy2,xvar);
    y2m2e3 = polyval(Hmx2EIxy2,xvar);
    y2m3e3 = polyval(HmxEIxy2,xvar);
    y2me3 = [];
    for ii=1:length(xvar),
        yy23 = [y2m1e3(ii) y2m2e3(ii) y2m3e3(ii)];
        y2me3 = [y2me3 (polyval(polyfit(xx,yy23,2),mtmr))];
    end %for2
    %mt/m0
    y2me = [];
    for ii=1:length(xvar),
        yy21 = [y2me1(ii) y2me2(ii)];
        yy22 = [y2me2(ii) y2me3(ii)];
        if ( (EItEIr <= 1) & (EItEIr > 0.5) )
            yy2 = polyval(polyfit(xx1,yy21,1),EItEIr);
        elseif ( (EItEIr <= 0.5) & (EItEIr >= 0) )
            yy2 = polyval(polyfit(xx2,yy22,1),EItEIr);
        end
        y2me = [y2me yy2];
    end %for2
%3RD MODE: (fit a curve for m_tip/m_root and then a curve for EI_tip/IE_root)
```

```
    %EIt/EI0 = 1
    y3m1e1 = polyval(HmEIy3,xvar);
    y3m2e1 = polyval(Hmx2EIy3,xvar);
    y3m3e1 = polyval(HmxEIy3,xvar);
    y3me1 = [];
    for ii=1:length(xvar),
        yy31 = [y3m1e1(ii) y3m2e1(ii) y3m3e1(ii)];
        y3me1 = [y3me1 (polyval(polyfit(xx,yy31,2),mtmr))];
    end %for2
    %EIt/EI0 = 0.5
    y3m1e2 = polyval(HmEIx2y3,xvar);
    y3m2e2 = polyval(Hmx2EIx2y3,xvar);
    y3m3e2 = polyval(HmxEIx2y3,xvar);
    y3me2 = [];
    for ii=1:length(xvar),
        yy32 = [y3m1e2(ii) y3m2e2(ii) y3m3e2(ii)];
        y3me2 = [y3me2 (polyval(polyfit(xx,yy32,2),mtmr))];
    end %for2
    %EIt/EI0 = 0
    y3m1e3 = polyval(HmEIxy3,xvar);
    y3m2e3 = polyval(Hmx2EIxy3,xvar);
    y3m3e3 = polyval(HmxEIxy3,xvar);
    y3me3 = [];
    for ii=1:length(xvar),
        yy33 = [y3m1e3(ii) y3m2e3(ii) y3m3e3(ii)];
        y3me3 = [y3me3 (polyval(polyfit(xx,yy33,2),mtmr))];
    end %for2
    %mt/m0
    y3me = [];
    for ii=1:length(xvar),
        yy31 = [y3me1(ii) y3me2(ii)];
        yy32 = [y3me2(ii) y3me3(ii)];
        if ( (EItEIr <= 1) & (EItEIr > 0.5) )
            yy3 = polyval(polyfit(xx1,yy31,1),EItEIr);
        elseif ( (EItEIr <= 0.5) & (EItEIr >= 0) )
            yy3 = polyval(polyfit(xx2,yy32,1),EItEIr);
        end
        y3me = [y3me yy3];
    end %for2

case {'C','c'}
%1ST MODE: (fit a curve for m_tip/m_root and then a curve for EI_tip/IE_root)
    %EIt/EI0 = 1
    y1m1e1 = polyval(CmEIy1,xvar);
    y1m2e1 = polyval(Cmx2EIy1,xvar);
    y1m3e1 = polyval(CmxEIy1,xvar);
    y1me1 = [];
    for ii=1:length(xvar),
        yy11 = [y1m1e1(ii) y1m2e1(ii) y1m3e1(ii)];
        y1me1 = [y1me1 (polyval(polyfit(xx,yy11,2),mtmr))];
    end %for2
    %EIt/EI0 = 0.5
    y1m1e2 = polyval(CmEIx2y1,xvar);
    y1m2e2 = polyval(Cmx2EIx2y1,xvar);
    y1m3e2 = polyval(CmxEIx2y1,xvar);
    y1me2 = [];
    for ii=1:length(xvar),
        yy12 = [y1m1e2(ii) y1m2e2(ii) y1m3e2(ii)];
        y1me2 = [y1me2 (polyval(polyfit(xx,yy12,2),mtmr))];
    end %for2
    %EIt/EI0 = 0
    y1m1e3 = polyval(CmEIxy1,xvar);
    y1m2e3 = polyval(Cmx2EIxy1,xvar);
    y1m3e3 = polyval(CmxEIxy1,xvar);
    y1me3 = [];
    for ii=1:length(xvar),
        yy13 = [y1m1e3(ii) y1m2e3(ii) y1m3e3(ii)];
        y1me3 = [y1me3 (polyval(polyfit(xx,yy13,2),mtmr))];
    end %for2
    %mt/m0
    y1me = [];
    for ii=1:length(xvar),
        yy11 = [y1me1(ii) y1me2(ii)];
        yy12 = [y1me2(ii) y1me3(ii)];
        if ( (EItEIr <= 1) & (EItEIr > 0.5) )
            yy1 = polyval(polyfit(xx1,yy11,1),EItEIr);
        elseif ( (EItEIr <= 0.5) & (EItEIr >= 0) )
            yy1 = polyval(polyfit(xx2,yy12,1),EItEIr);
        end
```

```
        y1me = [y1me yy1];
    end %for2
%2ND MODE: (fit a curve for m_tip/m_root and then a curve for EI_tip/IE_root)
    %EIt/EI0 = 1
    y2m1e1 = polyval(CmEIy2,xvar);
    y2m2e1 = polyval(Cmx2EIy2,xvar);
    y2m3e1 = polyval(CmxEIy2,xvar);
    y2me1 = [];
    for ii=1:length(xvar),
        yy21 = [y2m1e1(ii) y2m2e1(ii) y2m3e1(ii)];
        y2me1 = [y2me1 (polyval(polyfit(xx,yy21,2),mtmr))];
    end %for2
    %EIt/EI0 = 0.5
    y2m1e2 = polyval(CmEIx2y2,xvar);
    y2m2e2 = polyval(Cmx2EIx2y2,xvar);
    y2m3e2 = polyval(CmxEIx2y2,xvar);
    y2me2 = [];
    for ii=1:length(xvar),
        yy22 = [y2m1e2(ii) y2m2e2(ii) y2m3e2(ii)];
        y2me2 = [y2me2 (polyval(polyfit(xx,yy22,2),mtmr))];
    end %for2
    %EIt/EI0 = 0
    y2m1e3 = polyval(CmEIxy2,xvar);
    y2m2e3 = polyval(Cmx2EIxy2,xvar);
    y2m3e3 = polyval(CmxEIxy2,xvar);
    y2me3 = [];
    for ii=1:length(xvar),
        yy23 = [y2m1e3(ii) y2m2e3(ii) y2m3e3(ii)];
        y2me3 = [y2me3 (polyval(polyfit(xx,yy23,2),mtmr))];
    end %for2
    %mt/m0
    y2me = [];
    for ii=1:length(xvar),
        yy21 = [y2me1(ii) y2me2(ii)];
        yy22 = [y2me2(ii) y2me3(ii)];
        if ( (EItEIr <= 1) & (EItEIr > 0.5) )
            yy2 = polyval(polyfit(xx1,yy21,1),EItEIr);
        elseif ( (EItEIr <= 0.5) & (EItEIr >= 0) )
            yy2 = polyval(polyfit(xx2,yy22,1),EItEIr);
        end
        y2me = [y2me yy2];
    end %for2
%3RD MODE: (fit a curve for m_tip/m_root and then a curve for EI_tip/IE_root)
    %EIt/EI0 = 1
    y3m1e1 = polyval(CmEIy3,xvar);
    y3m2e1 = polyval(Cmx2EIy3,xvar);
    y3m3e1 = polyval(CmxEIy3,xvar);
    y3me1 = [];
    for ii=1:length(xvar),
        yy31 = [y3m1e1(ii) y3m2e1(ii) y3m3e1(ii)];
        y3me1 = [y3me1 (polyval(polyfit(xx,yy31,2),mtmr))];
    end %for2
    %EIt/EI0 = 0.5
    y3m1e2 = polyval(CmEIx2y3,xvar);
    y3m2e2 = polyval(Cmx2EIx2y3,xvar);
    y3m3e2 = polyval(CmxEIx2y3,xvar);
    y3me2 = [];
    for ii=1:length(xvar),
        yy32 = [y3m1e2(ii) y3m2e2(ii) y3m3e2(ii)];
        y3me2 = [y3me2 (polyval(polyfit(xx,yy32,2),mtmr))];
    end %for2
    %EIt/EI0 = 0
    y3m1e3 = polyval(CmEIxy3,xvar);
    y3m2e3 = polyval(Cmx2EIxy3,xvar);
    y3m3e3 = polyval(CmxEIxy3,xvar);
    y3me3 = [];
    for ii=1:length(xvar),
        yy33 = [y3m1e3(ii) y3m2e3(ii) y3m3e3(ii)];
        y3me3 = [y3me3 (polyval(polyfit(xx,yy33,2),mtmr))];
    end %for2
    %mt/m0
    y3me = [];
    for ii=1:length(xvar),
        yy31 = [y3me1(ii) y3me2(ii)];
        yy32 = [y3me2(ii) y3me3(ii)];
        if ( (EItEIr <= 1) & (EItEIr > 0.5) )
            yy3 = polyval(polyfit(xx1,yy31,1),EItEIr);
        elseif ( (EItEIr <= 0.5) & (EItEIr >= 0) )
            yy3 = polyval(polyfit(xx2,yy32,1),EItEIr);
```

```
        end
        y3me = [y3me yy3];
    end %for2

otherwise
    disp('ERROR: Please enter a proper cantHing input fot YNTEMOSHNR() function!');
    disp(' ');
end %switch1

%Outputs:==================================================================
if (nargout > 3)
    varargout = {xvar};
end
y1NR = y1me;
y2NR = y2me;
y3NR = y3me;

if (plotFlag > 0)
    figure; hold on;
    if (plotFlag ~= 1)
        subplot(3,1,3); hold on;
    end
    plot(xvar,y3NR,'m-'); grid on;
    if (plotFlag ~= 1)
        legend('3rd Bending Mode Shape',0);
    end
    if (plotFlag ~= 1)
        subplot(3,1,2); hold on;
    end
    plot(xvar,y2NR,'r-'); grid on;
    if (plotFlag ~= 1)
        legend('2nd Bending Mode Shape',0);
    end
    if (plotFlag ~= 1)
        subplot(3,1,1); hold on;
    end
    plot(xvar,y1NR,'b-'); grid on;
    if (plotFlag ~= 1)
        legend('1st Bending Mode Shape',0);
    end
    if (plotFlag == 1)
        legend('3rd Bending Mode Shape','2nd Bending Mode Shape','1st Bending Mode
Shape',0);
    end
    title('NONROTATING BEAM FIRST THREE BENDING MODE SHAPES');
    text(xvar(2),0.8*y1NR(length(y1NR)),{['Hinged/Cantilever: '
cantHing];['m_t_i_p/m_r_o_o_t = ' num2str(mtmr)]; ...
            ['EI_t_i_p/EI_r_o_o_t = ' num2str(EItEIr)];['x-range = [ ' num2str(xvar(1)) '
: ' ...
                num2str((1/(length(xvar)-1))) ' : ' num2str(xvar(length(xvar))) ' ]']});
    hold off;
end %if1

%===================================================================
%Subfunction:
function [] = plotme()
teta = [0.001:0.1:13*pi];
figure; whitebg(gcf,'red');
axis([-7 7 -7 7]);
polar(teta,teta.^0.5,'yp-'); grid off;
title({'ERROR!';'PLEASE INPUT AT LEAST FOUR INPUT ARGUMENTS TO YNTEMOSHNR() FUNCTION!'});

return
```

## D. MATLAB® CODE FOR *MAKELINE* UTILITY FUNCTION

```
function [polyCoef] = makeline(inVec,inX,varargin)

%[polyCoef] = MAKELINE(inVec,inX <,cutoffr<,cutofft>>)
%This function returns the polynomial coefficients of the line approximated for values
given in a vector.
%Inputs:
%       inVec is a vector containing the values which are to be approximated by a first-
order polynomial.
```

```
%       inX is the corresponding x-values for inVec from zero to blade tip.
%           cutoffr is the distance from the root which will be excluded from the approximation. Default is zero.
%           cutofft is the distance from the tip which will be excluded from the approximation. Default is zero.
%Output:
%        polyCoef is a 1-by-2 vector which contains the polynomial coefficients fo the approximate line.
%This function is intended to be used for Yntema method which is about bending frequencies of rotating beams.
%<<<<<<<<<<<<<<1st.Lt.H.E.AKIN, Master of Science in Aeronautical Engineering,>>>>>>>>>>>>>>>>>>>>>>>>>>>
%<<<<<<<<<<<<<<<<<<<<<<Naval Postgraduate School, CA, USA, September 2002.>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%-------------------------------------------------------------------------------
tempVec = inVec(:);
tempX = inX(:);
if (nargin-2) == 0
    cutoffr = 0;
    cutofft = 0;
elseif (nargin-2) == 1
    cutoffr = varargin{1};
    cutofft = 0;
elseif (nargin-2) == 2
    cutoffr = varargin{1};
    cutofft = varargin{2};
else
    cutoffr = 0;
    cutofft = 0;
end %if1
tt = 0;
while (tt < length(tempX)),
    tt = tt + 1;
    if (tempX(tt) <= cutoffr)
        tempX(tt) = [];
        tempVec(tt) = [];
        tt = tt - 1;
    elseif (tempX(tt) >= (inX(length(inX)) - cutofft))
        tempX(tt) = [];
        tempVec(tt) = [];
        tt = tt - 1;
    end %if2
end %for1

polyCoef = polyfit(tempX,tempVec,1);
```

## E. MATLAB® CODE FOR *LINKMASS* UTILITY FUNCTION

```
function [massout,varargout] = linkmass(massinR,varargin)

% [massout <,xnew>] = LINKMASS(massinR <,xinR <,nint>>)
%This function rearranges the mass values in a vector to their corresponding places in a longer vector.
%The middle values are assigned zero. The first and last values remain as the first and last values.
%For example, if you have a mass vector of length 11 and if you want to rearrange it into a vector of
%length 101, this function relocates the original mass values as the 1st, 11th, 21st,...,101st elements
%of new vector. Other elements are filled with zero.
%This function is a part of the computer code written for Yntema method about bending frequencies of rotating beams.
%Inputs:
%       massinR is a vector containing mass values at any station along the beam.
%        xinR is a vector of actual/normalized x-values corresponding to each mass value. You can opt not
%             to input this argument if they are linearly spaced. But you must input if they are nonlinear.
%            Default is [0:1/(length(massinR)-1):1]. You must input 0 (zero) for xinR if you want to input
%           a nint argument but x-values; then xinR is defaulted.
```

```
%        nint is the number of intervals of the output vector which contains rearranged
mass values. For
%           the example above, nint would be 100. Default is 100.
%       The lengths of massinR and xinR vectors must be equal.
%Outputs:
%        massout is a vector containing the rearranged mass values. If any two new x-
values are the same
%           distance from the original x-value, the first new x-value is used to relocate
the mass value.
%            For example, if the mass value is originally at x=2 and the massout vector
has no x-value at
%             2 but at 1.9 and 2.1, the mass is relocated to x=1.9, assuming 1.9 comes
before 2.1 in the
%           xnew vector (type HELP MIN for more details).
%       xnew is a vector of corresponding x-values newly arranged.
%
%<<<<<<<<<<<<<<1st.Lt.H.E.AKIN,      Master      of      Science      in      Aeronautical
Engineering,>>>>>>>>>>>>>>>>>>>>>>>>>>>
%<<<<<<<<<<<<<<<<<<<<<Naval      Postgraduate      School,      CA,      USA,      September
2002.>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

if ((nargin ~= 1) & ((nargin ~= 2) & (nargin ~= 3)))
    error('Please input at least one (no more than three) input argument for the function
LINKMASS()!');
end %if1
massin = massinR(:);
nintmassin = length(massin) - 1;
if ( ((nargin == 2) | (nargin == 3)) & (sum(sum(varargin{1} ~= 0))) )
    xinR = varargin{1};
    xin = xinR(:);
else
    xinR = [0:1/nintmassin:1];
    xin = xinR(:);
end %if1
if (length(massin) ~= length(xin))
    error('The length of massinR and xinR vectors must be equal!');
end %if1
if (nargin == 3)
    nintin = varargin{2};
else
    nintin = 100;
    disp('ATTENTION: Number of intervals is asumed to be 100.');
    disp(' ');
end %if1
if (length(massin) >= nintin)
    error('Please input a number-of-intervals which is more than the length of the input
vector');
end %if1

xnew = [xin(1):((xin(length(xin))-xin(1)) / nintin):xin(length(xin))].';
mtemp = zeros(size(xnew)); %initializing
xxin = 1;
while (xxin <= length(xin)),
    [tempD,xind] = min(abs(xin(xxin)-xnew)); %if two minimums, MIN returns first one's
indice!
    mtemp(xind) = massin(xxin);
    xxin = xxin + 1;
end %while1

if (size(massinR,1) == 1)
    mtemp = mtemp.';
    xnew = xnew.';
end %if1

massout = mtemp;
if (nargout == 2)
    varargout = {xnew};
end %if1


%end of file LINKMASS()
```

## F. MATLAB® CODE FOR *LINKSPOR* UTILITY FUNCTION

```
function [outVec,varargout] = linkspor(inVec,bLen,nIntr)

% [outVec <,outOrd <,outVecX>>] = LINKSPOR(inVec,bLen,nIntr) function:
%           Recalculates the values of the vector 'inVec' with the number of linear
intervals 'nIntr'.
%      Use this function when you have n-number of values but need more than n number of
values.
%       This function linearly fills the interval between any two consecutive values of
'inVec'
%      with values corresponding to 'nIntr'.
%          For example, if you have 11 values, which correspond to a linear interval
x=0:bLen/10:bLen,
%      but need 101 values (nIntr=100), then you will get a vector of length(outVec)=101
corresponding
%      to an interval of x=0:bLen/100:bLen. The middle values of 'outVec' will correspond
to a line
%      fitted between two consecutive 'inVec' values.
%         'inVec' values are assumed to be 'y' values corresponding to a linear interval
of x-values
%         between 0 and 1. First value of 'inVec' corresponds to x=0, its last value
corresponds
%      to x=bLen, and the middle values must be linearly spaced in the computations.
%        'bLen' is the length of the blade.
%                'nIntr' is the desired number of intervals for the range of
x=[0:(bLen/nIntr):bLen]. It must
%       not be less than length(inVec). It might be equal to length(inVec), but then the
function would
%       return the same values as of 'inVec'. Use a higher 'nIntr' if you need a better
precision.
%        Optional 'outVecX' is the range of x values corresponding to 'outVec' values.
%        Optional 'outOrd' is the order of the polynomial which approximates the 'outVec'
values as
%       precise as possible. It incorporates the number of turn-directions (CW-to-CCW or
CCW-to-CW)
%      changing along the curve.
%          If only two output arguments are specified, only 'outOrd' is output together
with 'outVec'.
%        If no output arguments are specified, only the values of 'outVec' are output.
%        The 'outVec' and 'outVecX' vectors are the same size as 'inVec'.
%<<<<<<<<<<<<<<1st.Lt.H.E.AKIN, Master of Science in Aeronautical
Engineering,>>>>>>>>>>>>>>>>>>>>>>>>>>
%<<<<<<<<<<<<<<<<<<<<Naval Postgraduate School, CA, USA, September
2002.>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

mlen = length(inVec);
yvar = [inVec(1)];
xvar = [0];
deltax = bLen/(mlen-1);

for kk = 2:mlen, %filling spaces linearly
    tx = [xvar(length(xvar))+(bLen/nIntr):(bLen/nIntr):(kk-1)*deltax];
    ty = polyfit([(kk-2) (kk-1)].*deltax,[inVec(kk-1) inVec(kk)],1);
    ytemp = polyval(ty,tx);
    if (size(inVec,1) == 1)
        xvar = [xvar tx];
        yvar = [yvar ytemp];
    elseif (size(inVec,2) == 1)
        xvar = [xvar tx];
        yvar = [yvar ytemp];
    else
        disp('ERROR: PLEASE INPUT A VECTOR FOR inVec NOT A MATRIX!...');
        disp('       Although an early termination is not performed,');
        disp('       you will most probably get a run-time error!');
        disp(' ');
        errordlg(['PLEASE INPUT A VECTOR FOR inVec, DO NOT INPUT A MATRIX! ' ...
            'Although an early termination is not performed, ' ...
            'you will most probably get a run-time error!']);
    end %if2
end %for1
if (size(inVec,2) == 1)
    xvar = xvar';
    yvar = yvar';
end

if length(yvar) == nIntr %if there is one less value than required
    if (size(inVec,1) == 1)
```

```
        yvar = [yvar inVec(length(inVec))];
        xvar = [xvar bLen];
    elseif (size(inVec,2) == 1)
        yvar = [yvar ; inVec(length(inVec))];
        xvar = [xvar ; bLen];
    end
end %if1

cnt = 2; %(counting turn-points); cnt=2 => at least a second order curve
for mm = 3:length(yvar),
    if ( (abs(yvar(mm-1)) > abs(yvar(mm-2))) & (abs(yvar(mm)) < abs(yvar(mm-1))) )
        cnt = cnt + 1;
    end
    if ( (abs(yvar(mm-1)) < abs(yvar(mm-2))) & (abs(yvar(mm)) > abs(yvar(mm-1))) )
        cnt = cnt + 1;
    end
end
outVec = yvar;
nout = max(nargout,1)-1;
tempVarOut = {cnt xvar};
for nn = 1:nout,
    varargout{nn} = tempVarOut{nn};
end

if (size(yvar) ~= size(inVec))
    disp('ERROR: In LINKSPOR() function, the size of outVec-vector is not the same as of
inVec-vector!');
end %if1
if ( (size(xvar) ~= size(inVec)) & (nout == 2) )
    disp('ERROR: In LINKSPOR() function, the size of optional outVecX-vector is not the
same as of inVec-vector!');
end %if1
```

# APPENDIX D. MATLAB® CODE FOR GRAPHICAL USER INTERFACE (GUI)

```
function varargout = yntemaGUI(varargin)
% YNTEMAGUI Application M-file for yntemaGUI.fig
%    FIG = YNTEMAGUI launch yntemaGUI GUI.
%    YNTEMAGUI('callback_name', ...) invoke the named callback.

% Last Modified by GUIDE v2.0 11-Sep-2002 23:02:46
global YNTEMAILKKEZMICALISIYOR

if nargin == 0  % LAUNCH GUI

    fig = openfig(mfilename,'reuse');

    % Generate a structure of handles to pass to callbacks, and store it.
    handles = guihandles(fig);
    guidata(fig, handles);

    %popupmenu initiations:
    flagSouth_popupmenu_Callback(handles.flagSouth_popupmenu,[],handles)
    ch_popupmenu_Callback(handles.ch_popupmenu,[],handles)
    melast_popupmenu_Callback(handles.melast_popupmenu,[],handles)
    optionalInputs_radiobutton_Callback(handles.optionalInputs_radiobutton,[],handles)
    %         optionalInputs_radiobutton_Callback(handles.optionalInputs_radiobutton,  [],
handles)

    %defaulting for first run:
    if YNTEMAILKKEZMICALISIYOR == []
        disp('Values defaulted!');
        handles.ch_popupmenu = 'C';
        handles.melast_popupmenu = 1e7;
        handles.flagSouth_popupmenu = 0;
        handles.eoffset_edit = 0;
        handles.blength_edit = 200;
        handles.maLen0_edit = 0.00033;
        handles.minert0_edit = 1.19;
        handles.maLenT_edit = 0.00033;
        handles.minertT_edit = 1.19;
        handles.massTip_edit = 0;
        handles.rtrSpdRng_edit = [0:5:700];
        handles.opRtrSpd_edit = 470;
        handles.alfa_edit = 0;
        handles.optionalInputs_radiobutton = 0;
        guidata(fig,handles);
    end

    if nargout > 0
        varargout{1} = fig;
    end

    YNTEMAILKKEZMICALISIYOR = [YNTEMAILKKEZMICALISIYOR 1];

elseif ischar(varargin{1}) % INVOKE NAMED SUBFUNCTION OR CALLBACK

    try
        if (nargout)
            [varargout{1:nargout}] = feval(varargin{:}); % FEVAL switchyard
        else
            feval(varargin{:}); % FEVAL switchyard
        end
    catch
        disp(lasterr);
    end

    global YNTEMAILKKEZMICALISIYOR
    YNTEMAILKKEZMICALISIYOR = [YNTEMAILKKEZMICALISIYOR 2];

end


%| ABOUT CALLBACKS:
%| GUIDE automatically appends subfunction prototypes to this file, and
%| sets objects' callback properties to call them through the FEVAL
```

```
%| switchyard above. This comment describes that mechanism.
%|
%| Each callback subfunction declaration has the following form:
%| <SUBFUNCTION_NAME>(H, EVENTDATA, HANDLES, VARARGIN)
%|
%| The subfunction name is composed using the object's Tag and the
%| callback type separated by '_', e.g. 'slider2_Callback',
%| 'figure1_CloseRequestFcn', 'axis1_ButtondownFcn'.
%|
%| H is the callback object's handle (obtained using GCBO).
%|
%| EVENTDATA is empty, but reserved for future use.
%|
%| HANDLES is a structure containing handles of components in GUI using
%| tags as fieldnames, e.g. handles.figure1, handles.slider2. This
%| structure is created at GUI startup using GUIHANDLES and stored in
%| the figure's application data using GUIDATA. A copy of the structure
%| is passed to each callback.  You can store additional information in
%| this structure at GUI startup, and you can change the structure
%| during callbacks.  Call guidata(h, handles) after changing your
%| copy to replace the stored original so that subsequent callbacks see
%| the updates. Type "help guihandles" and "help guidata" for more
%| information.
%|
%| VARARGIN contains any extra arguments you have passed to the
%| callback. Specify the extra arguments by editing the callback
%| property in the inspector. By default, GUIDE sets the property to:
%| <MFILENAME>('<SUBFUNCTION_NAME>', gcbo, [], guidata(gcbo))
%| Add any extra arguments after the last argument, before the final
%| closing parenthesis.



% --------------------------------------------------------------------
function varargout = ch_popupmenu_Callback(h, eventdata, handles, varargin)

val=get(h,'Value');
switch val
case 1
    cantHing = 'C';
    handles.ch_popupmenu = cantHing;
case 2
    cantHing = 'H';
    handles.ch_popupmenu = cantHing;
otherwise
    cantHing = 'C';
    handles.ch_popupmenu = cantHing;
end

guidata(h,handles)

% --------------------------------------------------------------------
function varargout = melast_popupmenu_Callback(h, eventdata, handles, varargin)

val1=get(h,'Value');
switch val1
case 1
    melast = 1e7;
    handles.melast_popupmenu = melast;
case 2
    melast = 2e7; %??????????????????????
    handles.melast_popupmenu = melast;
case 3
    melast = 1.5e7; %??????????????????????
    handles.melast_popupmenu = melast;
case 4
    melast = 3e7;
    handles.melast_popupmenu = melast;
otherwise
    melast = 1e7;
    handles.melast_popupmenu = melast;
end

guidata(h,handles)

% --------------------------------------------------------------------
function varargout = flagSouth_popupmenu_Callback(h, eventdata, handles, varargin)

val2=get(h,'Value');
```

```
switch val2
case 1
    flagSouth = 0;
    handles.flagSouth_popupmenu = flagSouth;
case 2
    flagSouth = 1;
    handles.flagSouth_popupmenu = flagSouth;
case 3
    flagSouth = 2;
    handles.flagSouth_popupmenu = flagSouth;
case 4
    flagSouth = 3;
    handles.flagSouth_popupmenu = flagSouth;
otherwise
    flagSouth = 0;
    handles.flagSouth_popupmenu = flagSouth;
end

guidata(h,handles)

% --------------------------------------------------------------------
function varargout = eoffset_edit_Callback(h, eventdata, handles, varargin)

global YNTEMAILKKEZMICALISIYOR

eoffsetget = get(h,'String');
if isempty(eoffsetget) | isempty(YNTEMAILKKEZMICALISIYOR)
    eoffset = 0;
else
    eoffset = str2num(eoffsetget);
end
handles.eoffset_edit = eoffset;

guidata(h,handles)

% --------------------------------------------------------------------
function varargout = blength_edit_Callback(h, eventdata, handles, varargin)

global YNTEMAILKKEZMICALISIYOR

blengthget = get(h,'String');
if isempty(blengthget) | isempty(YNTEMAILKKEZMICALISIYOR)
    blength = 200;
else
    blength = str2num(blengthget);
end
handles.blength_edit = blength;

guidata(h,handles)

% --------------------------------------------------------------------
function varargout = maLen0_edit_Callback(h, eventdata, handles, varargin)

global YNTEMAILKKEZMICALISIYOR

maLen0get = get(h,'String');
if isempty(maLen0get) | isempty(YNTEMAILKKEZMICALISIYOR)
    maLen0 = 0.00033;
else
    maLen0 = str2num(maLen0get);
end
handles.maLen0_edit = maLen0;

guidata(h,handles)

% --------------------------------------------------------------------
function varargout = minert0_edit_Callback(h, eventdata, handles, varargin)

global YNTEMAILKKEZMICALISIYOR

minert0get = get(h,'String');
if isempty(minert0get) | isempty(YNTEMAILKKEZMICALISIYOR)
    minert0 = 1.19;
else
    minert0 = str2num(minert0get);
end
handles.minert0_edit = minert0;

guidata(h,handles)
```

```
% ------------------------------------------------------------------
function varargout = maLenT_edit_Callback(h, eventdata, handles, varargin)

global YNTEMAILKKEZMICALISIYOR

maLenTget = get(h,'String');
if isempty(maLenTget) | isempty(YNTEMAILKKEZMICALISIYOR)
    maLenT = 0;
else
    maLenT = str2num(maLenTget);
end
handles.maLenT_edit = maLenT;

guidata(h,handles)

% ------------------------------------------------------------------
function varargout = minertT_edit_Callback(h, eventdata, handles, varargin)

global YNTEMAILKKEZMICALISIYOR

minertTget = get(h,'String');
if isempty(minertTget) | isempty(YNTEMAILKKEZMICALISIYOR)
    minertT = 0;
else
    minertT = str2num(minertTget);
end
handles.minertT_edit = minertT;

guidata(h,handles)

% ------------------------------------------------------------------
function varargout = massTip_edit_Callback(h, eventdata, handles, varargin)

global YNTEMAILKKEZMICALISIYOR

massTipget = get(h,'String');
if isempty(massTipget) | isempty(YNTEMAILKKEZMICALISIYOR)
    massTip = 0;
else
    massTip = str2num(massTipget);
end
handles.massTip_edit = massTip;

guidata(h,handles)

% ------------------------------------------------------------------
function varargout = rtrSpdRng_edit_Callback(h, eventdata, handles, varargin)

global YNTEMAILKKEZMICALISIYOR

rtrSpdRngget = get(h,'String');
if isempty(rtrSpdRngget) | isempty(YNTEMAILKKEZMICALISIYOR)
    rtrSpdRng = 0;
else
    rtrSpdRng = str2num(rtrSpdRngget);
end
handles.rtrSpdRng_edit = rtrSpdRng;

guidata(h,handles)

% ------------------------------------------------------------------
function varargout = opRtrSpd_edit_Callback(h, eventdata, handles, varargin)

global YNTEMAILKKEZMICALISIYOR

opRtrSpdget = get(h,'String');
if isempty(opRtrSpdget) | isempty(YNTEMAILKKEZMICALISIYOR)
    opRtrSpd = 0;
else
    opRtrSpd = str2num(opRtrSpdget);
end
handles.opRtrSpd_edit = opRtrSpd;

guidata(h,handles)

% ------------------------------------------------------------------
function varargout = alfa_edit_Callback(h, eventdata, handles, varargin)
```

```
global YNTEMAILKKEZMICALISIYOR

alfaget = get(h,'String');
if isempty(alfaget)  | isempty(YNTEMAILKKEZMICALISIYOR)
    alfa = 0;
else
    alfa = str2num(alfaget);
end
handles.alfa_edit = alfa;

guidata(h,handles)

% --------------------------------------------------------------------
function varargout = optionalInputs_radiobutton_Callback(h, eventdata, handles, varargin)

global YNTEMAILKKEZMICALISIYOR

val3 = get(h,'Value');
if ((val3 ~= 0) & (val3 ~= 1)) | isempty(YNTEMAILKKEZMICALISIYOR)
    val3 = 0;
end
handles.optionalInputs_radiobutton = val3;

guidata(h,handles)

% --------------------------------------------------------------------
function varargout = ok_pushbutton_Callback(h, eventdata, handles, varargin)

global YNTEMAILKKEZMICALISIYOR
disp('==================== START OF YNTEMAGUI() FILE ==========================');

%initializing diary file:
dirynte = dir(pwd);
flagdirec = 0;
for ddd = 1:size(dirynte,1)
    chkvar111 = struct2cell(dirynte(ddd));
    if  ((size(chkvar111{1},2)  ==  17)  &  ((chkvar111{1}  ==  'yntemaguirundiary')  &
(isdir('yntemaguirundiary'))))
        flagdirec = 1;
    end
end
if flagdirec == 0
    eval('mkdir yntemaguirundiary',...
        'disp(''The diary folder for this run of YNTEMAGUI could not be created!'')');
end
simdi = datestr(now);
simdi = [simdi(1:14) simdi(16:17) simdi(19:20)];
eval('diary([pwd ''\yntemaguirundiary\'' simdi ''.txt''])',...
    'disp(''Diary will not be saved for this run of YNTEMAGUI()!'')');

%
flagSouthc = handles.flagSouth_popupmenu;
cantHingc = handles.ch_popupmenu;
melastc = handles.melast_popupmenu;
eoffsetc = handles.eoffset_edit;
blengthc = handles.blength_edit;
maLen0c = handles.maLen0_edit;
minert0c = handles.minert0_edit;
maLenTc = handles.maLenT_edit;
minertTc = handles.minertT_edit;
massTipc = handles.massTip_edit;
rtrSpdRngc = handles.rtrSpdRng_edit;
rtrSpdRngc = rtrSpdRngc(:).';
opRtrSpdc = handles.opRtrSpd_edit;
alfac = handles.alfa_edit;

valRB = handles.optionalInputs_radiobutton;
if valRB ~= 1
    massTipc = 0;
    maLenTc = maLen0c;
    minertTc = minert0c;
    rtrSpdRngc = [0:5:700];
    opRtrSpdc = 470;
    alfac = 0;
end

reset(gca)
```

```
[w1,w2,w3,w0,rtrspd,coeffs,w1k,w2k,w3k,w0k]                                              =
yntemag(flagSouthc,cantHingc,eoffsetc,blengthc,melastc,maLen0c,minert0c,'mtip',maLenTc,'I
tip',minertTc,'Mt',massTipc,'AoA',alfac,[rtrSpdRngc opRtrSpdc]);

omega_1 = num2str(w1(1));
set(handles.w1_text, 'String', omega_1)
omega_1rps = num2str(w1(1)*(pi/30));
set(handles.w1rps_text, 'String', omega_1rps)
omega_2 = num2str(w2(1));
set(handles.w2_text, 'String', omega_2)
omega_2rps = num2str(w2(1)*(pi/30));
set(handles.w2rps_text, 'String', omega_2rps)
omega_3 = num2str(w3(1));
set(handles.w3_text, 'String', omega_3)
omega_3rps = num2str(w3(1)*(pi/30));
set(handles.w3rps_text, 'String', omega_3rps)
omega_0 = num2str(w0(1));
set(handles.w0_text, 'String', omega_0)
omega_0rps = num2str(w0(1)*(pi/30));
set(handles.w0rps_text, 'String', omega_0rps)

omegak_1 = num2str(w1k(1));
set(handles.w1k_text, 'String', omegak_1)
omegak_1rps = num2str(w1k(1)*(pi/30));
set(handles.w1krps_text, 'String', omegak_1rps)
omegak_2 = num2str(w2k(1));
set(handles.w2k_text, 'String', omegak_2)
omegak_2rps = num2str(w2k(1)*(pi/30));
set(handles.w2krps_text, 'String', omegak_2rps)
omegak_3 = num2str(w3k(1));
set(handles.w3k_text, 'String', omegak_3)
omegak_3rps = num2str(w3k(1)*(pi/30));
set(handles.w3krps_text, 'String', omegak_3rps)
if isempty(w0k)
    w0k = 0;
end
omegak_0 = num2str(w0k(1));
set(handles.w0k_text, 'String', omegak_0)
omegak_0rps = num2str(w0k(1)*(pi/30));
set(handles.w0krps_text, 'String', omegak_0rps)

a_0 = num2str((coeffs{1}));
set(handles.a0_text, 'String', a_0)
a_1 = num2str(coeffs{2});
set(handles.a1_text, 'String', a_1)
a_2 = num2str(coeffs{3});
set(handles.a2_text, 'String', a_2)
a_3 = num2str(coeffs{4});
set(handles.a3_text, 'String', a_3)

K_0 = num2str(coeffs{5});
set(handles.K0_text, 'String', K_0)
K_1 = num2str(coeffs{6});
set(handles.K1_text, 'String', K_1)
K_2 = num2str(coeffs{7});
set(handles.K2_text, 'String', K_2)
K_3 = num2str(coeffs{8});
set(handles.K3_text, 'String', K_3)

K_00 = num2str(coeffs{9});
set(handles.K00_text, 'String', K_00)
K_01 = num2str(coeffs{10});
set(handles.K01_text, 'String', K_01)
K_02 = num2str(coeffs{11});
set(handles.K02_text, 'String', K_02)
K_03 = num2str(coeffs{12});
set(handles.K03_text, 'String', K_03)

K_10 = num2str(coeffs{13});
set(handles.K10_text, 'String', K_10)
K_11 = num2str(coeffs{14});
set(handles.K11_text, 'String', K_11)
K_12 = num2str(coeffs{15});
set(handles.K12_text, 'String', K_12)
K_13 = num2str(coeffs{16});
set(handles.K13_text, 'String', K_13)

%displaying w_n:
disp(['w_0123 = [ ' omega_0 '   ' omega_1 '   ' omega_2 '   ' omega_3 ' ] CPM']);
```

```
disp(['        = [ ' omega_0rps '    ' omega_1rps '    ' omega_2rps '    ' omega_3rps ' ]
rad/sec']);
%displaying w_n_ksi:
disp(['w_0123_ksi = [ ' omegak_0 '    ' omegak_1 '    ' omegak_2 '    ' omegak_3 ' ] CPM']);
disp(['           = [ ' omegak_0rps '    ' omegak_1rps '    ' omegak_2rps '    ' omegak_3rps
' ] rad/sec']);
%displaying a_n
disp(['a_0123 = [ ' a_0 '    ' a_1 '    ' a_2 '    ' a_3 ' ]']);
%displaying K_n
disp(['K_0123 = [ ' K_0 '    ' K_1 '    ' K_2 '    ' K_3 ' ]']);
%displaying K_0_n
disp(['K_0_0123 = [ ' K_00 '    ' K_01 '    ' K_02 '    ' K_03 ' ]']);
%displaying K_1_n
disp(['Kh_1_0123 = [ ' K_10 '    ' K_11 '    ' K_12 '    ' K_13 ' ]']);
disp('===================== END OF YNTEMAGUI() FILE ============================');

clear global YNTEMAILKKEZMICALISIYOR

diary off;
% ----------------------------------------------------------------------
function varargout = cancel_pushbutton_Callback(h, eventdata, handles, varargin)
```

THIS PAGE INTENTIONALLY LEFT BLANK

# APPENDIX E. EXAMPLE CODES FOR THE VERIFICATION AND VALIDATION OF MATLAB® CODES

## A. CODE FOR A VERIFICATION OF *YNTEMA* FUNCTION

```
flagSouth = {0 1 2 3};%h
cantHing = {'C' 'H'};%jj
eoffset = {0 0.1 9};%k
blength = {0 150};%m
melast = {0 1e7};%n
maLen0 = {0 0.00033};%o
minert0 = {0 1.19};%p
maLenT = {[] 0 0.00011};%q
minertT = {[] 0 1.19};%r
massTip = {[] 0 0.0495};%s
rtrSpd = {[],0,420,[0:10:500],[[0:10:500] 420]};%t
alfa = {[],0,0.15,[0:0.01:0.30],[0.15:0.01:0.40]};%u
figure;
for h = 1:4,
    for jj=1:2,
        for k=1:3,
            diary([pwd date ' - ' num2str(h) num2str(jj) num2str(k)    '.txt']);
             for m=1:2,
                 for n=1:2,
                     for o=1:2,
                         for p=1:2,
                             for q=1:3,
                                for r=1:3,
                                    for s=1:3,
                                        for t=1:5,
                                            for u=1:5,
disp([h jj k m n o p q r s t u]);
[w1,w2,w3,w0,rtrSpdRng,coefss,w1k,w2k,w3k,w0k]=yntema(flagSouth{h},cantHing{jj},
eoffset{k},blength{m},melast{n},maLen0{o},minert0{p},'mtip',maLenT{q},'Itip',
minertT{r},'Mt',massTip{s},rtrSpd{t},'AoA',alfa{u});
                                            end
                                        end
                                    end
                                end
                            end
                            clc;
                        end
                    end
                end
            end
            diary off;
%         end
%     end
% end
disp(['Total Time = ' num2str(cputime-runstart) ' sec.']);

clear;
```

## B. CODE FOR A VALIDATION OF *YNTEMA* FUNCTION

```
%YNTEMATRIAL.M: For a verification of YNTEMA() function.
ch = 'H'; %or ch = 'C';
if upper(ch(1))=='H'
    cho = 'HINGED'; n=4;
else
    cho = 'CANTILEVER'; n=3;
end
mtmr = [0:0.1:1];
EItEIr = [0,0.5,1];
Mt = [0:.01:.2];
%==========================================================
%For Yntema figures 11~16:
```

```
for jj = 1:3,
    for ii = 1:11,
        [w1,w2,w3,w0,rtrSpd,coeffs] = yntema(0,cho(1),0,100,1e7,0.001,1.19,...
            'mtip',0.001*mtmr(ii),'Itip',1.19*EItEIr(jj));
        a0(jj,ii) = coeffs{1}; a1(jj,ii) = coeffs{2};
        a2(jj,ii) = coeffs{3}; a3(jj,ii) = coeffs{4};
        K00(jj,ii) = coeffs{9}; K01(jj,ii) = coeffs{10};
        K02(jj,ii) = coeffs{11}; K03(jj,ii) = coeffs{12};
        Kh10(jj,ii) = coeffs{13}; Kh11(jj,ii) = coeffs{14};
        Kh12(jj,ii) = coeffs{15}; Kh13(jj,ii) = coeffs{16};
        clear w* r*
    end
end
%For Yntema figures 17~20:
for hh = 1:11,
    [w1,w2,w3,w0,rtrSpd,coeffs] = yntema(0,cho(1),0,100,1e7,0.001,1.19,'Mt',Mt(hh));
    a0M(hh) = coeffs{1}; a1M(hh) = coeffs{2}; a2M(hh) = coeffs{3}; a3M(hh) = coeffs{4};
    K00M(hh) = coeffs{9};K01M(hh) = coeffs{10};K02M(hh) = coeffs{11};K03M(hh)=coeffs{12};
    clear w* r*
end
%===========================================================
figure(1);subplot(3,1,3); hold on;
plot(mtmr,a1(1,:),'b-','linewidth',1.5); hold on;
plot(mtmr,a1(2,:),'r-','linewidth',1.5);
plot(mtmr,a1(3,:),'g-','linewidth',1.5);
ylabel('a_1'); xlabel('m_t/m_r');grid minor;
legend('EI_t/EI_r = 0',' = .5',' = 1');
subplot(3,1,2);
plot(mtmr,a2(1,:),'b-','linewidth',1.5);hold on;
plot(mtmr,a2(2,:),'r-','linewidth',1.5);
plot(mtmr,a2(3,:),'g-','linewidth',1.5);
ylabel('a_2');grid minor;
legend('EI_t/EI_r = 0',' = .5',' = 1');
subplot(3,1,1);
plot(mtmr,a3(1,:),'b-','linewidth',1.5); hold on;
plot(mtmr,a3(2,:),'r-','linewidth',1.5);
plot(mtmr,a3(3,:),'g-','linewidth',1.5);
ylabel('a_3');grid minor;
legend('EI_t/EI_r = 0',' = .5',' = 1');
title([cho ', m_0=0.001 lbm/in, I_0=1.19 in^4, E=1e7 psi']);
figure(2);subplot(3,1,3);
plot(mtmr,K01(1,:),'b-','linewidth',1.5); hold on;
plot(mtmr,K01(2,:),'r-','linewidth',1.5);
plot(mtmr,K01(3,:),'g-','linewidth',1.5);
ylabel('K_0_1'); xlabel('m_t/m_r');grid minor;
legend('EI_t/EI_r = 0',' = .5',' = 1');
subplot(3,1,2);
plot(mtmr,K02(1,:),'b-','linewidth',1.5); hold on;
plot(mtmr,K02(2,:),'r-','linewidth',1.5);
plot(mtmr,K02(3,:),'g-','linewidth',1.5);
ylabel('K_0_2');grid minor;
legend('EI_t/EI_r = 0',' = .5',' = 1');
subplot(3,1,1);
plot(mtmr,K03(1,:),'b-','linewidth',1.5); hold on;
plot(mtmr,K03(2,:),'r-','linewidth',1.5);
plot(mtmr,K03(3,:),'g-','linewidth',1.5);
ylabel('K_0_3');grid minor;
legend('EI_t/EI_r = 0',' = .5',' = 1');
title([cho ', m_0=0.001 lbm/in, I_0=1.19 in^4, E=1e7 psi']);
figure(3);
if n==4
    subplot(n,1,4);
    plot(mtmr,Kh10(1,:),'b-','linewidth',1.5); hold on;
    plot(mtmr,Kh10(2,:),'r-','linewidth',1.5);
    plot(mtmr,Kh10(3,:),'g-','linewidth',1.5);
    ylabel('K_1_0'); xlabel('m_t/m_r');grid minor;
    legend('EI_t/EI_r = 0',' = .5',' = 1');
end
subplot(n,1,3);
plot(mtmr,Kh11(1,:),'b-','linewidth',1.5); hold on;
plot(mtmr,Kh11(2,:),'r-','linewidth',1.5);
plot(mtmr,Kh11(3,:),'g-','linewidth',1.5);
ylabel('K_1_1'); grid minor;
if n==3; xlabel('m_t/m_r');end
legend('EI_t/EI_r = 0',' = .5',' = 1');
subplot(n,1,2);
plot(mtmr,Kh12(1,:),'b-','linewidth',1.5); hold on;
plot(mtmr,Kh12(2,:),'r-','linewidth',1.5);
plot(mtmr,Kh12(3,:),'g-','linewidth',1.5);
```

```
ylabel('K_1_2');grid minor;
legend('EI_t/EI_r = 0',' = .5',' = 1');
subplot(n,1,1);
plot(mtmr,Kh13(1,:),'b-','linewidth',1.5); hold on;
plot(mtmr,Kh13(2,:),'r-','linewidth',1.5);
plot(mtmr,Kh13(3,:),'g-','linewidth',1.5);
ylabel('K_1_3');grid minor;
legend('EI_t/EI_r = 0',' = .5',' = 1');
title([cho ', m_0=0.001 lbm/in, I_0=1.19 in^4, E=1e7 psi']);
figure(4);subplot(3,1,3); hold on;
plot(Mt./(0.001*100),a1M(1,:),'b-','linewidth',1.5); hold on;
ylabel('\theta_1^2'); xlabel('r');grid minor;
subplot(3,1,2);
plot(Mt./(0.001*100),a2M(1,:),'b-','linewidth',1.5); hold on;
ylabel('\theta_2^2');grid minor;
subplot(3,1,1);
plot(Mt./(0.001*100),a3M(1,:),'b-','linewidth',1.5); hold on;
ylabel('\theta_3^2');grid minor;
title([cho ', m_0=0.001 lbm/in, I_0=1.19 in^4, E=1e7 psi']);
figure(5);subplot(3,1,3); hold on;
plot(Mt./(0.001*100),K01M(1,:),'b-','linewidth',1.5); hold on;
ylabel('K_0_1'); xlabel('r');grid minor;
subplot(3,1,2);
plot(Mt./(0.001*100),K02M(1,:),'b-','linewidth',1.5); hold on;
ylabel('K_0_2');grid minor;
subplot(3,1,1);
plot(Mt./(0.001*100),K03M(1,:),'b-','linewidth',1.5); hold on;
ylabel('K_0_3');grid minor;

title([cho ', m_0=0.001 lbm/in, I_0=1.19 in^4, E=1e7 psi']);
```

## C. CODE FOR A VALIDATION OF THE *YNTERYGH* FUNCTION

```
%YNTERYGHTRIAL.M: verification for ynterygh.m
rtfxty = 'CH';
for tt=1:2,
    ch = rtfxty(tt);
    mtmr = [0:0.1:1];
    EItEIr = [0,0.5,1];
    L = 200;
    for jj = 1:3,
        for ii = 1:11,
            [y1NR,y2NR,y3NR]=yntemoshnr(0,ch,mtmr(ii),EItEIr(jj));
            [w1,a1(jj,ii),K1(jj,ii),K01(jj,ii),K11(jj,ii)]                          =
YNTERYGH(y1NR','L',L,'mx',0.00033.*[1;mtmr(ii)],'EIx',1.19e7.*[1:-0.1:EItEIr(jj)]');
            [w2,a2(jj,ii),K2(jj,ii),K02(jj,ii),K12(jj,ii)]                          =
YNTERYGH(y2NR','L',L,'mx',0.00033.*[1;mtmr(ii)],'EIx',1.19e7.*[1:-0.1:EItEIr(jj)]');
            [w3,a3(jj,ii),K3(jj,ii),K03(jj,ii),K13(jj,ii)]                          =
YNTERYGH(y3NR','L',L,'mx',0.00033.*[1;mtmr(ii)],'EIx',1.19e7.*[1:-0.1:EItEIr(jj)]');
            [w1r,w2r,w3r,w0r,rotspd,coeffs{jj,ii}]                                  =
YNTEMA(0,ch,0,L,1e7,0.00033,1.19,'mtip',0.00033*mtmr(ii),'Itip',1.19*EItEIr(jj));
        end
    end
    figure;
    subplot(3,1,3); hold on;
    plot(mtmr,a1(1,:),'b-','linewidth',1.5); hold on;
    plot(mtmr,a1(2,:),'r-','linewidth',1.5);
    plot(mtmr,a1(3,:),'g-','linewidth',1.5);
    ylabel('a1'); xlabel('m_t/m_r');
    grid minor;
    legend('EI_t/EI_r = 0',' = .5',' = 1');
    subplot(3,1,2);
    plot(mtmr,a2(1,:),'b-','linewidth',1.5);hold on;
    plot(mtmr,a2(2,:),'r-','linewidth',1.5);
    plot(mtmr,a2(3,:),'g-','linewidth',1.5);
    ylabel('a2');
    grid minor;
    legend('EI_t/EI_r = 0',' = .5',' = 1');
    subplot(3,1,1);
    plot(mtmr,a3(1,:),'b-','linewidth',1.5); hold on;
    plot(mtmr,a3(2,:),'r-','linewidth',1.5);
    plot(mtmr,a3(3,:),'g-','linewidth',1.5);
```

```
ylabel('a3');
grid minor;
legend('EI_t/EI_r = 0',' = .5',' = 1');
if upper(ch(1)) == 'C'
    title('CANTILEVER, m_0=0.00033 lbm/in, I_0=1.19 in^4, E=1e7 psi');
else
    title('HINGED, m_0=0.00033 lbm/in, I_0=1.19 in^4, E=1e7 psi');
end
figure;
subplot(3,1,3);
plot(mtmr,K01(1,:),'b-','linewidth',1.5); hold on;
plot(mtmr,K01(2,:),'r-','linewidth',1.5);
plot(mtmr,K01(3,:),'g-','linewidth',1.5);
ylabel('K01'); xlabel('m_t/m_r');
grid minor;
legend('EI_t/EI_r = 0',' = .5',' = 1');
subplot(3,1,2);
plot(mtmr,K02(1,:),'b-','linewidth',1.5); hold on;
plot(mtmr,K02(2,:),'r-','linewidth',1.5);
plot(mtmr,K02(3,:),'g-','linewidth',1.5);
ylabel('K02');
grid minor;
legend('EI_t/EI_r = 0',' = .5',' = 1');
subplot(3,1,1);
plot(mtmr,K03(1,:),'b-','linewidth',1.5); hold on;
plot(mtmr,K03(2,:),'r-','linewidth',1.5);
plot(mtmr,K03(3,:),'g-','linewidth',1.5);
ylabel('K03');
grid minor;
legend('EI_t/EI_r = 0',' = .5',' = 1');
if upper(ch(1)) == 'C'
    title('CANTILEVER, m_0=0.00033 lbm/in, I_0=1.19 in^4, E=1e7 psi');
else
    title('HINGED, m_0=0.00033 lbm/in, I_0=1.19 in^4, E=1e7 psi');
end
figure;
subplot(3,1,3);
plot(mtmr,K11(1,:)*L,'b-','linewidth',1.5); hold on;
plot(mtmr,K11(2,:)*L,'r-','linewidth',1.5);
plot(mtmr,K11(3,:)*L,'g-','linewidth',1.5);
ylabel('K11'); xlabel('m_t/m_r');
grid minor;
legend('EI_t/EI_r = 0',' = .5',' = 1');
subplot(3,1,2);
plot(mtmr,K12(1,:)*L,'b-','linewidth',1.5); hold on;
plot(mtmr,K12(2,:)*L,'r-','linewidth',1.5);
plot(mtmr,K12(3,:)*L,'g-','linewidth',1.5);
ylabel('K12');
grid minor;
legend('EI_t/EI_r = 0',' = .5',' = 1');
subplot(3,1,1);
plot(mtmr,K13(1,:)*L,'b-','linewidth',1.5); hold on;
plot(mtmr,K13(2,:)*L,'r-','linewidth',1.5);
plot(mtmr,K13(3,:)*L,'g-','linewidth',1.5);
ylabel('K13');
grid minor;
legend('EI_t/EI_r = 0',' = .5',' = 1');
if upper(ch(1)) == 'C'
    title('CANTILEVER, m_0=0.00033 lbm/in, I_0=1.19 in^4, E=1e7 psi');
else
    title('HINGED, m_0=0.00033 lbm/in, I_0=1.19 in^4, E=1e7 psi');
end
figure;
subplot(3,1,3); hold on;
for mm=1:3,
    for kk = 1:11,
        vara1(mm,kk) = (a1(mm,kk)-coeffs{mm,kk}{2})/a1(mm,kk)*100;
    end
end
plot(mtmr,vara1(1,:),'b-','linewidth',1.5); hold on;
plot(mtmr,vara1(2,:),'r-','linewidth',1.5);
plot(mtmr,vara1(3,:),'g-','linewidth',1.5);
ylabel('Error Percentage for a1'); xlabel('m_t/m_r');
grid minor;
legend('EI_t/EI_r = 0',' = .5',' = 1');
subplot(3,1,2);
for mm=1:3,
    for kk = 1:11,
        vara2(mm,kk) = (a2(mm,kk)-coeffs{mm,kk}{3})/a2(mm,kk)*100;
```

```
    end
end
plot(mtmr,vara2(1,:),'b-','linewidth',1.5);hold on;
plot(mtmr,vara2(2,:),'r-','linewidth',1.5);
plot(mtmr,vara2(3,:),'g-','linewidth',1.5);
ylabel('Error Percentage for a2');
grid minor;
legend('EI_t/EI_r = 0',' = .5',' = 1');
subplot(3,1,1);
for mm=1:3,
    for kk = 1:11,
        vara3(mm,kk) = (a3(mm,kk)-coeffs{mm,kk}{4})/a3(mm,kk)*100;
    end
end
plot(mtmr,vara3(1,:),'b-','linewidth',1.5); hold on;
plot(mtmr,vara3(2,:),'r-','linewidth',1.5);
plot(mtmr,vara3(3,:),'g-','linewidth',1.5);
ylabel('Error Percentage for a3');
grid minor;
legend('EI_t/EI_r = 0',' = .5',' = 1');
if upper(ch(1)) == 'C'
    title('CANTILEVER, m_0=0.00033 lbm/in, I_0=1.19 in^4, E=1e7 psi');
else
    title('HINGED, m_0=0.00033 lbm/in, I_0=1.19 in^4, E=1e7 psi');
end
figure;
subplot(3,1,3);
for mm=1:3,
    for kk = 1:11,
        varK01(mm,kk) = (K01(mm,kk)-coeffs{mm,kk}{10})/K01(mm,kk)*100;
    end
end
plot(mtmr,varK01(1,:),'b-','linewidth',1.5); hold on;
plot(mtmr,varK01(2,:),'r-','linewidth',1.5);
plot(mtmr,varK01(3,:),'g-','linewidth',1.5);
ylabel('Error Percentage for K01'); xlabel('m_t/m_r');
grid minor;
legend('EI_t/EI_r = 0',' = .5',' = 1');
subplot(3,1,2);
for mm=1:3,
    for kk = 1:11,
        varK02(mm,kk) = (K02(mm,kk)-coeffs{mm,kk}{11})/K02(mm,kk)*100;
    end
end
plot(mtmr,varK02(1,:),'b-','linewidth',1.5); hold on;
plot(mtmr,varK02(2,:),'r-','linewidth',1.5);
plot(mtmr,varK02(3,:),'g-','linewidth',1.5);
ylabel('Error Percentage for K02');
grid minor;
legend('EI_t/EI_r = 0',' = .5',' = 1');
subplot(3,1,1);
for mm=1:3,
    for kk = 1:11,
        varK03(mm,kk) = (K03(mm,kk)-coeffs{mm,kk}{12})/K03(mm,kk)*100;
    end
end
plot(mtmr,varK03(1,:),'b-','linewidth',1.5); hold on;
plot(mtmr,varK03(2,:),'r-','linewidth',1.5);
plot(mtmr,varK03(3,:),'g-','linewidth',1.5);
ylabel('Error Percentage for K03');
grid minor;
legend('EI_t/EI_r = 0',' = .5',' = 1');
if upper(ch(1)) == 'C'
    title('CANTILEVER, m_0=0.00033 lbm/in, I_0=1.19 in^4, E=1e7 psi');
else
    title('HINGED, m_0=0.00033 lbm/in, I_0=1.19 in^4, E=1e7 psi');
end
figure;
subplot(3,1,3);
for mm=1:3,
    for kk = 1:11,
        varK11(mm,kk) = (K11(mm,kk)*L-coeffs{mm,kk}{14})/K11(mm,kk)/L*100;
    end
end
plot(mtmr,varK11(1,:),'b-','linewidth',1.5); hold on;
plot(mtmr,varK11(2,:),'r-','linewidth',1.5);
plot(mtmr,varK11(3,:),'g-','linewidth',1.5);
ylabel('Error Percentage for K11'); xlabel('m_t/m_r');
grid minor;
```

```
    legend('EI_t/EI_r = 0',' = .5',' = 1');
    subplot(3,1,2);
    for mm=1:3,
        for kk = 1:11,
            varK12(mm,kk) = (K12(mm,kk)*L-coeffs{mm,kk}{15})/K12(mm,kk)/L*100;
        end
    end
    plot(mtmr,varK12(1,:),'b-','linewidth',1.5); hold on;
    plot(mtmr,varK12(2,:),'r-','linewidth',1.5);
    plot(mtmr,varK12(3,:),'g-','linewidth',1.5);
    ylabel('Error Percentage for K12');
    grid minor;
    legend('EI_t/EI_r = 0',' = .5',' = 1');
    subplot(3,1,1);
    for mm=1:3,
        for kk = 1:11,
            varK13(mm,kk) = (K13(mm,kk)*L-coeffs{mm,kk}{16})/K13(mm,kk)/L*100;
        end
    end
    plot(mtmr,varK13(1,:),'b-','linewidth',1.5); hold on;
    plot(mtmr,varK13(2,:),'r-','linewidth',1.5);
    plot(mtmr,varK13(3,:),'g-','linewidth',1.5);
    ylabel('Error Percentage for K13');
    grid minor;
    legend('EI_t/EI_r = 0',' = .5',' = 1');
    if upper(ch(1)) == 'C'
        title('CANTILEVER, m_0=0.00033 lbm/in, I_0=1.19 in^4, E=1e7 psi');
    else
        title('HINGED, m_0=0.00033 lbm/in, I_0=1.19 in^4, E=1e7 psi');
    end

end
```

# LIST OF REFERENCES

1. Yntema, R. T., *Simplified Procedures and Charts for the Rapid Estimation of Bending Frequencies of Rotating Beams,* NACA Technical Note 3459, Langley Field, VA, 1955.

2. Hartog, J. P. Den, *Mechanical Vibrations*, Dover Publications, Inc., New York, 1985.

3. Fung, Y. C., *An Introduction to the Theory of Aeroelasticity,* Dover Publications, Inc., New York, 1969.

4. Hartog, J. P. Den, *Strength of Materials,* Dover Publications, Inc., New York, 1961.

5. Bisplinghoff, R. L., Ashley, H. A., Halfman, R. L., *Aeroelasticity,* Dover Publications, Inc., New York, 1996.

6. Timoshenko, S. P., *History of Strength of Materials,* Dover Publications, Inc., New York, 1983.

7. Hartog, J. P. Den, *Advanced Strength of Materials,* Dover Publications, Inc., New York, 1987.

8. Hartog, J. P. Den, *Mechanics,* Dover Publications, Inc., New York, 1961.

9. Gerstenberger, W. and Wood, E. R., *Analysis of Helicopter Aeroelastic Characteristics in High-Speed Flight,* AIAA Journal, Vol. 1, No. 10, pp 2366-2381, November 1963.

10. Nicholson, R. K. Jr., *Computer Code for Interactive Rotorcraft Preliminary Design Using a Harmonic Balance Method for Rotor Trim,* Naval Postgraduate School Master's Thesis, Monterey, CA, 1993.

11. Wirth, W. M., *Linear Modeling of Rotorcraft for Stability Analysis and Preliminary Design,* Naval Postgraduate School Master's Thesis, Monterey, CA, 1993.

12. Eccles, D. M., *A Validation of the Joint Army/Navy Rotorcraft Analysis ans Design Software by Comparison with H-34 and UUH-60A Flight Test,* Naval Postgraduate School Master's Thesis, Monterey, CA, 1995.

13. Lapacik, C. F., *Development of Graphical User Interface for Joint Army/Navy Rotorcraft Analysis and Design (JANRAD) software,* Naval Postgraduate School Master's Thesis, Monterey, CA, 1998.

14. Hucke, W. L., *Performance Enhancements to Joint Army/Navy Rotorcraft Analysis and Design (JANRAD) Software and Graphical User Interface (GUI),* Naval Postgraduate School Master's Thesis, Monterey, CA, 1998.

15. MATLAB® 6 Software Documentation, *Using MATLAB®,* ch. 13, The MathWorks Inc., 2000.

16. Hanselman, D. C. and Littlefeild, B., *Mastering MATLAB®,* Prentice Hall, 1996.

17. Marchand, P., *Graphics and GUI's with MATLAB®,* CRC Press, 1996.

18. MATLAB® 6 Software Documentation, *Building GUI's with MATLAB®,* The MathWorks Inc., 2000.

19. Dean, L., *Building a Graphical User Interface with MATLAB® 5,* The MathWorks Inc., 1997.

20. Dean, L., *Design of Callbacks,* MATLAB® Digest, Volume 5, No.3, December 1997.

# INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Kara Kuvvetleri Komutanligi
   Ankara, TURKIYE

4. K.H.O. Savunma Bilimleri Enstitusu
   Ankara, TURKIYE

5. Prof. E. Roberts Wood
   Naval Postgraduate School
   Monterey, CA

6. CDR Mark A. Couch
   Naval Postgraduate School
   Monterey, CA

7. Prof. Fevzi Unal
   I.T.U. Ucak ve Uzay Bilimleri Fakultesi Dekani
   Maslak, Istanbul, TURKIYE

8. O.D.T.U. Havacilik Muhendisligi Bolumu
   Ankara, TURKIYE

9. Anadolu Universitesi Sivil Havacilik Okulu
   Eskisehir, TURKIYE

10. Erciyes Universitesi Sivil Havacilik Okulu
    Kayseri, TURKIYE

11. Turkish Aerospace Industries Inc.
    Ankara, TURKIYE

12. 901. Hv.Arc.A.Dp. ve Fb.K.ligi
    Guvercinlik, Ankara, TURKIYE

13. Hakki Erdem Akin
    Ankara, TURKIYE

14. Dogan Ozturk
    Ankara, TURKIYE