

# Free Vibrations of Nonuniform Timoshenko Beams II

C.H. von Kerczek

## 1. Introduction:

I present here some vibratory characteristics (eigenvalues and eigenfunctions) of Timoshenko beams (T-beams) with variable cross section shape and/or a variable elastic property along the length of the beam. In this study I have developed a 2nd order finite difference method for solving these problems. In the case of uniform prismatic beams the eigenvalue problem can be solved exactly. I use these exact solutions to validate the finite difference solution of the differential equations.

## 2. Formulation of the T-beam theory:

Timoshenko beam (T-beam) theory is a well known extension of the Classical Euler-Bernoulli (E-B) beam theory. EB- theory is taught in elementary Mechanics of Materials courses in most engineering curricula. T-beam theory is taught in advanced mechanics of Materials courses. Therefore we will not derive the T-beam theory here, but only state the relevant equations, scaling and notation. A derivation of the T-beam theory can be found in Reference [1].

The the beam lies along the x axis in the x-y-z coordinate system with its left end at  $x=0$  and its right end at  $x=L$ . The forces and the motion of the beam are only in the x-y plane with  $y(x,t)$  denoting the displacement of the neutral axis of the beam. The beam is characterized by a cross section area equation  $S(x,y,z)$  which is symmetric across the x-y plane;  $S(x,y,z)=S(x,y,-z)$ . The beam material is characterized by an elastic modulus  $E(x)$ , a poison ratio  $\nu$  and a mass density  $\rho$ . The differential equations governing the deflection  $y(x,t)$  and the shear angle  $f(x,t)$  of the beam are

$$(1a) \quad \rho A \frac{\partial^2 y}{\partial t^2} = \frac{\partial}{\partial x} \left[ \kappa A G \left( \frac{\partial y}{\partial x} - f \right) \right] + p(x, t)$$

$$(1b) \quad \rho I \frac{\partial^2 f}{\partial t^2} = \frac{\partial}{\partial x} \left[ E I \frac{\partial f}{\partial x} \right] + \kappa A G \left( \frac{\partial y}{\partial x} - f \right)$$

where  $t$  denotes time,  $G = \frac{E}{2(1+\nu)}$ ,  $\kappa$  is a cross section shape factor,  $p(x,t)$  is the force distribution exciting the beam,  $A(x)$  is the cross section area and  $I(x)$  is the cross section area 2nd moment about the z axis passing through the centroid of the area.

$$A(x) = \iint_{S(x,y,z)} dydz \quad \text{and} \quad I(x) = \iint_{S(x,y,z)} y^2 dydz$$

The beam's elastic modulus  $E$ , cross section area  $A$  and 2nd moment  $I$  can vary along  $x$ . Say  $E(x)$ ,  $A(x)$  and  $I(x)$  have the reference values  $E_0, A_0$  and  $I_0$  respectively at some location on the beam. Then we write

$$(2a,b,c) \quad E = E_0 E'(x), \quad A = A_0 A'(x) \quad \text{and} \quad I = I_0 I'(x)$$

so that  $E'$ ,  $A'$  and  $I'$  are dimensionless shape functions of  $x$ . If  $E'$ ,  $A'$  and  $I'$  are constant in  $x$ , then they are each equal to 1. Now apply the following scalings:

$$(3) \quad p = F_0 p'/L, \quad x = Lx', \quad y = Ly', \quad t = t' \sqrt{\rho L^2/E_0}$$

and substitute (2) and (3) into equations (1) and dispense with the primes. There results

$$(4a) \quad \frac{\partial^2 y}{\partial t^2} = \frac{\alpha}{A} \frac{\partial}{\partial x} \left[ EA \left( \frac{\partial y}{\partial x} - f \right) \right] + \frac{q}{A}$$

$$(4b) \quad \frac{\partial^2 f}{\partial t^2} = \frac{1}{I} \frac{\partial}{\partial x} \left( EI \frac{\partial f}{\partial x} \right) + \alpha RE \frac{A}{I} \left( \frac{\partial y}{\partial x} - f \right)$$

where  $q = p' \frac{F_0}{E_0 A_0} = p \frac{L}{E_0 A_0}$  (dimensional  $p$ ) is a rescaled force distribution,  $\alpha = \frac{\kappa}{2(1+\nu)}$

and  $R = \frac{L^2 A_0}{I_0}$ .  $R$  is the *natural slenderness ratio* for this problem (the inverse of the square of the dimensionless radius of gyration of the beam reference cross section).

Reconsider the dimensionless time variable  $t$ . Rescale it as  $t = \sqrt{R} \tau$  and substitute this into equations (4) to obtain

$$(5a) \quad \frac{1}{R} \frac{\partial^2 y}{\partial \tau^2} = \frac{\alpha}{A} \frac{\partial}{\partial x} \left[ EA \left( \frac{\partial y}{\partial x} - f \right) \right] + \frac{q}{A}$$

$$(5b) \quad \frac{1}{R} \frac{\partial^2 f}{\partial \tau^2} = \frac{1}{I} \frac{\partial}{\partial x} \left( EI \frac{\partial f}{\partial x} \right) + \alpha RE \frac{A}{I} \left( \frac{\partial y}{\partial x} - f \right)$$

Multiply equation (5a) by  $R$  and rearrange it to obtain

$$(6a) \quad \frac{\partial^2 y}{\partial \tau^2} = \frac{1}{A} \frac{\partial}{\partial x} \left[ \alpha REA \left( \frac{\partial y}{\partial x} - f \right) \right] + \frac{qR}{A}$$

Then multiply equation (5b) by  $I$  and rearrange it to obtain

$$(6b) \quad \alpha REA \left( \frac{\partial y}{\partial x} - f \right) = \frac{I}{R} \frac{\partial^2 f}{\partial \tau^2} - \frac{\partial}{\partial x} \left( EI \frac{\partial f}{\partial x} \right)$$

In equation (6b) as  $R \rightarrow \infty$   $f \rightarrow \frac{\partial y}{\partial x}$ , so that substituting (6b) into the right side of (6a) and taking the limit  $R \rightarrow \infty$  one obtains

$$(7) \quad A(x) \frac{\partial^2 y}{\partial \tau^2} = \frac{-\partial^2}{\partial x^2} \left( EI \frac{\partial^2 y}{\partial x^2} \right) + q(x, \tau) R .$$

The force term  $qR$  in equation (7) *remains finite* as  $R \rightarrow \infty$ . In the above scaling we have  $qR = \rho' \frac{F_0}{E_0 A_0} \frac{L^2 A_0}{I_0} = \rho' \frac{F_0 L^2}{E_0 I_0} = \rho \frac{L^3}{E_0 I_0} = w$ . The last equality is the normal scaling for the E-B beam equations. So by replacing  $qR$  by  $w$  in equation (7) we have recovered the E-B beam theory. Furthermore, we have the new scaling of dimensional time  $t = T\tau$ , where

$$T = \sqrt{\frac{\rho L^4 A_0}{I_0 E_0}} .$$

Equations (5) with  $q=0$  are our basic working equations for vibration of T-beams. These equations are supplemented by an appropriate combination of boundary conditions (BCs)

$$(8) \quad (a) \ y(x_l)=0, \ (b) \ f(x_l)=0, \ (c) \ f'(x_l)=0, \ (d) \ y'(x_l)-f(x_l)=0$$

where  $x_l$  denotes the location of the boundary condition, usually  $x_l=0$  and  $1$ .

In T-beam theory a cantilever end does not impose condition  $y'(x_l)=0$  as in E-B theory. Instead the shear angle  $f$  is made zero there (condition b). Likewise, in T-beam theory the condition of no bending moment and no shear force at a free end requires conditions (c) and (d) respectively instead of the conditions  $y''(x_l)=0$  and  $y'''(x_l)=0$  as in E-B theory.

### 3. The Calculation of Vibration Characteristics of T-beams:

To compute the vibration characteristics of T-beams we set  $q=0$  and assume the time variation in equations (6) to be of the form

$$(9) \quad \begin{bmatrix} y \\ f \end{bmatrix} = \begin{bmatrix} Y(x) \\ F(x) \end{bmatrix} \exp(i\omega\tau) .$$

Then equations (6) become our eigenvalues (evs),  $\omega^2/R$ , and eigenfunctions (efs), ( $Y$  and  $F$ ) problem

$$(10a) \quad -\frac{\omega^2}{R} Y = \frac{1}{A} \frac{d}{dx} \left[ \alpha EA \left( \frac{dY}{dx} - F \right) \right]$$

and

$$(10b) \quad -\frac{1}{R} \omega^2 F = \frac{1}{I(x)} \frac{d}{dx} \left( EI \frac{dF}{dx} \right) + \frac{\alpha R E A}{I(x)} \left( \frac{dY}{dx} - F \right) .$$

There are many numerical ways to calculate the evs and efs of equations (10). We will do this by a second order finite difference (FD) matrix method which is very easy to implement.

### 3.1: FD matrix method:

First equations (10) are expanded as

$$(11a) \quad \alpha E \frac{d^2 Y}{dx^2} + \alpha \frac{(EA)'}{A} \frac{dY}{dx} - \alpha E \frac{dF}{dx} - \alpha \frac{(EA)'}{A} F = -\beta^4 Y$$

$$(11b) \quad E \frac{d^2 F}{dx^2} + \frac{(EI)'}{I} \frac{dF}{dx} + \alpha R \frac{EA}{I} \frac{dY}{dx} - \alpha R \frac{EA}{I} F = -\beta^4 F$$

where (...) denotes the x-derivative of the quantity in the parenthesis.

My implementation partitions the interval [0,1] into a uniform N point grid as  $x_j \in [0=x_1, x_2, \dots, x_N=1]$  where  $h=x_j-x_{j-1}=\text{constant}$  for  $j=2,3,\dots,N$ . Furthermore I assign the grid point values of Y and F to be  $Y_j=Y(x_j)$  and  $F_j=F(x_j)$  for  $j=1,2,\dots,N$ . Equations (11) are evaluated at each *interior* grid point  $j=2,3,\dots,N-1$  with the derivatives replaced by finite difference formulas, for example for Y,

$$(12) \quad \left( \frac{dY}{dx} \right)_j = \frac{Y_{j-1} - Y_{j+1}}{2h} \quad \text{and} \quad \left( \frac{d^2 Y}{dx^2} \right)_j = \frac{Y_{j-1} - 2Y_j + Y_{j+1}}{h^2} .$$

Now I construct matrix derivative operators that obtain the derivatives at all the interior grid points simultaneously. Define the N component row differencing operators

$$(13a) \quad d1_j = [0, 0, \dots, -1, 0, 1, \dots, 0, 0]$$

$$(13b) \quad d2_j = [0, 0, \dots, 1, -2, 1, \dots, 0, 0]$$

for  $j=2,3,\dots,N-1$ , where the first non-zero element in these row operators are at the position  $j-1$ . Define the vector of discrete values of Y and F as

$$(14) \quad \begin{bmatrix} \vec{Y} \\ \vec{F} \end{bmatrix} = [Y_1, Y_2, \dots, Y_N, F_1, F_2, \dots, F_N]^T ,$$

Where the superscript T denotes transpose the row vector into a column vector.

The first and second discrete derivatives of Y and F at the *interior points*  $j=2,3,\dots,N-1$

can now be computed by the simple vector operations

$$(15) \quad \frac{d}{dx} \begin{bmatrix} \vec{Y} \\ \vec{F} \end{bmatrix}_j = \frac{1}{2h} \begin{bmatrix} d1_j \vec{Y} \\ d1_j \vec{F} \end{bmatrix} \quad \text{and} \quad \frac{d^2}{dx^2} \begin{bmatrix} \vec{Y} \\ \vec{F} \end{bmatrix}_j = \frac{1}{h^2} \begin{bmatrix} d2_j \vec{Y} \\ d2_j \vec{F} \end{bmatrix} \quad \text{for } j=2,3,\dots,N-1.$$

For the time being let  $By_1, By_N, Bf_1, Bf_N$  each be N component row vectors that implement the some type of condition on Y and F at  $x_1=0$  and  $x_N=1$  respectively. For example say we want to evaluate the derivative  $Y'(0)=Y'_1$ . The central differencing formula (13a) cannot be used at the end point  $j=1$  because there is no value  $Y_0$ , but it is possible to use a quadratic (to maintain  $O(h^2)$  accuracy) forward differencing formula given as the N element row vector  $By_1=[-3,4,1,0,\dots,0]$ . Presently we will leave the  $By_1, By_N, Bf_1, Bf_N$  open to later implement the BCs. But an example of just taking the first derivative of a discrete function, say  $g_j=\sin(\pi x_j)$  for  $x_j \in [0,0.1,0.2,\dots,0.9,1]$  ( $N=11$ ) we construct the matrix derivative operator D1 as

$$(16) \quad D1 \vec{g} = \frac{1}{2h} \begin{bmatrix} By_1 \\ d1_2 \\ d1_2 \\ \cdot \\ \cdot \\ d1_{N-1} \\ By_N \end{bmatrix} \vec{g}$$

where  $By_N=[0,0,\dots,0,1,-4,3]$  and D1 is a NxN matrix. Formula (16) is the first derivative operator that produces the derivative values at all the grid points at once. The Figure 1 below shows a comparison of the exact  $g'(x)$  with the discrete derivative  $g'_j$  computed by D1 at the discrete points  $x_j$ .

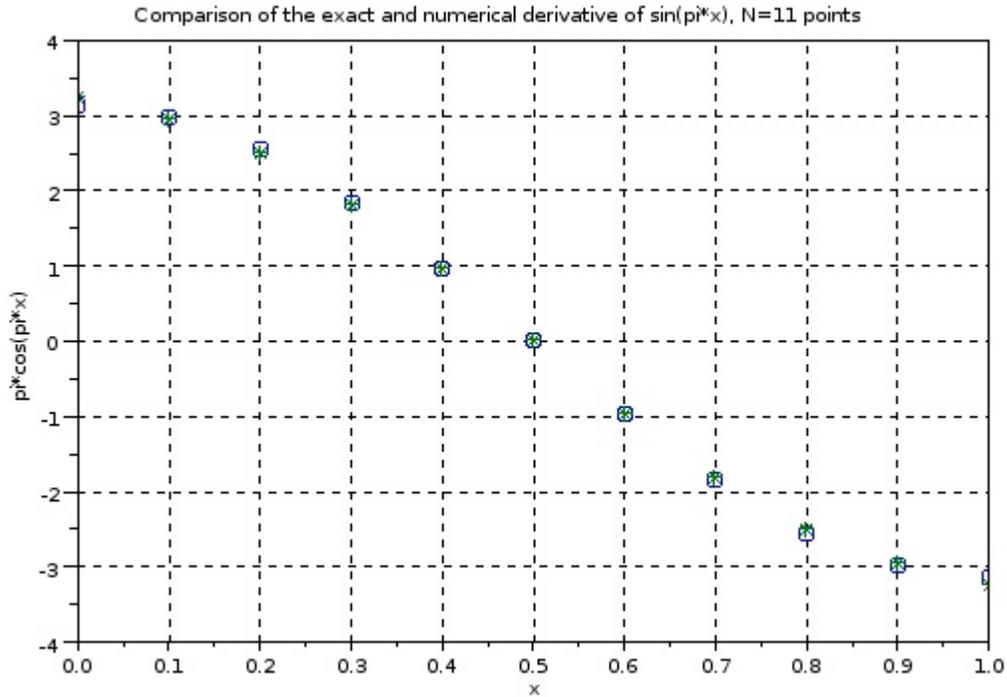


Figure 1 A comparison of the numerical vs the exact derivative of  $\sin(\pi x)$ .  $\circ$  are the exact values and  $*$  are the numerical values obtained using formula (16).

It can be seen in Figure 1 that formula (16) performs quite well except at the end points which are not quite as accurate. This is of no consequence since this can be made much more accurate by making  $N$  larger, which will be almost always the case. Typically, we will not need to compute the derivatives at the end points since BCs specify exactly what is to be imposed at the ends. These conditions will be specified by the end vectors  $By_1, By_N, Bf_1, Bf_N$ . Before dealing with these let us construct the matrix difference equations for the *interior points* of the interval.

Define the matrices

$$(17) \quad D1 = \frac{1}{2h} \begin{bmatrix} \vec{0} \\ d1_2 \\ d1_3 \\ \vdots \\ \vdots \\ d1_{N-1} \\ \vec{0} \end{bmatrix}, \quad D21 = \frac{1}{h^2} \begin{bmatrix} By_1 \\ d2_2 \\ d2_3 \\ \vdots \\ \vdots \\ d2_{N-1} \\ By_N \end{bmatrix}, \quad D22 = \frac{1}{h^2} \begin{bmatrix} Bf_1 \\ d2_2 \\ d2_3 \\ \vdots \\ \vdots \\ d2_{N-1} \\ Bf_N \end{bmatrix} \quad \text{and} \quad l1 = \begin{bmatrix} \vec{0} \\ \vec{1}_2 \\ \vec{1}_3 \\ \vdots \\ \vdots \\ \vec{1}_{N-1} \\ \vec{0} \end{bmatrix}$$

where  $\vec{0}$  denotes a row of N zeros,  $\vec{1}_j$  denotes a row of zeros except at position j where there is a 1. I have placed the special end conditions in the first and last entries of the 2<sup>nd</sup> derivative matrix for convenience and that is why I placed rows of zeros in the first and last position of the other matrices.

Now evaluate all the coefficient functions in equations (11) at the grid points and for the sake of compactness define the vectors

$$(18) \quad \vec{e}=[E_j], \quad \vec{g}=\left[\left(\frac{(EA)'}{A}\right)_j\right], \quad \vec{q}=\left[\left(\frac{(EI)'}{I}\right)_j\right], \quad \vec{s}=\left[\left(\frac{(EA)}{I}\right)_j\right]$$

where in each bracket the terms run from j=1 to j=N.

With the preliminary setup of the matrices in (17) and the row vectors in (18) it is rather easy to set up the finite difference form of equations (11) by simply replacing the analytical terms by the appropriate matrices. Thus the matrix finite difference representation of equations (11) is

$$(19) \quad \begin{bmatrix} \alpha \text{diag}(\vec{e})D21 + \alpha \text{diag}(\vec{g})D1 & -\alpha \text{diag}(\vec{e})D1 - \alpha \text{diag}(\vec{g})I1 \\ \alpha R \text{diag}(\vec{s})D1 & \text{diag}(\vec{e})D22 + \text{diag}(\vec{q})D1 - \alpha R \text{diag}(\vec{s})I1 \end{bmatrix} \begin{bmatrix} \vec{Y} \\ \vec{F} \end{bmatrix} = -\omega^2 \begin{bmatrix} \vec{Y} \\ \vec{F} \end{bmatrix}$$

where  $\text{diag}(\vec{e})$  makes a diagonal matrix out of the vector  $\vec{e}$  and similarly for the other diag terms. Let us call the matrix on the left side of (19)  $B_m$ . Then all that needs to be done is to find the eigenvalues (and eigenvectors if desired) of  $B_m$ . This is routine and simple to implement in Scilab as will be shown in Appendix A.

What remains is to be specific about the boundary vectors  $By_1, By_N, Bf_1, Bf_N$ . These are the only things, besides the coefficient vectors that need to be changed from one case to another. To be specific I show the case of the cantilever beam. The boundary conditions for the cantilever beam are  $Y(0)=Y_1=0$ ,  $F(0)=F_1=0$ ,  $F'(1)=F'_N=0$  and  $Y'(1)-F(1)=Y'_N-F_N=0$ .

The boundary conditions at x=0 are easily implemented by simply making

$$(20a) \quad By_1=[1/\alpha h^2, 0, 0, \dots, 0] \text{ and } Bf_1=[1/h^2, 0, 0, \dots, 0].$$

What this does is make the first equations at j=1 and the (N+1)<sup>th</sup>, when used in (19),

$$Y_1 = -\beta^4 Y_1 \text{ and } F_1 = -\beta^4 F_1.$$

This may seem a bit artificial, but it forces  $Y_1$  and  $F_1$  to be zero for any eigenvalue  $\beta \neq 1$ . If any of the eigenvalues  $\beta=1$  (which is in fact what happens to two eigenvalues), then we disregard them and their corresponding eigenvectors. These spurious modes are due to this implementation of the BCs and are easily identified because they remain exactly 1 regardless

of the step size  $h$  (the number of grid points  $N$ ) used.

The BCs at  $x=1$  are implemented a bit differently. Notice that equations (11) applied at the point  $x=1$  reduce to

$$(21a) \quad \alpha E \frac{d^2 Y}{dx^2} = -\beta^4 Y$$

$$(21b) \quad E \frac{d^2 F}{dx^2} = -\beta^4 F$$

in view of the two BCs. These two equations are then implemented using the 2<sup>nd</sup> derivative evaluated at  $x_N$  by a cubic interpolation polynomial through the points  $x_{N-3}$ ,  $x_{N-2}$ ,  $x_{N-1}$ ,  $x_N$ . This yields for the boundary terms  $B_{y_N}$  and  $B_{f_N}$

$$(22) \quad B_{y_N} \text{ and } B_{f_N} = [0, 0, \dots, 0, -1, 4, -5, 2] \text{ .}$$

This completes the FD matrix equation (19).

I also use exactly the set of equations (19) to do static deflection simply by changing the right hand side eigenvalue vector to whatever discrete force I am interested in. This will also be illustrated in the applications section.

### 3.2: An Exact Solution:

It is always useful to have an exact solution available to test numerical implementations against. In equations (10) or (11), a uniform prismatic beam has  $E$ ,  $A$  and  $I$  constant equal to 1. Hence the derivatives of these properties are zero and the equations reduce to

$$(23a) \quad \alpha \frac{d^2 Y}{dx^2} - \alpha \frac{dF}{dx} = -\beta^4 Y$$

$$(23b) \quad \frac{d^2 F}{dx^2} + \alpha R \frac{dY}{dx} - \alpha R F = -\beta^4 F \text{ .}$$

These are linear equations with constant coefficients and thus have solutions of the form

$$(24) \quad \begin{bmatrix} Y \\ F \end{bmatrix} \propto \vec{C} \exp(\lambda x)$$

where  $\vec{C}$  is a two component complex vector to be determined, along with  $\lambda$ , by the differential equations (23).

By substituting (24) into equations (23) the eigenvalue problem

$$(25) \quad \begin{bmatrix} \alpha\lambda^2 + \beta^4 & -\alpha\lambda \\ \alpha R\lambda & \lambda^2 - \alpha R + \beta^4 \end{bmatrix} \vec{C} = A(\lambda, \beta) \vec{C} = 0$$

results.

System (25) only has solutions for which the determinant of the matrix is zero. For each value of  $\beta = \sqrt{\omega}$  we must find the four roots  $\lambda$  and the corresponding four eigenvectors  $\vec{C}$ . Then the four corresponding functions (24) are solutions of the differential equations (23). But such solutions must satisfy the appropriate BCs of the beam under consideration. These boundary conditions will determine which of the solutions from (24) and (25) satisfy all the conditions of the problem. Below is the example of the cantilever beam.

Example: The Uniform Prismatic Cantilever Beam:

Assume for a specified value of  $\beta$  equation (25) is solved for the four roots  $\lambda_1, \lambda_2, \lambda_3, \lambda_4$  and the corresponding eigenvectors  $\vec{C}_1, \vec{C}_2, \vec{C}_3, \vec{C}_4$ . Each of these eigenvalues and eigenvectors is a function of  $\beta$ . The BCs for the cantilever beam are  $y(0)=0$ ,  $f(0)=0$ ,  $f'(1)=0$  and  $y'(1)-f(1)=0$ . The solution of the differential equation is

$$(26) \quad \begin{bmatrix} Y \\ F \end{bmatrix} = B_1 \vec{C}_1 \exp(\lambda_1 x) + B_2 \vec{C}_2 \exp(\lambda_2 x) + B_3 \vec{C}_3 \exp(\lambda_3 x) + B_4 \vec{C}_4 \exp(\lambda_4 x)$$

where  $B_1, B_2, B_3, B_4$  are complex constants that are determined by the BCs. By applying the BCs to (26) one obtains the set of equations

$$(27a) \quad Y(0) = B_1 C_{1,1} + B_2 C_{1,2} + B_3 C_{1,3} + B_4 C_{1,4} = 0$$

$$(27b) \quad F(0) = B_1 C_{2,1} + B_2 C_{2,2} + B_3 C_{2,3} + B_4 C_{2,4} = 0$$

$$(27c) \quad F'(1) = B_1 \lambda_1 C_{2,1} \exp(\lambda_1) + B_2 \lambda_2 C_{2,2} \exp(\lambda_2) + B_3 \lambda_3 C_{2,3} \exp(\lambda_3) + B_4 \lambda_4 C_{2,4} \exp(\lambda_4) = 0$$

$$(27d) \quad Y'(1) - F(1) = B_1 (\lambda_1 C_{1,1} - C_{2,1}) \exp(\lambda_1) + B_2 (\lambda_2 C_{1,2} - C_{2,2}) \exp(\lambda_2) + B_3 (\lambda_3 C_{1,3} - C_{2,3}) \exp(\lambda_3) + B_4 (\lambda_4 C_{1,4} - C_{2,4}) \exp(\lambda_4) = 0$$

where the vectors  $\vec{C}_j = [C_{j,1}, C_{j,2}, C_{j,3}, C_{j,4}]$  for  $j=1$  and  $2$ . Remember all the  $C$ 's and all the  $\lambda$ 's are functions of  $\beta$ . Thus in order for there to be nontrivial values of the  $B$ 's that satisfy these equations the determinant of the coefficient matrix of this system of equations must be zero. That is, we must find the value of  $\beta$  so that

$$(28) \quad \text{the } \det(Q(\beta)) = 0 \text{ where } Q(\beta) =$$

$$\begin{bmatrix} C_{1,1} & C_{1,2} & C_{1,3} & C_{1,4} \\ C_{2,1} & C_{2,2} & C_{2,3} & C_{2,4} \\ \lambda_1 C_{2,1} \exp(\lambda_1) & \lambda_2 C_{2,2} \exp(\lambda_2) & \lambda_3 C_{2,3} \exp(\lambda_3) & \lambda_4 C_{2,4} \exp(\lambda_4) \\ (\lambda_1 C_{1,1} - C_{2,1}) \exp(\lambda_1) & (\lambda_2 C_{1,2} - C_{2,2}) \exp(\lambda_2) & (\lambda_3 C_{1,3} - C_{2,3}) \exp(\lambda_3) & (\lambda_4 C_{1,4} - C_{2,4}) \exp(\lambda_4) \end{bmatrix} \cdot$$

Technically, one should reduce the equations (27) to real and imaginary parts and choose one or the other to be the vector  $\begin{bmatrix} Y \\ F \end{bmatrix}$  and then solve for  $\beta$  and the appropriate B constants. This is what one would do by hand calculations. We are only interested in the ultimate eigenvalues  $\beta$  that solves our problem and having available the mathematical software Scilab (or Matlab or similar software) that does complex arithmetic automatically, we just simply search  $\det(|Q(\beta)|)$  for those values of  $\beta$  that makes this determinant equal to zero. Briefly, the algorithm consists of the following steps:

Choose a large sequence of values of  $\beta$   
 for each  $\beta$  compute the eigenvalues  $\lambda_1, \lambda_2, \lambda_3, \lambda_4$   
 and eigenvectors  $\vec{C}_1, \vec{C}_2, \vec{C}_3, \vec{C}_4$  by solving  
 equation (20) (easily in Scilab) and  
 then compute  $\det(|Q(\beta)|)$ .

Plot  $\log(\det(|Q(\beta)|))$ .

At the zeros of  $\det(|Q(\beta)|)$  its log plot should dip steeply towards  $-\infty$ . this will not happen, but one will see very sharp downwards spikes at the eigenvalues of  $\beta$ .

This procedure will be demonstrated in the applications. Appendix B gives the Scilab implementation of this method.

#### 4. Application to a Cantilever Beam:

*Example 1: A uniform prismatic beam:*

First I test my FD eigenvalue solver (19) against the exact eigenvalue solver (28) by considering a uniform prismatic cantilever beam for various aspect ratio (dimensional  $L/h$  or dimensionless  $R$ ) beams. These beams all have rectangular cross section of width  $w$  and height  $h$ . One can easily compute that  $R=12/h^2$  and from the literature  $\kappa=0.87$  so that with the Poisson ratio  $\nu=0.33$ , we have  $\alpha \approx 1/3$ .

The Figures 2a-e below plot  $\log(|\det(Q(\beta))|)$  vs  $\beta$  for five values of  $R$ . On these plot the points denoted by the circles o and connected by stems are the eigenvalues produced by solving the FD equations (19). I used  $N=401$  points in the FD solver (19) so that I'm pretty

confident that the first 20 or so eigenvalues produced are accurate to at least 4 or 5 significant figures, especially the first 10 or so. Of course the plot is probably only accurate to 2 or three figures. I plotted about 5000 points of the values of  $\log(|\det(Q(\beta))|)$  to make sure I capture the dips where  $\det(Q)$  is near zero. This computation goes very quickly. In any case, these plots are more than adequate to verify that my FD method (19) does obtain accurate eigenvalues of vibrating T-beams.

Figure 2a

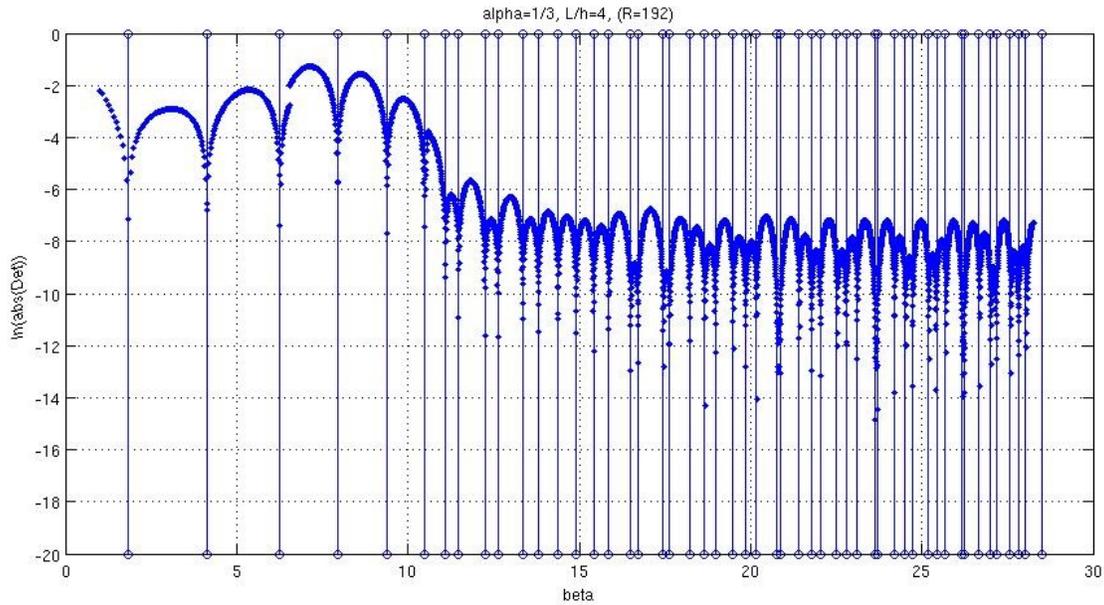


Figure 2b

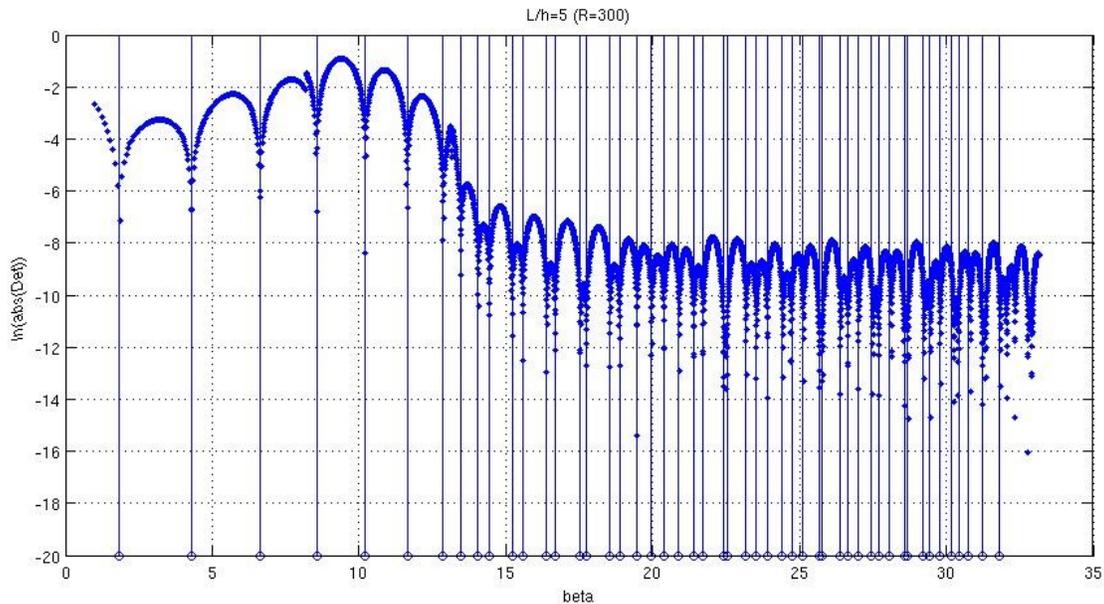


Figure 2c

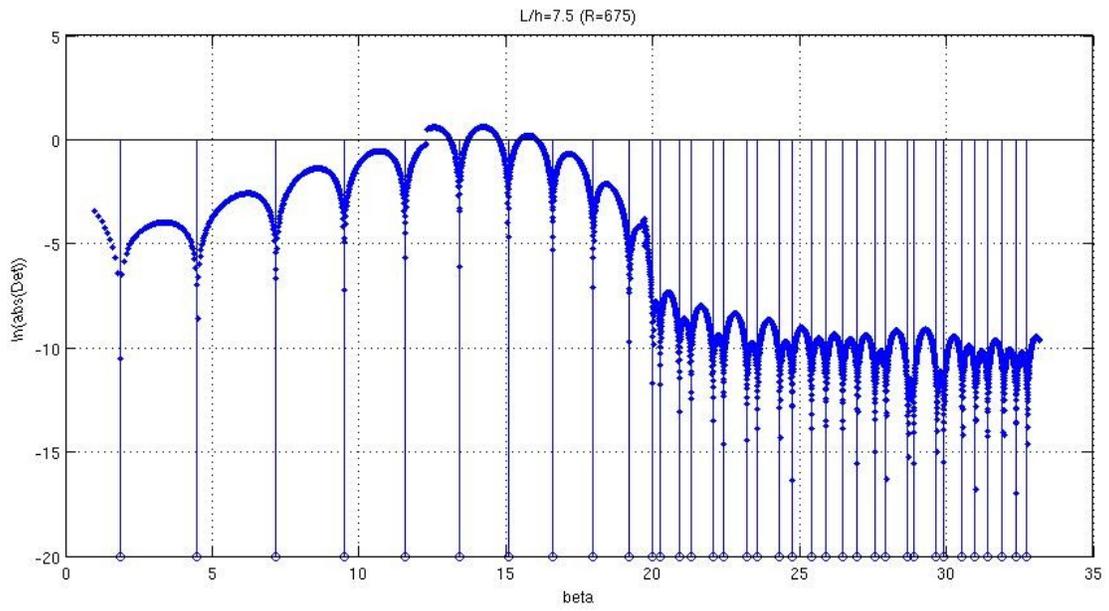


Figure 2d

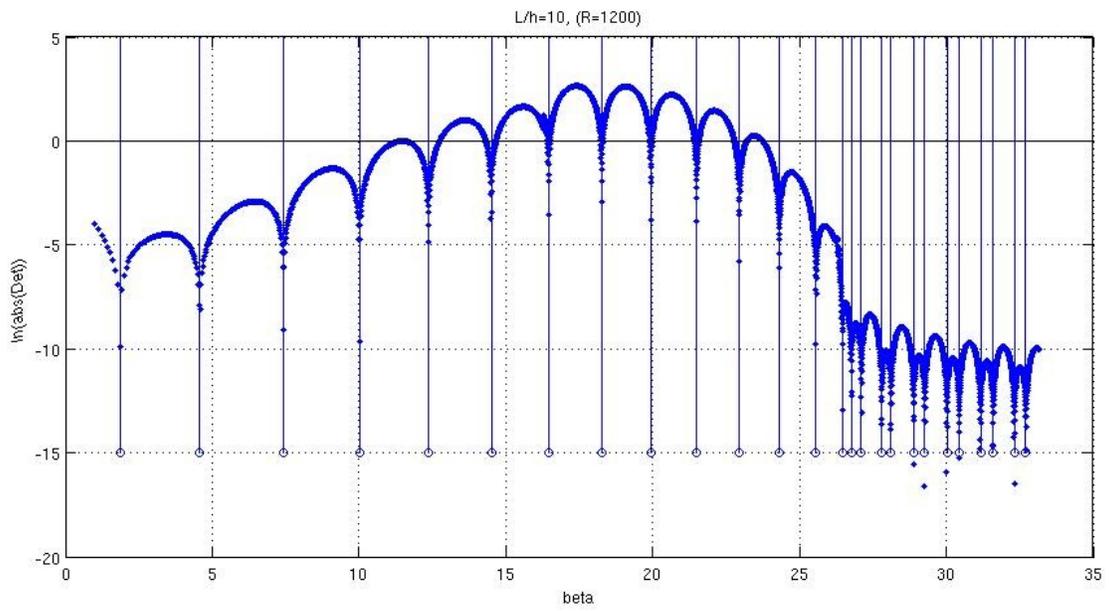
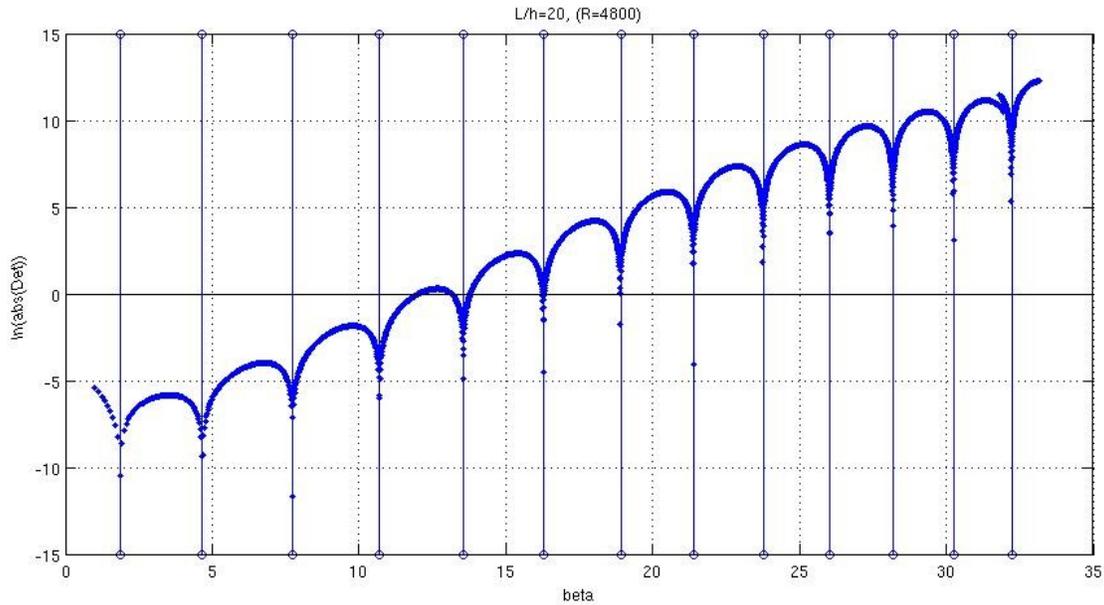


Figure 2e



It is interesting to note that plots of  $\log(|\det(Q)|)$  all have a discontinuity at a certain value of  $\beta$ , which moves to lower values as  $R$  decreases. This is due to a phase change in the eigenvalue relation (the phase function) (25). It is quite remarkable how well the large number of FD eigenvalues match the exact eigenvalues. The FD method will always produce  $N$  eigenvalues, but only about the first 10% of these will be accurate approximations of exact eigenvalues. The T-beam equations are self adjoint so that the eigenvalues are real. Furthermore the equation system is positive definite so that all the eigenvalues are positive and hence are bounded below. The FD method will approximate a finite number ( $N$ ) of the lower eigenvalues (those whose corresponding eigenfunctions can be resolved by the  $N$  finite values of the grid). Of course only a few of these, the lowest modes, whose eigenfunctions have the least number of oscillations, will be accurate.

Another interesting item revealed in the above figures is that T-beam theory predicts that at some point in the eigenvalue spectrum eigenvalues begin to pair up, as can be clearly seen at the lower values of  $R$  ( $R < 4800$ ). Actually this will occur at all finite values of  $R$ , but the place in terms of  $\beta$  where such pairing occurs increases with  $R$ . This pairing of eigenvalues always occurs after the discontinuity in the  $\log(|\det(Q)|)$  curves. The pairing corresponds to flexural deformation dominated vibratory modes and shear angle dominated vibratory modes of similar wave length (distance between nodes) proportional to  $1/\beta$ .

Finally, we note that the eigenvalue spectrum of the EB beam will be very similar (almost identical for the lower eigenvalues) as the  $R=4800$  ( $L/h=20$ ) T-beam spectrum, except that the EB spectrum has no discontinuities in  $\log(|\det(Q)|)$  and proceeds to  $\beta \rightarrow \infty$  uniformly because it is easy to show that for a uniform prismatic cantilever beam the eigenvalues are accurately  $\beta \approx (2n-1)\pi/2$  for  $n > 6$ .

In conclusion I am confident that my FD method can accurately capture the lower (depending on N) eigenvalues of a T-beam.

*Example 2: A Prismatic Cantilever Beam with Variable Elastic Parameter E(x):*

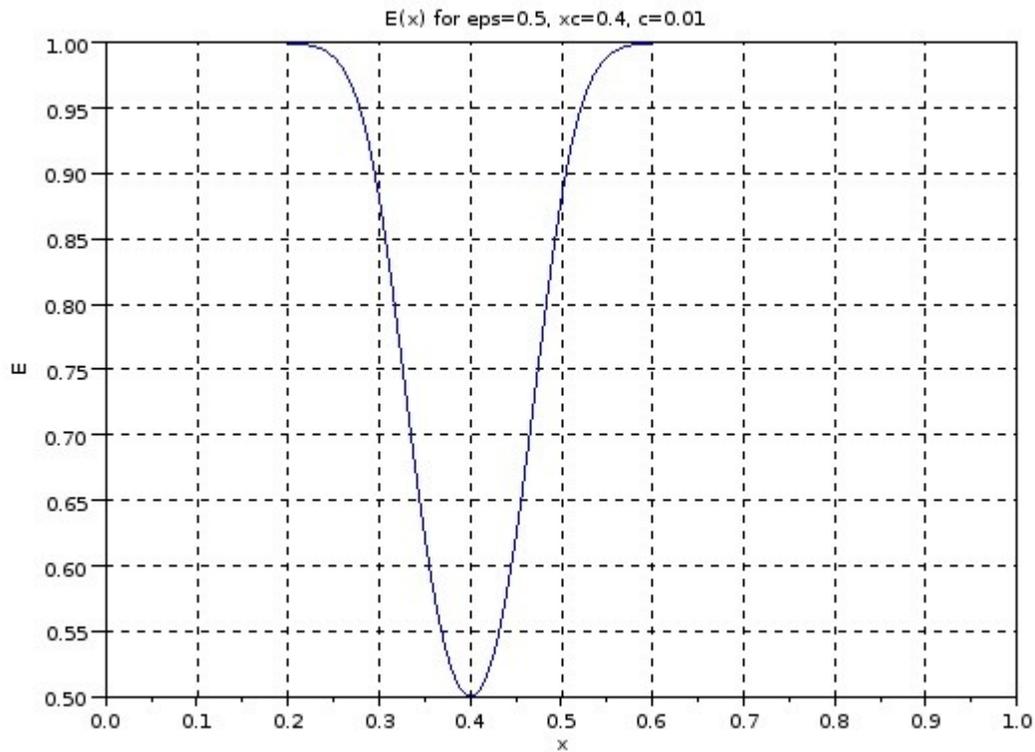
Say we have a rectangular cross section beam that has suffered some kind of internal weakening. For example, say a part of the beam was exposed to an intense fire that left its shape nominally the same, but changed its material property E, and hence its stiffness. Before reloading this beam we would like to know to what extent the weakening has occurred. Perhaps one way to test the beam in situ nondestructively is to tap it to set up vibrations in it. The response of the free vibrations can be accurately measured. That is, we can measure the *frequencies (eigenvalues)* of the free vibrations. By comparing the measured frequencies to the calculated frequencies of the undamaged beam, it may be possible to determine the location and extent of beam weakening. Thereby determining its suitability for continued use. So in this section I examine the effects of a variable elastic property E(x) on the vibrations of cantilever beams.

Let's assume that the variability of E can be characterized by the three parameter shape function

$$(28) \quad E(x) = 1 - \epsilon \left( 1 - \tanh \left( \frac{(x - x_c)^2}{c} \right) \right)$$

where  $\epsilon$  is the magnitude of the decrease of E(x),  $x_c$  is the location of the maximum decrease and c is a measure of the width in x of the weakening. The smaller the value of c, the smaller the width of the weakening. Figure 3 below is a plot of an example E(x)

Figure 3



Function (28) has been programmed in the scilab function  $Tbeamf(x,q)$ . I show two sample calculations below.

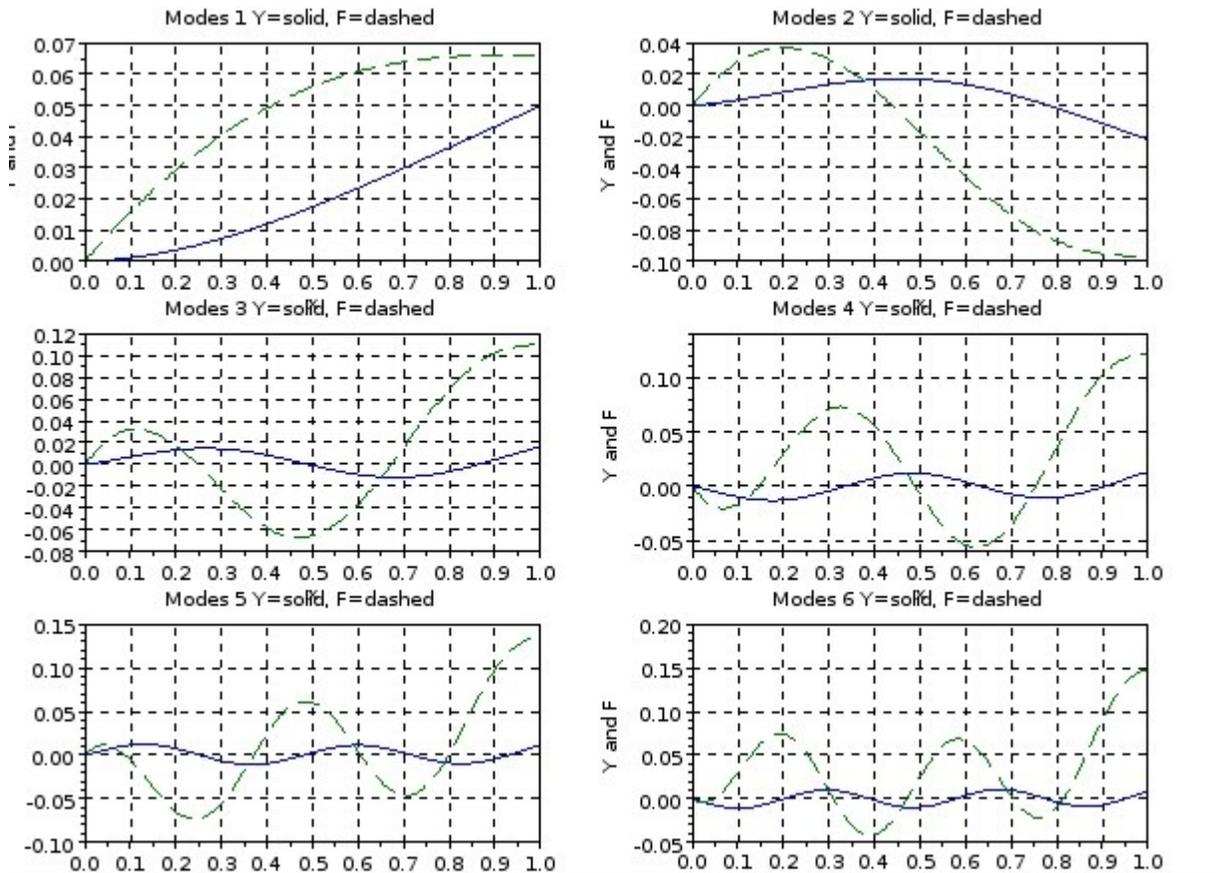
*Sample 1:* Uniform prismatic beam with  $\alpha=1/3$ ,  $R=300$  ( $L/h=5$ ),  $\varepsilon=0$ .  $N=301$  for this calculation.

Table 1: First 10 EVs for  $\alpha=1/3$ ,  $R=300$  ( $L/h=5$ ),  $\varepsilon=0$ .

EV#	$\beta$
1	1.8475
2	4.2952
3	6.6355
4	8.5588
5	10.214
6	11.643
7	12.871
8	13.467
9	14.059
10	14.443

I note that the 1st six eigenvalues in this case agree to  $\pm 1$  in the 4th significant figure with a previous calculation using a shooting method reported in Reference [2].

Figure 4: The Eigenfunctions for the first 6 Eigenvalues in Table 1:



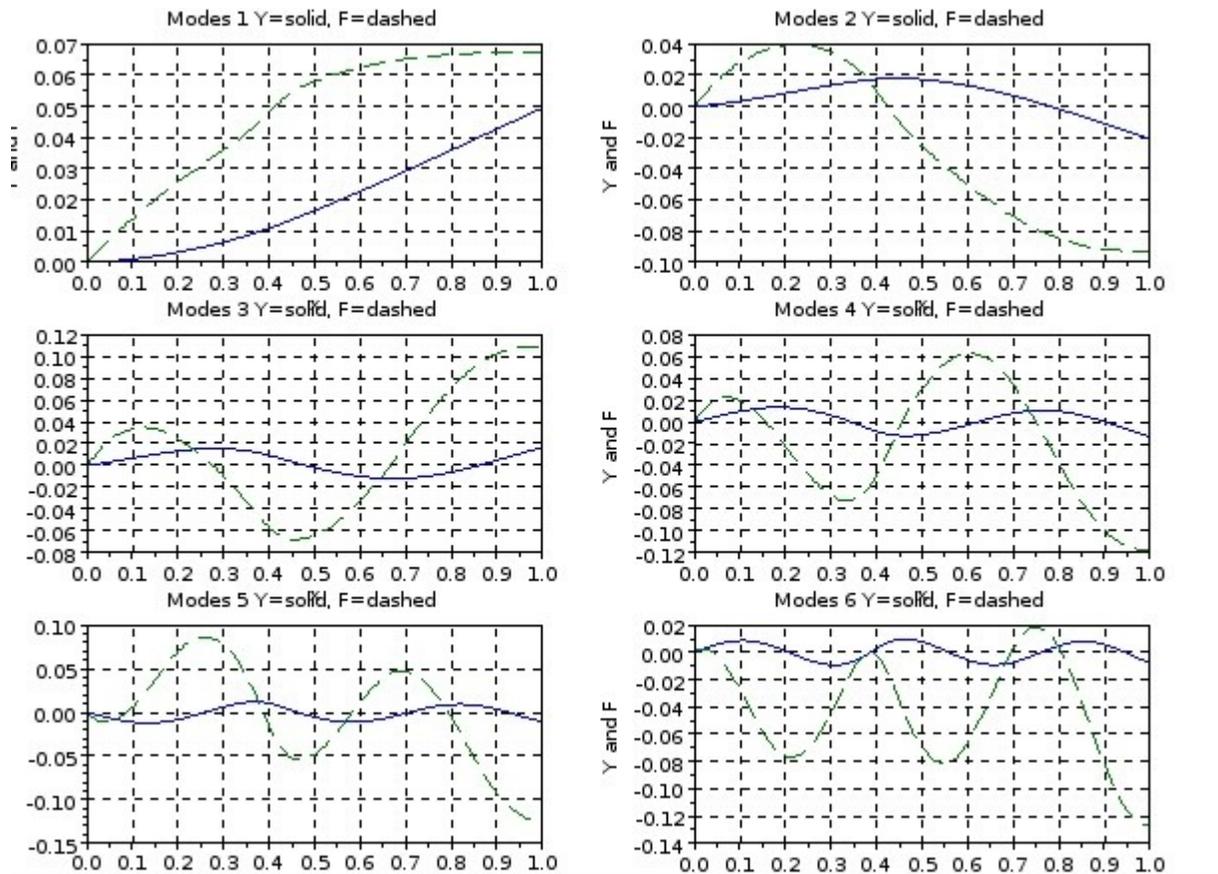
Sample 2: The same beam as Sample 1 except with weakening at  $x_c=0.4$ ,  $\epsilon=0.5$ ,  $c=0.01$  as in Figure 3.  $N=301$ .

Table 3: First 10 EVs for  $\alpha=1/3$ ,  $R=300$  ( $L/h=5$ ),  $x_c=0.4$ ,  $\epsilon=0.5$ ,  $c=0.01$

EV#	$\beta$
1	1.7990
2	4.1486
3	6.5043
4	8.3468
5	9.9392
6	11.243
7	12.580

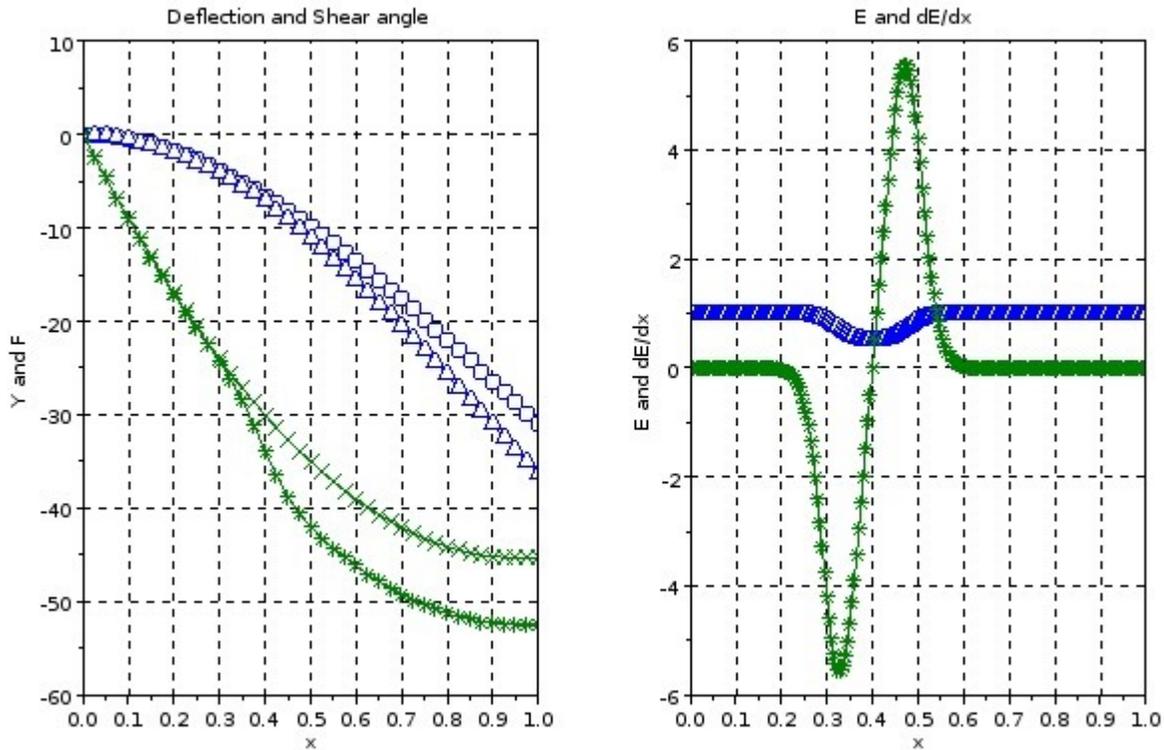
8	13.018
9	13.577
10	14.210

Figure 5: The Eigenfunctions for the first 6 Eigenvalues in Table 2:



*Sample 3:* As a matter of interest I have computed the static deflection due to a point load of dimensionless magnitude 100, located at  $x=0.95$  for the above beam with and without the damage. The deflection  $Y(x)$  and the shear angle  $F(x)$  are shown for both cases in Figure 6 below.

Figure 6: Static deflection of beam with  $R=300$  ( $L/h=5$ )  
 Beam loaded by point load  $P=-100$  at  $x=0.95$ .  
 $\Delta\Delta\Delta Y$ ,  $*** F$  for  $x_c=0.4$ ,  $\varepsilon=0.5$ ,  $c=0.01$ . The right  
 figure shows the  $E(x)$  and  $dE/dx$  distributions for this case.  
 $\square\square\square Y$ ,  $xxx F$  for intact beam,  $\varepsilon=0.0$ ,  $E=1$ .



Not surprisingly, the deflection is greater for the weakened beam. It is interesting, and physically not unexpected, that the shear angle steepens considerably in the weakened region. The Appendix B giving the FD program will also explain how static deflection is calculated and selected to operate the program.

*Example 3: Uniformly Elastic, Tapered Beams:*

As before we consider a rectangular cross section beam with  $\alpha=1/3$ . For such a beam recall that the dimensional cross section area is  $A_0A(x)$ , second moment is  $I_0I(x)$  and elastic constant is  $E_0E(x)$  where the  $A(x)$ ,  $I(x)$  and  $E(x)$  are simply dimensionless shape functions that are each equal to 1 at the location  $x_0$ . Now

$$(29) \quad A_0 = wh \text{ and } I_0 = \frac{1}{12} wh^3$$

for a rectangular cross section beam, where  $w$  is the dimensional width and  $h$  is the dimensional height of the cross section. For the shape functions  $w$  and  $h$  are dimensionless.

There are two ways the beam can be tapered. One way is for  $w$  to vary and the other is for  $h$  to vary. Say  $w$  alone varies. Then both  $A(x)$  and  $I(x)$  would vary in exactly the same way. The reference area is at  $x=0$ , and the variation is dimensionless  $w(x)$ , the both  $A(x)$  and  $I(x)$  would vary like  $w(x)$  with  $w(0)=1$ .

On the other hand if  $w$  is fixed and  $h$  varies, then dimensionless  $h(0)=1$  and  $A(x)=h(x)$  and  $I(x)=h^3(x)$ .

For this example let us take a vertically tapered beam in which  $h$  varies. Say the beam tapers linearly from  $h=1$  at  $x=0$  to  $h=0.5$  at  $x=1$ . Then

$$(30) \quad h(x)=1-0.5x$$

so that

$$(31) \quad A(x)=1-0.5x \quad \text{and} \quad I(x)=(1-0.5x)^3 .$$

The five coefficient vectors (18), which I repeat here for convenience,

$$\vec{e}=[E_j], \quad \vec{g}=\left[\left(\frac{(EA)'}{A}\right)_j\right], \quad \vec{q}=\left[\left(\frac{(EI)'}{I}\right)_j\right], \quad \vec{s}=\left[\left(\frac{(EA)}{I}\right)_j\right]$$

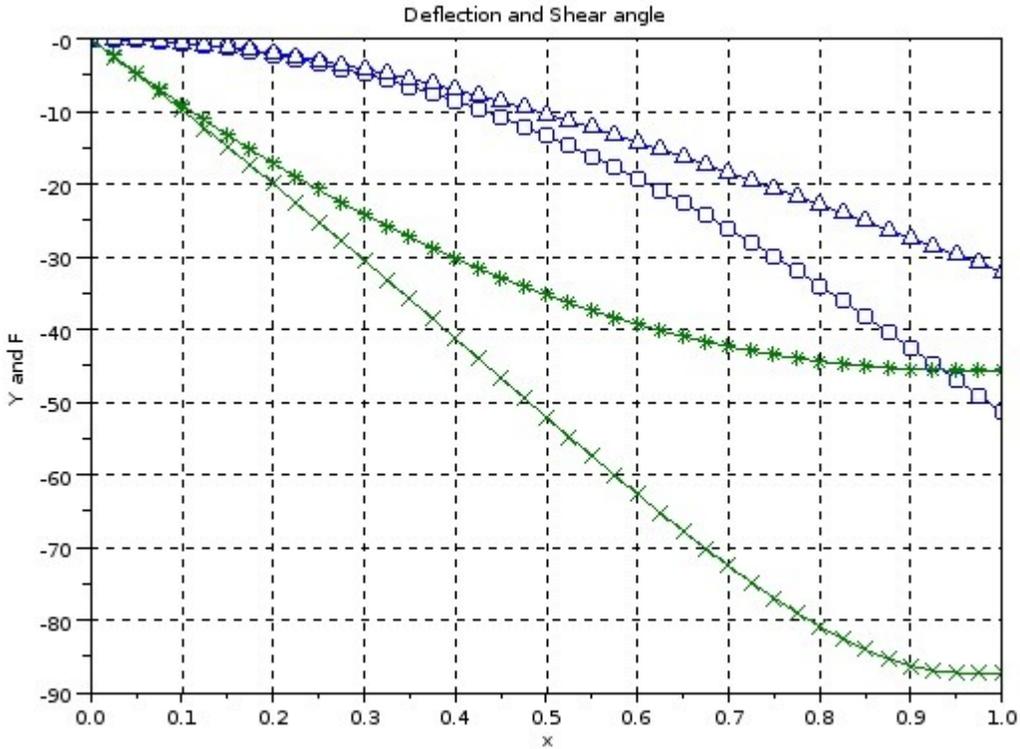
become

$$(32) \quad E(x)=1, \quad \frac{(EA)'}{A}=\frac{-0.5}{1-0.5x}, \quad \frac{(EI)'}{I}=\frac{-1.5}{1-0.5x}, \quad \frac{EA}{I}=\frac{1}{(1-0.5x)^2} .$$

In the Scilab program there is an input string variable *Ec*ase which will select which type of beam variation to use. In the previous examples the variation was only for  $E$  and the *Ec*ase selector was put in as *Ec*ase= 'Evar'. We programed the functions (32) *Tbeamfct* as *Ec*ase='htap'.

First I show the effects of this taper on the  $L/h=5$  beam under the static load of Sample 3 above. This is compared in Figure 7 below to a the equivalent prismatic beam. As expected the tapered beam deflects considerably more under the given load. This is hardly surprising. A little more interesting is the largely linear behavior of the shear angle  $F$ . Explanation?

Figure 7.  $\square\square\square$  Y and  $\times\times\times$  F tapered beam  
 $\Delta\Delta\Delta$  Y and  $\ast\ast\ast$  F prismatic beam. (L/h=5)

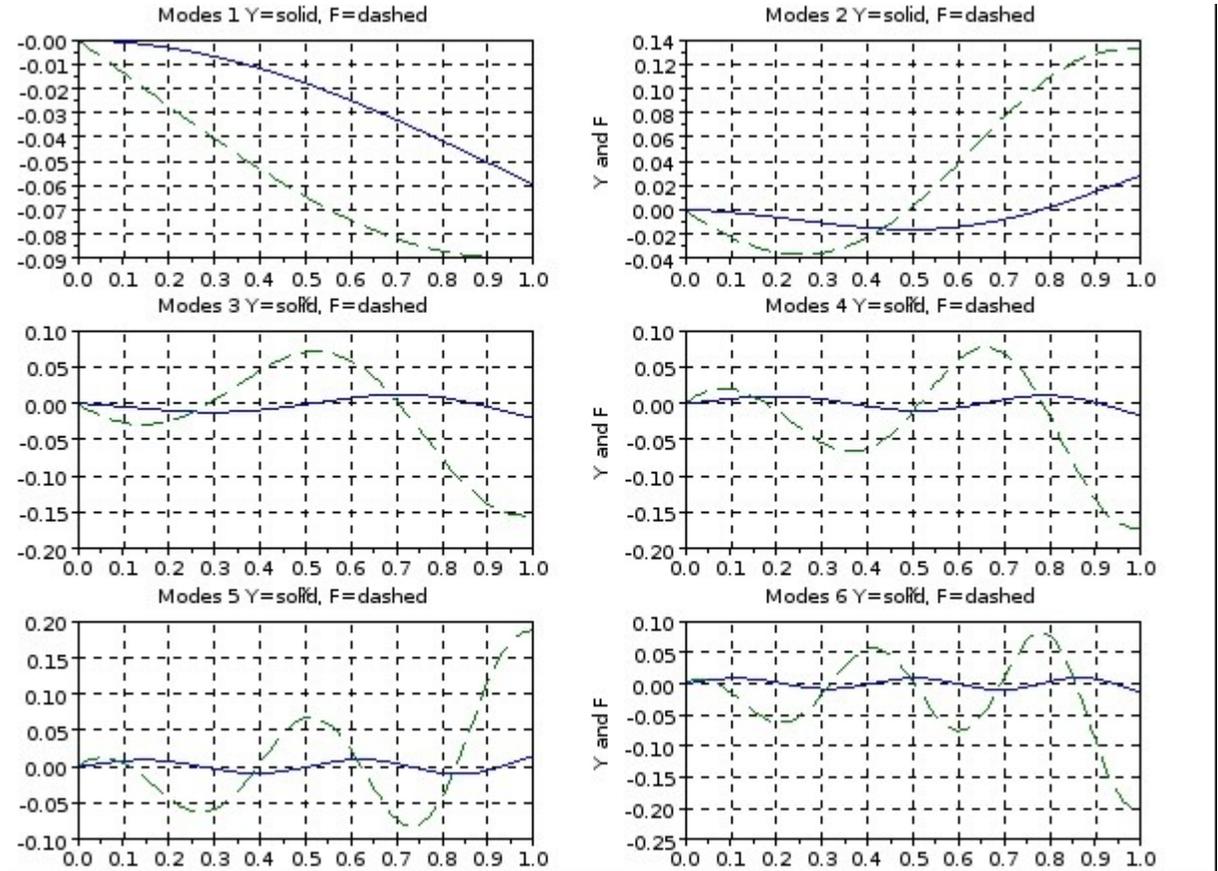


Now I compute the EVs for the tapered beam. The eigenvalues are listed in Table 4 below and the eigenfunctions are plotted in Figure 8.

Table 4: First 10 Eigenvalues of a Tapered beam  
 $h(x)=(1-cx)$ , L/h=5, (R=300); L/h=20, (R=4800).

EV#	$\beta$ , R=300	$\beta$ , R=4800
1	1.9321	1.9556
2	4.0477	4.2700
3	6.1469	6.8266
4	8.0013	9.3777
5	9.6542	11.891
6	11.139	14.347
7	12.486	16.736
8	13.714	19.052
9	14.796	21.293

Figure 8: Eigenfunctions corresponding to the eigenvalues in Table 4 for  $R=300$ .



I'm not showing the eigenfunctions for  $R=4800$  because they do not differ in shape very much from those at  $R=300$ . The case with  $R=4800$ ,  $L/h=20$ , is essentially an EB beam as far as the 1st 10 eigenvalues are concerned to at least 4 decimal places. But notice the eigenvalue difference between the two cases. The taper has a pretty drastic effect on the higher eigenvalues. This should be expected.

*Example 4: A beam notched on top and bottom:*

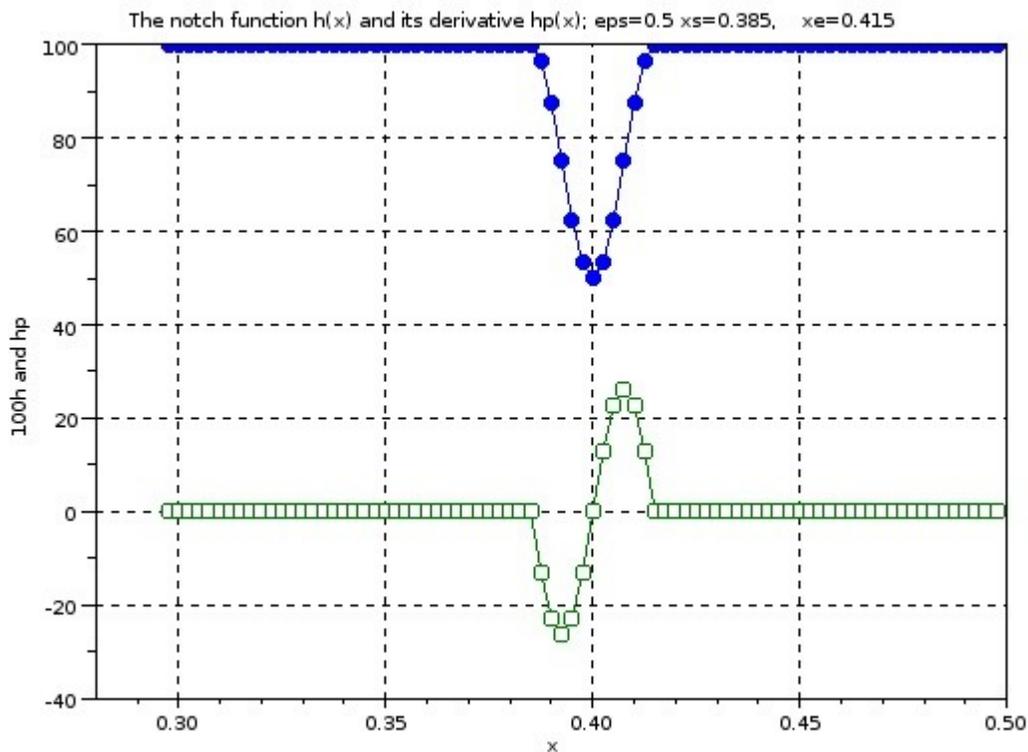
Let us consider a beam that has a sharp notched symmetrical on top and bottom. This might simulate a severe crack in the beam. The crack is sufficiently open (wide lengthwise) and does not close during the vibration. This notch or crack can be approximated by the function

$$(33) \quad h(x) = 1 - \epsilon \begin{cases} 0 & \text{for } x < x_s \\ \left( \sin\left(\frac{\pi}{2} \frac{x - x_s}{x_e - x_s}\right) \right)^2 & \text{for } x_s \leq x < x_e \\ 0 & \text{for } x_e < x \end{cases} .$$

The derivative of this function is

$$(34) \quad hp(x) = 1 - \epsilon \begin{cases} 0 & \text{for } x < x_s \\ \frac{\pi}{x_e - x_s} \sin\left(\frac{\pi}{2} \frac{x - x_s}{x_e - x_s}\right) \cos\left(\frac{\pi}{2} \frac{x - x_s}{x_e - x_s}\right) & \text{for } x_s \leq x \leq x_e \\ 0 & \text{for } x_e < x \end{cases} .$$

The functions  $h(x)$  and  $hp(x)$  are illustrated in Figure 9 below for  $\epsilon=0.5$ ,  $x_s=0.39$ ,  $x_e=0.41$ .



This particular case of a notch function has a width of  $d(x)=(x_e-x_s) = 0.02$ , so that its width compared to beam length is 2%. The above plot utilizes  $N=401$  points in  $[0,1]$  and as one can see it barely resolves this function. I'm not sure an even narrower notch would be adequately resolved in the numerical method unless I use a larger value of  $N$ . I often use  $N=501$  points,

but this only produces marginally more accuracy. It should be noted that the eigenvalue calculation is for a matrix of size  $2N$ , so that this begins to take considerable time (about 10 minutes for each case with  $N=501$ ). My computer is a small relatively slow notebook computer that runs at 1.7 Ghz. Furthermore, my numerical FD method is 2nd order accurate in step size. Thus to produce one more significant figure of accuracy, I would have to essentially double  $N$ . In conclusion I will confine the notch to the width  $=(x_e-x_s)\geq 0.02$  and  $N\leq 401$ .

I have made the calculation for this notched beam with  $\epsilon=0.5$ ,  $x_s=0.39$  and  $x_e=0.41$  for a  $L/h=10$  and  $L/h=5$  cantilever beam. I used a  $N=401$  point grid. First Figure 10 shows the deflection of the  $L/h=10$  beam with the force -100 at the point  $x=0.95$  of the beam.

Figure 10

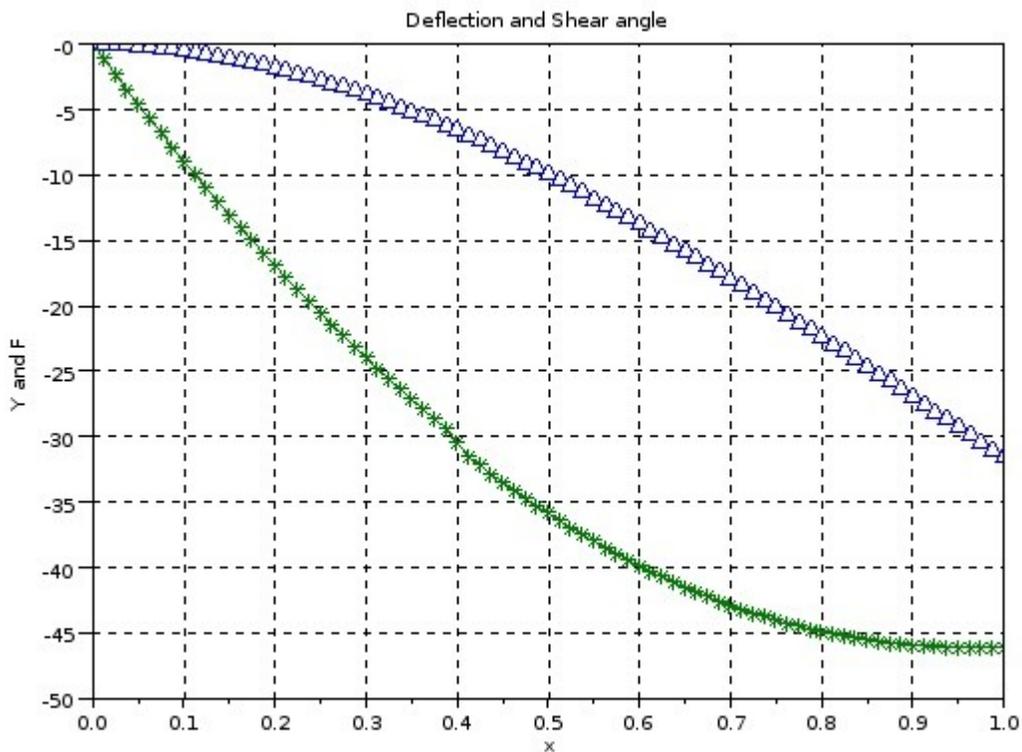


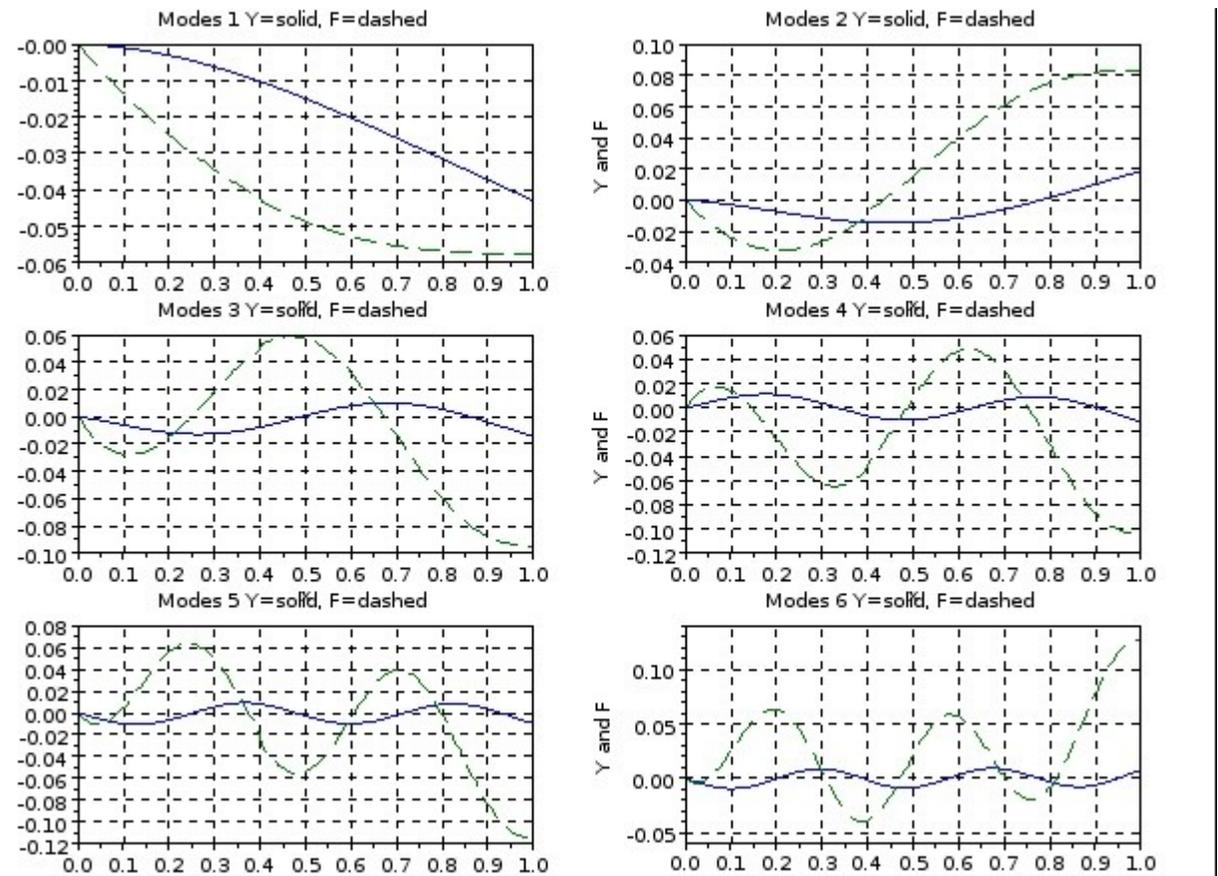
Table 5 below gives the first 10 eigenvalues of this notched beam for  $L/h=10$  and  $L/h=5$  respectively.

Table 5

EV#	$\beta$ , R1200	$\beta$ , R=300
1	1.8617	1.8379
2	4.5513	4.2794
3	7.3987	6.6192
4	9.9749	8.5368
5	12.3164	10.1907
6	14.4625	11.6207
7	16.4185	12.8681
8	18.2244	13.5082
9	19.8978	13.0711
10	21.4589	14.4319

I only show the eigenfunctions for the case R=300. The most notable feature, upon close examination of modes 5 and 6 is an almost discontinuity in the shear angle  $F$  at the point  $x=0.4$  of the notch. The eigenfunctions for the case R=1200 are similar to the prismatic R=1200 beam, as are the eigenvalues.

Figure 11: The eigenfunctions corresponding to the first 6 R=300 eigenvalues in Table 5.



### 5: Brief Concluding Remarks:

I have presented a fairly robust 2nd order finite difference method to compute many eigenvalues and eigenvectors of nonuniform, non-prismatic Timoshenko beams. This method agrees very well for the first 20-30 eigenvalues with the exact eigenvalue spectrum for uniform prismatic T-beams over a range of aspect ratios  $R$  (192 to 4800,  $L/h=2$  to  $L/h=20$ ). The many examples given here indicate the range of applicability. The Scilab programs given in the appendix allow the reader to compute static deflection and eigenvalue spectra for many different types of beams. Although all my examples are for cantilever beams, which I regard the most interesting, I think the reader should be able, on the basis of the description of the implementation given above, and a few notes in the Appendixes for the programs, implement other supports for the beam.

I add in passing that I have worked with Professor Panos Charalambides of University of Maryland Baltimore County and his Graduate student Valery in a project to compare EB beam theory, T-beam theory and exact 2D elasticity beam theory (by FEM) prediction of eigenvalues of prismatic beams with  $E(x)$  variation. The agreement of my T-beam theory results with the FEM results over a range of  $E(x)$  distributions and for  $L/h=5$  and  $10$  are astounding (within 0.1% across the board).

## 6: References:

1. Graff, Karl F., **Wave Motion in Elastic Solids**, Dover publications, (1991).
2. Von Kerczek, C. **Free Vibration of Nonuniform Timoshenko beams I**, [www.cvkjournal.wordpress.com](http://www.cvkjournal.wordpress.com), (2011).

## Appendix A: The Scilab FD Program:

Overall the Scilab program is self explanatory. Anyone with some familiarity with Matlab or Octave could easily read the program and with the aid of the above descriptions figure out how to use. I present the program here and will append some notes about it at the end of this appendix.

### Program *TbeamE.sce*

```
clear
stacksize('max')
//
// FD T-beam program for nonuniform, non-prismatic beams.
// Calculates static deflection for specified load or
// about 0.1N accurate eigenvalues and eigenvectors for N>201.
//
//-----
// 1. Input
////////// proble selectors
fn='eigv';
%fn='forc';
Ecase='hnot';
//////////
a=0; b=1; N=401; M=12;
alpha=1/3; R=300;
//outf='EVR192_5_25.dat'; If EV output file is to be made
q=[0.5,0.39,0.41]; // Parameter vector as used in function Tbeamf.
//          Check the Tbeamf function for the requirements
//          of each case.
//-----
// Auxiliary functions used
//
function [E, A, EA, EAp, EIp]=Tbeamf(x, q, Ecase)
// Note: The equations only utilize E, EI, EA, (EA)', and (EI)'in
// the combinations E, (EA)/A, (EI)/I and (EA)/A. A and I never
// appear separately.
N=length(x);
select Ecase
    case 'Evar'
xl=q(1); eps=q(2); c=q(3);
for j=1:N
th(j)=tanh(((x(j)-xl)^2)/c);
E(j)=1-eps*(1-th(j));
Ep(j)=(2*eps/c)*(x(j)-xl)*(1-th(j)^2);
end
EA=E;
EIp=Ep;
```

```

EAp=Ep;
A=ones(1,N);
case 'htap'
    c=q(1)
E=ones(1,N);
A=(1-c*x);
EAp=-c./(1-c*x);
Elp=-3*c./(1-c*x);
EA=(1)./(1-c*x).^2);
case 'hnot'
E=ones(1,N);
eps=q(1);
xs=q(2);
xe=q(3);
dx=(xe-xs);
for j=1:N
    if x(j)<xs then
        h(j)=1; hp(j)=0;
    elseif x(j)<=xe
        h(j)=1-eps*(sin(%pi*(x(j)-xs)/dx))^2;
        hp(j)=-eps*(sin(%pi*(x(j)-xs)/dx))*(cos(%pi*(x(j)-xs)/dx));
        hp(j)=hp(j)*%pi/dx;
    else
        h(j)=1; hp(j)=0;
    end
end
A=h;
EA=(1)/h.^2;
EAp=hp./h;
Elp=3*hp./h;
end
endfunction
//
function d=delta(x, xs)
N=length(x);
for j=2:N-1
    if x(j-1)<=xs & xs<x(j)
        d(j)=2/(x(j+1)-x(j-1));
    else
        d(j)=0;
    end
end
d(1)=0; d(N)=0;
endfunction
//-----
//
// 2. Start of main calculation
//
x=linspace(a,b,N); h=x(2)-x(1); hs=h*h; ht=2*h; h2=h/2;
N2=2*N;
//
// a. Force distribution for static deflection problem
//
if fn=='forc'
    f1=0; f2=100; xs1=0.5; xs2=0.95;

```

```

    p=hs*(f2*delta(x,xs2)+f1*delta(x,xs1))/R;
    P=[p',zeros(1,N)'];
else
end
//
// b. Setup of the differential equations matrix
//
[E,Aa,EA,EAp,Elp]=Tbeamf(x,q,Ecase);
EM=diag(E); EAM=diag(EA);
ElpM=diag(Elp); EApM=diag(EAp);
// BC implementation
EMa=EM; EMa(1,1)=1/alpha;
EAM(1,1)=0; EApM(1,1)=0;
ElpM(1,1)=0;
D1=zeros(N,N); D2=zeros(N,N);
//
for j=2:N-1
    D1(j,j-1)=-1;
    D1(j,j+1)=1;
    D2(j,j-1)=1;
    D2(j,j)=-2;
    D2(j,j+1)=1;
end
I1=eye(N,N);
I1(1,1)=0; I1(N,N)=0;
// BC implementation
By1=[1,zeros(1,N-1)];
Bf1=[1,zeros(1,N-1)];
ByN=[zeros(1,N-4),-1,4,-5,2];
BfN=[zeros(1,N-4),-1,4,-5,2];
D2(1,:)=By1;
D2(N,:)=ByN;
//
// Assembly of FD equation matrix
Bg=[alpha*EMa*D2+h2*alpha*EApM*D1,-h2*alpha*EMa*D1-hs*alpha*EApM*I1;...
    h2*alpha*R*EAM*D1,EM*D2+h2*ElpM*D1-hs*alpha*R*EAM*I1];
//
// c. Solve the eigenvalue problem
//
if fn=='eigv'
//Ev=spec(Bg);
[Vv,Ev]=spec(Bg);
k=0;
for j=1:N2
    if abs(Ev(j,j))~=1
        k=k+1;
        Em(k)=(abs(Ev(j,j))*R/hs)^(1/4);
    end
end
NM=length(Em);
[evs,kx]=gsort(Em,'g','i');
evso=evs(1:M);
disp(evso)
//write(outf,evso,'f10.6');
for m=1:6

```

```

k=0;
for j=1:2:N
    jj=j+N;
    k=k+1;
    Wv(k,m)=Vv(j,kx(m));
    Uv(k,m)=Vv(jj,kx(m));
    y(k)=x(j);
end
end
subplot(3,2,1)
plot(y,Wv(:,1),y,Uv(:,1),'--')
xlabel('Modes 1 Y=solid, F=dashed','x','Y and F')
xgrid
subplot(3,2,2)
plot(y,Wv(:,2),y,Uv(:,2),'--')
xlabel('Modes 2 Y=solid, F=dashed','x','Y and F')
xgrid
subplot(3,2,3)
plot(y,Wv(:,3),y,Uv(:,3),'--')
xlabel('Modes 3 Y=solid, F=dashed','x','Y and F')
xgrid
subplot(3,2,4)
plot(y,Wv(:,4),y,Uv(:,4),'--')
xlabel('Modes 4 Y=solid, F=dashed','x','Y and F')
xgrid
subplot(3,2,5)
plot(y,Wv(:,5),y,Uv(:,5),'--')
xlabel('Modes 5 Y=solid, F=dashed','x','Y and F')
xgrid
subplot(3,2,6)
plot(y,Wv(:,6),y,Uv(:,6),'--')
xlabel('Modes 6 Y=solid, F=dashed','x','Y and F')
xgrid
//dlmwrite(evdata, evs(kx(NM):kx(NM-M)))
else
//
// d. Solve the static deflection problem
//
    Aa=[Aa;ones(N,1)];
    P=P./Aa;
    z=Bg\P;
    k=0;
    for n=1:5:N
        nn=n+N;
        k=k+1;
        f(k)=z(n);
        g(k)=z(nn);
        y(k)=x(n);
    end
    NN=length(y);
    plot(y,f,'-^',y,g,'-*')
    xlabel('Deflection and Shear angle','x','Y and F')
    hl=legend('deflection','shear angle',1)
    xgrid

```

```
end
```

### *Operational Notes:*

1. At the start of input there are three selectors: `fn='eigv'` for EV calculation or `'forc'` for static deflection calculation. Toggle this by commenting the `'forc'` in or out.

The `Ecase` selector is for activating the beam property calculation in the function `Tbeamf`. You can add whatever properties you want (the `E`, `A`, `EA`, `EI` functions) by inserting as many `cases` as you want before the last `end` statement. Follow the form for the cases already programmed. The case names must have 4 letters.

2. There is a numerical delta function called `delta(x,xs)` that produces a simple triangle over the interval  $[x_{j-1}, x_{j+1}]$  where `xs` lies in the first half of this interval. The height is such that the area of the triangle is always 1. That is, it self adjusts to the grid. Of course the larger the `N`, the closer the true delta function is simulated. I use this delta function to put in a variety of point loads. See section (a) of the program to see how I do this. If you want a different loading function, just write a *function* for it and insert it along with my other functions. Then modify section (a) according to your wishes. Remember, in Scilab functions must precede the place where they are called. If your an expert program you will probably write a separate function library.

3. If you want to have different BCs go to the sections in the program where I have the comment *BC implementation*. You will have to read Section 3 of the paper and carefully scrutinize how I handled the cantilever beam BCs. Then figure out what you need to do. If you have questions, leave a comment on my blog.

I think the rest is self explanatory if you study the paper and the code carefully. Comment on my blog with questions, suggestions or just comments.