REFERENCES

Craighead, H.G., 2000, "Nanoelectromechanical systems", **Science**, Vol. 290, No. 5496, pp. 1532-1535.

Ekinci, K.L. and Roukes, M.L., 2005, "Nanoelectromechanical systems", **Review of Scientific Instruments**, Vol. 76, No. 6, pp. 061101-1-061101-12.

Eringen, A.C., 1983, "On differential equations of nonlocal elasticity and solutions of screw dislocation and surface waves", **Journal of Applied Physics**, Vol. 54, No. 9, pp. 4703-4711.

Eringen, A.C., 2002, **Nonlocal Continuum Field Theories**, Springer-Verlag, New York.

Eltaher, M.A., Emam, S.A. and Mahmoud, F.F., 2012, "**Free vibration analysis of functionally graded size-dependent nanobeams", Applied Mathematics and Computation**, Vol. 218, pp. 7406-7420.

Eltaher, M.A., Alshorbagy, A.E. and Mahmoud, F.F., 2013, "**Vibration Analysis of Euler-Bernoulli Nanobeams by using Finite Element method", Applied Mathematical Modeling**, Vol. 37, pp. 4787-4797.

Fu, Y., Zhang, J. and Jiang, Y., 2010, "Influences of the surface energies on the nonlinear static and dynamic behaviors of nanobeams, **Physica E**, Vol. 42, pp. 2268-2273.

Ghannadpour, S.A.M., Mohammadi, B. and Fazilati, J., **2013, "Bending, buckling and vibration problems of nonlocal Euler beams using Ritz method, Composite Structures**, Vol. 96, pp. 584-589.

Gurtin, M.E. and Murdoch, A.I., 1978, "Surface stress in solids", **International Journal of Solids and Structures**, Vol. 14, pp. 431-440.

He, J. and Lilley, C.M., 2008, "Surface effect on the elastic behavior of static bending nanowires", **Nano Letters**, Vol. 8, No. 7, pp. 1798-1802.

He, J. and Lilley, C.M., 2008, "Surface stress effect on the bending resonance of nanowires with different boundary conditions", **Applied Physics Letters**, Vol. 93, pp. 263108-263108-3.

Hui, D.H. and Wang, G.F., 2011, "**Surface effects on the vibration and buckling of double nanobeam systems", Journal of Nanomaterials**, Vol. 2011, pp. 518706-518713.

Jing, G.Y., Duan, H.L., Sun, X.M., Zhang, Z.S., Xu, J., Li, Y.D., Wang, J.X. and Yu, D.P., 2006, "Surface effects on elastic properties of silver nanowires: contact atomic-force microscopy", **Physical Review B**, Vol. 73, No. 23, pp. 235409-235415.

Juntarasaid, C., Pulngern, T. and Chucheepsakul, S., 2012, "Bending and buckling of nanowires including the effects of surface stress and nonlocal elasticity", **Physica E**, Vol. 46, pp. 68-76.

Karnovsky, I.A., 2004, **Free Vibrations of Beams and Frames: Eigenvalues and Eigenfunctions**, McGraw Hill Professional.

Lee, H.L. and Chang, W.J., 2010, "surface effects on frequency analysis of nanotubes using nonlocal Timoshenko beam theory", **Journal of Applied Physics**, Vol. 108, pp. 093503.

Lee, H.L. and Chang, W.J., 2010, "surface and small-scale effects on vibration analysis of a nonuniformnanocantilever beam", **Physica E**, Vol. 43, pp. 466-469.

Liu, C. and Rajapzikse, R. K. N. D., 2010, "Continuum models incorporating surface energy for static and dynamic response of nanoscale beams", **IEEE Transactions on Nanotechnology**, Vol. 9, No. 4, pp. 422-431.

Lu, P., Lee, H.P., Lu, C. and Zhang, P.Q., 2006, **"Dynamic properties of flexural beams using a nonlocal elasticity model", Journal of Applied Physics**, Vol. 99, pp. 073510-073510-9.

Mahmouud, F.F., Eltaher, M.A., Alshorbagy, A.E. and Meletis, E.I., 2012, "Static analysis of nanobeams including surface stress effects by nonlocal finite element", **Journal Mechanical Science and Technology**, Vol. 26, pp. 3555-3563.

Malekzadeh, P. and Shojaee, M., 2013,"Surface and nonlocal effects on the nonlinear free vibration of non-uniform nanobeams, **Composite: Part B**, Vol. 52, pp. 84-92.

Miller, R.E. and Shenoy, V.B., 2000, "Size dependent elastic properties of structural elements", **Nanotechnology**, Vol. 11, No. 3, pp. 139-147.

Peddieson, J., Buchanan, G.R. and McNitt, R.P., 2003, "Application of nonlocal continuum models to nanotechnology", **International Journal of Engineering Science**, Vol. 41, No. 3-5, pp. 305-412.

Phadikar, J.K. and Pradhan, S.C., 2010, "Variational formulation and finite element analysis of nonlocal elastic nanobeam and Nanoplates", **Computational Materials Science**, Vol. 49, No. 3, pp. 492-499.

Reddy, J.N. and Pang, S.D., 2008, "Nonlocal continuum theories of beams for the analysis of carbon nanotubes", **Journal of Applied Physics**, Vol. 103, No. 2, pp. 023511.

Thongyothee, C., Chucheepsakul, S. and Li, T., 2013, "Nonlocal elasticity theory for free vibration of single-walled carbon nanotubes", **Advanced Materials Research**, Vol. 747, pp. 257-260.

Wang, G.F. and Feng, X.Q., 2007, "**Effects of surface elasticity and residual surface tension on the natural frequency of microbeams", Applied Physics Letter**, Vol. 90, No. 23, pp. 231904-231904-3.

Yang, Y. and Lim, C.W., 2012, "Non-classical stiffness strengthening size effects for free vibration of a nonlocal nanostructure", **International Journal of Mechanical Sciences**, Vol. 54, No. 1, pp. 57-68.

APPENDIX A

PROGRAM DETAILS

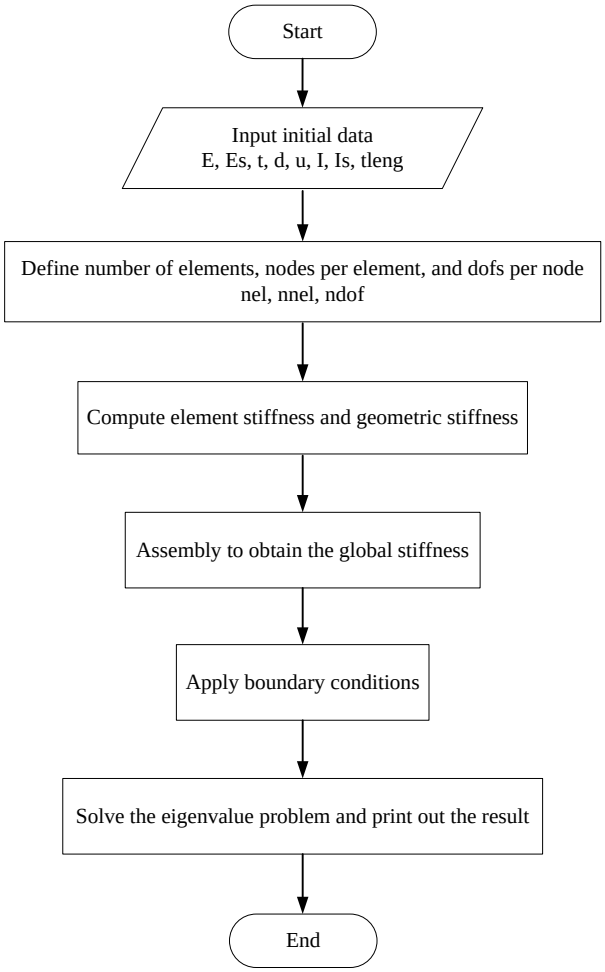

Figure A.1 Algorithm of the natural frequencies solution

```
%=========================================================%
% To find the natural frequencies of Pinned-Pinned nanobeams %
%=========================================================%
% MATLAB codes for Finite Element Analysis                 %
% Variable descriptions                                    %
%   k = element stiffness matrix                           %
%   m = element mass matrix                                %
%   kk = system stiffness matrix                           %
%   mm = system mass matrix                                %
%   index = a vector containing system dofs associated with each element %
%   bcdof = a vector containing dofs associated with boundary conditions %
%   bcval = a vector containing boundary condition values associated with %
%           the dofs in 'bcdof'                            %
%---------------------------------------------------------------------------------------------%

clear
format long e
nel=40;                     % number of elements
nnel=2;                     % number of nodes per element
ndof=2;                     % number of dofs per node
nnode=(nnel-1)*nel+1;       % total number of nodes in system
sdof=nnode*ndof;            % total system dofs

E=76e9 ;                    % Modulus of elasticity(N/m^2)
Es=1.22;                    % Sucface modulus of elasticity(N/m)
t=0.89;                     % Residual surface tension(N/m)
P=0;                        % Buckling Load (N)
d=50e-9;                    % Diameter of nanowire(m)
n=0.04;                     % Nonlocal parameter(eoa/L)^2
rho=10500;                  % mass density
I=(pi*(d^4)/64);            % Moment of inertia of nanowire(m^4)
Is=(pi*(d^3)/8);            % Perimeter moment of inertia nonowire(m^3)
tleng=1000e-9;              % total length
leng=tleng/nel;             % uniform mesh (equal size of elements)
area=(pi*(d^2)/4);          % cross-sectional area
H=2*t*d;
K=((E*I)+(Es*Is)-((n*(tleng^2)*(P-H))));

kk=zeros(sdof,sdof);        % initialization of system stiffness matrix
mm=zeros(sdof,sdof);        % initialization of system mass matrix
index=zeros(nel*ndof,1);    % initialization of index vector


%---------------------------------------------------------------------------------------------%
% boundary conditions
%---------------------------------------------------------------------------------------------%
% 1.Pinned-Pinned
bcdof(1)=1;                 % deflection at node 1 is constrained
bcdof(2)=2*nel+1;           % deflection at the last node is constrained
```

```
% 2.clamped-clamped
%bcdof(1)=1;      bcdof(2)=2;          % deflection and slope at node 1 is constrained
%bcdof(3)=sdof-1; bcdof(4)=sdof;  % deflection and slope at last node is constrained

% 3.clamped-Free at x=0
%bcdof(1)=1;                % deflection at node 1 is constrained
%bcdof(2)=2;                % slope at node 1 is constrained

% 4.clamped-Pinned
%bcdof(1)=1;                % deflection at node 1 is constrained
%bcdof(2)=2;                % slope at node 1 is constrained
%bcdof(3)=2*nel+1;          % deflection at last node is constrained

% 5.Free-Free

% 6.Pinned-Free
%bcdof(1)=1;                % deflection at node 1 is constrained

% 7.Clamped-Sliding
%bcdof(1)=1;                % deflection at node 1 is constrained
%bcdof(2)=2;                % slope at node 1 is constrained
%bcdof(3)=sdof;             % slope at last node is constrained

% 8.Pinned-Sliding
%bcdof(1)=1;                % deflection at node 1 is constrained
%bcdof(2)=sdof;             % slope at last node is constrained

% 9.Free-Sliding
%bcdof(1)=sdof;             % slope at last node is constrained

% 10.Sliding-Sliding
%bcdof(1)=2;                % slope at node 1 is constrained
%bcdof(2)=sdof;             % slope at last node is constrained

%-------------------------------------------------------------------------------------------------%

for iel=1:nel    % loop for the total number of elements

index=feeldof1(iel,nnel,ndof);  % extract system dofs associated with element

[k,m]=febeam1(K,P,H,n,leng,tleng,area,rho,1); % compute element stiffness & mass
matrix

kk=feasmbl1(kk,k,index); % assemble element stiffness matrices into system matrix

mm=feasmbl1(mm,m,index); % assemble element mass matrices into system matrix

end

[kk,mm]=feaplycs(kk,mm,bcdof);  % apply the boundary conditions
```

```
fsol=eig(kk,mm);   % solve the eigenvalue problem
fsol=sqrt(fsol)

%_____________________________________________________________________%


%-----------------------------------------------------------------------------------------------%
function [index]=feeldof1(iel,nnel,ndof)
%-----------------------------------------------------------------------------------------------%
%  Purpose:
%     Compute system dofs associated with each element in one-
%     dimensional problem
%
%  Synopsis:
%     [index]=feeldof1(iel,nnel,ndof)
%
%  Variable Description:
%     index - system dof vector associated with element "iel"
%     iel - element number whose system dofs are to be determined
%     nnel - number of nodes per element
%     ndof - number of dofs per node
%-----------------------------------------------------------------------------------------------%

 edof = nnel*ndof;
 start = (iel-1)*(nnel-1)*ndof;

   for i=1:edof
     index(i)=start+i;
   end


%-----------------------------------------------------------------------------------------------%
function [k,m]=febeam1(K,P,H,n,leng,tleng,area,rho,ipt)
%-----------------------------------------------------------------------------------------------%
%  Purpose:
%     Stiffness and mass matrices for Hermitian beam element
%     nodal dof {v_1 theta_1 v_2 theta_2}
%
%  Synopsis:
%     [k,m]=febeam1(el,xi,leng,area,rho,ipt)
%
%  Variable Description:
%     k - element stiffness matrix (size of 4x4)
%     m - element mass matrix (size of 4x4)
%     el - elastic modulus
%     xi - second moment of inertia of cross-section
%     leng - element length
%     area - area of beam cross-section
%     rho - mass density (mass per unit volume)
%     ipt = 1: consistent mass matrix
```

```
%           2: lumped mass matrix
%           otherwise: diagonal mass matrix
%-------------------------------------------------------------------------------------------------%

% stiffness matrix

 k=[((K*12/(leng)^3)-((P-H)*36/(30*leng)))     ((K*6/(leng)^2)-((P-H)*3/30))
(-(K*12/(leng)^3)-(-(P-H)*36)/(30*leng))    ((K*6/(leng)^2)-((P-H)*3)/30);...
   ((K*6/(leng)^2))-((P-H)*3/30)              ((K*4/leng)-((4*leng*(P-H)))/30)
(-(K*6/(leng)^2)-(-(P-H)*3/30))             ((K*2/leng)-(-(P-H)*leng/30));...
   (-(K*12/(leng)^3)-(-(P-H)*36/(30*leng)))   (-(K*6/(leng)^2)-(-(P-H)*3/30))
(K*12/(leng)^3)-((P-H)*36/(30*leng))        (-(K*6/(leng)^2)-(-(P-H)*3/30));...
   (K*6/(leng)^2)-((P-H)*3/30)                (K*2/leng)-(-(P-H)*leng/30)
(-(K*6/(leng)^2)-(-(P-H)*3/30))             ((K*4/leng)-((P-H)*4*leng/30))];

% consistent mass matrix

 if ipt==1

  mm=rho*area*leng;
  mmm=rho*area*n*(tleng^2);
    m=[((mm*156/420)+(mmm*36/(30*leng)))        ((mm*22*leng/420)+(mmm*3/30))
((mm*54/420)-(mmm*36/(30*leng)))             (-(mm*13*leng/420)+(mmm*3/30));...
      ((mm*22*leng/420)+(mmm*3/30))
((mm*4*(leng^2)/420)+(mmm*4*leng/30))      ((mm*13*leng/420)-(mmm*3/30))
(-(mm*3*(leng^2)/420)-(mmm*1*leng/30));...
      ((mm*54/420)-(mmm*36/(30*leng)))         ((mm*13*leng/420)-(mmm*3/30))
((mm*156/420)+(mmm*36/(30*leng)))           (-(mm*22*leng/420)-(mmm*3/30));...
      (-(mm*13*leng/420)+(mmm*3/30))           (-(mm*3*(leng^2)/420)-
(mmm*1*leng/30))     (-(mm*22*leng/420)-(mmm*3/30))
((mm*4*(leng^2)/420)+(mmm*4*leng/30))];

% lumped mass matrix

 elseif ipt==2

    m=zeros(4,4);
    mass=rho*area*leng;
    m=diag([mass/2  0  mass/2  0]);

% diagonal mass matrix

 else

    m=zeros(4,4);
    mass=rho*area*leng;
    m=mass*diag([1/2  leng^2/78  1/2  leng^2/78]);

end
```

```
%-------------------------------------------------------------------------------------------------%
function [kk]=feasmbl1(kk,k,index)
%-------------------------------------------------------------------------------------------------%
%  Purpose:
%     Assembly of element matrices into the system matrix
%  Synopsis:
%     [kk]=feasmbl1(kk,k,index)
%  Variable Description:
%     kk - system matrix
%     k  - element matri
%     index - d.o.f. vector associated with an element
%-------------------------------------------------------------------------------------------------%

 edof = length(index);
 for i=1:edof
   ii=index(i);
     for j=1:edof
       jj=index(j);
         kk(ii,jj)=kk(ii,jj)+k(i,j);
     end
 end

%-------------------------------------------------------------------------------------------------%
function [kk,mm]=feaplycs(kk,mm,bcdof)
%-------------------------------------------------------------------------------------------------%
%  Purpose:
%     Apply constraints to eigenvalue matrix equation
%     [kk]{x}=lamda[mm]{x}
%  Synopsis:
%     [kk,mm]=feaplycs(kk,mm,bcdof)
%  Variable Description:
%     kk - system stiffness matrix before applying constraints
%     mm - system mass matrix before applying constraints
%     bcdof - a vector containging constrained d.o.f
%-------------------------------------------------------------------

 n=length(bcdof);
 sdof=size(kk);

 for i=1:n
    c=bcdof(i);
    for j=1:sdof
       kk(c,j)=0;
       kk(j,c)=0;
       mm(c,j)=0;
       mm(j,c)=0;
    end

    mm(c,c)=1;
 end
```

```
%=========================================================%
% To find the natural frequencies of Clamped-Clamped nanobeams %
%=========================================================%
% MATLAB codes for Finite Element Analysis                %
% Variable descriptions                                   %
%   k = element stiffness matrix                          %
%   m = element mass matrix                               %
%   kk = system stiffness matrix                          %
%   mm = system mass matrix                               %
%   index = a vector containing system dofs associated with each element %
%   bcdof = a vector containing dofs associated with boundary conditions %
%   bcval = a vector containing boundary condition values associated with %
%           the dofs in 'bcdof'                           %
%---------------------------------------------------------%

clear
format long e
nel=40;                     % number of elements
nnel=2;                     % number of nodes per element
ndof=2;                     % number of dofs per node
nnode=(nnel-1)*nel+1;       % total number of nodes in system
sdof=nnode*ndof;            % total system dofs

E=76e9 ;                    % Modulus of elasticity(N/m^2)
Es=1.22;                    % Sucface modulus of elasticity(N/m)
t=0.89;                     % Residual surface tension(N/m)
P=0;                        % Buckling Load (N)
d=50e-9;                    % Diameter of nanowire(m)
n=0.04;                     % Nonlocal parameter(eoa/L)^2
rho=10500;                  % mass density
I=(pi*(d^4)/64);            % Moment of inertia of nanowire(m^4)
Is=(pi*(d^3)/8);            % Perimeter moment of inertia nonowire(m^3)
tleng=1000e-9;              % total length
leng=tleng/nel;             % uniform mesh (equal size of elements)
area=(pi*(d^2)/4);          % cross-sectional area
H=2*t*d;
K=((E*I)+(Es*Is)-((n*(tleng^2)*(P-H))));

kk=zeros(sdof,sdof);        % initialization of system stiffness matrix
mm=zeros(sdof,sdof);        % initialization of system mass matrix
index=zeros(nel*ndof,1);    % initialization of index vector


%---------------------------------------------------------%
% boundary conditions
%---------------------------------------------------------%
% 1.Pinned-Pinned
%bcdof(1)=1;                % deflection at node 1 is constrained
%bcdof(2)=2*nel+1;          % deflection at the last node is constrained
```

```
% 2.clamped-clamped
bcdof(1)=1;      bcdof(2)=2;         % deflection and slope at node 1 is constrained
bcdof(3)=sdof-1; bcdof(4)=sdof;  % deflection and slope at last node is constrained

% 3.clamped-Free at x=0
%bcdof(1)=1;                % deflection at node 1 is constrained
%bcdof(2)=2;                % slope at node 1 is constrained

% 4.clamped-Pinned
%bcdof(1)=1;                % deflection at node 1 is constrained
%bcdof(2)=2;                % slope at node 1 is constrained
%bcdof(3)=2*nel+1;          % deflection at last node is constrained

% 5.Free-Free

% 6.Pinned-Free
%bcdof(1)=1;                % deflection at node 1 is constrained

% 7.Clamped-Sliding
%bcdof(1)=1;                % deflection at node 1 is constrained
%bcdof(2)=2;                % slope at node 1 is constrained
%bcdof(3)=sdof;             % slope at last node is constrained

% 8.Pinned-Sliding
%bcdof(1)=1;                % deflection at node 1 is constrained
%bcdof(2)=sdof;             % slope at last node is constrained

% 9.Free-Sliding
%bcdof(1)=sdof;             % slope at last node is constrained

% 10.Sliding-Sliding
%bcdof(1)=2;                % slope at node 1 is constrained
%bcdof(2)=sdof;             % slope at last node is constrained

%-------------------------------------------------------------------------------------------------%

for iel=1:nel    % loop for the total number of elements

index=feeldof1(iel,nnel,ndof);  % extract system dofs associated with element

[k,m]=febeam1(K,P,H,n,leng,tleng,area,rho,1); % compute element stiffness & mass
matrix

kk=feasmbl1(kk,k,index); % assemble element stiffness matrices into system matrix

mm=feasmbl1(mm,m,index); % assemble element mass matrices into system matrix

end

[kk,mm]=feaplycs(kk,mm,bcdof);  % apply the boundary conditions
```

```
fsol=eig(kk,mm);   % solve the eigenvalue problem
fsol=sqrt(fsol)

%_____________________________________________________________________%


%-----------------------------------------------------------------------------------------------%
function [index]=feeldof1(iel,nnel,ndof)
%-----------------------------------------------------------------------------------------------%
%  Purpose:
%     Compute system dofs associated with each element in one-
%     dimensional problem
%
%  Synopsis:
%     [index]=feeldof1(iel,nnel,ndof)
%
%  Variable Description:
%     index - system dof vector associated with element "iel"
%     iel - element number whose system dofs are to be determined
%     nnel - number of nodes per element
%     ndof - number of dofs per node
%-----------------------------------------------------------------------------------------------%

 edof = nnel*ndof;
 start = (iel-1)*(nnel-1)*ndof;

   for i=1:edof
     index(i)=start+i;
   end


%-----------------------------------------------------------------------------------------------%
function [k,m]=febeam1(K,P,H,n,leng,tleng,area,rho,ipt)
%-----------------------------------------------------------------------------------------------%
%  Purpose:
%     Stiffness and mass matrices for Hermitian beam element
%     nodal dof {v_1 theta_1 v_2 theta_2}
%
%  Synopsis:
%     [k,m]=febeam1(el,xi,leng,area,rho,ipt)
%
%  Variable Description:
%     k - element stiffness matrix (size of 4x4)
%     m - element mass matrix (size of 4x4)
%     el - elastic modulus
%     xi - second moment of inertia of cross-section
%     leng - element length
%     area - area of beam cross-section
%     rho - mass density (mass per unit volume)
%     ipt = 1: consistent mass matrix
```

```
%           2: lumped mass matrix
%           otherwise: diagonal mass matrix
%-------------------------------------------------------------------------------------------------%

% stiffness matrix

 k=[((K*12/(leng)^3)-((P-H)*36/(30*leng)))     ((K*6/(leng)^2)-((P-H)*3/30))
(-(K*12/(leng)^3)-(-(P-H)*36)/(30*leng))    ((K*6/(leng)^2)-((P-H)*3)/30);...
   ((K*6/(leng)^2))-((P-H)*3/30)             ((K*4/leng)-((4*leng*(P-H)))/30)
(-(K*6/(leng)^2)-(-(P-H)*3/30))            ((K*2/leng)-(-(P-H)*leng/30));...
   (-(K*12/(leng)^3)-(-(P-H)*36/(30*leng)))   (-(K*6/(leng)^2)-(-(P-H)*3/30))
(K*12/(leng)^3)-((P-H)*36/(30*leng))       (-(K*6/(leng)^2)-(-(P-H)*3/30));...
   (K*6/(leng)^2)-((P-H)*3/30)               (K*2/leng)-(-(P-H)*leng/30)
(-(K*6/(leng)^2)-(-(P-H)*3/30))            ((K*4/leng)-((P-H)*4*leng/30))];

% consistent mass matrix

 if ipt==1

  mm=rho*area*leng;
  mmm=rho*area*n*(tleng^2);
    m=[((mm*156/420)+(mmm*36/(30*leng)))       ((mm*22*leng/420)+(mmm*3/30))
((mm*54/420)-(mmm*36/(30*leng)))           (-(mm*13*leng/420)+(mmm*3/30));...
      ((mm*22*leng/420)+(mmm*3/30))
((mm*4*(leng^2)/420)+(mmm*4*leng/30))      ((mm*13*leng/420)-(mmm*3/30))
(-(mm*3*(leng^2)/420)-(mmm*1*leng/30));...
      ((mm*54/420)-(mmm*36/(30*leng)))         ((mm*13*leng/420)-(mmm*3/30))
((mm*156/420)+(mmm*36/(30*leng)))          (-(mm*22*leng/420)-(mmm*3/30));...
      (-(mm*13*leng/420)+(mmm*3/30))           (-(mm*3*(leng^2)/420)-
(mmm*1*leng/30))     (-(mm*22*leng/420)-(mmm*3/30))
((mm*4*(leng^2)/420)+(mmm*4*leng/30))];

% lumped mass matrix

 elseif ipt==2

    m=zeros(4,4);
    mass=rho*area*leng;
    m=diag([mass/2  0  mass/2  0]);

% diagonal mass matrix

 else

    m=zeros(4,4);
    mass=rho*area*leng;
    m=mass*diag([1/2  leng^2/78  1/2  leng^2/78]);

end
```

```
%-------------------------------------------------------------------------------------------------%
function [kk]=feasmbl1(kk,k,index)
%-------------------------------------------------------------------------------------------------%
%  Purpose:
%     Assembly of element matrices into the system matrix
%  Synopsis:
%     [kk]=feasmbl1(kk,k,index)
%  Variable Description:
%     kk - system matrix
%     k  - element matri
%     index - d.o.f. vector associated with an element
%-------------------------------------------------------------------------------------------------%

 edof = length(index);
 for i=1:edof
   ii=index(i);
     for j=1:edof
       jj=index(j);
         kk(ii,jj)=kk(ii,jj)+k(i,j);
     end
 end

%-------------------------------------------------------------------------------------------------%
function [kk,mm]=feaplycs(kk,mm,bcdof)
%-------------------------------------------------------------------------------------------------%
%  Purpose:
%     Apply constraints to eigenvalue matrix equation
%     [kk]{x}=lamda[mm]{x}
%  Synopsis:
%     [kk,mm]=feaplycs(kk,mm,bcdof)
%  Variable Description:
%     kk - system stiffness matrix before applying constraints
%     mm - system mass matrix before applying constraints
%     bcdof - a vector containging constrained d.o.f
%-------------------------------------------------------------------

 n=length(bcdof);
 sdof=size(kk);

 for i=1:n
    c=bcdof(i);
    for j=1:sdof
       kk(c,j)=0;
       kk(j,c)=0;
       mm(c,j)=0;
       mm(j,c)=0;
    end

    mm(c,c)=1;
 end
```

```
%=========================================================%
% To find the natural frequencies of Clamped-Free nanobeams %
%=========================================================%
% MATLAB codes for Finite Element Analysis                 %
% Variable descriptions                                    %
%   k = element stiffness matrix                           %
%   m = element mass matrix                                %
%   kk = system stiffness matrix                           %
%   mm = system mass matrix                                %
%   index = a vector containing system dofs associated with each element %
%   bcdof = a vector containing dofs associated with boundary conditions %
%   bcval = a vector containing boundary condition values associated with %
%           the dofs in 'bcdof'                            %
%---------------------------------------------------------------------------------------------%

clear
format long e
nel=40;                     % number of elements
nnel=2;                     % number of nodes per element
ndof=2;                     % number of dofs per node
nnode=(nnel-1)*nel+1;       % total number of nodes in system
sdof=nnode*ndof;            % total system dofs

E=76e9 ;                    % Modulus of elasticity(N/m^2)
Es=1.22;                    % Sucface modulus of elasticity(N/m)
t=0.89;                     % Residual surface tension(N/m)
P=0;                        % Buckling Load (N)
d=50e-9;                    % Diameter of nanowire(m)
n=0.04;                     % Nonlocal parameter(eoa/L)^2
rho=10500;                  % mass density
I=(pi*(d^4)/64);            % Moment of inertia of nanowire(m^4)
Is=(pi*(d^3)/8);            % Perimeter moment of inertia nonowire(m^3)
tleng=1000e-9;              % total length
leng=tleng/nel;             % uniform mesh (equal size of elements)
area=(pi*(d^2)/4);          % cross-sectional area
H=2*t*d;
K=((E*I)+(Es*Is)-((n*(tleng^2)*(P-H))));

kk=zeros(sdof,sdof);        % initialization of system stiffness matrix
mm=zeros(sdof,sdof);        % initialization of system mass matrix
index=zeros(nel*ndof,1);    % initialization of index vector


%---------------------------------------------------------------------------------------------%
% boundary conditions
%---------------------------------------------------------------------------------------------%
% 1.Pinned-Pinned
%bcdof(1)=1;                % deflection at node 1 is constrained
%bcdof(2)=2*nel+1;          % deflection at the last node is constrained
```

```
% 2.clamped-clamped
%bcdof(1)=1;      bcdof(2)=2;         % deflection and slope at node 1 is constrained
%bcdof(3)=sdof-1; bcdof(4)=sdof;  % deflection and slope at last node is constrained

% 3.clamped-Free at x=0
bcdof(1)=1;                 % deflection at node 1 is constrained
bcdof(2)=2;                 % slope at node 1 is constrained

% 4.clamped-Pinned
%bcdof(1)=1;                % deflection at node 1 is constrained
%bcdof(2)=2;                % slope at node 1 is constrained
%bcdof(3)=2*nel+1;          % deflection at last node is constrained

% 5.Free-Free

% 6.Pinned-Free
%bcdof(1)=1;                % deflection at node 1 is constrained

% 7.Clamped-Sliding
%bcdof(1)=1;                % deflection at node 1 is constrained
%bcdof(2)=2;                % slope at node 1 is constrained
%bcdof(3)=sdof;             % slope at last node is constrained

% 8.Pinned-Sliding
%bcdof(1)=1;                % deflection at node 1 is constrained
%bcdof(2)=sdof;             % slope at last node is constrained

% 9.Free-Sliding
%bcdof(1)=sdof;             % slope at last node is constrained

% 10.Sliding-Sliding
%bcdof(1)=2;                % slope at node 1 is constrained
%bcdof(2)=sdof;             % slope at last node is constrained

%-------------------------------------------------------------------------------------------------%

for iel=1:nel    % loop for the total number of elements

index=feeldof1(iel,nnel,ndof);  % extract system dofs associated with element

[k,m]=febeam1(K,P,H,n,leng,tleng,area,rho,1); % compute element stiffness & mass
matrix

kk=feasmbl1(kk,k,index); % assemble element stiffness matrices into system matrix

mm=feasmbl1(mm,m,index); % assemble element mass matrices into system matrix

end

[kk,mm]=feaplycs(kk,mm,bcdof);  % apply the boundary conditions
```

```
fsol=eig(kk,mm);   % solve the eigenvalue problem
fsol=sqrt(fsol)

%_____________________________________________________________________%


%-----------------------------------------------------------------------------------------------%
function [index]=feeldof1(iel,nnel,ndof)
%-----------------------------------------------------------------------------------------------%
%  Purpose:
%     Compute system dofs associated with each element in one-
%     dimensional problem
%
%  Synopsis:
%     [index]=feeldof1(iel,nnel,ndof)
%
%  Variable Description:
%     index - system dof vector associated with element "iel"
%     iel - element number whose system dofs are to be determined
%     nnel - number of nodes per element
%     ndof - number of dofs per node
%-----------------------------------------------------------------------------------------------%

 edof = nnel*ndof;
 start = (iel-1)*(nnel-1)*ndof;

   for i=1:edof
     index(i)=start+i;
   end


%-----------------------------------------------------------------------------------------------%
function [k,m]=febeam1(K,P,H,n,leng,tleng,area,rho,ipt)
%-----------------------------------------------------------------------------------------------%
%  Purpose:
%     Stiffness and mass matrices for Hermitian beam element
%     nodal dof {v_1 theta_1 v_2 theta_2}
%
%  Synopsis:
%     [k,m]=febeam1(el,xi,leng,area,rho,ipt)
%
%  Variable Description:
%     k - element stiffness matrix (size of 4x4)
%     m - element mass matrix (size of 4x4)
%     el - elastic modulus
%     xi - second moment of inertia of cross-section
%     leng - element length
%     area - area of beam cross-section
%     rho - mass density (mass per unit volume)
%     ipt = 1: consistent mass matrix
```

```
%           2: lumped mass matrix
%           otherwise: diagonal mass matrix
%-------------------------------------------------------------------------------------------------%

% stiffness matrix

 k=[((K*12/(leng)^3)-((P-H)*36/(30*leng)))     ((K*6/(leng)^2)-((P-H)*3/30))
(-(K*12/(leng)^3)-(-(P-H)*36)/(30*leng))    ((K*6/(leng)^2)-((P-H)*3)/30);...
   ((K*6/(leng)^2))-((P-H)*3/30)             ((K*4/leng)-((4*leng*(P-H)))/30)
(-(K*6/(leng)^2)-(-(P-H)*3/30))            ((K*2/leng)-(-(P-H)*leng/30));...
   (-(K*12/(leng)^3)-(-(P-H)*36/(30*leng)))   (-(K*6/(leng)^2)-(-(P-H)*3/30))
(K*12/(leng)^3)-((P-H)*36/(30*leng))       (-(K*6/(leng)^2)-(-(P-H)*3/30));...
   (K*6/(leng)^2)-((P-H)*3/30)               (K*2/leng)-(-(P-H)*leng/30)
(-(K*6/(leng)^2)-(-(P-H)*3/30))            ((K*4/leng)-((P-H)*4*leng/30))];

% consistent mass matrix

 if ipt==1

  mm=rho*area*leng;
  mmm=rho*area*n*(tleng^2);
    m=[((mm*156/420)+(mmm*36/(30*leng)))       ((mm*22*leng/420)+(mmm*3/30))
((mm*54/420)-(mmm*36/(30*leng)))           (-(mm*13*leng/420)+(mmm*3/30));...
      ((mm*22*leng/420)+(mmm*3/30))
((mm*4*(leng^2)/420)+(mmm*4*leng/30))      ((mm*13*leng/420)-(mmm*3/30))
(-(mm*3*(leng^2)/420)-(mmm*1*leng/30));...
      ((mm*54/420)-(mmm*36/(30*leng)))         ((mm*13*leng/420)-(mmm*3/30))
((mm*156/420)+(mmm*36/(30*leng)))          (-(mm*22*leng/420)-(mmm*3/30));...
      (-(mm*13*leng/420)+(mmm*3/30))          (-(mm*3*(leng^2)/420)-
(mmm*1*leng/30))     (-(mm*22*leng/420)-(mmm*3/30))
((mm*4*(leng^2)/420)+(mmm*4*leng/30))];

% lumped mass matrix

 elseif ipt==2

    m=zeros(4,4);
    mass=rho*area*leng;
    m=diag([mass/2  0  mass/2  0]);

% diagonal mass matrix

 else

    m=zeros(4,4);
    mass=rho*area*leng;
    m=mass*diag([1/2  leng^2/78  1/2  leng^2/78]);

end
```

```
%-------------------------------------------------------------------------------------------------%
function [kk]=feasmbl1(kk,k,index)
%-------------------------------------------------------------------------------------------------%
%  Purpose:
%     Assembly of element matrices into the system matrix
%  Synopsis:
%     [kk]=feasmbl1(kk,k,index)
%  Variable Description:
%     kk - system matrix
%     k  - element matri
%     index - d.o.f. vector associated with an element
%-------------------------------------------------------------------------------------------------%

 edof = length(index);
 for i=1:edof
   ii=index(i);
     for j=1:edof
       jj=index(j);
         kk(ii,jj)=kk(ii,jj)+k(i,j);
     end
 end

%-------------------------------------------------------------------------------------------------%
function [kk,mm]=feaplycs(kk,mm,bcdof)
%-------------------------------------------------------------------------------------------------%
%  Purpose:
%     Apply constraints to eigenvalue matrix equation
%     [kk]{x}=lamda[mm]{x}
%  Synopsis:
%     [kk,mm]=feaplycs(kk,mm,bcdof)
%  Variable Description:
%     kk - system stiffness matrix before applying constraints
%     mm - system mass matrix before applying constraints
%     bcdof - a vector containging constrained d.o.f
%-------------------------------------------------------------------

 n=length(bcdof);
 sdof=size(kk);

 for i=1:n
    c=bcdof(i);
    for j=1:sdof
       kk(c,j)=0;
       kk(j,c)=0;
       mm(c,j)=0;
       mm(j,c)=0;
    end

    mm(c,c)=1;
 end
```

```
%=========================================================%
% To find the natural frequencies of Clamped-Pinned nanobeams %
%=========================================================%
% MATLAB codes for Finite Element Analysis                %
% Variable descriptions                                   %
%   k = element stiffness matrix                          %
%   m = element mass matrix                               %
%   kk = system stiffness matrix                          %
%   mm = system mass matrix                               %
%   index = a vector containing system dofs associated with each element %
%   bcdof = a vector containing dofs associated with boundary conditions %
%   bcval = a vector containing boundary condition values associated with %
%           the dofs in 'bcdof'                           %
%---------------------------------------------------------%

clear
format long e
nel=40;                     % number of elements
nnel=2;                     % number of nodes per element
ndof=2;                     % number of dofs per node
nnode=(nnel-1)*nel+1;       % total number of nodes in system
sdof=nnode*ndof;            % total system dofs

E=76e9 ;                    % Modulus of elasticity(N/m^2)
Es=1.22;                    % Sucface modulus of elasticity(N/m)
t=0.89;                     % Residual surface tension(N/m)
P=0;                        % Buckling Load (N)
d=50e-9;                    % Diameter of nanowire(m)
n=0.04;                     % Nonlocal parameter(eoa/L)^2
rho=10500;                  % mass density
I=(pi*(d^4)/64);            % Moment of inertia of nanowire(m^4)
Is=(pi*(d^3)/8);            % Perimeter moment of inertia nonowire(m^3)
tleng=1000e-9;              % total length
leng=tleng/nel;             % uniform mesh (equal size of elements)
area=(pi*(d^2)/4);          % cross-sectional area
H=2*t*d;
K=((E*I)+(Es*Is)-((n*(tleng^2)*(P-H))));

kk=zeros(sdof,sdof);        % initialization of system stiffness matrix
mm=zeros(sdof,sdof);        % initialization of system mass matrix
index=zeros(nel*ndof,1);    % initialization of index vector


%---------------------------------------------------------%
% boundary conditions
%---------------------------------------------------------%
% 1.Pinned-Pinned
%bcdof(1)=1;                % deflection at node 1 is constrained
%bcdof(2)=2*nel+1;          % deflection at the last node is constrained
```

```
% 2.clamped-clamped
%bcdof(1)=1;      bcdof(2)=2;         % deflection and slope at node 1 is constrained
%bcdof(3)=sdof-1; bcdof(4)=sdof;  % deflection and slope at last node is constrained

% 3.clamped-Free at x=0
%bcdof(1)=1;                % deflection at node 1 is constrained
%bcdof(2)=2;                % slope at node 1 is constrained

% 4.clamped-Pinned
bcdof(1)=1;                % deflection at node 1 is constrained
bcdof(2)=2;                % slope at node 1 is constrained
bcdof(3)=2*nel+1;          % deflection at last node is constrained

% 5.Free-Free

% 6.Pinned-Free
%bcdof(1)=1;                % deflection at node 1 is constrained

% 7.Clamped-Sliding
%bcdof(1)=1;                % deflection at node 1 is constrained
%bcdof(2)=2;                % slope at node 1 is constrained
%bcdof(3)=sdof;             % slope at last node is constrained

% 8.Pinned-Sliding
%bcdof(1)=1;                % deflection at node 1 is constrained
%bcdof(2)=sdof;             % slope at last node is constrained

% 9.Free-Sliding
%bcdof(1)=sdof;             % slope at last node is constrained

% 10.Sliding-Sliding
%bcdof(1)=2;                % slope at node 1 is constrained
%bcdof(2)=sdof;             % slope at last node is constrained

%-------------------------------------------------------------------------------------------------%

for iel=1:nel    % loop for the total number of elements

index=feeldof1(iel,nnel,ndof);  % extract system dofs associated with element

[k,m]=febeam1(K,P,H,n,leng,tleng,area,rho,1); % compute element stiffness & mass
matrix

kk=feasmbl1(kk,k,index); % assemble element stiffness matrices into system matrix

mm=feasmbl1(mm,m,index); % assemble element mass matrices into system matrix

end

[kk,mm]=feaplycs(kk,mm,bcdof);  % apply the boundary conditions
```

```
fsol=eig(kk,mm);   % solve the eigenvalue problem
fsol=sqrt(fsol)

%_____________________________________________________________________%


%-----------------------------------------------------------------------------------------------%
function [index]=feeldof1(iel,nnel,ndof)
%-----------------------------------------------------------------------------------------------%
%  Purpose:
%     Compute system dofs associated with each element in one-
%     dimensional problem
%
%  Synopsis:
%     [index]=feeldof1(iel,nnel,ndof)
%
%  Variable Description:
%     index - system dof vector associated with element "iel"
%     iel - element number whose system dofs are to be determined
%     nnel - number of nodes per element
%     ndof - number of dofs per node
%-----------------------------------------------------------------------------------------------%

 edof = nnel*ndof;
 start = (iel-1)*(nnel-1)*ndof;

   for i=1:edof
     index(i)=start+i;
   end


%-----------------------------------------------------------------------------------------------%
function [k,m]=febeam1(K,P,H,n,leng,tleng,area,rho,ipt)
%-----------------------------------------------------------------------------------------------%
%  Purpose:
%     Stiffness and mass matrices for Hermitian beam element
%     nodal dof {v_1 theta_1 v_2 theta_2}
%
%  Synopsis:
%     [k,m]=febeam1(el,xi,leng,area,rho,ipt)
%
%  Variable Description:
%     k - element stiffness matrix (size of 4x4)
%     m - element mass matrix (size of 4x4)
%     el - elastic modulus
%     xi - second moment of inertia of cross-section
%     leng - element length
%     area - area of beam cross-section
%     rho - mass density (mass per unit volume)
%     ipt = 1: consistent mass matrix
```

```
%           2: lumped mass matrix
%           otherwise: diagonal mass matrix
%-------------------------------------------------------------------------------------------------%

% stiffness matrix

 k=[((K*12/(leng)^3)-((P-H)*36/(30*leng)))     ((K*6/(leng)^2)-((P-H)*3/30))
(-(K*12/(leng)^3)-(-(P-H)*36)/(30*leng))    ((K*6/(leng)^2)-((P-H)*3)/30);...
   ((K*6/(leng)^2))-((P-H)*3/30)             ((K*4/leng)-((4*leng*(P-H)))/30)
(-(K*6/(leng)^2)-(-(P-H)*3/30))            ((K*2/leng)-(-(P-H)*leng/30));...
   (-(K*12/(leng)^3)-(-(P-H)*36/(30*leng)))   (-(K*6/(leng)^2)-(-(P-H)*3/30))
(K*12/(leng)^3)-((P-H)*36/(30*leng))       (-(K*6/(leng)^2)-(-(P-H)*3/30));...
   (K*6/(leng)^2)-((P-H)*3/30)               (K*2/leng)-(-(P-H)*leng/30)
(-(K*6/(leng)^2)-(-(P-H)*3/30))            ((K*4/leng)-((P-H)*4*leng/30))];

% consistent mass matrix

 if ipt==1

  mm=rho*area*leng;
  mmm=rho*area*n*(tleng^2);
    m=[((mm*156/420)+(mmm*36/(30*leng)))       ((mm*22*leng/420)+(mmm*3/30))
((mm*54/420)-(mmm*36/(30*leng)))           (-(mm*13*leng/420)+(mmm*3/30));...
      ((mm*22*leng/420)+(mmm*3/30))
((mm*4*(leng^2)/420)+(mmm*4*leng/30))      ((mm*13*leng/420)-(mmm*3/30))
(-(mm*3*(leng^2)/420)-(mmm*1*leng/30));...
      ((mm*54/420)-(mmm*36/(30*leng)))         ((mm*13*leng/420)-(mmm*3/30))
((mm*156/420)+(mmm*36/(30*leng)))          (-(mm*22*leng/420)-(mmm*3/30));...
      (-(mm*13*leng/420)+(mmm*3/30))          (-(mm*3*(leng^2)/420)-
(mmm*1*leng/30))     (-(mm*22*leng/420)-(mmm*3/30))
((mm*4*(leng^2)/420)+(mmm*4*leng/30))];

% lumped mass matrix

 elseif ipt==2

    m=zeros(4,4);
    mass=rho*area*leng;
    m=diag([mass/2  0  mass/2  0]);

% diagonal mass matrix

 else

    m=zeros(4,4);
    mass=rho*area*leng;
    m=mass*diag([1/2  leng^2/78  1/2  leng^2/78]);

end
```

```
%-------------------------------------------------------------------------------------------------%
function [kk]=feasmbl1(kk,k,index)
%-------------------------------------------------------------------------------------------------%
%  Purpose:
%     Assembly of element matrices into the system matrix
%  Synopsis:
%     [kk]=feasmbl1(kk,k,index)
%  Variable Description:
%     kk - system matrix
%     k  - element matri
%     index - d.o.f. vector associated with an element
%-------------------------------------------------------------------------------------------------%

 edof = length(index);
 for i=1:edof
   ii=index(i);
     for j=1:edof
       jj=index(j);
         kk(ii,jj)=kk(ii,jj)+k(i,j);
     end
 end

%-------------------------------------------------------------------------------------------------%
function [kk,mm]=feaplycs(kk,mm,bcdof)
%-------------------------------------------------------------------------------------------------%
%  Purpose:
%     Apply constraints to eigenvalue matrix equation
%     [kk]{x}=lamda[mm]{x}
%  Synopsis:
%     [kk,mm]=feaplycs(kk,mm,bcdof)
%  Variable Description:
%     kk - system stiffness matrix before applying constraints
%     mm - system mass matrix before applying constraints
%     bcdof - a vector containging constrained d.o.f
%-------------------------------------------------------------------

 n=length(bcdof);
 sdof=size(kk);

 for i=1:n
    c=bcdof(i);
    for j=1:sdof
       kk(c,j)=0;
       kk(j,c)=0;
       mm(c,j)=0;
       mm(j,c)=0;
    end

    mm(c,c)=1;
 end
```

```
%=========================================================%
% To find the natural frequencies of Clamped-Sliding nanobeams %
%=========================================================%
% MATLAB codes for Finite Element Analysis                %
% Variable descriptions                                   %
%   k = element stiffness matrix                          %
%   m = element mass matrix                               %
%   kk = system stiffness matrix                          %
%   mm = system mass matrix                               %
%   index = a vector containing system dofs associated with each element %
%   bcdof = a vector containing dofs associated with boundary conditions %
%   bcval = a vector containing boundary condition values associated with %
%           the dofs in 'bcdof'                           %
%---------------------------------------------------------%

clear
format long e
nel=40;                     % number of elements
nnel=2;                     % number of nodes per element
ndof=2;                     % number of dofs per node
nnode=(nnel-1)*nel+1;       % total number of nodes in system
sdof=nnode*ndof;            % total system dofs

E=76e9 ;                    % Modulus of elasticity(N/m^2)
Es=1.22;                    % Sucface modulus of elasticity(N/m)
t=0.89;                     % Residual surface tension(N/m)
P=0;                        % Buckling Load (N)
d=50e-9;                    % Diameter of nanowire(m)
n=0.04;                     % Nonlocal parameter(eoa/L)^2
rho=10500;                  % mass density
I=(pi*(d^4)/64);            % Moment of inertia of nanowire(m^4)
Is=(pi*(d^3)/8);            % Perimeter moment of inertia nonowire(m^3)
tleng=1000e-9;              % total length
leng=tleng/nel;             % uniform mesh (equal size of elements)
area=(pi*(d^2)/4);          % cross-sectional area
H=2*t*d;
K=((E*I)+(Es*Is)-((n*(tleng^2)*(P-H))));

kk=zeros(sdof,sdof);        % initialization of system stiffness matrix
mm=zeros(sdof,sdof);        % initialization of system mass matrix
index=zeros(nel*ndof,1);    % initialization of index vector


%---------------------------------------------------------%
% boundary conditions
%---------------------------------------------------------%
% 1.Pinned-Pinned
%bcdof(1)=1;                % deflection at node 1 is constrained
%bcdof(2)=2*nel+1;          % deflection at the last node is constrained
```

```
% 2.clamped-clamped
%bcdof(1)=1;      bcdof(2)=2;         % deflection and slope at node 1 is constrained
%bcdof(3)=sdof-1; bcdof(4)=sdof;  % deflection and slope at last node is constrained

% 3.clamped-Free at x=0
%bcdof(1)=1;                % deflection at node 1 is constrained
%bcdof(2)=2;                % slope at node 1 is constrained

% 4.clamped-Pinned
%bcdof(1)=1;                % deflection at node 1 is constrained
%bcdof(2)=2;                % slope at node 1 is constrained
%bcdof(3)=2*nel+1;          % deflection at last node is constrained

% 5.Free-Free

% 6.Pinned-Free
%bcdof(1)=1;                % deflection at node 1 is constrained

% 7.Clamped-Sliding
bcdof(1)=1;                % deflection at node 1 is constrained
bcdof(2)=2;                % slope at node 1 is constrained
bcdof(3)=sdof;             % slope at last node is constrained

% 8.Pinned-Sliding
%bcdof(1)=1;                % deflection at node 1 is constrained
%bcdof(2)=sdof;             % slope at last node is constrained

% 9.Free-Sliding
%bcdof(1)=sdof;             % slope at last node is constrained

% 10.Sliding-Sliding
%bcdof(1)=2;                % slope at node 1 is constrained
%bcdof(2)=sdof;             % slope at last node is constrained

%-------------------------------------------------------------------------------------------------%

for iel=1:nel    % loop for the total number of elements

index=feeldof1(iel,nnel,ndof);  % extract system dofs associated with element

[k,m]=febeam1(K,P,H,n,leng,tleng,area,rho,1); % compute element stiffness & mass
matrix

kk=feasmbl1(kk,k,index); % assemble element stiffness matrices into system matrix

mm=feasmbl1(mm,m,index); % assemble element mass matrices into system matrix

end

[kk,mm]=feaplycs(kk,mm,bcdof);  % apply the boundary conditions
```

```
fsol=eig(kk,mm);   % solve the eigenvalue problem
fsol=sqrt(fsol)

%_____________________________________________________________________%


%-----------------------------------------------------------------------------------------------%
function [index]=feeldof1(iel,nnel,ndof)
%-----------------------------------------------------------------------------------------------%
%  Purpose:
%     Compute system dofs associated with each element in one-
%     dimensional problem
%
%  Synopsis:
%     [index]=feeldof1(iel,nnel,ndof)
%
%  Variable Description:
%     index - system dof vector associated with element "iel"
%     iel - element number whose system dofs are to be determined
%     nnel - number of nodes per element
%     ndof - number of dofs per node
%-----------------------------------------------------------------------------------------------%

 edof = nnel*ndof;
 start = (iel-1)*(nnel-1)*ndof;

   for i=1:edof
     index(i)=start+i;
   end


%-----------------------------------------------------------------------------------------------%
function [k,m]=febeam1(K,P,H,n,leng,tleng,area,rho,ipt)
%-----------------------------------------------------------------------------------------------%
%  Purpose:
%     Stiffness and mass matrices for Hermitian beam element
%     nodal dof {v_1 theta_1 v_2 theta_2}
%
%  Synopsis:
%     [k,m]=febeam1(el,xi,leng,area,rho,ipt)
%
%  Variable Description:
%     k - element stiffness matrix (size of 4x4)
%     m - element mass matrix (size of 4x4)
%     el - elastic modulus
%     xi - second moment of inertia of cross-section
%     leng - element length
%     area - area of beam cross-section
%     rho - mass density (mass per unit volume)
%     ipt = 1: consistent mass matrix
```

```
%           2: lumped mass matrix
%           otherwise: diagonal mass matrix
%-------------------------------------------------------------------------------------------------%

% stiffness matrix

 k=[((K*12/(leng)^3)-((P-H)*36/(30*leng)))     ((K*6/(leng)^2)-((P-H)*3/30))
(-(K*12/(leng)^3)-(-(P-H)*36)/(30*leng))    ((K*6/(leng)^2)-((P-H)*3)/30);...
   ((K*6/(leng)^2))-((P-H)*3/30)             ((K*4/leng)-((4*leng*(P-H)))/30)
(-(K*6/(leng)^2)-(-(P-H)*3/30))            ((K*2/leng)-(-(P-H)*leng/30));...
   (-(K*12/(leng)^3)-(-(P-H)*36/(30*leng)))   (-(K*6/(leng)^2)-(-(P-H)*3/30))
(K*12/(leng)^3)-((P-H)*36/(30*leng))       (-(K*6/(leng)^2)-(-(P-H)*3/30));...
   (K*6/(leng)^2)-((P-H)*3/30)               (K*2/leng)-(-(P-H)*leng/30)
(-(K*6/(leng)^2)-(-(P-H)*3/30))            ((K*4/leng)-((P-H)*4*leng/30))];

% consistent mass matrix

 if ipt==1

  mm=rho*area*leng;
  mmm=rho*area*n*(tleng^2);
    m=[((mm*156/420)+(mmm*36/(30*leng)))       ((mm*22*leng/420)+(mmm*3/30))
((mm*54/420)-(mmm*36/(30*leng)))           (-(mm*13*leng/420)+(mmm*3/30));...
      ((mm*22*leng/420)+(mmm*3/30))
((mm*4*(leng^2)/420)+(mmm*4*leng/30))      ((mm*13*leng/420)-(mmm*3/30))
(-(mm*3*(leng^2)/420)-(mmm*1*leng/30));...
      ((mm*54/420)-(mmm*36/(30*leng)))         ((mm*13*leng/420)-(mmm*3/30))
((mm*156/420)+(mmm*36/(30*leng)))          (-(mm*22*leng/420)-(mmm*3/30));...
      (-(mm*13*leng/420)+(mmm*3/30))          (-(mm*3*(leng^2)/420)-
(mmm*1*leng/30))     (-(mm*22*leng/420)-(mmm*3/30))
((mm*4*(leng^2)/420)+(mmm*4*leng/30))];

% lumped mass matrix

 elseif ipt==2

    m=zeros(4,4);
    mass=rho*area*leng;
    m=diag([mass/2  0  mass/2  0]);

% diagonal mass matrix

 else

    m=zeros(4,4);
    mass=rho*area*leng;
    m=mass*diag([1/2  leng^2/78  1/2  leng^2/78]);

end
```

```
%-------------------------------------------------------------------------------------------------%
function [kk]=feasmbl1(kk,k,index)
%-------------------------------------------------------------------------------------------------%
%  Purpose:
%     Assembly of element matrices into the system matrix
%  Synopsis:
%     [kk]=feasmbl1(kk,k,index)
%  Variable Description:
%     kk - system matrix
%     k  - element matri
%     index - d.o.f. vector associated with an element
%-------------------------------------------------------------------------------------------------%

 edof = length(index);
 for i=1:edof
   ii=index(i);
     for j=1:edof
       jj=index(j);
         kk(ii,jj)=kk(ii,jj)+k(i,j);
     end
 end

%-------------------------------------------------------------------------------------------------%
function [kk,mm]=feaplycs(kk,mm,bcdof)
%-------------------------------------------------------------------------------------------------%
%  Purpose:
%     Apply constraints to eigenvalue matrix equation
%     [kk]{x}=lamda[mm]{x}
%  Synopsis:
%     [kk,mm]=feaplycs(kk,mm,bcdof)
%  Variable Description:
%     kk - system stiffness matrix before applying constraints
%     mm - system mass matrix before applying constraints
%     bcdof - a vector containging constrained d.o.f
%-------------------------------------------------------------------

 n=length(bcdof);
 sdof=size(kk);

 for i=1:n
    c=bcdof(i);
    for j=1:sdof
       kk(c,j)=0;
       kk(j,c)=0;
       mm(c,j)=0;
       mm(j,c)=0;
    end

    mm(c,c)=1;
 end
```

```
%=========================================================%
% To find the natural frequencies of Sliding-Pinned nanobeams %
%=========================================================%
% MATLAB codes for Finite Element Analysis                %
% Variable descriptions                                   %
%   k = element stiffness matrix                          %
%   m = element mass matrix                               %
%   kk = system stiffness matrix                          %
%   mm = system mass matrix                               %
%   index = a vector containing system dofs associated with each element %
%   bcdof = a vector containing dofs associated with boundary conditions %
%   bcval = a vector containing boundary condition values associated with %
%           the dofs in 'bcdof'                           %
%---------------------------------------------------------%

clear
format long e
nel=40;                     % number of elements
nnel=2;                     % number of nodes per element
ndof=2;                     % number of dofs per node
nnode=(nnel-1)*nel+1;       % total number of nodes in system
sdof=nnode*ndof;            % total system dofs

E=76e9 ;                    % Modulus of elasticity(N/m^2)
Es=1.22;                    % Sucface modulus of elasticity(N/m)
t=0.89;                     % Residual surface tension(N/m)
P=0;                        % Buckling Load (N)
d=50e-9;                    % Diameter of nanowire(m)
n=0.04;                     % Nonlocal parameter(eoa/L)^2
rho=10500;                  % mass density
I=(pi*(d^4)/64);            % Moment of inertia of nanowire(m^4)
Is=(pi*(d^3)/8);            % Perimeter moment of inertia nonowire(m^3)
tleng=1000e-9;              % total length
leng=tleng/nel;             % uniform mesh (equal size of elements)
area=(pi*(d^2)/4);          % cross-sectional area
H=2*t*d;
K=((E*I)+(Es*Is)-((n*(tleng^2)*(P-H))));

kk=zeros(sdof,sdof);        % initialization of system stiffness matrix
mm=zeros(sdof,sdof);        % initialization of system mass matrix
index=zeros(nel*ndof,1);    % initialization of index vector


%---------------------------------------------------------%
% boundary conditions
%---------------------------------------------------------%
% 1.Pinned-Pinned
%bcdof(1)=1;                % deflection at node 1 is constrained
%bcdof(2)=2*nel+1;          % deflection at the last node is constrained
```

```
% 2.clamped-clamped
%bcdof(1)=1;      bcdof(2)=2;         % deflection and slope at node 1 is constrained
%bcdof(3)=sdof-1; bcdof(4)=sdof;  % deflection and slope at last node is constrained

% 3.clamped-Free at x=0
%bcdof(1)=1;                % deflection at node 1 is constrained
%bcdof(2)=2;                % slope at node 1 is constrained

% 4.clamped-Pinned
%bcdof(1)=1;                % deflection at node 1 is constrained
%bcdof(2)=2;                % slope at node 1 is constrained
%bcdof(3)=2*nel+1;          % deflection at last node is constrained

% 5.Free-Free

% 6.Pinned-Free
%bcdof(1)=1;                % deflection at node 1 is constrained

% 7.Clamped-Sliding
%bcdof(1)=1;                % deflection at node 1 is constrained
%bcdof(2)=2;                % slope at node 1 is constrained
%bcdof(3)=sdof;             % slope at last node is constrained

% 8.Pinned-Sliding
bcdof(1)=1;                % deflection at node 1 is constrained
bcdof(2)=sdof;             % slope at last node is constrained

% 9.Free-Sliding
%bcdof(1)=sdof;             % slope at last node is constrained

% 10.Sliding-Sliding
%bcdof(1)=2;                % slope at node 1 is constrained
%bcdof(2)=sdof;             % slope at last node is constrained

%-------------------------------------------------------------------------------------------------%

for iel=1:nel    % loop for the total number of elements

index=feeldof1(iel,nnel,ndof);  % extract system dofs associated with element

[k,m]=febeam1(K,P,H,n,leng,tleng,area,rho,1); % compute element stiffness & mass
matrix

kk=feasmbl1(kk,k,index); % assemble element stiffness matrices into system matrix

mm=feasmbl1(mm,m,index); % assemble element mass matrices into system matrix

end

[kk,mm]=feaplycs(kk,mm,bcdof);  % apply the boundary conditions
```

```
fsol=eig(kk,mm);   % solve the eigenvalue problem
fsol=sqrt(fsol)

%_____________________________________________________________________%


%-----------------------------------------------------------------------------------------------%
function [index]=feeldof1(iel,nnel,ndof)
%-----------------------------------------------------------------------------------------------%
%  Purpose:
%     Compute system dofs associated with each element in one-
%     dimensional problem
%
%  Synopsis:
%     [index]=feeldof1(iel,nnel,ndof)
%
%  Variable Description:
%     index - system dof vector associated with element "iel"
%     iel - element number whose system dofs are to be determined
%     nnel - number of nodes per element
%     ndof - number of dofs per node
%-----------------------------------------------------------------------------------------------%

 edof = nnel*ndof;
 start = (iel-1)*(nnel-1)*ndof;

   for i=1:edof
     index(i)=start+i;
   end


%-----------------------------------------------------------------------------------------------%
function [k,m]=febeam1(K,P,H,n,leng,tleng,area,rho,ipt)
%-----------------------------------------------------------------------------------------------%
%  Purpose:
%     Stiffness and mass matrices for Hermitian beam element
%     nodal dof {v_1 theta_1 v_2 theta_2}
%
%  Synopsis:
%     [k,m]=febeam1(el,xi,leng,area,rho,ipt)
%
%  Variable Description:
%     k - element stiffness matrix (size of 4x4)
%     m - element mass matrix (size of 4x4)
%     el - elastic modulus
%     xi - second moment of inertia of cross-section
%     leng - element length
%     area - area of beam cross-section
%     rho - mass density (mass per unit volume)
%     ipt = 1: consistent mass matrix
```

```
%           2: lumped mass matrix
%           otherwise: diagonal mass matrix
%-------------------------------------------------------------------------------------------------%

% stiffness matrix

 k=[((K*12/(leng)^3)-((P-H)*36/(30*leng)))     ((K*6/(leng)^2)-((P-H)*3/30))
(-(K*12/(leng)^3)-(-(P-H)*36)/(30*leng))    ((K*6/(leng)^2)-((P-H)*3)/30);...
   ((K*6/(leng)^2))-((P-H)*3/30)             ((K*4/leng)-((4*leng*(P-H)))/30)
(-(K*6/(leng)^2)-(-(P-H)*3/30))            ((K*2/leng)-(-(P-H)*leng/30));...
   (-(K*12/(leng)^3)-(-(P-H)*36/(30*leng)))   (-(K*6/(leng)^2)-(-(P-H)*3/30))
(K*12/(leng)^3)-((P-H)*36/(30*leng))       (-(K*6/(leng)^2)-(-(P-H)*3/30));...
   (K*6/(leng)^2)-((P-H)*3/30)               (K*2/leng)-(-(P-H)*leng/30)
(-(K*6/(leng)^2)-(-(P-H)*3/30))            ((K*4/leng)-((P-H)*4*leng/30))];

% consistent mass matrix

 if ipt==1

  mm=rho*area*leng;
  mmm=rho*area*n*(tleng^2);
    m=[((mm*156/420)+(mmm*36/(30*leng)))       ((mm*22*leng/420)+(mmm*3/30))
((mm*54/420)-(mmm*36/(30*leng)))           (-(mm*13*leng/420)+(mmm*3/30));...
      ((mm*22*leng/420)+(mmm*3/30))
((mm*4*(leng^2)/420)+(mmm*4*leng/30))      ((mm*13*leng/420)-(mmm*3/30))
(-(mm*3*(leng^2)/420)-(mmm*1*leng/30));...
      ((mm*54/420)-(mmm*36/(30*leng)))         ((mm*13*leng/420)-(mmm*3/30))
((mm*156/420)+(mmm*36/(30*leng)))          (-(mm*22*leng/420)-(mmm*3/30));...
      (-(mm*13*leng/420)+(mmm*3/30))          (-(mm*3*(leng^2)/420)-
(mmm*1*leng/30))     (-(mm*22*leng/420)-(mmm*3/30))
((mm*4*(leng^2)/420)+(mmm*4*leng/30))];

% lumped mass matrix

 elseif ipt==2

    m=zeros(4,4);
    mass=rho*area*leng;
    m=diag([mass/2  0  mass/2  0]);

% diagonal mass matrix

 else

    m=zeros(4,4);
    mass=rho*area*leng;
    m=mass*diag([1/2  leng^2/78  1/2  leng^2/78]);

end
```

```
%-------------------------------------------------------------------------------------------------%
function [kk]=feasmbl1(kk,k,index)
%-------------------------------------------------------------------------------------------------%
%  Purpose:
%     Assembly of element matrices into the system matrix
%  Synopsis:
%     [kk]=feasmbl1(kk,k,index)
%  Variable Description:
%     kk - system matrix
%     k  - element matri
%     index - d.o.f. vector associated with an element
%-------------------------------------------------------------------------------------------------%

 edof = length(index);
 for i=1:edof
   ii=index(i);
     for j=1:edof
       jj=index(j);
         kk(ii,jj)=kk(ii,jj)+k(i,j);
     end
 end

%-------------------------------------------------------------------------------------------------%
function [kk,mm]=feaplycs(kk,mm,bcdof)
%-------------------------------------------------------------------------------------------------%
%  Purpose:
%     Apply constraints to eigenvalue matrix equation
%     [kk]{x}=lamda[mm]{x}
%  Synopsis:
%     [kk,mm]=feaplycs(kk,mm,bcdof)
%  Variable Description:
%     kk - system stiffness matrix before applying constraints
%     mm - system mass matrix before applying constraints
%     bcdof - a vector containging constrained d.o.f
%-------------------------------------------------------------------

 n=length(bcdof);
 sdof=size(kk);

 for i=1:n
    c=bcdof(i);
    for j=1:sdof
       kk(c,j)=0;
       kk(j,c)=0;
       mm(c,j)=0;
       mm(j,c)=0;
    end

    mm(c,c)=1;
 end
```

APPENDIX B

PUBLICATIONS

Chinnawut Juntarasaid Tawich Pulngern and Somchai Chucheepsakul, 2014, "Surface stress and non-local elasticity effects on the free vibration behavior of nanobeams" **KMUTT Research and Development Journal**, Vol.37, No.4 October-December, pp. 481-502.