# Euler Bernoulli Beams

**Numerical Analysis Project with Prof. Michael Karow**
**TU Berlin**
COSSE Summer semester 2020
**Group members**: Arvind Nayak, Fons van der Plans, Karolina Siemieniuk, Obin Sturm

## 1. Introduction

Structures are all around us. Constantly deflecting and deforming. To measure the performance of a structure deflections and deformations need to be investigated. There are many reason why those performance measures are important. One of them is safety, but the comfort of the users is a vital consideration as well. For example a floor needs to be stiff enough so that it does not deflect too much and users do not feel uncomfortable, even though the deflection might be safe. Another example would be buildings deflecting due to wind or bridges deflecting under the load of the cars going through it. There are various methods to calculate those parameters and this project is going to focus on the Euler Bernoulli beam theory. It is a model that was developed in the 18th century and is a method still used today to analyse the behaviour of bending elements.

In this project, we have analyzed the static and dynamic equations of simply supported and clamped beams based on the Euler Bernoulli beam theory. Finite element formulation has been used for the bending equations via cubic piecewise differentiable functions. Variational statement is adopted to derive the governing equations and element matrices. The mathematical model is presented in section 2. After that, the variational formulation and corresponding finite element basis are explained in detail in section 3. Section 4 is devoted to certain numerical experiments. The accuracy and reliability of this model is presented and verified comparing the results with the closed form solutions.
This project has been done using the Pluto editor, a reactive notebook environment for Julia. Hence, we also detail our implementation alongside the theory.

## 2. Governing equations

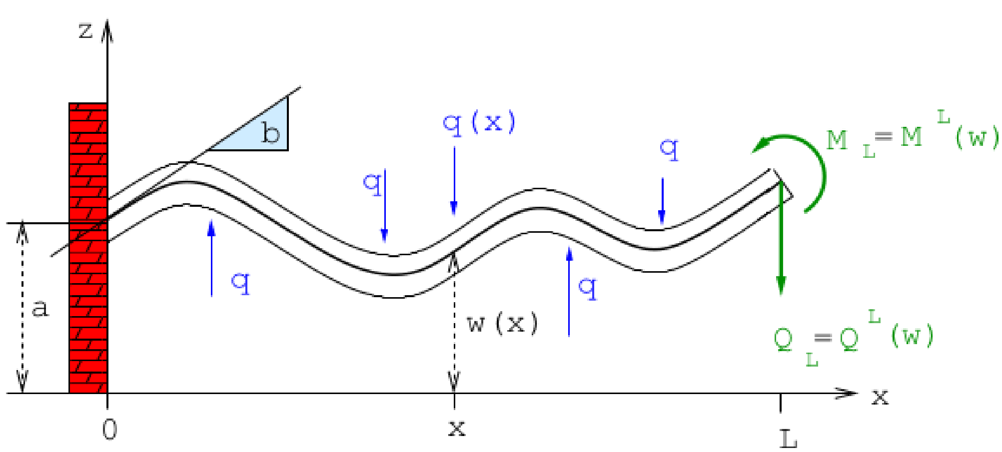



Fig:1 Cantilever beam domain with terminologies [Kar16]

The static bending equation is described below [Kar16]

$$(EIw'')''(x) = q(x), \quad x \in \Omega = [0, L], \ (q \in V, \ (EIw'') \in V), \tag{1}$$

where the set of piecewise twice differentiable functions is defined as
$V := C^{2,p}(0, L) = \{\phi : \Omega \to \mathbb{R} | \phi \in C(0, L), \phi' \in C(0, L), \phi'' \in C^p(0, L)\}$
and,
$w(x)$ - height of the neutral axis (bending curve) at $x$
$E = E(x)$ Young's module
$I = I(x)$ area moment of inertia: $I(x) = \int z^2 dydz$
$q(x)$ load (force density) at $x$
Further notation:
$M^x(w)$ = $EIw''(x)$ bending moment at $x$
$Q^x(w)$ = -$(EIw'')'(x)$ shear force at $x$
To get a unique solution of the beam equation, two essential and two natural boundary conditions need to be added. For a cantilever that is clamped at one end with tip laoding we see that,
$w(0) = a, \quad w'(0) = b, \quad Q^L(w) = Q_L, \quad M^L(w) = M_L$, where $a, b, Q_L, M_L \in \mathbb{R}$ are given.

In the case of a simply supoorted beam, these then become,
$w(0) = a_0, \quad w(L) = a_L, \quad M^0(w) = M_0, \quad M^L(w) = M_L$, where $a_0, a_L, M_0, M_L \in \mathbb{R}$ are given.

In the dynamic case the bending curve as well as all forces and boundary conditions depend on time. In particular the bending curve at time t is given by $w(x, t)$ governed by, [Kar16]

$$\begin{aligned} \ddot{w} + (EIw'')'' = q, \qquad x \in \Omega, \ t \in (0, \mathrm{T}] \\ w(x, 0) = w_0, \qquad x \in \Omega \end{aligned} \tag{2}$$

using initial data, $w_0 \in V$ and where $\mu = \mu(x)$ is the mass density (more precisely: the mass per unit length). $\ddot{w}$ denotes the second derivative of $w$ with respect to $t$ and all other definitions follow from (1). One important fact about this differential equation is the absence of a disspation term, implies that the transverse delfections of the beam will be undamped over time.

# 3 Weak formulation

## 3.1 Variational form

Using the strong form from the static case (1), we formulate the weak form, assuming that $w$ satisfies (1) and the appropriate boundary conditions. [Kar16]

$$\int_0^L EIw''\psi'' = \int_0^L q\psi + b(\psi), \qquad \forall \psi \in V, \tag{3}$$

where, $b(\psi) = Q_L\psi(L) - Q_0\psi(0) + M_L\psi'(L) - M_0\psi'(0)$.
Now, we choose $\psi \in V$ such that $\psi(0) = 1$, $\psi'(0) = \psi'(L) = 0$. Therefore, $b(\psi) = Q_0$.

The same procedure as in the static case (multiplication with and partial integration) yields the weak formulation:

$$\int_0^L \ddot{w}\psi + \int_0^L EIw''\psi'' = \int_0^L q(\cdot, t)\psi + b(\psi, t), \tag{4}$$

where, $b(\psi, t) = Q_L(t)\psi(L) - Q_0(t)\psi(0) + M_L(t)\psi'(L) - M_0(t)\psi'(0)$. [Kar16]

## 3.2 Galerkin's method

An approximate solution of w, wh needs to be computed. To calculate $w_h$, a finite dimensional subspace $V_h$ (Ansatz space) of the space $V$ needs to be chosen with basis $\phi_1 \dots, \phi_n : [0, L] \to \mathbb{R}$, where $w_h(x) = \sum_{k=1}^{n} w_k \phi_k(x), w_k \in R.$ [Kar16]. Inserting this Ansatz into the weak formulations (3) with Galerkin Bubnov [Jog15] scheme, where $\varphi = \phi_j,\ \ j = 1, \dots, n$

$$\int_0^L EI w_h'' \phi_j'' = \int_0^L q\phi_j + b(\phi j),\ \ j = 1, \dots, n \tag{5}$$

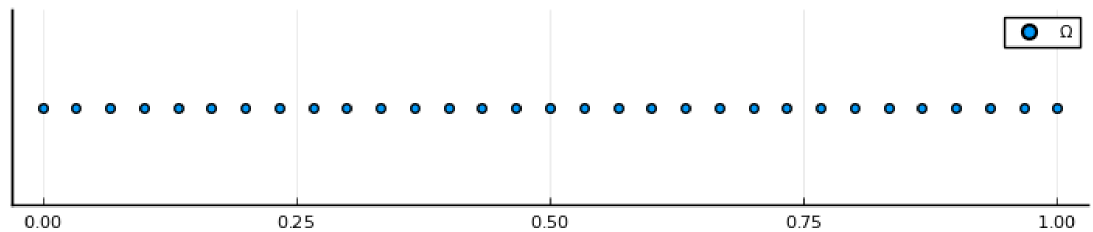
And in (4) for w the equivalent (time depentent) Galerkin ansatz, $w_h(x,t) = \sum_{k=1}^{N} w_k(t)\phi_k(x)$ and for $\varphi$ the same basis functions as above, $\phi_j, j = 1, \dots n$ we get a similar equation with one additional term representing the mass density.

$$\int_0^L \ddot{w}_h \phi_j + \int_0^L EI w_h'' \phi_j'' = \int_0^L q\phi_j + b(\phi j),\ \ j = 1, \dots, n \tag{6}$$

Let us now generate our discrete domain!

```
gen_elements (generic function with 1 method)
```

```
xh,ne,conn=gen_elements(L,N);
```




```
local2global (generic function with 1 method)
```

```
dofs=local2global(ne);
```

```
constant_dist (generic function with 1 method)
```

```
qv,mu,EI=constant_dist(ul,μ,ei); #taking constant distribution of q, EI and μ on domain
```

**Note:** For simplicity, we assume here that the beam has a constant bending stiffiness `EI` and mass diistribution `μ` throughout the domain. It is also assumed that the distributed load is uniform.

**Choice of Ansatz space**

$V_h$ was chosen to be the space of piecewise cubic polynomials, according to [Kar16].

$$V_h = \{\phi \in V | \phi|_{(x_i, x_{i+1})} \text{ is a polynomial of degree} \le 3, i = 1, \dots, N-1\}$$

Each $\phi \in V_h$ can be uniquely written as

$$\phi = \sum_{k=1}^{2N} u_k \phi_k = \sum_{i=1}^{N} (u_{2i-1}\phi_{2i-1} + u_{2i}\phi_{2i})$$

with the basis functions $\phi_i \in V_h$, for $i = 1, \dots, 2N$, define:

$$\phi_1(x) = \begin{cases} \bar{\phi}_1(\frac{x}{h}) & x \in [0,h] \\ 0 & \text{otherwise,} \end{cases}$$

$$\phi_2(x) = \begin{cases} h\bar{\phi}_2(\frac{x}{h}) & x \in [0,h] \\ 0 & \text{otherwise,} \end{cases}$$

for $i = 2, \dots, N-1$:

$$\phi_{2i-1}(x) = \begin{cases} \bar{\phi}_3(\frac{x-x_{i-1}}{h}) & x \in [x_{i-1}, x_i] \\ \bar{\phi}_1(\frac{x-x_i}{h}) & x \in [x_i, x_{i+1}] \\ 0 & \text{otherwise,} \end{cases}$$

$$\phi_{2i}(x) = \begin{cases} h\bar{\phi}_4(\frac{x-x_{i-1}}{h}) & x \in [x_{i-1}, x_i] \\ h\bar{\phi}_2(\frac{x-x_i}{h}) & x \in [x_i, x_{i+1}] \\ 0 & \text{otherwise,} \end{cases}$$

and,

$$\phi_{2N-1}(x) = \begin{cases} \bar{\phi}_1(\frac{x-x_{N-1}}{h}) & x \in [x_{N-1}, L] \\ 0 & \text{otherwise,} \end{cases}$$

$$\phi_{2N}(x) = \begin{cases} h\bar{\phi}_4(\frac{x-x_N}{h}) & x \in [x_{N-1}, L] \\ 0 & \text{otherwise,} \end{cases}$$

where,

$$\bar{\phi}_1(\xi) = 1 - 3\xi^2 + 2\xi^3, \qquad \bar{\phi}_2(\xi) = \xi(\xi-1)^2$$

,

$$\bar{\phi}_3(\xi) = 3\xi^2 - 2\xi^3, \qquad \bar{\phi}_4(\xi) = \xi^2(\xi-1)$$

```
basis1D (generic function with 1 method)
```

```
phi,phid,phidd= basis1D(xh,N);
```

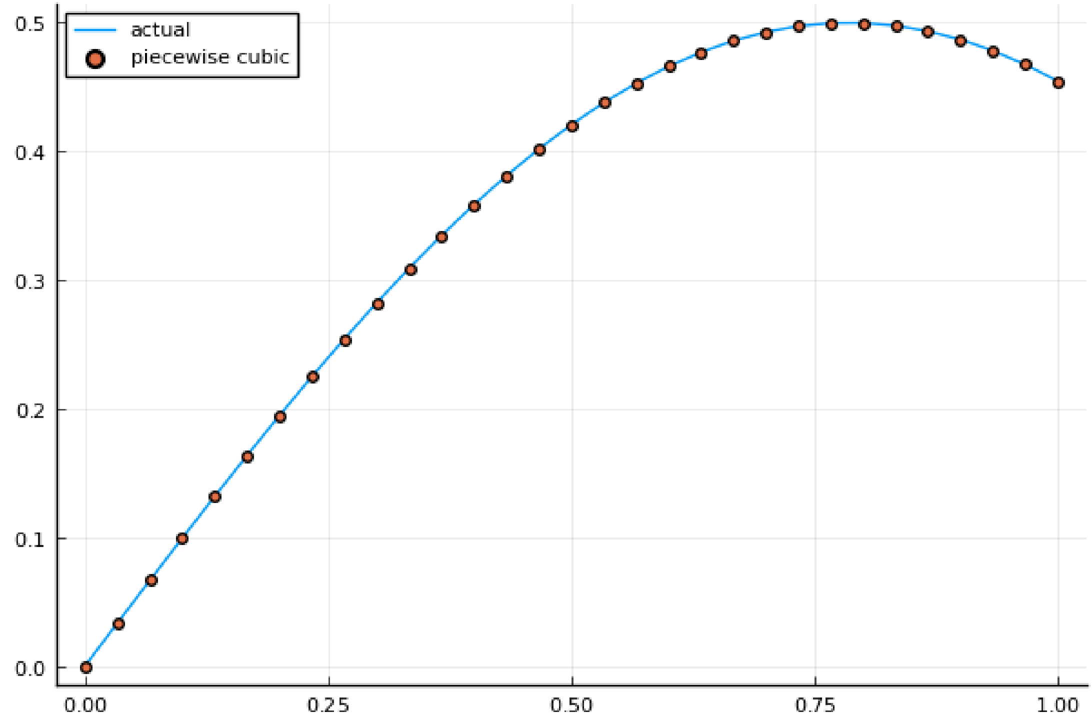
## Do you want to check whether our basis functions can approximate a function?

```
f (generic function with 1 method)
```

```
f(x)=cos(x)*sin(x) ##Give a function
```

Check ☑

```
check (generic function with 1 method)
```




# 3.3 Finite Element matrices and vectors

We consider a spatial discretizaton of $\Omega$ into "Finite Elements" $n_e = N-1$, which are disjoint subdomains of, $\Omega = \bigcup_{e=1}^{n_e}\Omega^e$ where, $\Omega^e : (x_i, x_{i+1})\ \&\ i = 1, ..n_e$. We write the weak form over interval $\Omega$ as the sum of contributions from each subdomain, $\Omega^e$ [Jog15].

Hence, (5) and (6) are reformulated as,

$$\sum_{k=1}^{2N}\Big(\sum_{e=1}^{n_e}\overbrace{\int_{x_i}^{x_{i+1}} EI\phi_k''\phi_j''}^{S_e}\Big)w_k = \sum_{e=1}^{n_e}\Big(\overbrace{\int_{x_i}^{x_{i+1}} q\phi_j}^{F_e}\Big) + Q_L\phi_j(L) - Q_0\phi_j(0) + M_L\phi_j'(L) - M_0\phi_j'(0) \qquad (7)$$

and for the dynamic case,

$$\sum_{k=1}^{2N}\Big(\sum_{e=1}^{n_e}\big(\overbrace{\int_{x_i}^{x_{i+1}} \mu\phi_k\phi_j}^{M_e} + \int_{x_i}^{x_{i+1}} EI\phi_k''\phi_j''\big)\Big)w_k = \Big(\sum_{e=1}^{n_e}\int_{x_i}^{x_{i+1}} q\phi_j\Big) + Q_L\phi_j(L) - Q_0\phi_j(0) + M_L\phi_j'(L) - M_0\phi_j'($$

Let us compute these "local/element" contributions now.

**Note:** We assume here, for simplicity that point loads on the beam can only occur on the boundaries

```
Plots.PyPlotBackend()
```

```
element_mass (generic function with 1 method)
```

```
function element_mass(mu_e,x_e,phi_e)
    dofs = length(phi_e)
    M_e = zeros(dofs,dofs)
    gp = [-sqrt((3/7) - (2/7)*sqrt(6/5)),sqrt((3/7) - (2/7)*sqrt(6/5)),-sqrt((3/7) +
(2/7)*sqrt(6/5)),sqrt((3/7) + (2/7)*sqrt(6/5))]
    gw = [(18+sqrt(30))/36,(18+sqrt(30))/36,(18-sqrt(30))/36,(18-sqrt(30))/36]
    h=x_e[2]-x_e[1]

    for i=1:length(gp)
        p=(0.5*gp[i]+0.5)*h + x_e[1]
```

```
            for j=1:dofs
                for k=1:dofs
                    M_e[j,k] += (0.5*mu_e*gw[i]*phi_e[j](p)*phi_e[k](p))
                end
            end
        end
        return M_e
    end
```

```
element_stiffness (generic function with 1 method)
```

```
    function element_stiffness(EI_e,x_e,phidd_e)
        dofs = length(phidd_e)
        K_e = zeros(dofs,dofs)
        gp = [-1/sqrt(3),1/sqrt(3)]
        gw = [1,1]
        h=x_e[2]-x_e[1]
        for i=1:length(gp)
            p=(0.5*gp[i]+0.5)*h + x_e[1]
            for j=1:dofs
                for k=1:dofs
                    K_e[j,k] += (0.5*(EI_e/h^3)*gw[i]*phidd_e[j](p)*phidd_e[k](p))
                end
            end
        end
        return K_e
    end
```

```
element_forces (generic function with 1 method)
```

```
    function element_forces(q_e,x_e,phi_e)
        dofs =length(phi_e)
        f_e = zeros(dofs)
        gp = [-sqrt(3/5),0,sqrt(3/5)]
        gw = [5/9,8/9,5/9]
        h=x_e[2]-x_e[1]
        for i=1:length(gp)
            p=(0.5*gp[i]+0.5)*h + x_e[1]
            for j=1:dofs
                f_e[j]+= (0.5*q_e*h*gw[i]*phi_e[j](p))
            end
        end
        return f_e
    end
```

In our implementations,for the $e^{th}$ element `element_mass()` calculates $M_e$, `element_stiffness()` calculates $S_e$ and `element_forces()` calculates $F_e$. We use appropriate Gauss quadrature integration rules to compute the matrices.

# 3.4 Assembly

The global stiffness matrix $S$ that is used to solve the system, is then obtained by an appropriate local to global transformation of the indices. This is kept track by the function `local2global()` in our implementation. Finally the global stiffness matrix is equivalent to, [Kar16]

$$S = \begin{bmatrix} s_{11} & \cdots & s_{12N} \\ \vdots & \ddots & \vdots \\ s_{2N1} & \cdots & s_{2N2N} \end{bmatrix} \quad \text{and} \quad s_{jk} = \int_0^L EI\phi_k''\phi_j''$$

$$S \in \mathbb{R}^{2N \times 2N}$$

is the stiffness matrix and is positive semi-definite for any $w \in \mathbb{R}^{2N}$.

On a similar note, the mass matrix which is used to calculate the load vector(also used in the dynamic case as described further) is formulated as follows. [Kar16]

$$M = \begin{bmatrix} m_{11} & \cdots & m_{12N} \\ \vdots & \ddots & \vdots \\ m_{2N1} & \cdots & m_{2N2N} \end{bmatrix} \quad \text{where,} \quad m_{ij} = \int_0^L \phi_i \phi_j$$

and the load vector is given by $\hat{q} = \begin{bmatrix} \int q\phi_1 \\ \vdots \\ \int q\phi_{2N} \end{bmatrix}$,

Incorporating these into the linear system results in the following extensions for the cantilever beam and for the simply supported beam, respectively.

## Static case

$$\begin{bmatrix} S & e_0 & d_0 \\ e_0^T & 0 & 0 \\ d_0^T & 0 & 0 \end{bmatrix} \begin{bmatrix} w \\ Q_0 \\ M_0 \end{bmatrix} = \begin{bmatrix} q + Q_L e_L + M_L d_L \\ a \\ b \end{bmatrix}$$

$$\begin{bmatrix} S & e_0 & -e_L \\ e_0^T & 0 & 0 \\ -e_L^T & 0 & 0 \end{bmatrix} \begin{bmatrix} w \\ Q_0 \\ Q_L \end{bmatrix} = \begin{bmatrix} q - M_0 d_0 + M_L d_L \\ -a_0 \\ -a_L \end{bmatrix}$$

where, $w = \begin{bmatrix} w_1 \\ \vdots \\ w_{2N} \end{bmatrix}$, $e_x = \begin{bmatrix} \phi_1(x) \\ \vdots \\ \phi_{2N}(x) \end{bmatrix}$, $\quad d_x = \begin{bmatrix} \phi_1'(x) \\ \vdots \\ \phi_{2N}'(x) \end{bmatrix}$

For all x: $w_h(x) = \mathbf{e}_x^T \mathbf{w}$, $w_h'(x) = \mathbf{d}_x^T \mathbf{w}$. [Kar16]

Function `assemble()` creates this linear equation system for the static case using sparse matrices.

```
assemble (generic function with 1 method)
```

```julia
function assemble(t,EI,Q,ML,qv,a,b,xh,ne,conn,phi,phidd,dofs)
    ndofs=length(phi);
    nphi= length(dofs[1]);
    q = zeros(ndofs)
    ii = zeros(Int64, ne, nphi, nphi); # sparse i-index
    jj = zeros(Int64, ne, nphi, nphi); # sparse j-index
    aa = zeros(ne, nphi, nphi); # entry of Galerkin matrix

    for e=1:ne
        sloc= element_stiffness(EI[e],xh[conn[e]],phidd[dofs[e]])
        floc= element_forces(qv[e],xh[conn[e]],phi[dofs[e]])
        q[dofs[e]]+=floc[:]
        for j=1:nphi
            for k=1:nphi
                ii[e,j,k] = dofs[e][j]; # local-to-global
                jj[e,j,k] = dofs[e][k]; # local-to-global
                aa[e,j,k] = sloc[j,k];
            end
        end
    end
    S = sparse(ii[:],jj[:],aa[:]);

```

```
    #BCs
    e0 = [phi[i](0) for i=1:ndofs];
    eL = [phi[i](L) for i=1:ndofs];
    d0 = [phid[i](0) for i=1:ndofs];
    dL = [phid[i](L) for i=1:ndofs];

    if t=="c"
        C = [e0 d0];
        q_e = [q + Q*eL + ML*dL; a; b];
    elseif t=="ss"
        C = [e0 -eL];
        q_e = [q - Q*d0 + ML*dL; a; -b];
    end

    #Extended matrix system with BCs
    S_e = [S C; C' zeros(2,2)];

    return S_e,q_e;
end
```

# Dynamic Case

For the dynamic case `dyn_assemble()` yields the following DAEs for a cantilever and a simply supported beam respetively,

$$\begin{bmatrix} M & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \ddot{\underline{w}} \\ \ddot{Q}_0 \\ \ddot{M}_0 \end{bmatrix} + \begin{bmatrix} S & \underline{e}_0 & \underline{d}_0 \\ \underline{e}_0^\top & 0 & 0 \\ \underline{d}_0^\top & 0 & 0 \end{bmatrix} \begin{bmatrix} \underline{w} \\ Q_0 \\ M_0 \end{bmatrix} = \begin{bmatrix} \underline{q} + Q_L \underline{e}_L + M_L \underline{d}_L \\ a \\ b \end{bmatrix}$$

$$\begin{bmatrix} M & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \ddot{\underline{w}} \\ \ddot{Q}_0 \\ \ddot{M}_0 \end{bmatrix} + \begin{bmatrix} S & e_0 & -e_L \\ e_0^T & 0 & 0 \\ -e_L^T & 0 & 0 \end{bmatrix} \begin{bmatrix} w \\ Q_0 \\ Q_L \end{bmatrix} = \begin{bmatrix} q - M_0 d_0 + M_L d_L \\ -a_0 \\ -a_L \end{bmatrix}$$

[Kar16]

```
dyn_assemble (generic function with 1 method)
```

```
function dyn_assemble(t,mu,EI,Q,ML,qv,a,b,xh,ne,conn,phi,phidd,dofs)
    ndofs=length(phi);
    nphi= length(dofs[1]);
    q = zeros(ndofs)
    ii = zeros(Int64, ne, nphi, nphi); # sparse i-index
    jj = zeros(Int64, ne, nphi, nphi); # sparse j-index
    aa = zeros(ne, nphi, nphi); # entry of Galerkin matrix
    bb = zeros(ne, nphi, nphi); #entry of mass matrix
    for e=1:ne
        sloc= element_stiffness(EI[e],xh[conn[e]],phidd[dofs[e]])
        mloc= element_mass(mu[e],xh[conn[e]],phi[dofs[e]])
        floc= element_forces(qv[e],xh[conn[e]],phi[dofs[e]])
        q[dofs[e]]= q[dofs[e]]+floc[:]
        for j=1:nphi
            for k=1:nphi
                ii[e,j,k] = dofs[e][j]; # local-to-global
                jj[e,j,k] = dofs[e][k]; # local-to-global
                aa[e,j,k] = sloc[j,k];
                bb[e,j,k] = mloc[j,k];
            end
        end
    end
    S = sparse(ii[:],jj[:],aa[:]);
    M = sparse(ii[:],jj[:],bb[:]);

```

```
    #BCs
    e0 = [phi[i](0) for i=1:ndofs];
    eL = [phi[i](L) for i=1:ndofs];
    d0 = [phid[i](0) for i=1:ndofs];
    dL = [phid[i](L) for i=1:ndofs];

    if t=="c"
        C = [e0 d0];
        q_e = [q + Q*eL + ML*dL; a; b];
    elseif t=="ss"
        C = [e0 -eL];
        q_e = [q - Q*d0 + ML*dL; a; -b];
    end

    #Extended matrix system with BCs
    S_e = [S C; C' zeros(2,2)];
    M_e = [M 0*C; 0*C' zeros(2,2)];

    return S_e,M_e,q_e;
end
```

The above DAEs are solved using the **Newmark's algorithm**. [Kar20] We list the algorithm for our case:

**Algorithm 1**: Newmark's method

Algorithm parameters: step size $\tau$, $\beta \in [0, 0.5]$, $\gamma \in [0, 1]$
Initialize $w_0 = w(t_0),\ \ \ddot{w}_0 = \ddot{w}(t_0)$
For $j = 0$ to $T$:
  Compute: $w_j^* = w_j + \dot{w}_j\tau_j + (0.5 - \beta)\ddot{w}\tau_j^2$
  Compute the solution $\ddot{w}_{j+1}$ using, $(M + \beta\tau_j^2 S)\ddot{w} = f - Sw_j^*$
  Compute $w_{j+1} = w_j^* + \beta\ddot{w}_{j+1}\tau_j^2$
End

The parameter values were chosen to be $\beta = 1/4$ and $\gamma = 1/2$, as specified in [Kar20]. Function `newmark_step()` implements this time stepping algorithm for us.

```
newmark_step (generic function with 1 method)
```

```
function newmark_step(Mtau,xh,w,tau,beta,gamma,M_e,S_e,phi)
    w_interp = zeros(length(xh),Mtau);
    udot = 0*w;
    udot[end-1:end] =w[end-1:end];
    udotdot =  0*w;
    udotdot[end-1:end] =w[end-1:end];
    for i = 1:Mtau
        ustar = w + udot*tau + (1/2 - beta).*udotdot.*tau^2;
        ustardot= udot + (1-gamma)*tau*udotdot;
        b= -S_e*ustar;
        udotdot = (M_e + (beta*tau^2)*S_e)\b;
        w = ustar + beta*tau^2*udotdot;
        udot = ustardot + gamma*tau*udotdot;

        w_interp[:,i] = sum((w[2*j-1]*phi[2*j-1].(xh) + w[2*j]*phi[2*j].(xh)) for j=1:length(xh));
    end
    return w_interp
end
```

## 3.5 Solving the system

```
begin
    if analysis=="static"
        S_e,q_e = assemble(t,EI,Q,ML,qv,a,b,xh,ne,conn,phi,phidd,dofs);
        w = S_e\q_e;
        w_interp = sum((w[2*j-1]*phi[2*j-1].(xh) + w[2*j]*phi[2*j].(xh)) for j=1:N);

    elseif analysis=="dynamic"
        S_e,M_e,q_e = dyn_assemble(t,mu,EI,Q,ML,qv,a,b,xh,ne,conn,phi,phidd,dofs);
        w = S_e\q_e;
        w_interp = newmark_step(Mtau,xh,w,tau,beta,gamma,M_e,S_e,phi);
    end
    nothing
end
```

# 4 Results and Post Processing

Here we present some pre compiled results of our implementation. The GUI explained below can be used for plotting other results.

## 4.1 The Static Case

We compare our computed FEM results with the closed form solutions obtained from literature. [Tim40], [Wik20] We investigate different loading scenarios and find that our computed FEM solution matches the closed form solutions accurately. Figures [2-7] show the corresponding results.

**Cantilever End load**

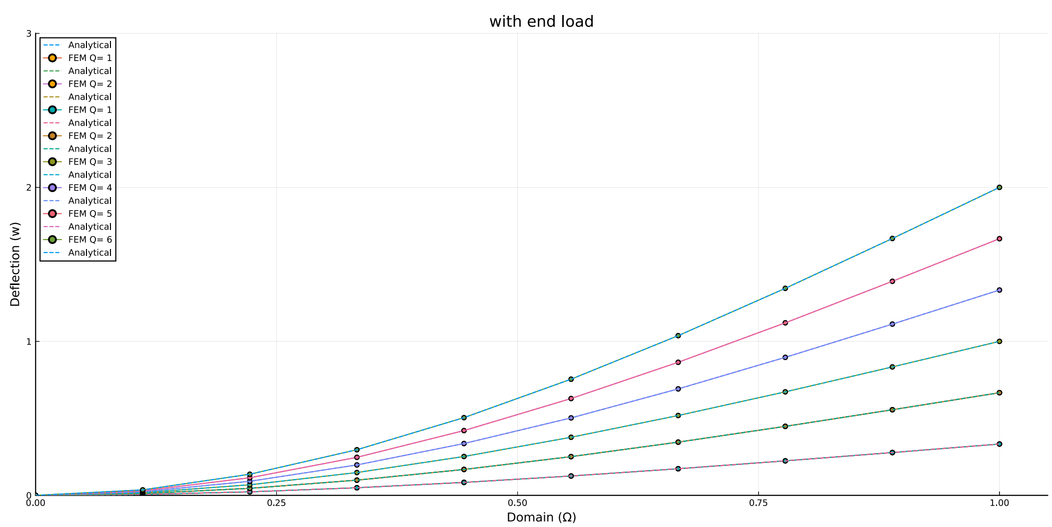



Fig 2: Cantilever end load, exact solution: $w(x) = (\frac{Q_L x^2}{6EI})(3L - x)$

**Cantilever End Moment**

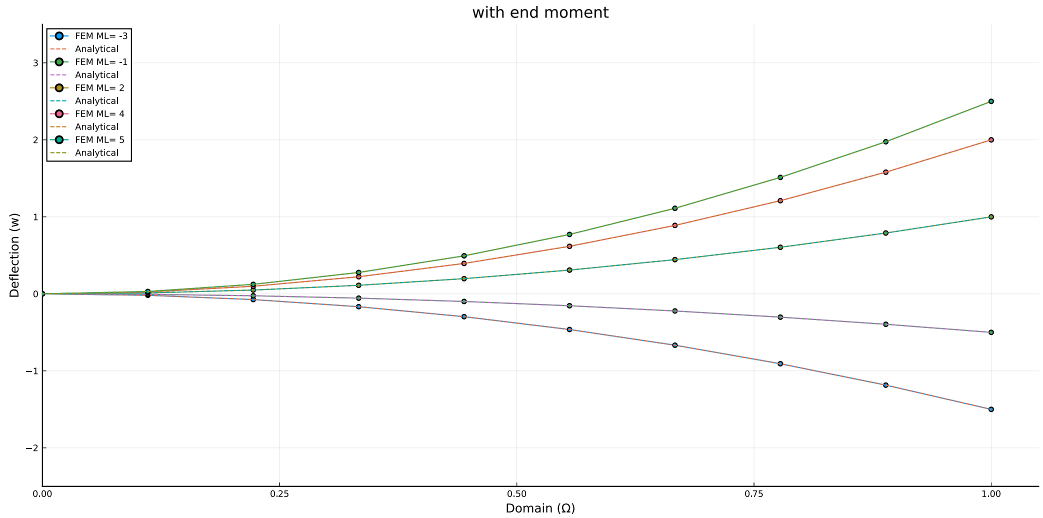



Fig 3: Cantilever end load, exact solution: $w(x) = \frac{M_L x^2}{2EI}$

### Cantilever Distributed Load

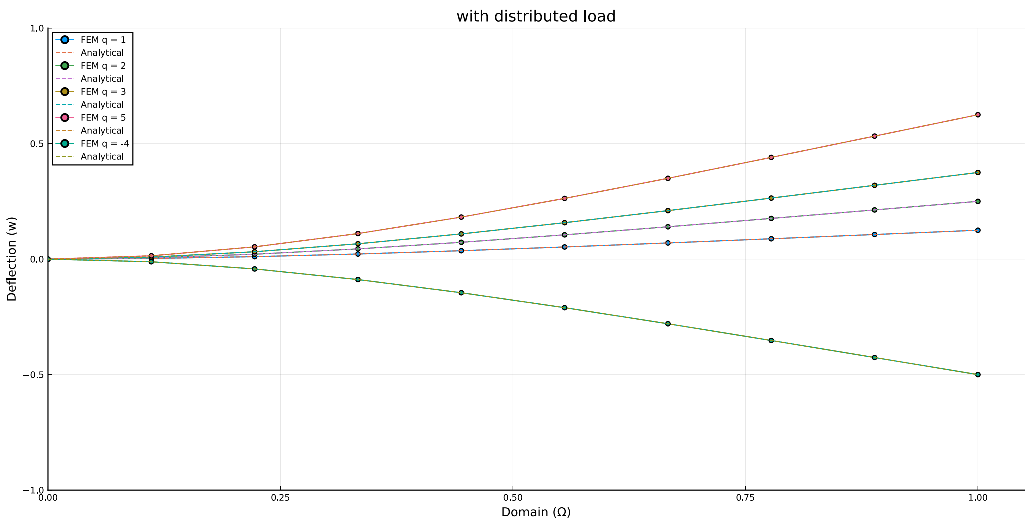



Fig 4: Cantilever dist load, exact solution: $w(x) = \frac{qx^2}{24EI}(6L^2 - 4Lx + x^2)$

### Cantilever End Moment and Load

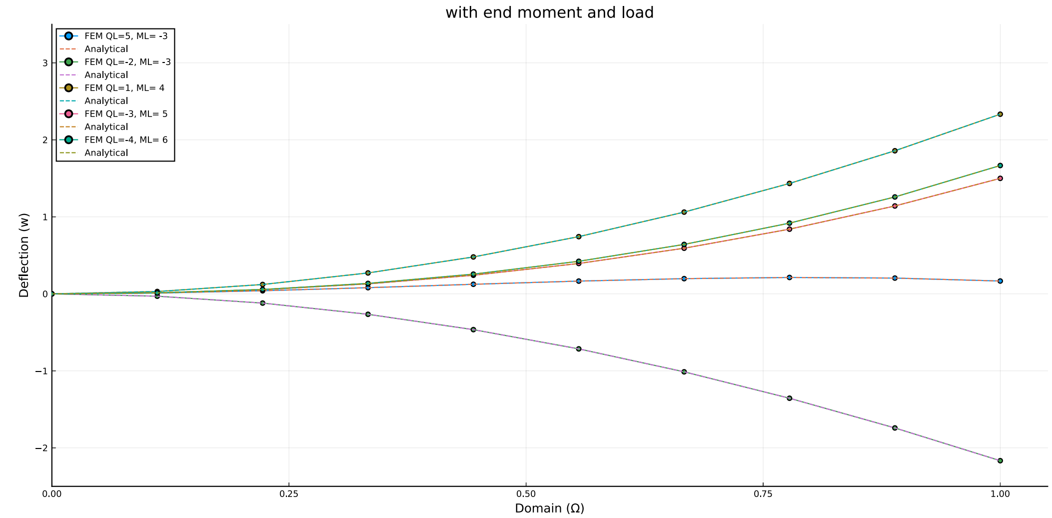



Fig 5: Cantilever combined moment and load at tip, exact solution: $w(x) = \frac{x^2}{6EI}(3M_L + 3LQ_L - Q_L x)$

### Simply Supported Beam with equal end moments

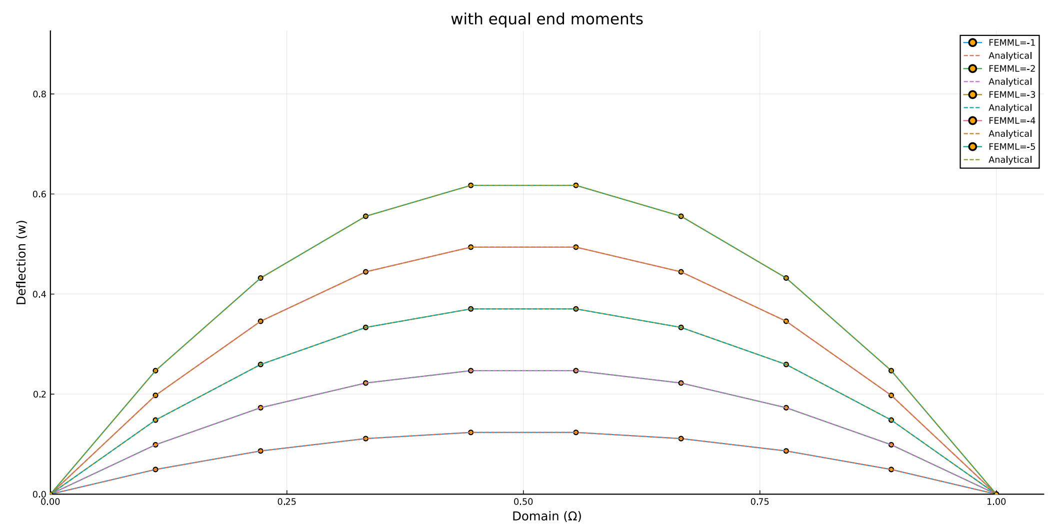



Fig 6: Simply Supported with equal end moments, exact solution: $w(x) = \frac{M_L x}{2EI}(L - x)$

### Simply Supported Beam with distributed loading

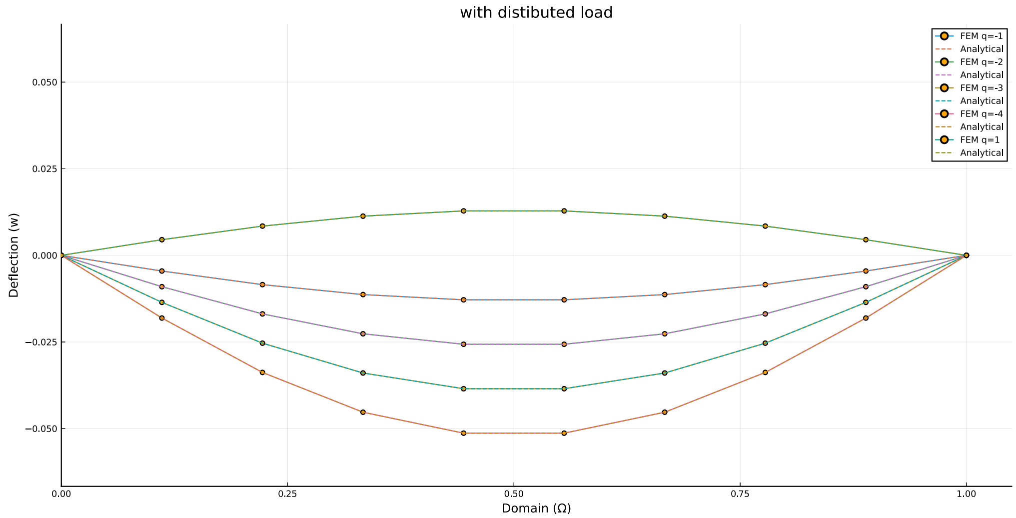



Fig 7: Simply Supported with uniform loading, exact solution: $w(x) = \frac{qx}{24EI}(L^3 - 2Lx^2 + x^3)$

# 4.2 The dynamic case

The dynamic equation is solved and plotted over time for both the cantilever and the simply supported beam.(See movie) The initial condition $w_0$ (Ref: (2)) is inputed through the computed solution of the static equation. We observe that due to the absence of a disspation term, the vibration movement of the beam will be non-damped over time. The movement of the beam will not stop, but also due to the fact that there is no external loading, it will be harmonic.[Jog15]]

**Computation of eigenmodes**

Consider this special case of undamped force free matrix equation,

$$M\ddot{w} + Sw = 0 \tag{9}$$

In the absence of a transverse load, we have the free vibration equation. This equation can be solved using a Fourier decomposition of the displacement into the sum of harmonic vibrations of the form $w = \bar{w}e^{i\omega t}$. Observe that (9) can be then rewritten as a Eigenvalue problem,

$$\left(M - \omega^2 S\right)\bar{w} = 0 \tag{10}$$

Every $j^{th}$ eigenvalue $\lambda_j = \omega_j^2$ can give us the j-th natural frequency $\omega_j$. The corresponding eigenvector $w_j$ can be used to calculate the displacement curve, called the mode shape.

Solutions to the undampened forced problem have unbounded displacements when the driving frequency matches a natural frequency $\omega_j$, i.e., the beam can resonate. The natural frequencies of a beam therefore correspond to the frequencies at which resonance can occur.[Jog15],[Wik20]

We plot the mode shape and list the corresponding natural frequency for the cantilever and the simply supported beams.

**Cantilever**

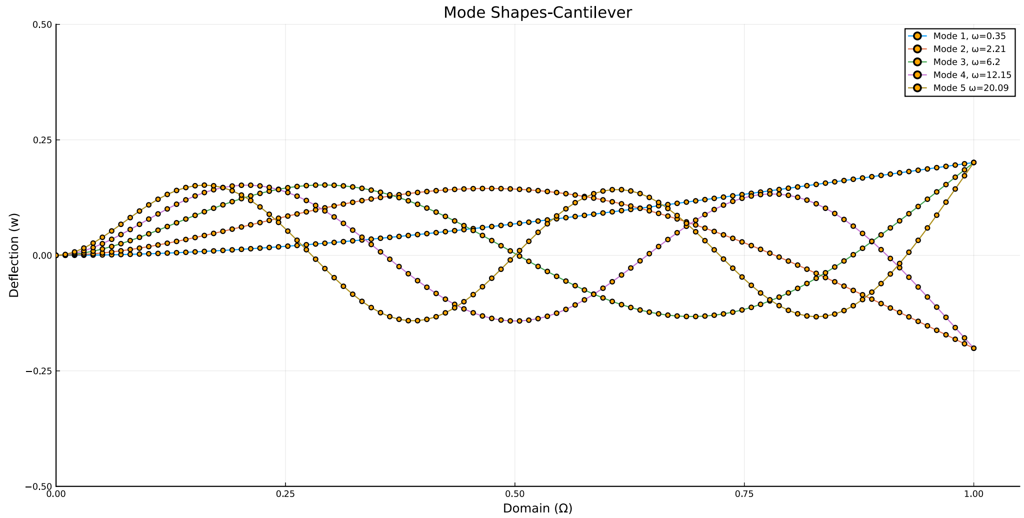



Fig 8: Mode shapes along with natural frequencies $\omega$ for a cantilever beam

**Simply Supported**

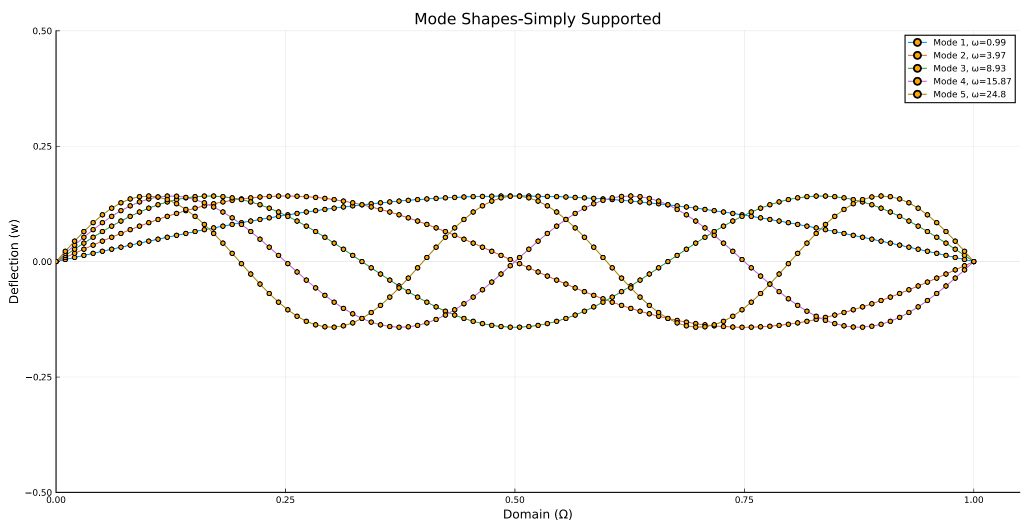



Fig 9: Mode shapes along with natural frequencies $\omega$ for the Simply Supported beam

# Parameters GUI

**In this section, the properties and loading characterstics of the beam can be selected in order to find the solution of the case that the user is interested in. This will accordingly update the code and the Notebook will calculate and display the solution for the chosen case.**

Analysis: Dynamic Type: Simply Supported

Length of the beam (L): 1 10

Number of nodes (N): 2 100

Bending modulus (EI) = 1

μ Mass/length = 1

**Boundary Conditions for a Simply Supported beam**

a0 = 0

aL = 0

Moment at x=0 (M0) = -1

Moment at x=L (ML) = 1

**Dynamic pameters**

Total time steps (T) = 200

stepping(τ) = 0.1

β = 0.25

γ = 0.5

UDL (q) = 0

### Let us analyze the results!

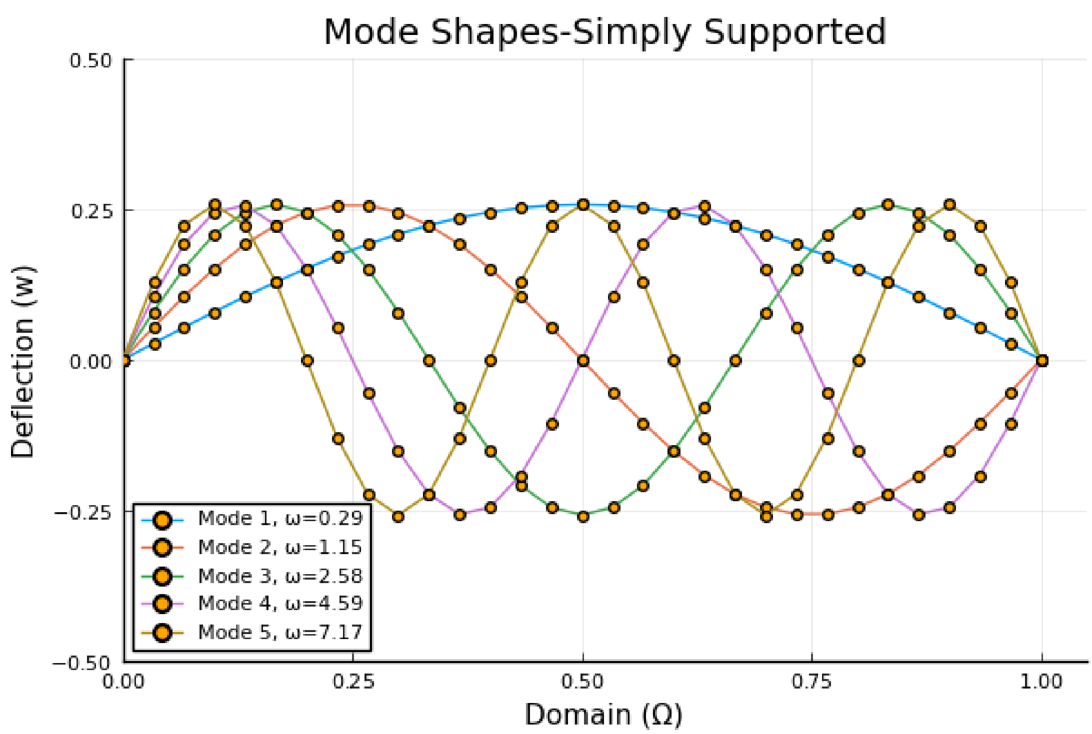
```
plotFEM (generic function with 4 methods)
```

```
check_analytical (generic function with 1 method)
```

Select the static case to see the static plots!

# The Dynamic Case - Free and Undamped Vibrations

# Plotting the first 5 modes



**Create Movie** ☐

```
"select the dynamic case to see this clip"
```

# 5 Conclusions

We have analyzed the Finite Element framework for solving the tranverse deflections for a Euler Bernoulli beam model under various boundary conditions. Our resulting finite element approximations on different static examples presented here, helps reproduce the standard closed form solutions that have been derived in the literature. We note that for free undamped vibrations, a harmonic motion is observed. As a logical extension, the first few natural frequencies of the beam could be calculated from the generalized eigenvalued problem that arose from this case. Finally, Graphical User Interface was developed for the user to create and analyze their own custom examples. While there is no limitation in our FEM implementation on either the magnitude or direction of the loads, it must be noted that we made the assumption that the beam can have point loads only on the tips. Furthermore, the beam is considered to be prismatic with uniform bending stiffness and mass density. In the dynamic example, we restricted ourselves to undamped free vibrations. It is of course relevant, to analyze further and look at damped and forced vibrations. A good extension to study Euler Bernoulli beams further will be to explore the above mentioned areas. Our model provides a basis on which this can be done.

# References

[Tim40]:

> S. Timoshenko. *Strength of Materials*, 2nd edition, Chapter 5: Deflection of Transversely loaded beams, 1940.

[Kar16]:

> Micheal Karow, *Bending of Bernoulli beams and FEM*.

[Kar20]:

> Micheal Karow, *The Newmark Method*.

[Jog15]:

> C.S Jog, *Introduction to the Finite Element Method*, Lecture Notes, Indian Institute of Science, 2015.

[Wik20]:

> Wikipedia contributors. (2020, August 28). Euler–Bernoulli beam theory. In Wikipedia, The Free Encyclopedia. Retrieved 14:10, October 1, 2020, from https://en.wikipedia.org/w/index.php?title=Euler%E2%80%93Bernoulli*beam*theory&oldid=975370990