

Dynamics of An Elastic Structural Beam With the **MATLAB** **pdepe** Function

Bill Greene

Version 0.2, July 23, 2018

This example shows how the structural dynamics of a beam can be analyzed with **pdepe**. The classical PDE for transient deflections of a beam is second order in time and fourth order in space. The form of PDE that **pdepe** accepts is first order in time and second order in space. One of the specific goals of this example is to show how the beam equation can be converted to a form acceptable to **pdepe**.

The classical equation for the deflection of a beam is

$$\frac{\partial^2}{\partial x^2}(EI \frac{\partial^2 w}{\partial x^2}) + N \frac{\partial^2 w}{\partial x^2} + \rho A \frac{\partial^2 w}{\partial t^2} = F \quad (1)$$

The variables in this equation are:

- w Transverse deflection of the beam
- x Coordinate along the beam axis
- t Time
- E Modulus of elasticity of the material
- I Moment of inertia of the beam cross section
- N Prescribed axial force in the beam
- ρ Density of the material
- A Cross sectional area of the beam
- F Distributed transverse loading on the beam

To convert this equation into a form acceptable to **pdepe**, we first define the following auxiliary variables

$$\frac{\partial^2 w}{\partial x^2} = K \quad (2)$$

and

$$\frac{\partial w}{\partial t} = v \quad (3)$$

With these two definitions equation (1) can be rewritten as this system of three PDE

$$\begin{pmatrix} \dot{w} \\ \rho A \dot{v} \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ -EI \frac{\partial^2 K}{\partial x^2} - N \frac{\partial^2 w}{\partial x^2} \\ -\frac{\partial^2 w}{\partial x^2} \end{pmatrix} + \begin{pmatrix} V \\ F \\ K \end{pmatrix} \quad (4)$$

The dependent variables in this system are

$$u = \begin{pmatrix} w \\ V \\ K \end{pmatrix} \quad (5)$$

1 MATLAB Code

1.1 PDE Definition

`pdepe` requires that the user write a function with the following signature to define the system of PDE.

```
[c,f,s] = pdefun(x,t,u,dudx)
```

A `pdefun` function defining equations (4) and (5) takes

1.2 Boundary Conditions

`pdepe` requires that the user write a function with the following signature to define the boundary conditions at the left and right ends of the spatial region.

```
[pl,ql,pr,qr] = bcfun(xl,ul,xr,ur,t)
```

In the structural analysis of beams, three types of boundary conditions are frequently considered: free, simple support, and clamped. The two ends of the beam may have any combination of these three basic types.

2 Examples

2.1 Free Vibration of a Simply Supported Beam

As a simple example we will consider a beam that is initially displaced but is otherwise unloaded and has simple support boundary conditions at both ends. The initial displacement is chosen to be a half sin wave over the length of the beam. This is the eigenvector for the lowest vibration frequency and this allows for a particularly simple analytical solution. We will compare the analytical solution with the solution obtained from `pdepe`.

The complete listing of the `MATLAB` code for this example is shown below.

```
function beamFreeVibration
E=200e9;
thick = .2;
width = .1;
I = width*thick^3/12;
EI=E*I;
A=thick*width;
L=10;
xMidPoint = L/2;
rho=7700; % density of steel, kg/m^3
m = 0;
amp=.1;
numElems=20; % need a node in the middle of the beam
elemLength = L/numElems;
numNodes = numElems + 1;

x = linspace(0,L,numNodes);
t=linspace(0, .75, 100);
%figure; plot(x, wa); grid;
pde = @(x,t,u,DuDx) beampde(x,t,u,DuDx, EI, rho, A);
bc = @(xl,ul,xr,ur,t) beambc(xl,ul,xr,ur,t);
ic = @(x) beamic(x, L, amp, EI);
```

```

tic
if(1)
options.vectorized = 'on';
sol = pde1d(m,pde,ic,bc,x,t, options);
else
sol = pdepe(m,pde,ic,bc,x,t);
end
toc
% plot the results

midNode = int32(numElems/2 + 1);
uMidPt = sol(:, midNode, 1);
max(uMidPt)
uAnalVibr = analFreeVibr(L, EI, rho*A, amp, x, t);
figure; plot(t, uMidPt, t, uAnalVibr(:,midNode), 'o'); grid;
figure; plot(x,sol(end,:,1), x, uAnalVibr(end, :), 'o'); grid;
s1 = sol(:,:,1);
max(abs(s1(:)-uAnalVibr(:)))
%uAnalVibr-s1
assertElementsAlmostEqual(uAnalVibr, s1, 'absolute', .05);
end

function [cr,fr,sr] = beampde(x,t,u,DuDx, EI, rho, A)
nx = length(x);
o=ones(1,nx);
z=zeros(1,nx);
cr = [o ; rho*A*o; z];
fr = [z; -DuDx(3,:); -EI*DuDx(1,:)];
f=1;
sr = [u(2,:); z; u(3,:)];
end

function [cr,fr,sr] = beampdeX(x,t,u,DuDx, EI, rho, A)
nx = length(x);
o=ones(1,nx);
z=zeros(1,nx);
cr = [o ; rho*A*o; z];
fr = [z; -EI*DuDx(3,:); -DuDx(1,:)];
f=1;
sr = [u(2,:); z; u(3,:)];
end

function u0 = beamic(x, L, amp, EI)
% half sin wave initial condition
s = sin(pi*x/L);
u0 = [amp*s; 0; -amp*EI*(pi/L)^2*s];
end

function [pl,ql,pr,qr] = beambc(xl,ul,xr,ur,t)
pl = [ul(1);ul(2);ul(3)];
ql = [0;0;0];
pr = [ur(1);ur(2);ur(3)];

```

```
    qr = [0;0;0];  
end  
  
function u=analFreeVibr(L, EI, rhoA, amp, x, t)  
    omega=(pi/L)^2*sqrt(EI/rhoA);  
    u=amp*cos(omega*t)*sin(pi*x/L);  
end
```