# 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

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

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 |
| $\rho$ | 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

$$\frac{\partial^2 w}{\partial x^2} = K \tag{2}$$

and

$$\frac{\partial w}{\partial t} = \dot{w} = V \tag{3}$$

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

$$\left\{ \begin{array}{c} \dot{w} \\ \rho A\dot{V} \\ 0 \end{array} \right\} = \left\{ \begin{array}{c} 0 \\ -EI\frac{\partial^2 K}{\partial x^2} - N\frac{\partial^2 w}{\partial x^2} \\ -\frac{\partial^2 w}{\partial x^2} \end{array} \right\} + \left\{ \begin{array}{c} V \\ F \\ K \end{array} \right\} \tag{4}$$

The dependent variables in this system are

$$u = \left\{ \begin{array}{c} w \\ V \\ K \end{array} \right\} \tag{5}$$

# 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)
```

The details of the input and output arguments to this function are defined in the pdepe documentation. A pdefun function defining equations (4) and (5) takes the following form

```
c = [1 rho*A 0]';
f = [0 -E*I*DuDx(3)-N*DuDx(1) -DuDx(1)]';
s = [u(2) F u(3)]';
```

## 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. In the code snippets below, $u, p$, and $q$ represent the value of the corresponding variable at either the left or right end.

### 1.2.1 Free

```
p = [0 0 u(3)]';
q = [1 1 0]';
```

### 1.2.2 Simple Support

```
p = [u(1) u(2) u(3)]';
q = [0 0 0]';
```

### 1.2.3 Clamped

```
p = [u(1) u(2) 0]';
q = [0 0 1]';
```

# 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;
F=0;
N=0;
xMidPoint = L/2;
rho=7700; % density of steel, kg/m^3
m = 0;
amp=.1;
numElems=20; % even number of elements gives a node in the middle of the beam
elemLength = L/numElems;
numNodes = numElems + 1;
x = linspace(0,L,numNodes);
t=linspace(0, .75, 100);

% use anonymous functions to pass parameters to functions required by pdepe
pde = @(x,t,u,DuDx) beampde(x,t,u,DuDx, EI, rho, A, F, N);
bc = @(xl,ul,xr,ur,t) beambc(xl,ul,xr,ur,t);
ic = @(x) beamic(x, L, amp);

sol = pdepe(m,pde,ic,bc,x,t);

% plot the results

midNode = int32(numElems/2 + 1);
uMidPt = sol(:, midNode, 1);
uAnalVibr = analFreeVibr(L, EI, rho*A, amp, x, t);
figure; plot(t, uMidPt, t, uAnalVibr(:,midNode), 'o'); grid;
xlabel Time; ylabel Displacement;
title('Displacement␣at␣Beam␣Mid-point␣as␣a␣Function␣of␣Time');
legend('pdepe', 'analytical');
figure; plot(x,sol(end,:,1), x, uAnalVibr(end, :), 'o'); grid;
xlabel x; ylabel Displacement;
title('Beam␣Displacement␣at␣the␣Final␣Time');
legend('pdepe', 'analytical');

end

function [cr,fr,sr] = beampde(x,t,u,DuDx, EI, rho, A, F, N)
cr = [1 rho*A 0]';
fr = [0 -EI*DuDx(3)-N*DuDx(1) -DuDx(1)]';
sr = [u(2) F u(3)]';
end

function u0 = beamic(x, L, amp)
  % half sin wave initial condition
  s = sin(pi*x/L);
```

```
  u0 = [amp*s; 0; -amp*(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
```

The results are shown in the following two figures. As can be seen, agreement between the `pdepe` and analytical solutions is very good.