MEMS 1060: Homework #9

Due on April 15, 2021 at 6:00pm

Professor Sammak Th 6:00PM

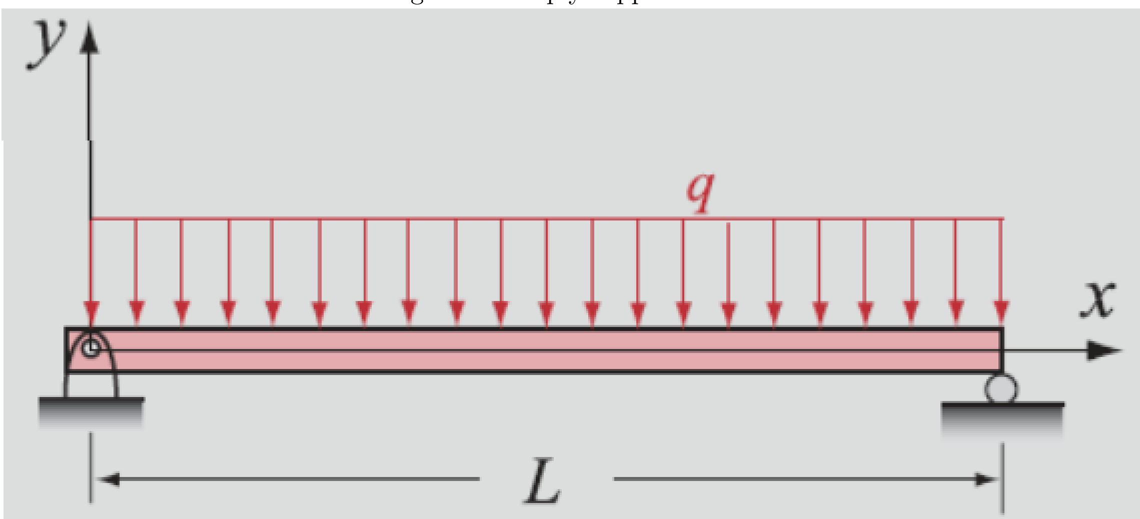
Made using LaTeX/Inkscape. Source files available upon request.

Shane Riley

Problem 1

Given a simply supported beam with a uniform distributed load along the entire length, determine and plot the deflection y, $\frac{dy}{dx}$, and $\frac{d^2y}{dx^2}$.

Figure 1: Simply supported beam.



Note: y' denotes the x-derivative of y.

$$EIy'' = [1 + (y')^2]^{1.5} * 0.5q(L * x - x^2)$$
$$y(0) = y(L) = 0$$
$$EI = 1.4 * 10^7 [Nm^2]$$
$$q = 10000[N/m]$$

Solution

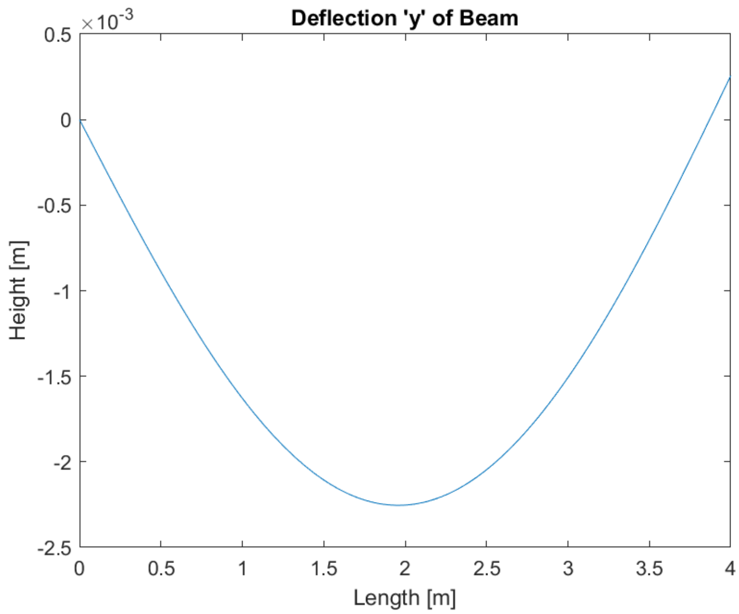
To determine a solution, we first recognize this to be a two-point boundary value problem, governed by a 2nd order, non-homogenous, nonlinear ODE. Since the problem is nonlinear, we will use the shooting method to solve to save some trouble. In the shooting method, we handle the problem as an IVP by defining a state vector and finding the derivative of that state in terms of state and input information. Expressing the differential equation as equal to $\frac{d^2y}{dx^2}$ is trivial, which makes this dimplr:

$$s = \begin{bmatrix} y \\ \frac{dy}{dx} \end{bmatrix}$$
$$\frac{ds}{dt} = \begin{bmatrix} \frac{dy}{dx} \\ \frac{d^2y}{dx^2} \end{bmatrix}$$
$$= \begin{bmatrix} s_2 \\ \frac{1}{EI}(1 + (s_2)^2)^{1.5} * \frac{q}{2}(Lx - x^2) \end{bmatrix}$$

To complete our IVP definition, we need an initial $\frac{dy}{dx}$ in addition to our initial y. By guessing an initial angle for the beam at $x = 0$, we can solve the BVP as if it were an IVP, and iterate our slope until the boundary condition at the far point is satisfied. We will use a convergence criteria of relative error as $\epsilon = 0.001$. We expect the initial $\frac{dy}{dx}$ to be negative but not very large in magnitude, so we will set our first two guesses as $W_1 = 0$ and $W_2 = -5$, and use bisection to iterate W.

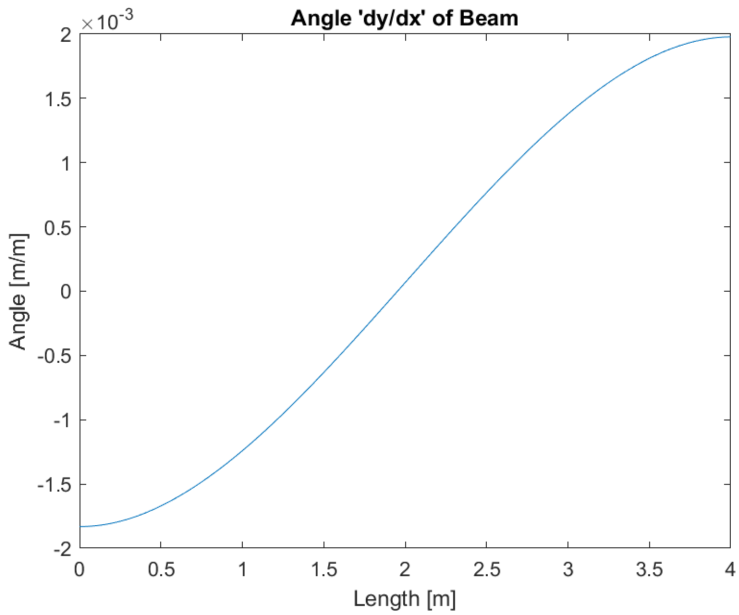
With the bounds, convergence criteria, and step size chosen, the proper slope is found on the 13th iteration. See the plots.

Figure 2: Beam Height.



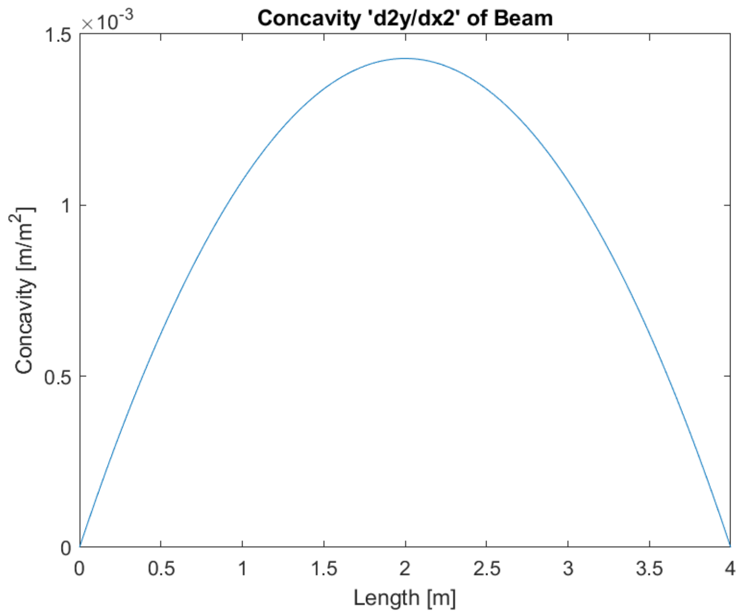

```
%% MEMS 1060 Homework 9
% Author: Shane Riley
% Date: 4/10/2021
format long
clear
clc
close all
%% Problem 1
% Handle a uniform distributed load BVP problem on a beam

EI = 1.4e7;       % [N-m^2]
q = 10000;       % [N/m]
L = 4;            % [m]
y_0 = 0;          % [m]
y_L = 0;          % [m]
h = 0.01;         % [m]
xspan = [0, L]; % [m]

% EQ:
% EI y'' = (1 + y'^2)^1.5 * 0.5q(L*x - x^2)
% (' is an x-derivative)
%
% BC:
```

Figure 3: Beam Angle.




```
% y(0) = y(L) = 0 (pin and roller)

% To solve, use the shooting method
% Convergence criteria and first guesses
epsilon = 0.001;  % [-]
max_iter = 1000;  % [-]
W_H = 0; % [m/m]
W_L = -5;  % [m/m]

% state vector is [y, y'] as 's'
sdot = @(x, s) [...
    s(2)
    (1 + (s(2))^2 )^(1.5)*0.5*q*(L*x - x^2)/EI
];
get_s0 = @(W) [...
    y_0
    W
];
absdiff = @(x1, x2) abs(x1-x2);

% Use average for first guess, and then walk the bounds inward using
% bisection
W = (W_H + W_L)/2;

for i=1:max_iter

    % Run guess
    [t, s] = myEulerExp(sdot, xspan, get_s0(W), h);
```

Figure 4: Beam Concavity.




```

    % Check convergence
    y_end_found = s(end,1);
    if (absdiff(y_end_found, y_L) < epsilon), break; end

    % Set up new W
    if (y_end_found > y_L)
        % Too high, bisect W and W_L
        Wnew = (W + W_L)/2;
        W_H = W;
        W = Wnew;

    else
        % Too low, bisect W and W_H
        Wnew = (W + W_H)/2;
        W_L = W;
        W = Wnew;
    end
end

% Precondition: convergence was reached

% Use final state matrix to compute y'' and add to state
acc = zeros(length(t), 1);
for j=1:length(t)
    state_change = sdot(t(j), s(j,:));
    acc(j) = state_change(2);
end
```

```
s = [s acc];

% Plots
figure(1)
plot(t,s(:,1));
title("Deflection 'y' of Beam");
xlabel("Length [m]");
ylabel("Height [m]");
print("images/figure2",'-dpng');

figure(2)
plot(t,s(:,2));
title("Angle 'dy/dx' of Beam");
xlabel("Length [m]");
ylabel("Angle [m/m]");
print("images/figure3",'-dpng');

figure(3)
plot(t,s(:,3));
title("Concavity 'd2y/dx2' of Beam");
xlabel("Length [m]");
ylabel("Concavity [m/m^2]");
print("images/figure4",'-dpng');

%% Supporting functions

function [t, s] = myEulerExp(sdot, tspan, s0, h)
% MYEULEREXP computes state information across a time interval using
% Explicit Euler
%
%   sdot: column vector as function handle to find change in state
%
%   tspan: row vector holding time span
%
%   s0: column vector holding initial state
%
%   h: step size

% Set up outputs
t = tspan(1):h:tspan(end);
s = zeros(length(t), length(s0));
s(1,:) = s0';

for i = 2:length(t)

    % Step
    time = t(i-1);
    state = s(i-1,:);
    s(i,:) = s(i-1,:) + sdot(time, state)' .* h;
```

```
end

end
```