APPM 2360: Civil Engineering Project 2

October 28, 2014

Souneth Ly (103036031) (L140) (R241)

Zhi Huang (102298433) (L150) (R242)

William Sear (101107069) (L160) (R251)

Abstract

This paper will analyze the behviour of different equations as possible models for two civil engineering applications. In this paper we will investigate the use of linear algebra techniques to find the forces acting on different nodes in a bridge. We then move on to explore both linear algebra and differential equations in order to find the deflection of a beam under uniform loading using a method called "linear shooting".

Contents

Part 1

Distribution of Forces on a Bridge

1.1 Physical Setup

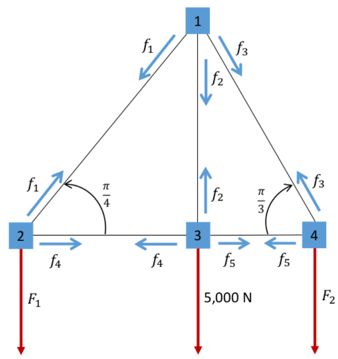


Figure 1: Distribution of Forces on Bridge

Based on the above figure 1 of forces acting on bridge joints we broke the forces into their horizontal and veritcal components to find the equations described in table 1.

Joint	Horizontal	Vertical
1	$\frac{\sqrt{2}}{2}f_1 + \frac{1}{2}f_3 = 0$	$\frac{\sqrt{2}}{2}f_1 + f_2 + \frac{\sqrt{3}}{2}f_3 = 0$
2	$\frac{\sqrt{2}}{2}f_1 + f_4 = 0$	$\frac{\sqrt{2}}{2}f_1 = F_1$
3	$f_4 + f_5 = 0$	$f_2 = 5,000N$
4	$\frac{f_3}{2} + f_5 = 0$	$\frac{\sqrt{3}}{2}f_2 = F_2$

Table 1: Forces Acting on Bridge

Based on the equations defined in the above table and the physical orientation of the joints described in figure 1 we created the below equation shown in table 2. It is important to note that the sign of each term was modified based on the orientation of the force relative to joint 1 where an upward force was positive and a rightward force was positive.

1.2 Linear Model

$$\begin{bmatrix} -\frac{\sqrt{2}}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 \\ -\frac{\sqrt{2}}{2} & -1 & -\frac{\sqrt{3}}{2} & 0 & 0 & 0 & 0 \\ \frac{\sqrt{2}}{2} & 0 & 0 & 1 & 0 & 0 & 0 \\ \frac{\sqrt{2}}{2} & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{2} & 0 & -1 & 0 & 0 \\ 0 & 0 & \frac{\sqrt{3}}{2} & 0 & 0 & 0 & -1 \end{bmatrix} * \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ F_1 \\ F_1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 5,000 \\ 0 \\ 0 \end{bmatrix}$$

$$A\vec{F} = \vec{b}$$

Table 2: Coefficient Matrix with Solution Matrix

In the above matrix equation we can see that A is an $8x7$ matrix while $\overrightarrow{b}$ is a $8x1$ vector. Matlab confirms these results. For the above matrix A it is impossible to calculate the determinant because A is not a square matrix (it has 8 rows but only 7 columns) and as a non-square matrix cannot have a determinant by definition A must not have a determinant.

1.3 Solution to Linear Model

Using the data from table 2 above it was possible to simplifiy the augmented matrix $[A|\overrightarrow{b}]$ to the row reduced echelon form (RREF) using row operations. The results are shown in the below table 3.

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} -2588.2 \\ 5000 \\ -3660.3 \\ 1830.1 \\ 1830.1 \\ -1830.1 \\ -3169.9 \\ 0 \end{bmatrix}$$

Table 3: RREF for Given Problem

Based on the table given above we can conclude that the RREF has a rank of 7 as there are 7 pivot columns in the reduced A matrix. We can also note that the above relation must be consistent as each pivot column has a value and the only column without a pivot equals zero, thus indicating that this system has unique solutions. Based on this solution we find the tensions and forces in the below table 4.

$$\begin{bmatrix} f_1 = -2588.2 \\ f_2 = 5000 \\ f_3 = -3660.3 \\ f_4 = 1830.1 \\ f_5 = 1830.1 \\ F_1 = -1830.1 \\ F_2 = -3169.9 \end{bmatrix}$$

Table 4: Tension and Force Values

We know based on the way in which our original matrices were constructed in table 2 that a positive force indicates tension while a negative force indicates compression. As such we can see that $f_1\,and\,f_3$ are compression forces while $f_{2,}\,f_4,\,and\,f_5$ are tension forces. As such we can conclude that f_2 is the largest tension force and f_1 is the largest compression force.

1.4 Solution with Free Force

In order to generalize the solution found in the above calculations to a bridge with some calculable force (F_3) being applied at node 3 we recalculated the values of each force using the same principles as above. The results are shown in table 5 below.

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{\sqrt{2}F_3}{\sqrt{3}+1} \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & F_3 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & -\frac{2F_3}{\sqrt{3}+1} \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & \frac{F_3}{\sqrt{3}+1} \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & \frac{F_3}{\sqrt{3}+1} \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & -\frac{F_3}{\sqrt{3}+1} \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -\frac{\sqrt{3}F_3}{\sqrt{3}+1} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Table 5: RREF for Free Variable

These values are consistent in sign with the values found in the previous section so we conclude that these force values are legitimate. F_1 and F_2 being negative is correct because F_1 and F_2 being negative it means that the ground underneath node 2 and node 4 is supporting those nodes. This makes sense as the ground is unlikely to pull the bridge down.

Part 2
Deflection of a Uniform Beam

2.1 Differential Equation for Deflection of a Uniform Beam

$$y^{''} = \frac{S}{EI}y + \frac{Qx}{2EI}(x - \mathbb{L});\ 0 \le x \le \mathbb{L};\ y(0) = 0;\ y(\mathbb{L}) = 0 \tag{1}$$

The above differential equation models the deflection of of a beam with uniform loading assuming S is the stress at the endpoints, E is the modulus of elasticity, I is the central moment of inertia, $\mathbb{L}$ is the length of the beam, and Q is the intensity of the uniform load.

Based on the equation we can conclude that it is second order, linear, non-homogenous, and its coefficients are constant (as no y term is modified by anything other than a constant).

2.2 Breaking Down the Beam Differential Equation

In order to better model the differential equation we break it into the form given in equation 2.

$$y(x) = y_1(x) + Cy_2(x) \tag{2}$$

Based on equation 2 we then define y_1 and y_2 as follows:

$$y_1 = y^{''} = P(x)y^{'} + q(x)y + r(x);\ a \le x \le b,\ y_1(a) = \alpha,\ y_1(b) = 0 \tag{3}$$

$$y_2 = y^{''} = P(x)y^{'} + q(x)y;\ a \le x \le b,\ y_1(a) = 0,\ y_1(a) = 1 \tag{4}$$

Based on definition of y_1 and y_2 we can conclude that y_1 represents a particular solution to the differential equation because there it is a non-homogenous solution due to the $r(x)$ term present. We can then conclude that y_2 is a homogenous solution to the differential equation because it lacks a $r(x)$ term. As we have a paticular solution (y_1) and a homogenous equation (y_2) and the homogenous equation is multiplied by some constant (C) this solution is a form of the nonhomogenous principle.

Based on the equations described above we can find the constant C by substituting the initial conditions as below:

Solution:

$y(x) = y_1(x) + Cy_2(x)$	
$y(a) = y_1(a) + Cy_2(a)$	$y(b) = y_1(b) + Cy_2(b)$
$y(a) = \alpha + C * 0$	$\beta = y_1(b) + Cy_2(b)$
$\alpha = \alpha$	$\beta - y_1(b) = Cy_2(b)$
XXXXX	$C = \frac{\beta - y_1(b)}{y_2(b)}$

Table 6: Solving for C

As can be seen above the value of C is $C = \frac{\beta - y_1(b)}{y_2(b)}$.

2.3 Preparing the Linear Shooting Method

In order to prepare the differential equation so that it can be solved using the linear shooting method we must first create sets of first order differential equations from our second order differential equation.

We do this by substituting $z(x) = y^{'}(x)$ this gives the below sets of equations:

Equation 3	Equation 4
$z(x) = y^{'}(x)$	$z(x) = y^{'}(x)$
$z^{'}(x) = P(x)z(x) + q(x)y + r(x)$	$z^{'}(x) = P(x)z(x) + q(x)y$
$y(a) = \alpha \ z(a) = 0$	$y(a) = 0 \ z(a) = 1$

Table 7: First Order Linear Solution Sets

2.4 Solving Using Linear Shooting

Given the initial conditions $\mathbb{L} = 360in$, Q=5lbs/in, $I = 600in^4$, and $S = 900lbs$ we calculated the deflection of the beam using linear shooting assuming that we used a step size of 12 inches. The graph is shown in the below figure 2.

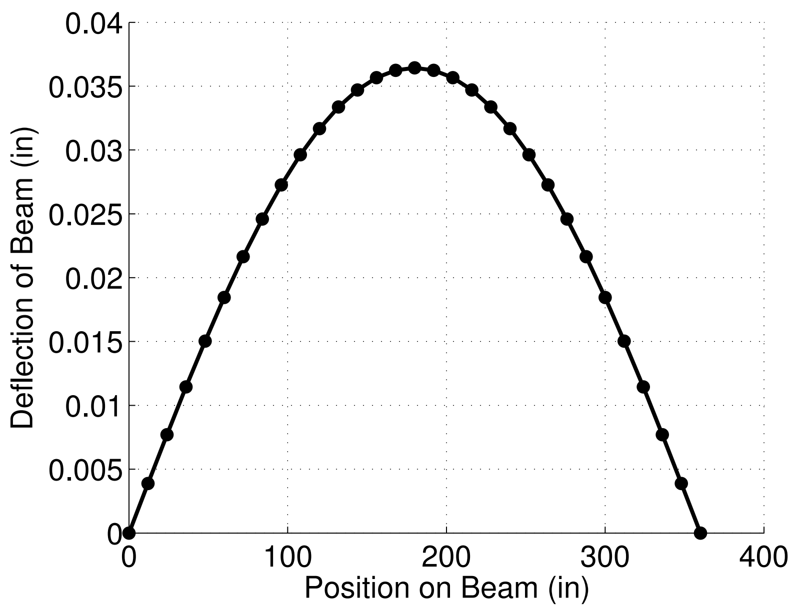


Figure 2: Linear Shooting Solution

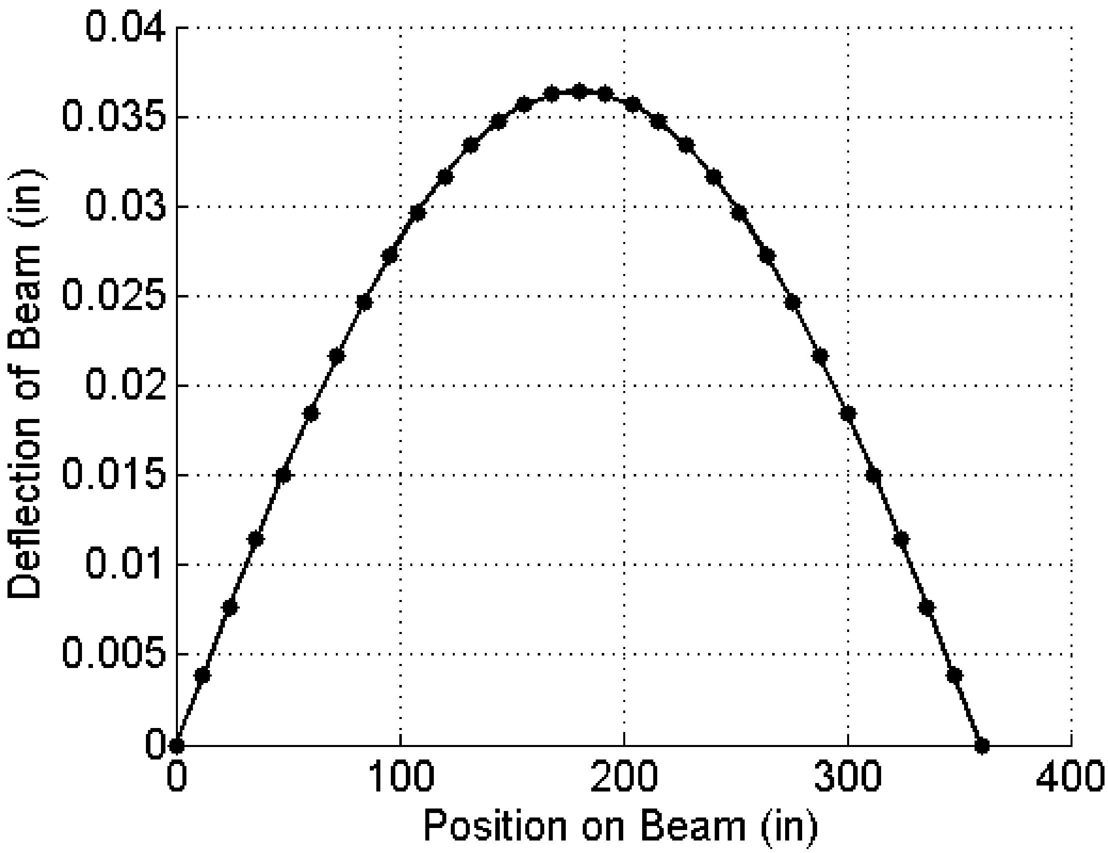
Based on the plot found in figure 2 we calculated the maximum deflection of the beam to be 0.0364 inches at a point 192 inches from the $(0, 0)$ point. This solves the differential equation for the deflection of a beam under uniform loading.

Part 3

Appendix A: Linear Shooting Code:

```
function linearShooting

%Solves the BVP y'' = p(x)y' + q(x)y + r(x), for a<x<b, with the boundary
%conditions y(a)=alpha and y(b)=beta.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%INPUTS.  Change these to adjust for the problem you are solving.
S=900;
L=360;
E=5e7;
I=600;
Q=5;
a = 0;  b = L;             %the endpoints of the interval, a<x<b.
h = 12;                    %space between points on x axis.
alpha = 0;  beta = 0;       %boundary values.  y(a)=alpha, y(b)=beta.
p = @(x) 0;     %continuous function
q = @(x)  S./(E.*I);      %positive continuous function
r = @(x)  Q*x./(2*E.*I).*(x-L);       %continuous function

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Main part of the code.  Solves numerically the two IVP systems with
%ode45, and then combines the results to form the solution y to the BVP.



t = a:h:b;

[ ~, y1 ] = ode45( @odefun1, t, [alpha,0] );
[ ~, y2 ] = ode45( @odefun2, t,     [0,1] );

y1 = y1(:,1);  y2 = y2(:,1);

y = y1 + (beta-y1(end)) / y2(end) * y2;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Plots the numerical solution y

figure(1), clf, hold('on')
plot( t, y, 'k', 'lineWidth', 2 )
[maxDeflectionValue,index] = max(y);
maxDeflectionPosition = index * h;

plot( t, y, 'k.', 'markerSize', 20 )
set( gca, 'fontSize', 15 )
xlabel('Position on Beam (in)'), ylabel('Deflection of Beam (in)')
grid('on')
drawnow, hold('off')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%The two ODE functions that are passed into ode45
```

```
    function u = odefun1(t,y)
        u = zeros(2,1);
        u(1) = y(2);
        u(2) = p(t)*y(2) + q(t)*y(1) + r(t);
    end

    function u = odefun2(t,y)
        u = zeros(2,1);
        u(1) = y(2);
        u(2) = p(t)*y(2) + q(t)*y(1);
    end

end
```



Published with MATLAB® R2013b