

University of California, Los Angeles, CA
Department of Civil & Environmental Engineering

CEE 15 Introduction to Computing for Civil Engineers
Summer Session 2009
Examination 1
Instructor: M. Asghar Bhatti

Name : _____

Problem	Max Points	Points Scored
1	20	
2	15	
3	15	
TOTAL	50	

General notes

Except for the function files, all other scripts for all exam problems must be in a *single* file. Add suitable section headings to clearly identify different problems and various parts in each problem. Include your name and email on the first page. Upload this file, and any associated function files, to the Exam 1 folder on the course web page.

Each student must work independently on the exam problems and must submit their own work for grading.

The only applications allowed during the exam time are suitable text editors and MATLAB. Use of web browsers, chat, or email clients is not allowed. The only exception will be the use of web browser to upload the file after you are done with your exam. Anyone violating this policy will be considered as *cheating on the exam* and will be dealt with using the University's established procedures for handling these cases.

No questions will be answered during the exam period. If a problem statement is confusing to you or some data is missing, make an assumption and clearly state your reasoning in your solution.

Problem 1 — 20 points

- (a) Using appropriate matlab function create a **row vector** a with 16 equally spaced elements with the first element 4 and the last 61.
- (b) Using appropriate matlab command create a **column vector** b in which the first element is 31, the elements decrease by -4 and the last element is -9 .
- (c) Use the matlab's `rand` function to create two random row vectors a and b each with 6 elements. Using these vectors create a 3×2 matrix c that contains element-wise product of entries in a and b .
- (d) Create a 4×5 matrix A and a 5×4 matrix B that contain random numbers between 0 and 10. Using these matrices create a matrix C that contains the product of A and B .

- (a) Using appropriate matlab function create a **row vector** a with 16 equally spaced elements with the first element 4 and the last 61.

```
a=linspace(4,61,16)
```

```
a =
Columns 1 through 4
         4         7.8        11.6        15.4
Columns 5 through 8
        19.2         23         26.8        30.6
Columns 9 through 12
        34.4        38.2         42         45.8
Columns 13 through 16
        49.6        53.4        57.2         61
```

- (b) Using appropriate matlab command create a **column vector** b in which the first element is 31, the elements decrease by -4 and the last element is -9 .

```
>> b=[31:-4:-9]'
```

```
b =
    31
    27
    23
    19
    15
    11
     7
     3
    -1
    -5
    -9
```

- (c) Use the matlab's `rand` function to create two random row vectors a and b each with 6 elements. Using these vectors create a 3×2 matrix c that contains element-wise product of entries in a and b .

```
a = rand(1,6)
b = rand(1,6)
c = reshape(a .* b, 3, 2)
```

```
a =
Columns 1 through 4
    0.011902    0.33712    0.16218    0.79428
Columns 5 through 6
    0.31122    0.52853
```

```
b =
Columns 1 through 4
    0.16565    0.60198    0.26297    0.65408
Columns 5 through 6
    0.68921    0.74815
```

```
c =
    0.0019716    0.51952
    0.20294    0.21449
    0.042649    0.39542
```

(d) Create a 4×5 matrix A and a 5×4 matrix B that contain random numbers between 0 and 10. Using these matrices create a matrix C that contains the product of A and B .

```
A = 10*rand(4,5)
B = 10*rand(5,4)
C = A*B
```

```
A =
Columns 1 through 4
    4.5054    1.5238    0.78176    0.046342
    0.83821    8.2582    4.4268    7.7491
    2.2898    5.3834    1.0665    8.173
    9.1334    9.9613    9.619    8.6869
Column 5
    0.84436
    3.9978
    2.5987
    8.0007
```

```
B =
    4.3141    1.3607    8.5303    0.75967
    9.1065    8.6929    6.2206    2.3992
    1.8185    5.797    3.5095    1.2332
    2.638    5.4986    5.1325    1.8391
    1.4554    1.4495    4.0181    2.3995
```

```
C =
    36.086    25.387    54.285    10.154
    113.13    146.99    129.89    49.753
    86.185    104.8    109.15    37.237
    182.17    214.15    250.37    77.873
```

Problem 2 — 15 points

Deflection of a beam supported by a spring and subjected to triangular loading is given by the following equation.

$$v(x) = -\frac{w_0 x}{360 E I k L} [120 E I L + k(3x^4 - 10L^2 x^2 + 7L^4)]$$

where w_0 is the loading, E is the Young's modulus, I is the moment of inertia, L is the beam length, k is stiffness of the spring, and x is the distance along the beam measured from the left end.

(a) Create a matlab function that takes arbitrary values of w_0 , L , E , I , and, k and returns deflection v at one or more specified values of x . Your function should handle the situation where the desired x locations are given as a vector. It should also check to make sure that the given values of x are between 0 and L .

(b) Test your function for a beam with $w_0 = 50$ lb/in, $L = 300$ in, $E = 29 \times 10^6$ lb/in², $I = 959$ in⁴, and $k = 75000$ lb/in. Report numerical values of the deflection at $x = L/4$, $L/2$, and $3L/4$.

(c) Using the same beam parameters as those in (b) use your function to compute and plot deflected shape of a beam. Use large enough points, say 100 points, for x locations to get a smooth plot of v . Put appropriate labels (with appropriate units) on the axes and give the plot a suitable title. Use matlab functions to determine the location along the beam where the maximum (absolute) deflection occurs and mark this location on the plot with a circle. Also place the text 'Max deflection' near the location of the maximum.

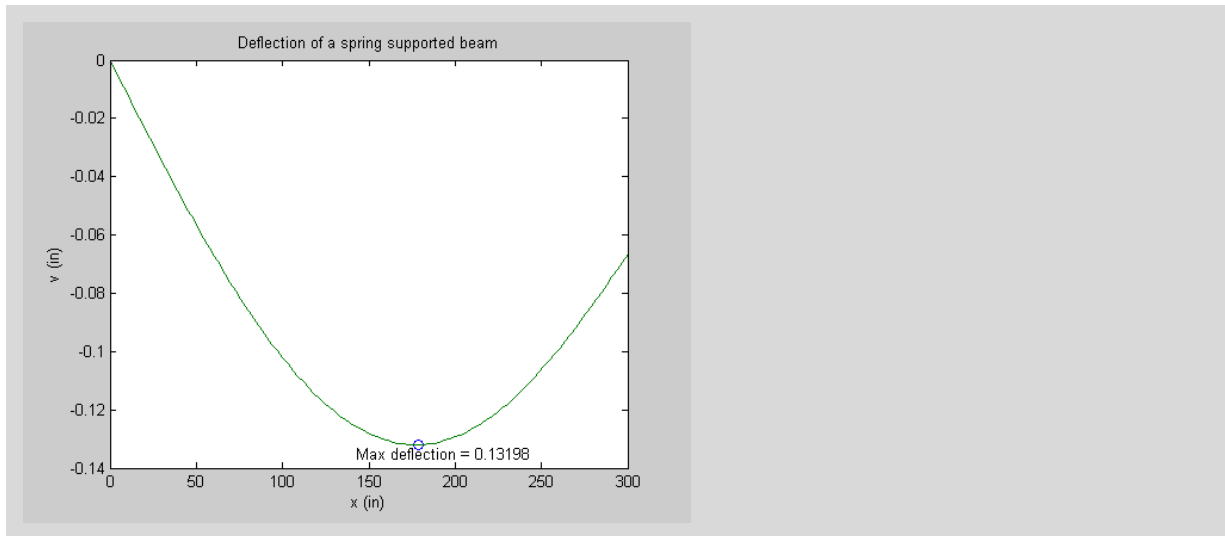
For grading submit your function, script for generating plot, and the resulting figure generated by matlab.

```
function v = springSupportedBeam(w0, L, k, E, I, x)
% Computes maximum moment, shear & deflection of a simply supported W beam
% Input
% w0 = distributed load intensity at x = 0
% L = span length
% k = spring constant
% I = Moment of inertia
% x = locations where deflections are desired
% Output
% v = deflection values
if x>=0 & x<=L
    v=-w0/(360*E*I*x*k*L) * x .* (120*E*I*x*L + k*(3*x.^4 - 10*L^2 * x.^2 + 7*L^4));
else
    disp ('Specified x values must be between 0 and L')
end

w0=50; L=25*12; k=75000; Ix = 959; E = 29000000;
x = [L/4, L/2, 3*L/4];
v = springSupportedBeam(w0, L, k, E, Ix, x)

v =
    -0.081255    -0.12814    -0.12051
```

```
x = linspace(0,L);  
v = springSupportedBeam(w0, L, k, E, Ix, x);  
[vmax, idMax] = max(abs(v));  
plot(x(idMax), v(idMax), 'o', x,v);  
xlabel('x (in)'); ylabel('v (in)');  
title('Deflection of a spring supported beam');  
text(0.8*x(idMax), 1.025*v(idMax), ['Max deflection = ', num2str(vmax)]);
```



Problem 3 — 15 points

(a) Write command(s) that take any vector a of numbers and returns a new vector b containing 0 for the negative numbers and the cubes of the positive numbers in a .

Example:

$$a = [1, 2, -1, 5, 6, 7, -4, 3, -2] \implies b = [1, 8, 0, 125, 216, 343, 0, 27, 0]$$

(b) Write command(s) that take any vector a of numbers and returns a new vector b containing every other element of a , starting with element 1.

Example:

$$a = [1, 2, -1, 5, 6, 7, -4, 3, -2] \implies b = [1, -1, 6, -4, -2]$$

(c) Write a single command that returns the total number of positive elements in any given matrix A .

Example:

$$A = [1, 2, -1, 5, 7; 6, 7, -4, 3, -2]; \implies 7$$

(a) Write command(s) that take any vector a of numbers and returns a new vector b containing 0 for the negative numbers and the cubes of the positive numbers in a .

```
a=[1, 2, -1, 5, 6, 7, -4, 3, -2];
naIndex = find(a<0);
b = a.^3; b(naIndex)=0;
b
```

```
b =
     1     8     0   125   216   343     0    27     0
```

(b) Write command(s) that take any vector a of numbers and returns a new vector b containing every other element of a , starting with element 1.

```
a=[1, 2, -1, 5, 6, 7, -4, 3, -2];
b=a([1:2:length(a)])
```

```
b =
     1    -1     6    -4    -2
```

(c) Write a single command that returns the total number of positive elements in any given matrix A .

```
A = [1, 2, -1, 5, 7; 6, 7, -4, 3, -2];
length(find(A(:)>0))
```

```
ans =
     7
```