College of Engineering and Computer Science
Mechanical Engineering Department

California State University Northridge

Mechanical Engineering 309
Numerical Analysis of Engineering Systems

Spring 2014 Number: 15237 Instructor: Larry Caretto

Solutions to Second Programming Assignment

First exercise

1. **Enter the matrices A, B, c, and r, shown on slide 19 and do the operations on that page to obtain the matrices D, E, ct, F, G, and H.**

```
>> format compact
>> r=[1 5 -4]
r =
     1     5    -4
>> c=[8;3;-2]
c =
     8
     3
    -2
>> B=[5 4;1 4;2 8]
B =
     5     4
     1     4
     2     8
>> A=[12 8 0;-7 -16 4;0 3 2]
A =
    12     8     0
    -7   -16     4
     0     3     2
>> D=[A B]
D =
    12     8     0     5     4
    -7   -16     4     1     4
     0     3     2     2     8
>> E=[B A]
E =
     5     4    12     8     0
     1     4    -7   -16     4
     2     8     0     3     2
>> ct = c'
ct =
     8     3    -2
>> F=[r ct]
F =
     1     5    -4     8     3    -2
>> F=[r; ct]
F =
     1     5    -4
     8     3    -2
>> G=[c A ct']
G =
     8    12     8     0     8
     3    -7   -16     4     3
    -2     0     3     2    -2
>> H=[G;D]
H =
     8    12     8     0     8
     3    -7   -16     4     3
    -2     0     3     2    -2
    12     8     0     5     4
```

Jacaranda Hall 3314 Mail Code Phone: N/A
Email: lcaretto@csun.edu 8348 Fax: 818.677.7062

```
    -7   -16     4     1     4
     0     3     2     2     8
```

2. Do all the subarray commands shown on slide 21.

```
>> H(2:4,3:5)
ans =
   -16     4     3
     3     2    -2
     0     5     4
>> H(:,2:4)
ans =
    12     8     0
    -7   -16     4
     0     3     2
     8     0     5
   -16     4     1
     3     2     2
>> H(1,:)
ans =
     8    12     8     0     8
>> H(2:4,3)
ans =
   -16
     3
     0
>> H(2:4,3)=[-1;-2;-3]
H =
     8    12     8     0     8
     3    -7    -1     4     3
    -2     0    -2     2    -2
    12     8    -3     5     4
    -7   -16     4     1     4
     0     3     2     2     8
```

3. Compute the following array operations using the arrays defined above: 4A, A/2, D+E, D-E, D.*E, and D./E. Compare the results to those on slides 30, 29, 32, and 33.

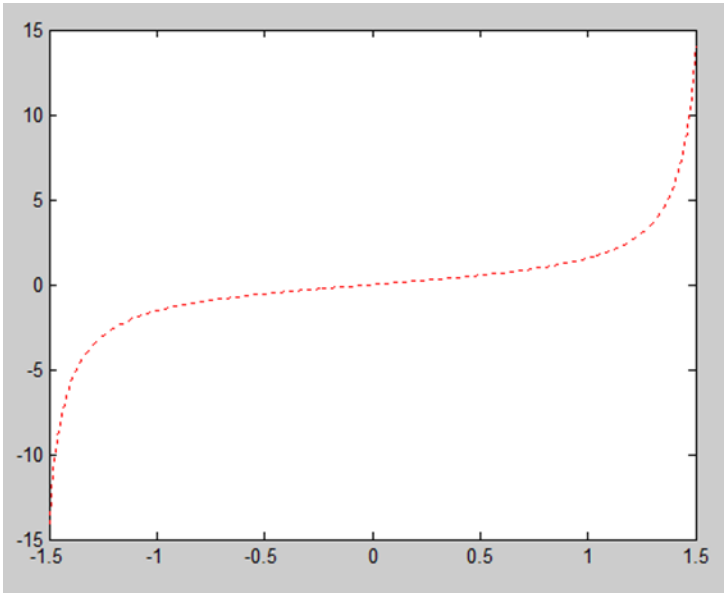
```
>> 4*A
ans =
    48    32     0
   -28   -64    16
     0    12     8
>> A/2
ans =
    6.0000    4.0000         0
   -3.5000   -8.0000    2.0000
         0    1.5000    1.0000
>> D+E
ans =
    17    12    12    13     4
    -6   -12    -3   -15     8
     2    11     2     5    10
>> D-E
ans =
     7     4   -12    -3     4
    -8   -20    11    17     0
    -2    -5     2    -1     6

>> D.*E
ans =
    60    32     0    40     0
    -7   -64   -28   -16    16
     0    24     0     6    16
>> D./E
ans =
```

```
    2.4000    2.0000         0    0.6250       Inf
   -7.0000   -4.0000   -0.5714   -0.0625    1.0000
         0    0.3750       Inf    0.6667    4.0000
```

4. **Using the commands on slide 26 as an example, obtain a plot of tan(x) *vs*. x for -1.5 ≤ x ≤ 1.5 with Δx = 0.01, using a dotted red line.**

The commands are shown below and the plot is shown at the right.

```
>>x=-1.5:0.01:1.5;
>>tanX = tan(x);
>>plot(x,tanX,'r:')
```



5. **Using the script on page 24 as an example, create a script to plot tan(x) *vs*. x for -1 ≤ x ≤ 1 with Δx = 0.01. Enter the help command for your script and execute the script.**

```
>> help tanPlot
tanPlot script file
plots tan(x) for -1.5 < x < 1.5
```

A copy of the script is shown below. Execution of the script produces the same plot found in part four.

```
%tanPlot script file
%plots tan(x) for -1.5 < x < 1.5
x=-1.5:0.01:1.5;
tanX = tan(x);
plot(x,tanX,'r:')
```

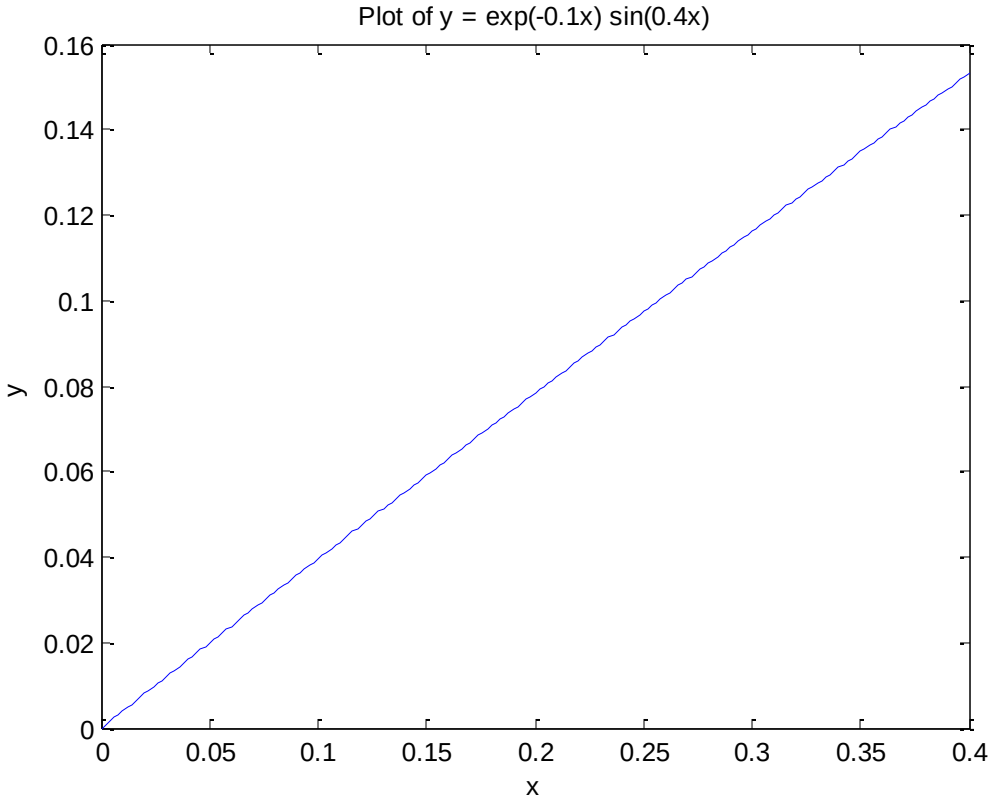
6. **Following the commands on slide 32, create a script that plots $e^{ax}\sin(bx)$ with a = –0.1 and b = 0.2**

A copy of the script, which is saved as the file name `expAxSinBx.m`, is shown below. Note the use of the `num2str` function of MATLAB that converts numbers to strings. This is used to create a plot title (not required for the assignment) that shows the numbers used in the plot.

```
%expAxSinBx plot
%commands for script file
x = linspace(0,0.4,201);
y=exp(a*x).*sin(b*x);
plot(x,y)
xlabel('x')
ylabel('y')
title(['Plot of y = exp(' num2str(a) 'x) sin(' num2str(b) 'x)'])
```

The following commands were used to run the script to get the desired plots and the result plot is shown after these commands.

```
>> a = -0.1;
>> b = 0.2;
expAxSinBx
```



7. **Use the `publish` command to produce an html file from the script you created in the previous step.**

   This produces the same result in an html file that you can copy to your submission file.

8. **Convert the script from the previous step into a function that accepts input values for a, b, x_{max} and N, where x_{max} is the maximum value for x (2 in the example on slide 32) and N is the number of points to be plotted. Note that you can create the time array by the command x = 0:Δt:t_{max}; where Δt = t_{max}/(N – 1). Alternatively you can use the command x = `linspace`(0, x_{max}, N). Note that this function to create a plot does not have to return any values. Run the function for x_{max} = 6 and N = 201 the four values of a and b shown above.**

   The function is shown below.

```
function [ x y ] = expaxSinbx( a, b, xMax, N )
%expaxSinbx plots function implied in title
%   Function generates two arrays, x and y
%   y = exp(a*x) .* sin(b*x)
%   x-y plot is generated for values of x and y

x = 0:xMax/(N-1):xMax;
y=exp(a*x).*sin(b*x);
plot(x,y)
xlabel('x')
ylabel('y')
title(['Plot of y = exp(' num2str(a) 'x) sin(' num2str(b) 'x)'])
end
```

   The commands to run the function for the data given are shown below:.

```
expaxSinbx(-1,12,6,201);
expaxSinbx(-1,6,6,201);
```

```
expaxSinbx(1,6,6,201);
expaxSinbx(1,12,6,201);
```

Second exercise

This exercise asks you to plot the emissive power of an ideal radiating surface, known as a black body, which is important for calculations of radiation heat transfer such as in the design of solar collectors. The emissive power $E_{b\lambda}$ in W/m^2-μm, which is given by the following equation, describes the radiation intensity (at a given wavelength), per unit area, per unit wavelength interval, as a function of temperature, T, and wavelength, λ.

$$E_{b\lambda} = \frac{C_1}{\lambda^5[e^{C_2/\lambda T} - 1]}$$

Where C_1 = 3.74177153x10^8 W·μm^4/m^2, is the first radiation constant; C_2 = 14387.770 μm·K, is the second radiation constant; λ is the wavelength in μm, and T is the temperature in kelvins.

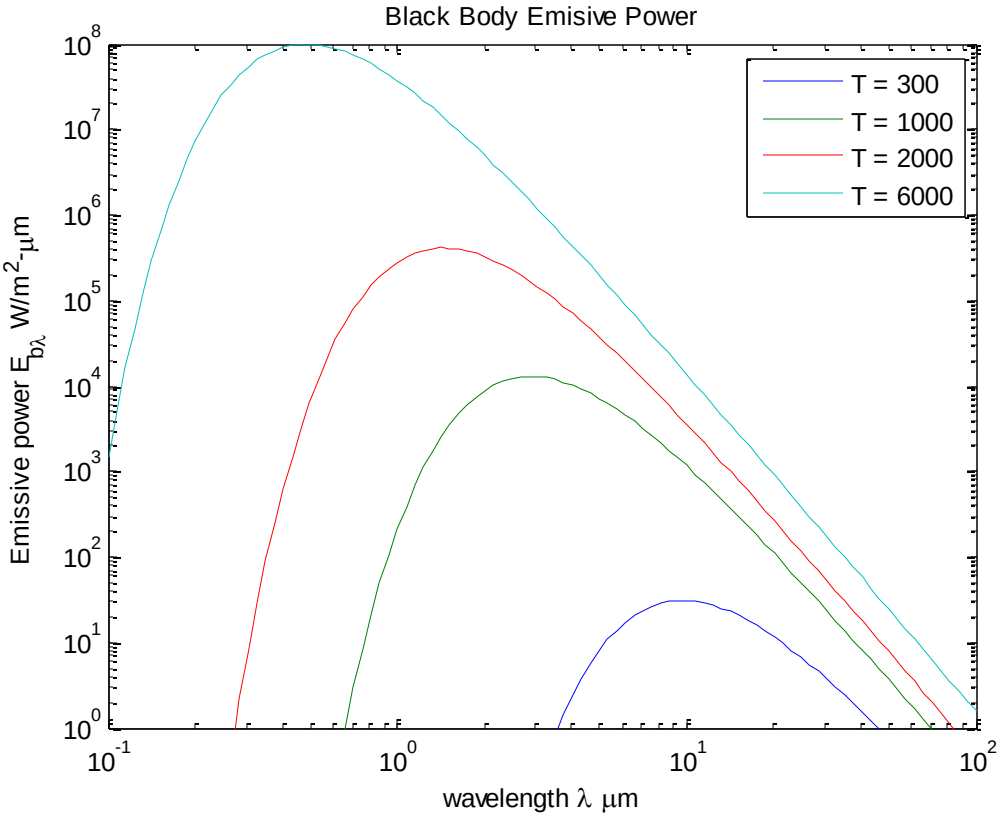
Use the help system for the `logSpace` command to learn how to create a wavelength array that has logarithmic spacing with 100 points between λ = 0.1 μm and λ = 100 μm.

Use help for the plot command to learn how to plot multiple lines on a single plot, how to place a legend on the plot, and how to set the axis scales using the command `axis([xmin, xmax, ymin, ymax])`. In this case we are given xmin = 0.1 and xmax = 100; pick reasonable values for ymin and ymax.

Using the λ array that you created with the `logSpace` command, generate a single log-log[1] plot of $E_{b\lambda}$ *versus* λ for T = 300 K, 1000 K, 2000 K, and 6000 K. Choose different colors, different line styles, or different markers to distinguish the four curves on your plot. Include a legend that shows the temperatures for each curve.

```
>> C1 = 3.74177153E8
>> C2 = 14387.770
>> T=300;
>> eb1=C1./((lambda.^5).*(exp(C2./(lambda*T)) - 1));
>> T=1000;
>> eb2=C1./((lambda.^5).*(exp(C2./(lambda*T)) - 1));
>> T=2000;
>> eb3=C1./((lambda.^5).*(exp(C2./(lambda*T)) - 1));
>> T=6000;
>> eb4=C1./((lambda.^5).*(exp(C2./(lambda*T)) - 1));
>> %Use MATLAB default plot colors by omitting plot options
>> loglog(lambda,eb1,lambda,eb2,lambda,eb3,lambda,eb4)
>> legend('T = 300', 'T = 1000', 'T = 2000', 'T = 6000')
>> axis([0.1 100 1 1e8])
>> %Commands for axis and chart labels shown below, not required for assignment
>> %Note MATLAB ability to use different symbols and superscript/subscript
>> xlabel('wavelength \lambda \mum')
>> ylabel('Emissive power E_{b\lambda} W/m^2-\mum')
>> title('Black Body Emissive Power')
```

[1] Note that the plot command always gives x and y as linear axes. The commands to give logarithmic axes have the same syntax as the plot command. They are **`semilogx`**, **`semilogy`**, and **`loglog`**. As you can probably guess, these commands, respectively, give a logarithmic x axis only, a logarithmic y axis only, and both axes logarithmic



Third exercise

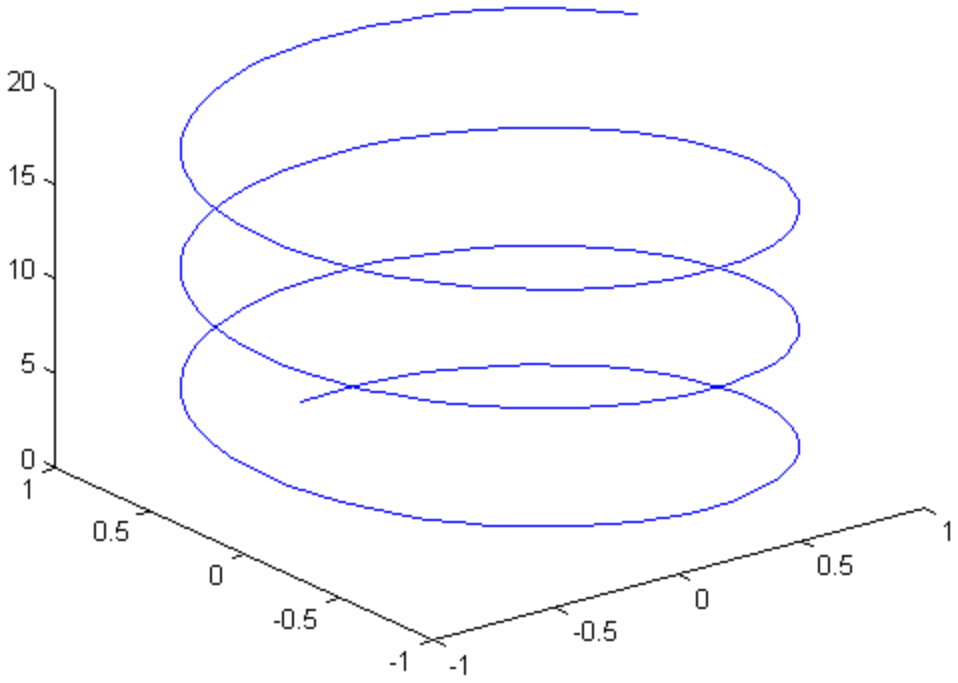
1. **Make a three-dimensional plot. Define a vector z with 201 points in the region $0 \le z \le 20$. Obtain x = sin(z), y = cos(z) then use the plot command plot3(x,y,z). Create a script and publish this in an html file.**

A copy of the publish file is shown below; the plot is shown on the next page..

```
z=linspace(0, 20, 201);
x = sin(z);
y = cos(z);
plot3(x,y,z)
```

Fourth exercise

Download the Excel file matlabInput.xlsx from the course web site and save it in your MATLAB directory. Use the File | Import | Data command to import the array in the workbook into an array x. Use the MATLAB functions size, sum, mean, median, std, max, and min to determine the corresponding properties of the data in the array. Note that these functions work on each column of the array. For some functions you can use a command like sum(sum(x)) to get the sum of the entire array. For other commands, like std, this approach will give an incorrect result. Use the commands on slide 50 to copy the x array into a one-dimensional column array, y. Compare the results for all the functions using two approaches sum(sum(x)) and sum(y). Which results are different?



The left column of the table below shows the use of repeated command, such as sum(sum(x)), on the two-dimensional array. The right column shows the commands to convert all data into a one-dimensional array before applying the functions one time. The only results that are different are those for the standard deviation (std) and the median.

```
>> size(x)
ans =
   200    89
>> sum(sum(x))
ans =
  8.8637e+003
>> mean(mean(x))
ans =
    0.4980
>> median(median(x))
ans =
    0.4999
>> std(std(x))
ans =
    0.0105
>> max(max(x))
ans =
    0.9999
>> min(min(x))
ans =
  4.0000e-005
```

```
>> y=x(:,1);
>> for k = 2:size(2) y=[y; x(:,k)]; end
>> size(y)
ans =
  17800     1
>> sum(y)
ans =
  8.8637e+003
>> mean(y)
ans =
    0.4980
>> median(y)
ans =
    0.4970
>> std(y)
ans =
    0.2897
>> max(y)
ans =
    0.9999
>> min(y)
ans =
  4.0000e-005
```

Fifth exercise

You can program if statements, count-controlled loops and for loops in MATLAB. The for loop has the following form

```
for <index> = <array>
      <statements>
end
```

Any MATLAB array specification can be used for the <array> part of the for command. The statements in the loop will be repeated for each statement in the array. The conventional count-controlled-loop notation with a start point, an end point, and an increment is expressed in MATLAB array notation as <start>:<increment>:<end>, where an increment of 1 can be omitted.

Write a script that uses a for loop over temperatures to create an array in which the first row is the wavelength array from the second exercise and the remaining rows are the black body radiation calculations for each temperature of the second exercise. (HINT: In this case you can use the following looping command: for T = [300 1000 2000 6000]. The logspace command you used to define lambda above gives you a row matrix.

To store the results in an array where lambda is the first row and the values of $E_{b\lambda}$ are in subsequent rows, initialize the matrix Eb = lambda before the for loop. Inside the for loop, after you compute $E_{b\lambda}$ for one temperature use the following statement to add the most recent value (say eblambda) to the array:

```
Eb = [Eb; Eblambda];
```

Do all these operations with a semicolon at the end of each command to avoid intermediate output. When you are finished use a subarray command to show all rows for the first set of columns that will fit onto the width of your command window.

The commands shown below are saved in the script file allEb.m.

```
C2=14387.770;
C1=3.74177153e8;
lambda=logspace(-1,2,100);
eb = lambda;
for T=[200 1000 2000 6000]
    eblambda =C1./((lambda.^5).*(exp(C2./(lambda*T)) - 1));
    eb=[eb;eblambda];
end
format compact
format shortg
eb(:,1:6)
```

The command to execute the script and the results are shown below. Note that the use of the format shortg command gives each individual array element without a scaling factor. With this format specification, the individual format of each array element (with or without an exponent) is selected by MATLAB to give five significant figures.

```
>> ans =
  Columns 1 through 7
         0.1      0.10723      0.11498      0.12328      0.13219      0.14175
 1.9443e-195   1.496e-181  1.3031e-168  1.4884e-156  2.5561e-145  7.4997e-135
   1.224e-49   1.4047e-45   8.3855e-42   2.7214e-38   5.0025e-35   5.4121e-32
  2.1401e-18   1.9256e-16   1.2496e-14   5.9794e-13   2.1533e-11   5.9488e-10
      1441.7       5119.5        16304        46907   1.2276e+05   2.9409e+05
```