Recitation Problems – Week 1

440:127 – Spring 2015 – S. J. Orfanidis

Please do the following problems during your recitation session, including any additional problems given to you by your TA. Within *48 hours* of your recitation session, you must upload the complete solutions of these problems to Sakai, as instructed by your TA.

1. Consider the row vectors:

   ```
   a = [5 2 1];
   b = [1 2 4];
   ```

   Perform the following element-wise operations and then try to understand them by re-doing them by hand:

   ```
   c1 = [a + 5;  b*5;  a + b;  a.*b;  a./b; a.\b]

   c2 = [1./a + b;  (1./a) + b;  1./(a + b)]

   c3 = [a.^2;  a.^2./a;  a.^2./a.^2;  a.^(2./a).^2]

   c4 = [a.^2./b;  (a.^2)./b;  a.^(2./b);  a./b.^2;  a./(b.^2);   (a./b).^2]
   ```

   What are your conclusions about the precedence of the operations `+ * / ^` ?

   In doing this problem, you may wish to create a script M-file and place all of the above commands into it. This will save you time retyping commands in the command window.

2. Create a *function* M-file, say, **abc.m**, that accepts two row vectors a, b of the same length as inputs, and calculates the four matrices c_1, c_2, c_3, c_4 as in Problem 1. It must have usage:

   ```
   [c1,c2,c3,c4] = abc(a,b)
   ```

   Run it on the example of Problem 1 and verify that you get the same results. See section 6.1 of the text on how to create function M-files, which must always start with the keyword **function**.

3. End-of-Chapter Problem 2.19. For both parts (a) and (b), calculate the temperatures using both the ideal gas law and the van der Waals equation. However, make the following changes: for part (a) use only five equally-spaced pressures in the range $0 \le P \le 400$, and for part (b), use nine equally-space volumes in the range $1 \le V \le 9$.

   Use the **char** and **num2str** functions to print your calculations for case (a) with a single Matlab command, as shown below, using two decimal places,

   ```
       P         T_i       T_vw
   --------------------------
     0.00       0.00    125.04
   100.00     601.36    689.74
   200.00    1202.72   1254.43
   300.00    1804.08   1819.12
   400.00    2405.44   2383.81
   ```

   and for part (b), print them as follows with a single Matlab command, using three decimal places,

```
  V        T_i         T_vw
-----------------------------
1.000    1322.994    1367.363
2.000    2645.989    2629.865
3.000    3968.983    3931.793
4.000    5291.978    5244.085
5.000    6614.972    6560.604
6.000    7937.966    7879.259
7.000    9260.961    9199.143
8.000   10583.955   10519.798
9.000   11906.950   11840.969
```

4. In a simplified version of Problem 4 of the week-1 homework set, consider the following table of stopping distances and stopping times for various car speeds (this version ignores the driver's reaction time), takne from the NJ Driver's Manual:

speed (mph)	braking distance (ft)	deceleration (ft/sec^2)	braking time (sec)
10	12	9.34	1.57
20	36	11.97	2.45
30	72	13.54	3.25
40	127	13.55	4.33
50	216	12.47	5.88
60	333	11.62	7.57
70	493	10.68	9.61

We know from physics that a car with initial speed v, subject to constant deceleration a, will stop in time t, and travel distance s, given by:

$$t = \frac{v}{a}, \qquad s = \frac{1}{2}at^2$$

a. Given the column vectors of speeds (column 1) and braking times (column 4), calculate the corresponding column vectors of braking distances and decelerations (columns 2 & 3). You will need to convert the speeds from units of MPH to ft/sec. Arranged the computed values into a 7×4 matrix, $A = [v, s, a, t]$, and print it in the command window (the values shown above are rounded).

b. Using the built-in functions **char** and **num2str**, generate a table of values as shown below using a single Matlab command:

```
speed        braking     deceleration   braking
(mph)     distance (ft)  (ft/sec^2)   time (sec)
----------------------------------------------
10.00         11.51          9.34          1.57
20.00         35.93         11.97          2.45
30.00         71.50         13.54          3.25
40.00        127.01         13.55          4.33
50.00        215.60         12.47          5.88
60.00        333.08         11.62          7.57
70.00        493.31         10.68          9.61
```

This can be done by using **char** to concatenate vertically the three header lines, and concatenate below those the entire matrix A, converted to a string using **num2str**, i.e.,

```
num2str(A,'%13.2f')
```

where the `13.2f` format specification allows for two decimal places and leaves enough blank spaces to fit the headers.

Recitation Problems – Week 2

440:127 – Spring 2015 – S. J. Orfanidis

Please work on the following problems during your recitation session, including any additional problems given to you by your TA. The objectives of this week's recitation are to get you familiar with some matrix operations, the use of the built-in functions **min**, **max**, **sum**, **cumsum**, **mean**, **sort**, **sortrows**, defining your own anonymous functions using function handles, finding maxima and minima using the built-in functions **max**, **min**, **fminbnd**, and **fzero**, and plotting your results.

Within 48hrs of your recitation session, please upload the complete solutions to these problems to sakai. Please follow your TA's instructions on how and where to upload your files. Please refer to the syllabus regarding the policies on uploading your recitation work and on recitation attendances.

1. Consider the 5×4 matrix:

$$A = \begin{bmatrix} 2 & 6 & 5 & 2 \\ 8 & 3 & 1 & 5 \\ 3 & 5 & 3 & 6 \\ 5 & 9 & 7 & 1 \\ 8 & 7 & 2 & 4 \end{bmatrix}$$

Use appropriate Matlab commands to answer the following questions.

a. With a single Matlab command, determine the maximum value in each column, and the row in which that maximum occurs. What happens when there are two entries in a column that are equal to the maximum of that column?

b. Using a single command, what is the minimum value in each row, and in which column does that minimum occur?

c. What is the maximum value in the entire matrix, and what is its row/column location?

d. What is the mean value of each column, what is the mean value of each row, and what is the mean value of the entire matrix?

e. What is the sum of each column, what is the sum of each row, and what is the sum of all entries?

f. What is the cumulative sum of each column, and what is the cumulative sum of each row?

g. With a single command, sort each column in ascending order, then sort the columns in descending order.

h. With a single command, sort the matrix A so that the third column is in descending order, but each row still retains its original data, in other words, your sorting operation should be like this:

$$A = \begin{bmatrix} 2 & 6 & 5 & 2 \\ 8 & 3 & 1 & 5 \\ 3 & 5 & 3 & 6 \\ 5 & 9 & 7 & 1 \\ 8 & 7 & 2 & 4 \end{bmatrix} \quad \Rightarrow \quad B = \begin{bmatrix} 5 & 9 & 7 & 1 \\ 2 & 6 & 5 & 2 \\ 3 & 5 & 3 & 6 \\ 8 & 7 & 2 & 4 \\ 8 & 3 & 1 & 5 \end{bmatrix}$$

i. With a single command, sort the matrix A so that its fourth row is in ascending order, but each column still retains its original data, in other words, your sorting operation should be like this:

$$A = \begin{bmatrix} 2 & 6 & 5 & 2 \\ 8 & 3 & 1 & 5 \\ 3 & 5 & 3 & 6 \\ 5 & 9 & 7 & 1 \\ 8 & 7 & 2 & 4 \end{bmatrix} \quad \Rightarrow \quad B = \begin{bmatrix} 2 & 2 & 5 & 6 \\ 5 & 8 & 1 & 3 \\ 6 & 3 & 3 & 5 \\ 1 & 5 & 7 & 9 \\ 4 & 8 & 2 & 7 \end{bmatrix}$$

2. Consider the following function $f(x)$ and its derivative $f'(x)$:

$$f(x) = x \cdot \sin(x)$$
$$f'(x) = \sin(x) + x \cdot \cos(x)$$

a. Using the three methods outlined on pp. 32–36 of the week-2 lecture notes, find the minimum of this function in the interval, $0 \le x \le 6$. In applying method-1 based on the function **min**, use 101 equally-spaced values of x in the interval $[0, 6]$. Compare the results from the three methods. You should find the following results from the three methods:

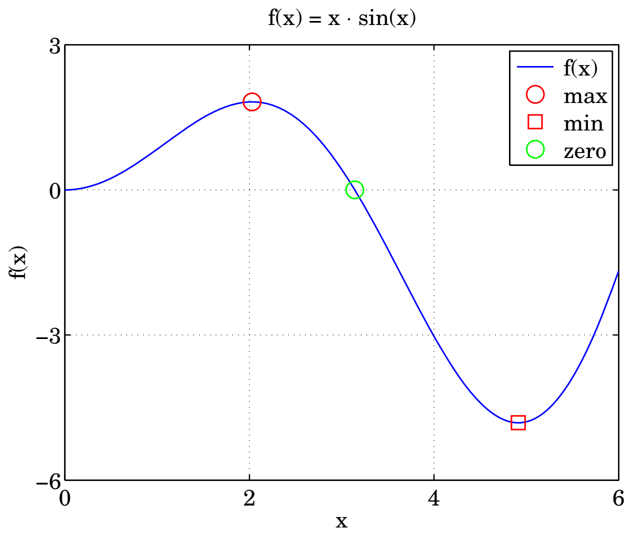
```
method   xmin          fmin
-----------------------------
  1     4.920000     -4.814349
  2     4.913176     -4.814470
  3     4.913180     -4.814470
```

Figure out how to print this table exactly as shown above, using either three calls to the **fprintf** function, or with a single Matlab command that uses a combination of the functions **char** and **num2str**.

How can you improve on the slight differences among the three methods?

b. Using the same three methods, find the maximum of this function in the interval $-1 \le x \le 4$. Compare the results from the three methods and make a similar table of results as in part (a).

c. Using your computations of the maximum and minimum arising from the **fminbnd** method, generate a plot of $f(x)$ over the interval, $0 \le x \le 6$ and annotate it exactly as shown below. Moreover, using the function **fzero**, find and place on the graph the x-point at which the $f(x)$ curve intersects the horizontal line, i.e., find the solution of the equation, $f(x) = 0$.



Recitation Problems – Week 3

440:127 – Spring 2015 – S. J. Orfanidis

Please do the following drill problems on matrix properties during your recitation session, including any additional problems given to you by your TA. Upload your completed work within 48hrs of your recitation session. Please follow your TA's instructions on how and where to upload your files.

1. Consider the 5×4 matrix:

$$A = \begin{bmatrix} 2 & 6 & 5 & 2 \\ 8 & 3 & 1 & 5 \\ 3 & 5 & 3 & 6 \\ 5 & 9 & 7 & 1 \\ 8 & 7 & 2 & 4 \end{bmatrix}$$

Using appropriate MATLAB operations do the following, each with a *single MATLAB command*, and without retyping the matrix elements of A:

a. Construct the matrix B whose columns are the cumulative sums of the columns of A.

b. Reconstruct A from B.

c. Repeat parts (a) and (b) when B is defined as the cumulative sum of the rows of A.

d. Set $B = A$, then, with a single command implement the operation:

$$A = \begin{bmatrix} 2 & 6 & 5 & 2 \\ 8 & 3 & 1 & 5 \\ 3 & 5 & 3 & 6 \\ 5 & 9 & 7 & 1 \\ 8 & 7 & 2 & 4 \end{bmatrix} \quad \Rightarrow \quad B = \begin{bmatrix} 2 & 6 & 5 & 2 \\ 8 & 0 & 0 & 5 \\ 3 & 0 & 0 & 6 \\ 5 & 0 & 0 & 1 \\ 8 & 7 & 2 & 4 \end{bmatrix}$$

e. Flip the matrix A upside-down. Then, flip A left-right. Combine the previous two operations into one that flips A both upside-down and left-right. Do not use the built-in functions **flipud**, **fliplr** in this part.

f. Swap the second and third columns of A. Swap the first and fifth rows of A.

g. Reshape A into the following matrices B and C:

$$A = \begin{bmatrix} 2 & 6 & 5 & 2 \\ 8 & 3 & 1 & 5 \\ 3 & 5 & 3 & 6 \\ 5 & 9 & 7 & 1 \\ 8 & 7 & 2 & 4 \end{bmatrix} \quad \Rightarrow \quad B = \begin{bmatrix} 2 & 8 & 9 & 3 & 5 \\ 8 & 6 & 7 & 7 & 6 \\ 3 & 3 & 5 & 2 & 1 \\ 5 & 5 & 1 & 2 & 4 \end{bmatrix}, \quad C = \begin{bmatrix} 2 & 8 & 3 & 5 & 8 \\ 6 & 3 & 5 & 9 & 7 \\ 5 & 1 & 3 & 7 & 2 \\ 2 & 5 & 6 & 1 & 4 \end{bmatrix}$$

Then, recover A from B and from C.

h. Transform A into the following matrix B:

$$A = \begin{bmatrix} 2 & 6 & 5 & 2 \\ 8 & 3 & 1 & 5 \\ 3 & 5 & 3 & 6 \\ 5 & 9 & 7 & 1 \\ 8 & 7 & 2 & 4 \end{bmatrix} \quad \Rightarrow \quad B = \begin{bmatrix} 2 & 0 & 6 & 0 & 5 & 0 & 2 \\ 8 & 0 & 3 & 0 & 1 & 0 & 5 \\ 3 & 0 & 5 & 0 & 3 & 0 & 6 \\ 5 & 0 & 9 & 0 & 7 & 0 & 1 \\ 8 & 0 & 7 & 0 & 2 & 0 & 4 \end{bmatrix}$$

Then, recover A from B.

2. You wish to take a loan to get a car, but from different dealers you are offered several possible choices of loan rates and loan durations that you need to evaluate, e.g., a longer loan is cheaper per month, but you end up paying more overall, and the car depreciates more. The payment per month, P, is computed by the following formula:

$$P = \frac{Ar(1+r)^M}{(1+r)^M - 1}$$

where A is the loan amount, M is the total number of months, r is the monthly rate, i.e., $r = \text{APR}/1200$, where APR is the annual percentage rate. Assume $A = 20000$, and consider the following choices of rates and loan durations corresponding to $[3, 4, 5, 6]$ year loans:

$$\text{APR} = [4, 5, 6] \quad \text{(annual percentage rate)}$$

$$M = [36, 48, 60, 72] \quad \text{(months)}$$

a. Using **meshgrid**, calculate all possible monthly payment amounts P with a single Matlab command, arranged in a 4×3 matrix, with the three columns corresponding to the three APR rates, and the four rows corresponding to the four durations.

b. Using at most three **fprintf** commands, print P in the manner shown below, where the rates are listed horizontally, and the months vertically:

```
       |   4%       5%       6%
------|-------------------------
36 mo | 590.48   599.42   608.44
48 mo | 451.58   460.59   469.70
60 mo | 368.33   377.42   386.66
72 mo | 312.90   322.10   331.46
```

Hint: To print a percent sign inside an fprintf format specification, use a double percent sign, %%, for example, `'%3.2f%%'`.

c. Using at most three **fprintf** commands, print the results also in a transposed fashion as shown below:

```
   |  36 mo    48 mo    60 mo    72 mo
---|----------------------------------
4% | 590.48   451.58   368.33   312.90
5% | 599.42   460.59   377.42   322.10
6% | 608.44   469.70   386.66   331.46
```

d. The total amount that you pay back for the loan is MP. Using the meshgrid matrices from part (a) and at most three **fprintf** commands, print a table of total payments as shown below:

```
      |    4%          5%          6%
------|-------------------------------
36 mo | 21257.27   21579.05   21903.79
48 mo | 21675.89   22108.12   22545.63
60 mo | 22099.83   22645.48   23199.36
72 mo | 22529.06   23191.10   23864.96
```

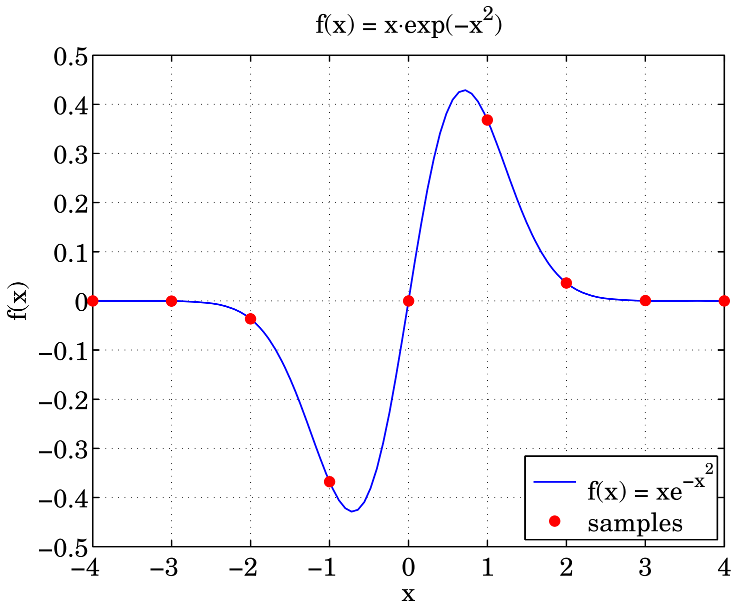
Recitation Problems – Week 4

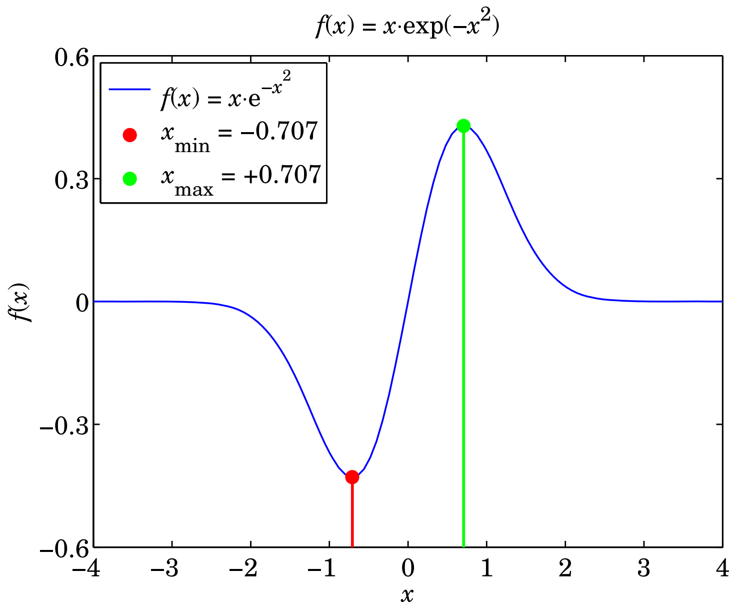
440:127 – Spring 2015 – S. J. Orfanidis

Please do the following problems from your textbook during your recitation session, including any additional problems given to you by your TA. Upload the complete solutions within 48hrs of your session.

1. Use appropriate MATLAB commands to reproduce *every* aspect of the two graphs shown below, i.e., axis limits, tickmarks, line styles, colors, markers, axis labels, title, and legends. All the information you need can be gleaned off the figure. Use the function **fminbnd** to determine the maximum and minimum points on the second graph. Notice also that the symbols x, f are in italics in the second graph and that the red and green dots and vertical lines are thicker than normal.

   Hint: `set(gca, 'xtick', tick_vector)`, also, to make a centered dot in the title, use `\cdot`.





2. The two functions $f(n)$ and $g(n)$ defined below are identically equal to each other for all non-negative integer values of n:

$$f(n) = \sum_{k=0}^{n} (k-1)^2 (0.8)^k = (0-1)^2 (0.8)^0 + (1-1)^2 (0.8)^1 + \cdots + (n-1)^2 (0.8)^n$$

$$g(n) = 145 - 4(n^2 + 8n + 36)(0.8)^n$$

   a. Using the above expression for $f(n)$ and the function **cumsum**, calculate the length-11 vector of values of $f(n)$ for $n = 10, 11, \dots, 20$, by means of a vectorized command (no for-loops). For example, you can calculate all 21 values for $0 \le n \le 20$ with a single command, and then extract the above subset of 11 values.

      Note that $f(n)$ need not be defined as a separate function of n—although, it can be. Also, the symbol $f(n)$ cannot denote a MATLAB array because n starts from $n = 0$.

   b. Then, calculate the vector of values of $g(n)$ implementing the given expression for $g(n)$. If you are not getting $f(n) = g(n)$, then please recheck your code.

   c. Using at most three **fprintf** commands (no loops), print your results in a form exactly as shown below:

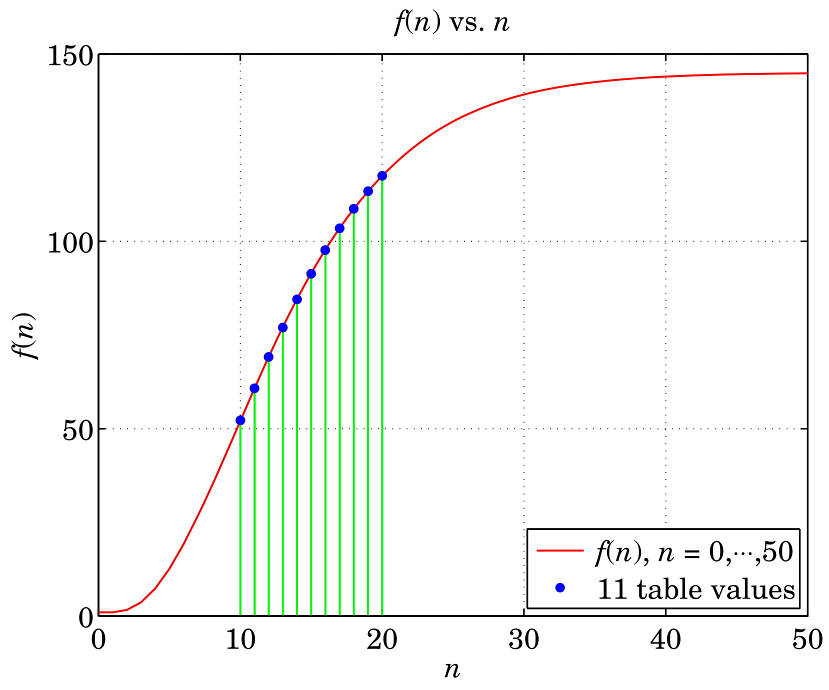
```
 n       f(n)          g(n)
---------------------------
10     52.228706     52.228706
11     60.818641     60.818641
12     69.133698     69.133698
13     77.050181     77.050181
14     84.482880     84.482880
15     91.379017     91.379017
16     97.712204     97.712204
17    103.476811    103.476811
18    108.682973    108.682973
19    113.352305    113.352305
20    117.514351    117.514351
```

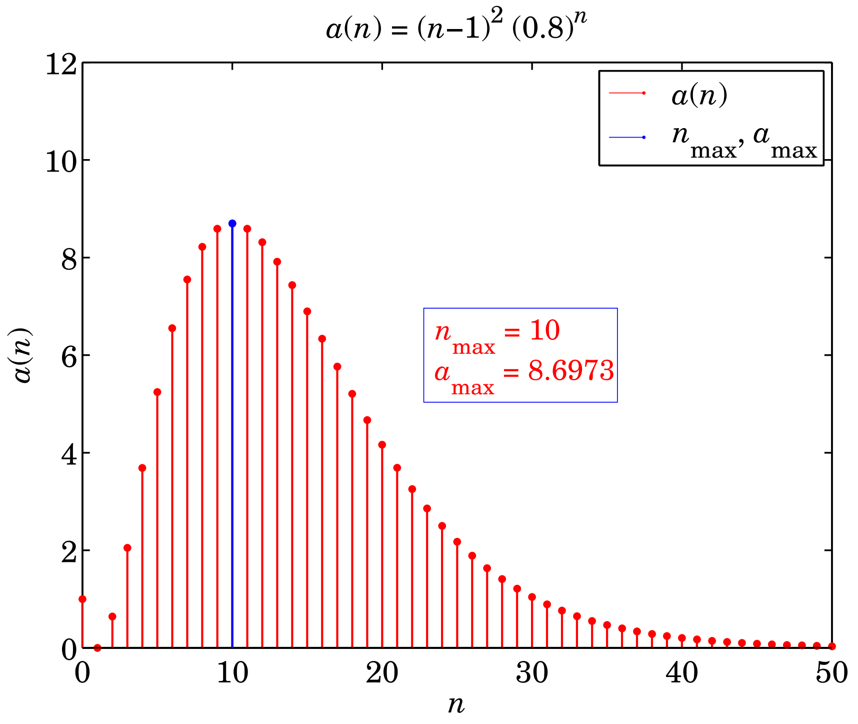
   d. Recalculate $f(n)$ for the longer range of values $n = 0:50$, and plot the results, including the 11 values of the above table, generating a graph exactly as the one shown below, including all colors, italic fonts, legends, and text.

      Hint: You need to use both **plot** and **stem** and hold the graph between.

   e. The sequence $f(n)$ is the cumulative sum of the sequence $a(n) = (n-1)^2 (0.8)^n$. Evaluate $a(n)$ for $n = 0:50$, and then, plot $a(n)$ vs. n using a **stem** plot, genenerating a graph that includes all the details and colors exactly shown below. The maximum point n_{max} and $a_{\text{max}} = a(n_{\text{max}})$ shown in blue on the graph must be computed with the aid of the MATLAB function **max**.

      Hint: You need to use the **text** function to create the boxed text that displays the maximum values.


$f(n)$ vs. n
$f(n)$
n
$f(n)$, $n = 0,\cdots,50$
11 table values
0
50
100
150
0
10
20
30
40
50

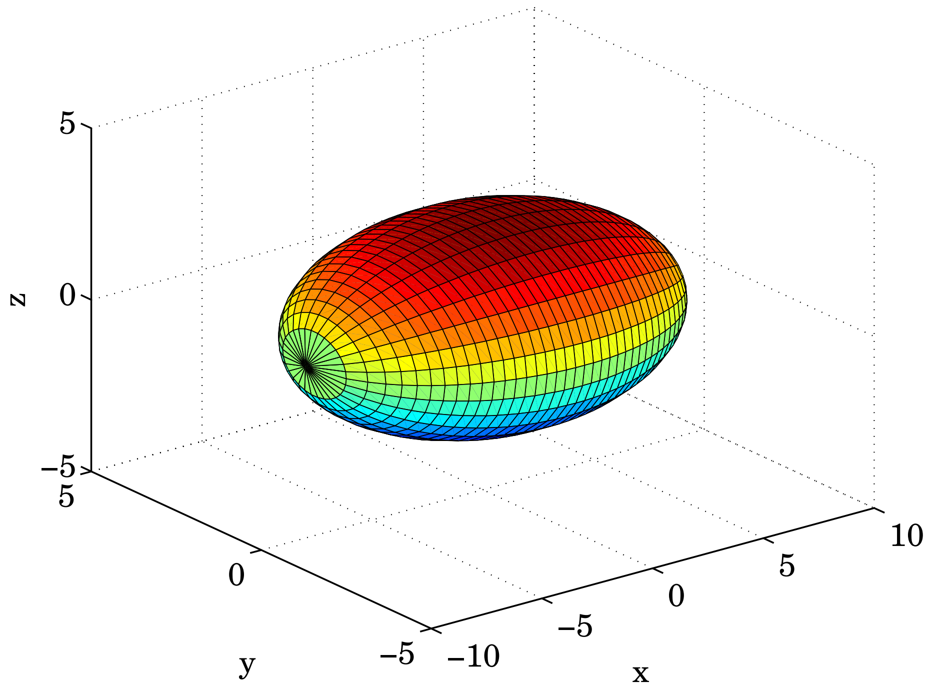

$a(n) = (n-1)^2\,(0.8)^n$
$a(n)$
n
$a(n)$
$n_{\max}$, $a_{\max}$
$n_{\max} = 10$
$a_{\max} = 8.6973$
0
2
4
6
8
10
12
0
10
20
30
40
50

3. An ellipse with semi-axes a, b is defined by the following equation on the x, z plane:

$$\frac{x^2}{a^2} + \frac{z^2}{b^2} = 1 \quad \Rightarrow \quad z = \pm b\sqrt{1 - \frac{x^2}{a^2}} \tag{1}$$

For the values $a = 8$, $b = 3$, keep the $+$ sign in Eq. (1), and rotate the curve about the x-axis, resulting in a surface of revolution called an *ellipsoid*. Generate a surface plot as shown below.

Hint: The variable x is restricted in the range $-a \le x \le a$.



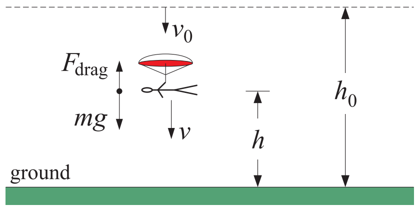
Recitation Problems – Weeks 5 & 6

440:127 – Spring 2015 – S. J. Orfanidis

Please do the following problems during your recitation session, including any additional problems given to you by your TA. Upload the complete solutions within 48hrs of the *end* your session.

1. A skydiver jumps off a plane from a height of h_0 meters, with an initial downward velocity v_0. The skydiver falls under the influence of the downward gravity force, mg, and an upward drag force, mentioned in Example 2.3 (p. 35) of the textbook, that depends quadratically on the downward velocity v, that is, $F_{\text{drag}} = \frac{1}{2}\rho C A v^2$. Assume the following numerical values:

$\rho = 1.2$	kg/m^3, air density
$g = 9.81$	m/sec^2, acceleration of gravity
$m = 75$	kg, skydiver's weight (mass)
$C = 1$	skydiver's drag coefficient
$A = 0.7$	m^2, skydiver's cross sectional area



We will see in a future homework that the equations of motion for this problem can be solved exactly, resulting in the following expressions for the height, downward velocity, and downward acceleration as functions of time,

$$\begin{aligned}
h(t) &= h_0 - h_c \ln\left[\cosh\left(\frac{t-t_0}{t_c}\right) + \frac{v_0}{v_c}\sinh\left(\frac{t-t_0}{t_c}\right)\right] \\
v(t) &= v_c \frac{\dfrac{v_0}{v_c} + \tanh\left(\dfrac{t-t_0}{t_c}\right)}{1 + \dfrac{v_0}{v_c}\tanh\left(\dfrac{t-t_0}{t_c}\right)} \\
a(t) &= g\left[1 - \frac{v^2(t)}{v_c^2}\right] = \frac{g(v_c^2 - v_0^2)}{\left[v_c\cosh\left(\dfrac{t-t_0}{t_c}\right) + v_0\sinh\left(\dfrac{t-t_0}{t_c}\right)\right]^2}
\end{aligned} \tag{1}$$

that are valid for $t \ge t_0$, where v_c, t_c, h_c are parameters defined as follows:

$$v_c = \sqrt{\frac{2mg}{\rho C A}}\,, \qquad t_c = \frac{v_c}{g}\,, \qquad h_c = v_c t_c = \frac{v_c^2}{g} \tag{2}$$

The quantity v_c is known as the critical or *terminal velocity*. The skydiver can control the value of v_c by changing the effective area A. For example, if she turns vertically up, then A decreases and v_c increases. Similarly, if just before reaching ground, she opens a parachute, thus suddenly increasing A, then v_c will decrease substantially.

a. Write a MATLAB function, **skydive**, that implements Eqs. (1), and has usage:

```
[h,v,a] = skydive(t,t0,h0,v0,vc);

% t  = time vector in seconds
% t0 = initial time
% h0 = initial height
% v0 = initial downward velocity
% vc = terminal velocity
%
% h = vector heights (same size as t)
% v = downward velocities (same size as t)
% a = downward accelerations (same size as t)
```

b. Assume that the skydiver jumps from a height of $h_0 = 2500$ m, with zero initial velocity $v_0 = 0$, at $t_0 = 0$, and is oriented so that her effective surface area is $A_0 = 0.7$ m^2. Calculate the terminal velocity v_{c0}, and then, using your function **skydive**, calculate the height h_1 and the speed v_1 at time $t_1 = 50$ seconds after the jump.

c. At the time instant, $t_1 = 50$ sec, the skydiver suddenly opens her parachute, which has a surface area of $A_1 = 50$ m^2. Calculate the new terminal velocity v_{c1} in m/sec.

   Use the values of t_1, h_1, v_1 as the initial values for the rest of the fall, for $t \geq t_1$. Thus, $h(t), v(t), a(t)$ can be calculated as follows for times before and after $t = t_1$:

```
    [h,v,a] = skydive(t,t0,h0,v0,vc0);    % for t <= t1
    [h,v,a] = skydive(t,t1,h1,v1,vc1);    % for t >= t1
```

   By appropriately passing the **skydive** function into **fzero**, caculate the times, say t_a, t_b, t_g, at which the skydiver is at heights of $h_a = 1500$ meters, $h_b = 250$ meters, and $h_g = 0$ (i.e. at ground), that is, use **fzero** to solve the following equations for the times t_a, t_b, t_g,

$$h(t_a) = h_a$$
$$h(t_b) = h_b$$
$$h(t_g) = h_g$$

   Hint: use the graph shown below to select proper search points for **fzero**.

d. Divide the time interval $0 \leq t \leq t_g$ into the two subintervals:

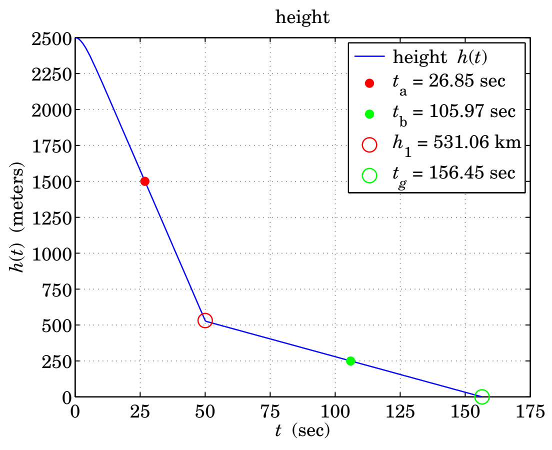
$$0 \leq t \leq t_1$$
$$t_1 \leq t \leq t_g$$

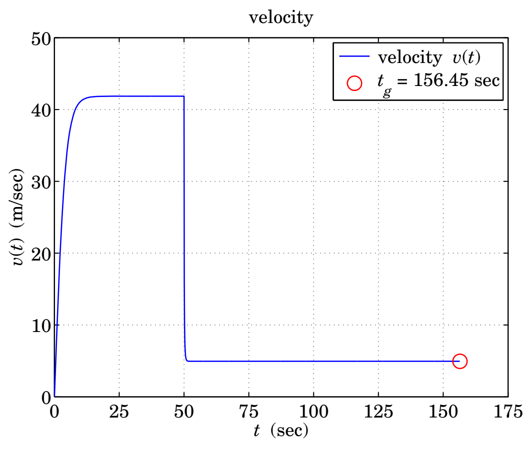
   For each subinterval, calculate the corresponding heights, $h(t)$ and concatenate the two subvectors into a single long vector. Do the same for $v(t)$ and $a(t)$. You may use a time increment of $\Delta t = 0.1$ sec in defining the time vectors spanning the above subintervals.

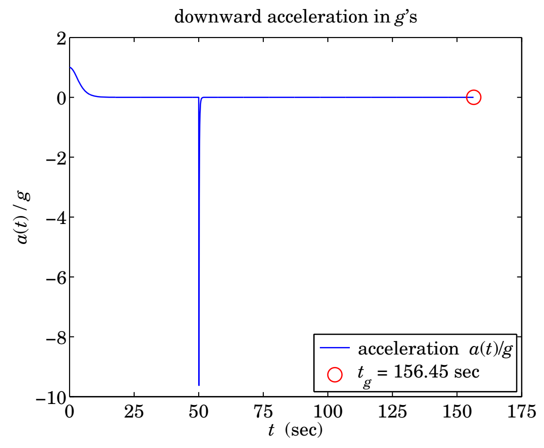
   On three separate graphs, plot $h(t), v(t), a(t)$ over the interval $0 \leq t \leq t_g$.

   Moreover, add the points (t_a, h_a), (t_b, h_b), (t_g, h_g), on the graph of $h(t)$, as well as the point t_g on the graphs of $v(t)$ and $a(t)$.

   In plotting $a(t)$ use units of g. The sudden upward deceleration at $t = t_1$ is due to the assumption that the parachute was opened instantaneously. A quick but more gradual opening can be used (see homework set 8) leading to more realistic g-forces on the skydiver.







2. The attached text data file, **rec0506a.dat**, contains the following student exam data:

```
  Name          netID     E1       E2       E2      |  AVE
--------------------------------------------------|-------
Apple,A.        a5123     85      100       90      |
Exxon,E.        e1567     20       58       65      |
Facebook,F.     f2321     68       45       92      |
Google,G.       g3455     85       87       90      |
Ibm,I.          i4678     86       88       89      |
Microsoft,M.    m2134     55       47       59      |
Twitter,T.      t5321     70       65       72      |
```

a. Open this file using **fopen**, skip over the two header lines using **fgetl**, and use a single **fscanf** command to read the three numerical columns into a 7×3 matrix of grades, say G. Rewind but do not close the file.

b. Then, use a single **textscan** command to read from the file the two text columns N, I, of student names and netIDs, each being a 7×1 cell array of strings. You may now close the opened file.

c. Using the function **mean**, compute the averages of the three exams for each student. The resulting column vector will be entered under the column labeled AVE in the above table. Enlarge the matrix G by appending to it the column AVE of averages. Then, compute the averages of each column, i.e., the average of each exam for the entire class, including the overall average for all exams and for all students.

d. Using the function **sortrows**, sort the exam grades according the column AVE of averages, in descending order. Then, open a new data file, say, **rec0506b.dat**, and using **fprintf**, write into it the sorted data including the original header lines and the computed averages. You may use a for-loop for printing the table data. The contents of this file should be as follows:

```
  Name          netID     E1       E2       E2      |  AVE
--------------------------------------------------|-------
Apple,A.        a5123     85      100       90      | 91.67
Ibm,I.          i4678     86       88       89      | 87.67
Google,G.       g3455     85       87       90      | 87.33
Twitter,T.      t5321     70       65       72      | 69.00
Facebook,F.     f2321     68       45       92      | 68.33
Microsoft,M.    m2134     55       47       59      | 53.67
Exxon,E.        e1567     20       58       65      | 47.67
--------------------------------------------------|-------
           exam_AV   =  67.00    70.00    79.57     | 72.19
```

Upload the new data file with your recitation report. Repeat this part using at most five **fprintf** commands (no for-loops).

Recitation Problems – Week 7

440:127 – Spring 2015 – S. J. Orfanidis

Please read the week-7 powerpoint lecture notes and Ch.8 of the text *prior* to going to your recitations sessions. Please do the following problems during your recitation session, including any additional problems given to you by your TA.

1. Consider the following matrices:

$$A = \begin{bmatrix} 6 & 7 & 7 & 8 \\ 1 & 7 & 1 & 7 \\ 8 & 4 & 3 & 3 \\ 9 & 6 & 1 & 9 \\ 7 & 2 & 1 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 9 & 4 & 2 \\ 2 & 1 & 6 \\ 1 & 8 & 3 \\ 5 & 3 & 6 \\ 1 & 5 & 7 \end{bmatrix}, \quad c = \begin{bmatrix} 9 \\ 1 \\ 8 \\ 3 \\ 6 \end{bmatrix}$$

   a. Using single-index notation, find the index numbers of the elements in each matrix that contain values strictly less than 7. Then, find the row and column numbers of these matrix elements.

   b. With a single command (for each matrix), replace all matrix elements that are strictly less than 7 by the value 70.

   c. Start with the original A, B, c matrices. Find the row and column numbers for the elements in each matrix that contain values strictly greater than 2 and strictly less than 5. Then, find the values of these matrix elements. Then, using the **length** and **find** commands, determine how many elements in each matrix satisfy this condition.

   d. Using the **length** and **find** commands, determine how many elements in each matrix are equal to 1, and then, using the **find** command, determine their row/column locations within each matrix. Then, replace all such elements by -1, and print the new matrices A, B, c.

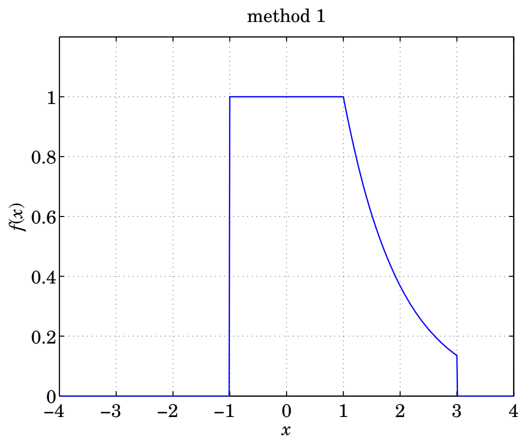
2. Consider the function:

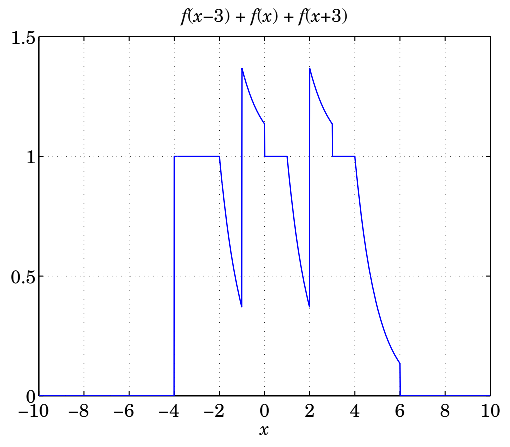
$$f(x) = \begin{cases} 1\,, & \text{if } |x| \le 1 \\ e^{1-x}\,, & \text{if } 1 < x \le 3 \\ 0\,, & \text{otherwise} \end{cases}$$

   a. Implement this function in MATLAB using the three methods outlined on pages 32–41 of the week-7 powerpoint lecture notes. For methods 2 & 3, you need to create appropriate function M-files that are to be uploaded to sakai with the rest of your work.

   b. Using the three different versions of your function, generate three separate graphs of the function $y = f(x)$ over the interval $-4 \le x \le 4$, using 801 equally-spaced points in that interval.

   c. Using your function, e.g., with method-1, generate a graph of the function replicated three times with function copies centered at $x = -3$, $x = 0$, and $x = 3$, exactly as shown below.





3. Consider the two functions,

$$y = f(x) = \exp(-x^2)$$

$$y = g(x) = \frac{x}{2}$$

We wish to carry out a Monte Carlo calculation to determine the areas under these curves over the range $0 \le x \le 2$. Following the discussion of a similar example in week-4 lecture notes, generate $N = 10^4$ random (x, y) pairs that are uniformly distributed inside the rectangle, $0 \le x \le 2$ and $0 \le y \le 1$.

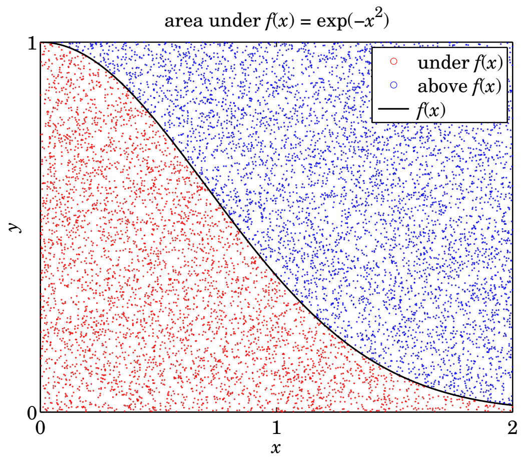
a. For each curve, use appropriate relational operators and the function **find** to determine those (x, y) pairs the lie under the curve, and make a scatter plot of them using red dots.

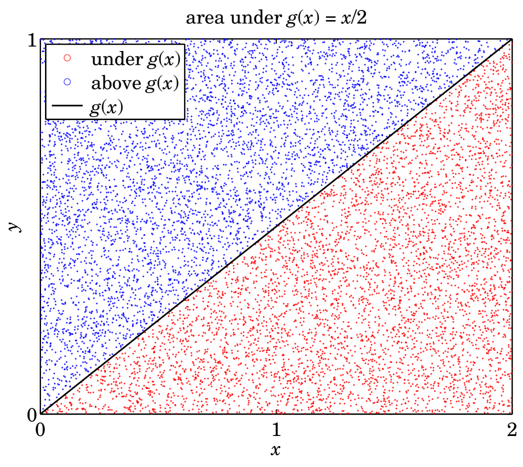
   Hold the graph, and determine those pairs that lie above the curve and make a scatter plot of them using blue dots. Also, add the curves themselve to the graph.

   Finally, determine the area under the curve from the proportion of the red dots.

   Compare the estimated areas with the exact areas given as follows, where **erf** is a built-in function:

$$A_f = \frac{\sqrt{\pi}}{2}\,\mathrm{erf}(2)\,, \qquad A_g = 1$$





b. Make a plot of both curves over $0 \le x \le 2$ and notice that they intersect at a point, say, $x = x_0$, which corresponds to the solution of the equation, $\exp(-x^2) = \frac{1}{2}x$. Determine the point x_0 using the function **fzero**.

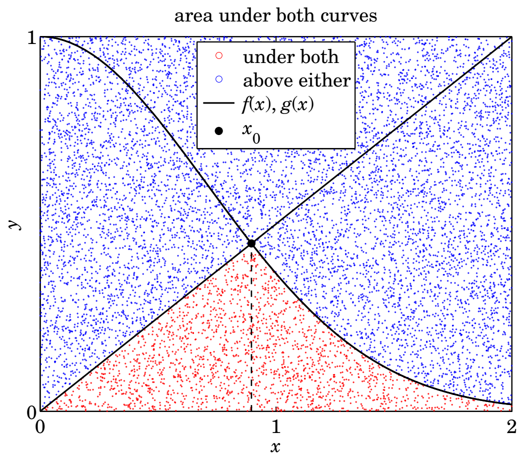
   The area under the $f(x)$ curve over the range $x_0 \le x \le 2$ can be evaluated by the formula:

$$\frac{\sqrt{\pi}}{2}\left[\mathrm{erf}(2) - \mathrm{erf}(x_0)\right]$$

   To that, add the area under the $g(x)$ curve over the range, $0 \le x \le x_0$, to determine the total area under both curves, i.e., over the range $0 \le x \le 2$.

c. Using the same set of (x, y) pairs of part (a), make a scatter plot with red dots of all pairs lying under both curves. Make a scatter plot with blue dots of all those pairs that are not under both curves. Add both curves to the graph, as well as the x_0 intersection point and produce a graph exactly as shown below.

   Estimate the area under both curves and compare it with that found in part (b).


area under both curves
under both
above either
$f(x)$, $g(x)$
x_0
y
x
0
1
0
1
2

Recitation Problems – Week 8

440:127 – Spring 2015 – S. J. Orfanidis

You may wish to get started on this set in advance of your session.

1. Suppose that you start a new savings account with an initial deposit of \$1,000 and that from then on you deposit \$1,200 a month. The account collects 3% annual interest. The accumulated balance at the end of the kth month can be represented by the recursion:

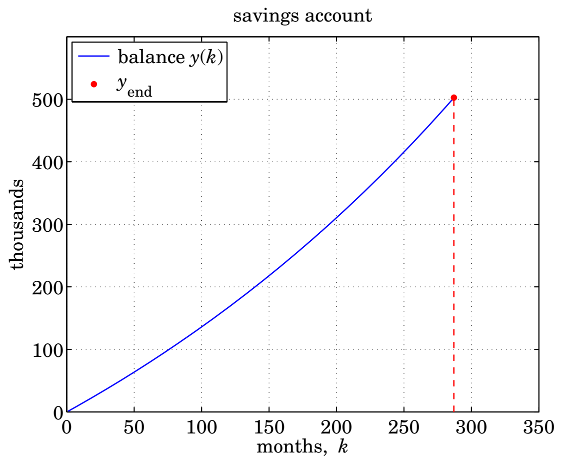
$$y(k) = y(k-1) + r \cdot y(k-1) + 1200\,, \quad k \geq 2$$

and initialized at $y(1) = 1000$, with effective monthly rate of $r = (3/100)/12 = 0.0025$.

   a. Using a for-loop determine how many months it would take to reach a desired balance of \$500,000.
   b. Repeat the previous part using a conventional while-loop.
   c. Repeat using a forever while-loop.
   d. Make a plot of the balance $y(k)$ versus k. Plot the horizontal axis in months, and the vertical axis in thousands of dollars. Add the ending balance on the graph.
   e. Write a function, **account**, that uses method (c) above, and given the initial balance y_0, annual percentage rate R, monthly deposit x, and desired final balance y_{max}, it returns the vector of monthly balances y until the goal of y_{max} is reached. It must have usage:

```
y = account(y0, R, x, ymax)
```

   Note that the length N of the array $y(k)$ is the required number of months to reach the savings goal, and $y(N)$ is the ending balance, which should be just a bit higher than y_{max}.



2. In the week-7 recitation, you solved the following equation using the built-in function **fzero**,

$$\exp(-x^2) = \frac{x}{2} \tag{1}$$

   a. Solve Eq. (1) iteratively by rearranging it in the equivalent way, $\exp(x^2) = 2/x$, or, $x^2 = \ln(2/x)$, or,

$$x = \sqrt{\ln\left(\frac{2}{x}\right)} \tag{2}$$

   and replacing it by the following iteration, initialized at some arbitrary value x_1 that lies somewhere in the range, $0 < x_1 \leq 1$, e.g., $x_1 = 1$,

$$x_{k+1} = \sqrt{\ln\left(\frac{2}{x_k}\right)}, \quad k = 1, 2, 3, \dots \tag{3}$$

Implement this iteration using a *forever while-loop* that exits when two successive iterates become closer to each other than some specified error tolerance such as, $tol = 10^{-10}$, that is, the iteration terminates when, $|x_{k+1} - x_k| < tol$.

Determine the number of iterations N that were required for convergence. The last value $x(N)$ is the required solution of Eq. (1). Save the iterates into a MATLAB array $x(k)$ and plot it versus the iteration index $1 \le k \le N$. Also plot the error difference $e_k = |x_{k+1} - x_k|$, for $1 \le k \le N-1$, using a **semilogy** plot and add the ending-point to the graph. See example graphs below.

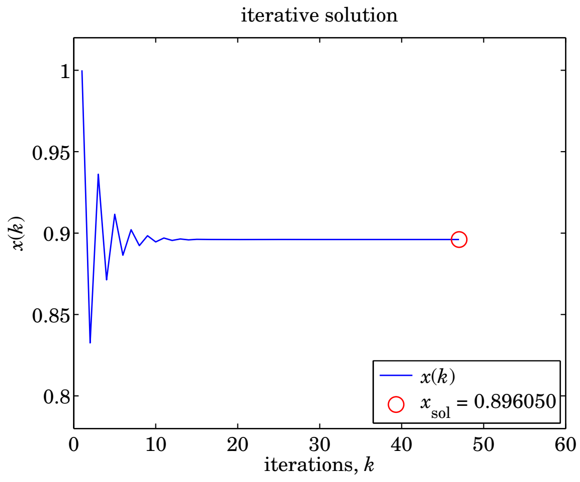
b. Implement the iteration of Eq. (3) using a *conventional while-loop*. Verify that this method produces the identical array $x(k)$ as method (a).

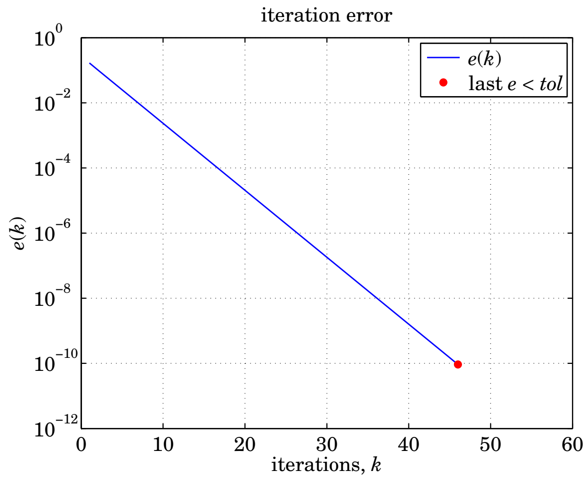
c. Solve Eq. (2) directly by using the built-in function **fzero**. Using trial-and-error determine an appropriate value for the error tolerance, *tol*, of part (a) such that the iterative and the **fzero** solutions agree in at least 12 decimal places.

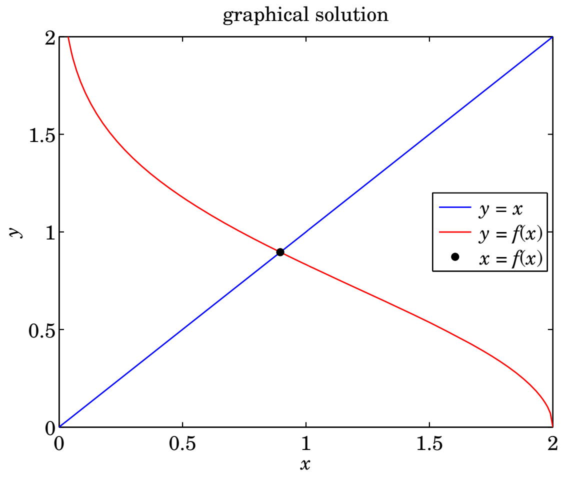
d. The solution of Eq. (1) lies at the intersection of the two curves:

$$y = x \quad \text{and} \quad y = f(x) = \sqrt{\ln\left(\frac{2}{x}\right)}$$

Plot both curves on the same graph over $0 < x < 2$ and place the solution point on the graph.







3. The following infinite series converges to the number 100:

$$\sum_{n=0}^{\infty} (n+1)\,(0.9)^n = 100$$

Define the partial sums, for $k \ge 0$,

$$S_k = \sum_{n=0}^{k} (n+1)\,(0.9)^n = (0+1)\,(0.9)^0 + (1+1)\,(0.9)^1 + (2+1)\,(0.9)^2 + \cdots + (k+1)\,(0.9)^k$$

which can be cast in the recursive form:

$$\begin{aligned} S_0 &= (0+1)\,(0.9)^0 = 1 \\ S_k &= S_{k-1} + (k+1)\,(0.9)^k\,, \quad k \ge 1 \end{aligned} \tag{4}$$

a. Define the following function $f(k)$ in Matlab as a single-line vectorized anonymous function and plot it vs. k in the interval $0 \le k \le 300$ using **semilogy**, noting that it rapidly decreases with k,

$$f(k) = (k+1)\,(0.9)^k\,, \quad k \ge 0$$

Because $f(k)$ represents the kth term in the above series, that is, $f(k) = S_k - S_{k-1}$, one can estimate the minimum value of k beyond which the difference $|S_k - S_{k-1}|$ will become smaller than a specified error tolerance, say, $tol = 10^{-10}$, by solving the following equation for k:

$$f(k) = tol \quad \Rightarrow \quad (k+1)\,(0.9)^k = tol$$

Solve this equation for k using the function **fzero**, and using the function **ceil**, round the result up to the next integer, denoted by, K. This represents the maximum number of terms that we need to use in the series S_k to achieve an accuracy of $tol = 10^{-10}$. Indicate the point at $k = K$ on the graph of $f(k)$.

b. Using a *forever while-loop* whose stopping condition is the inequality $|S_k - S_{k-1}| \le tol$, calculate S_k and determine the number K of iterations until the loop is exited.

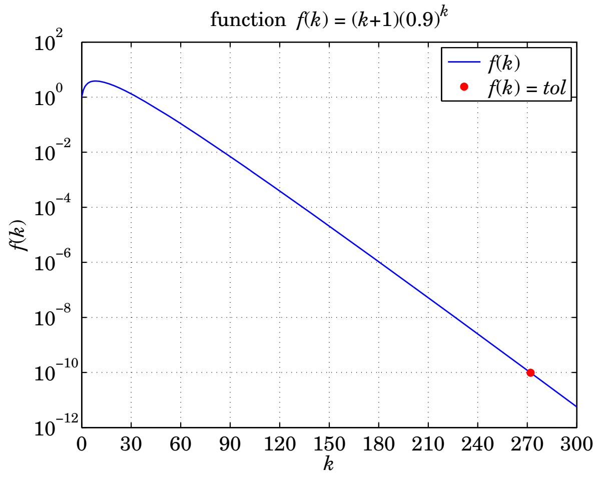
Save the iterates into a Matlab array, say $S(k+1)$, for $0 \le k \le K$, and plot it versus k, indicating the final computed value with a red marker, as well as the exact value of the limit. See example graph at end.

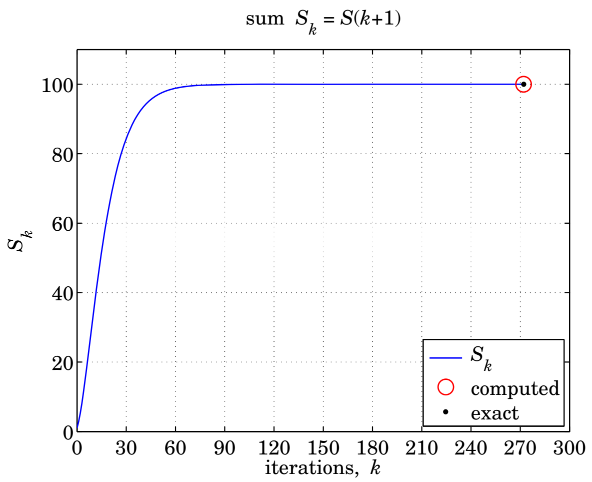
Note: The summation index $k = 0, 1, 2, \ldots$, is not a Matlab index because it starts from $k = 0$, and hence S_k is not a Matlab array. However, we can map it into a Matlab array by shifting the index, i.e., $S(k+1) = S_k$, for $k = 0, 1, 2, \ldots$. The recursion can then be rephrased in terms of the Matlab array:

Math Notation		Matlab Notation
$S_0 = (0+1)\,(0.9)^0 = 1$		$S(1) = (0+1)\,(0.9)^0 = 1$
$S_1 = S_0 + (1+1)\,(0.9)^1$	$\Rightarrow$	$S(2) = S(1) + (1+1)\,(0.9)^1$
$S_k = S_{k-1} + (k+1)\,(0.9)^k\,, \quad k \ge 1$		$S(k+1) = S(k) + (k+1)\,(0.9)^k\,, \quad k \ge 1$

c. Repeat part (b) using a *conventional while-loop*, whose continuation condition is the inequality, $|S_k - S_{k-1}| > tol$, that is, it will loop until this condition is violated. Determine the final value of k upon exit from the loop and compare it with the K of part (b).

d. Repeat parts (b-c) without saving the series values into an array $S(k)$, but rather using only two scalar variables S and S_{old} that are recycled in each iteration. Skip the plots of part (b) in this case, but in all cases do print out the final value of k and the final approximation error $E = |S - 100|$.


function $f(k) = (k+1)(0.9)^k$
$f(k)$
$f(k) = tol$
10^{2}
10^{0}
10^{-2}
10^{-4}
10^{-6}
10^{-8}
10^{-10}
10^{-12}
0 30 60 90 120 150 180 210 240 270 300
k


sum $S_k = S(k+1)$
S_k
computed
exact
100
80
60
40
20
0
0 30 60 90 120 150 180 210 240 270 300
iterations, k

Recitation Problems – Weeks 9 & 10

440:127 – Spring 2015 – S. J. Orfanidis

This recitation combines material from weeks 9 & 10. Please do the following problems, including any additional problems given to you by your TA.

1. The acceleration of a car is given by the following formula that includes an acceleration term A due to the engine torque and an air drag term that is proportional to the square of the velocity:

$$\frac{dv}{dt} = A - C\cdot v^2$$

It is desired to measure the 0–60 mph performance of the car, as well as its braking performance from 60 mph. To this end, the driver first applies maximum torque until the car reaches 60 mph and measures the corresponding time, say, t_{60}, and then, she applies the brakes until the car stops, and measures the stopping time, t_{stop}.

Let us discretize the time as, $t_n = (n-1)T$, where $n = 1,2,3,\dots$, and $T = 0.001$ sec is a small time step. Then, the above differential equation may be replaced by the following difference equation:

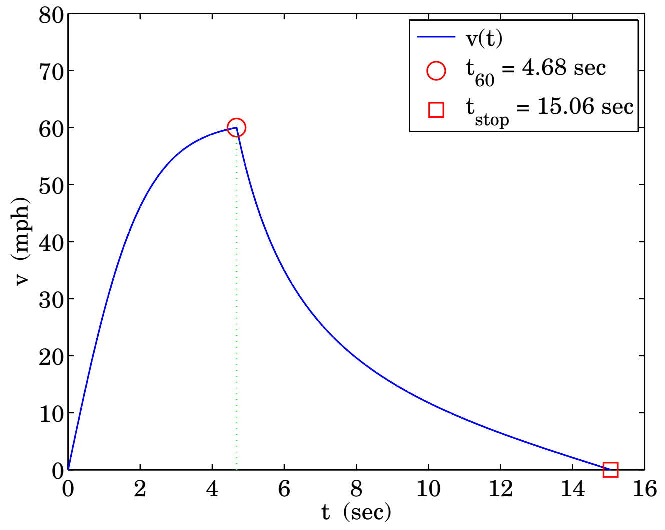
$$\frac{v(n+1)-v(n)}{T} = A(n)-C\cdot v^2(n) \quad\Rightarrow\quad \boxed{v(n+1) = v(n)+T\cdot\big[A(n)-C\cdot v^2(n)\big]} \tag{1}$$

where $v(n)$ is in units of mph,[†] the drag constant is chosen as $C = 0.008$, and the initial acceleration and subsequent braking action can be modeled by:

$$A(n) = \begin{cases} 30\,, & \text{if } t_n \le t_{60} \\ -2\,, & \text{if } t_{60} < t_n \le t_{\text{stop}} \end{cases}$$

a. Using a *forever* while-loop and starting with zero initial velocity, iterate Eq. (1) until $v(n)$ reaches 60 mph, and determine the iteration index, say, n_{60} when the loop exits, and the corresponding 0–60 mph time, $t_{60} = (n_{60}-1)T$. As you iterate Eq. (1) in part (a), save the computed velocity and time, $v(n), t(n)$, into arrays.

b. Then, continue with another forever while-loop that starts at $n = n_{60}$, and iterates until the velocity is reduced to zero. Determine the index upon exit from the loop, say, n_{stop}, and the total stopping time, $t_{\text{stop}} = (n_{\text{stop}}-1)T$. Append the new arrays $v(n), t(n)$ into the arrays of part (a) and make a plot exactly as the one shown below.



[†]Note that $v^2(n)$ stands for $[v(n)]^2$.

2. Consider the linear system:

$$\begin{aligned} 2x_1 - x_2 + 2x_3 &= 2 \\ x_1 + 2x_2 + 2x_3 &= 4 \\ -x_2 + 4x_3 &= 12 \end{aligned}$$

a. Cast this linear system in matrix form, $A\mathbf{x} = \mathbf{b}$, and solve it using the backslash operator.

b. Using the definition of the Jacobi iteration parameters, $B, \mathbf{c}$, given on p. 51 of the week-10 lecture notes, cast the above linear system in the form:

$$\mathbf{x} = B\mathbf{x} + \mathbf{c}\,, \qquad \text{where} \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

Use a forever while-loop as outlined on p. 52 of the lecture notes, to solve this system iteratively. Use error tolerance, $tol = 10^{-12}$, and initial value, $\mathbf{x}_0 = [1, 1, 1]'$. Determine the number of iterations, say, K, it took to converge, and verify the converged solution.

b. Within your while-loop of part (b), save the iterates into a $3\times K$ matrix and plot its components versus $0 \le k \le K-1$, producing a graph as shown at the end.

c. The iteration converges if the so-called spectral radius is less than one, computed in MATLAB by:

$$\rho = \texttt{max(abs(eig(B)))}$$

Calculate ρ and verify that it is less than one in this example.

d. The error $E(k)$ of the linear system at the kth iteration is defined in terms of the kth iterate $\mathbf{x}(k)$ by the norm:

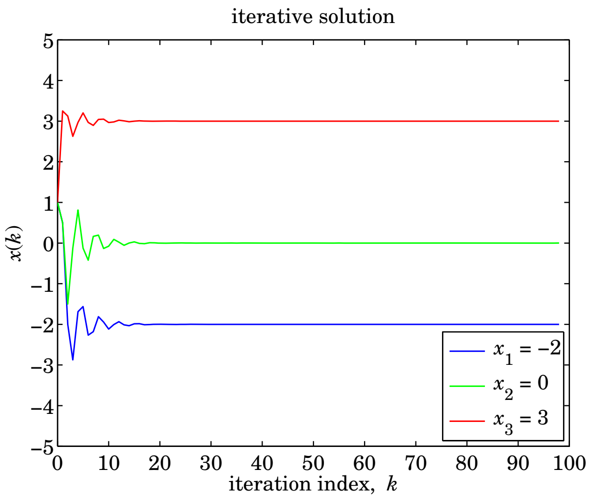
$$E(k) = \text{norm}\big(A\mathbf{x}(k) - \mathbf{b}\big)$$

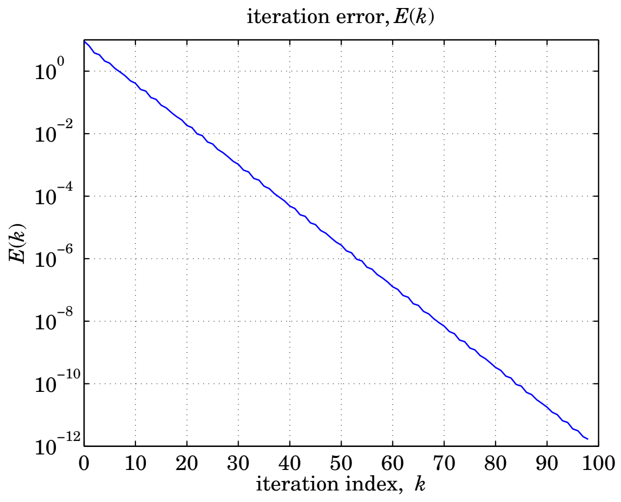
Calculate and plot $E(k)$ for $0 \le k \le K-1$, using a **semilogy** plot as shown at the end. Explain why the last value of $E(k)$ is not quite as small as the specified error tolerance *tol*.

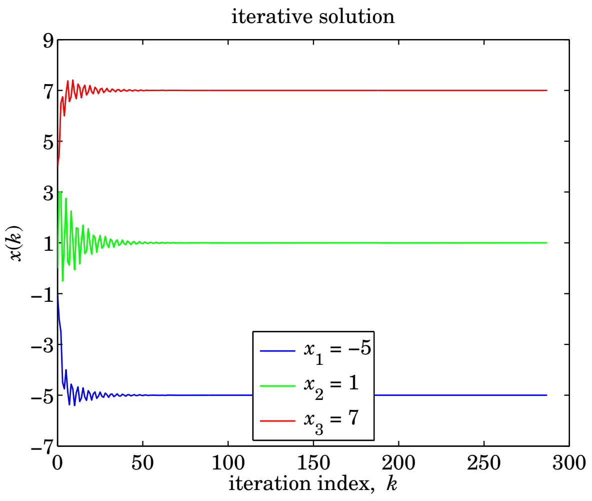
e. Repeat parts (a-d) for the following linear system:

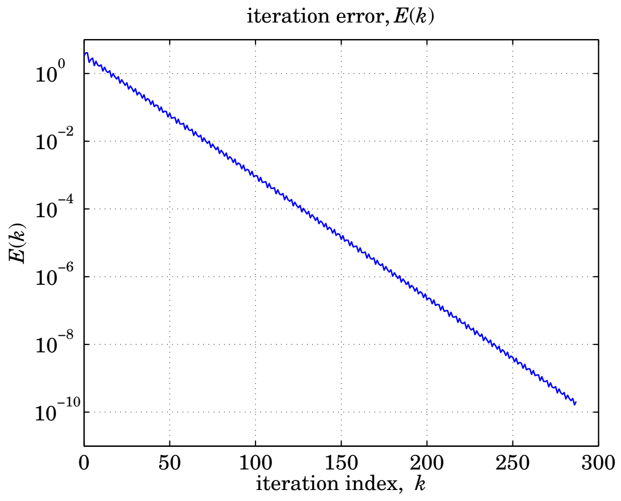
$$\begin{bmatrix} 1 & -1 & 2 \\ 1 & 1 & 2 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 8 \\ 10 \\ 2 \end{bmatrix}$$

Note that this system may require some preliminary re-arrangement. Here, use, $tol = 10^{-10}$.


iterative solution
x(k)
iteration index, k
$x_1 = -2$
$x_2 = 0$
$x_3 = 3$


iteration error, $E(k)$
$E(k)$
iteration index, k


iterative solution
x(k)
iteration index, k
$x_1 = -5$
$x_2 = 1$
$x_3 = 7$


iteration error, $E(k)$
$E(k)$
iteration index, k

Recitation Exercises – Week 11

440:127 – Spring 2015 – S. J. Orfanidis

Please do the following problems during your recitation session.

1. The viscosity of air and other gases can be modeled as follows as a function of temperature:

$$\nu = \frac{T^{3/2}}{aT + b} \qquad \text{(Sutherland's formula)} \tag{1}$$

where T is the temperature in absolute units, i.e., related to degrees Celsius by $T = 273.15 + T_C$, and a, b are constants. This relationship can be written in the following forms:

$$\frac{T^{3/2}}{\nu} = aT + b \tag{2}$$

$$\frac{1}{\nu} = aT^{-1/2} + bT^{-3/2} \tag{3}$$

The following data are given, where T_C is in degrees Celcius and ν in milli-N·s/m^2:

T_{Ci}	-20	0	20	40	70	100	200	300	400	500
ν_i	1.63	1.71	1.82	1.87	2.03	2.17	2.53	2.98	3.32	3.64

a. Using the form of Eq. (2) and the **polyfit** function, perform a least-squares fit to determine the parameters a, b. Then, use these parameters in Eq. (1) and plot ν versus T_C in the interval $-20 \le T_C \le 520$ $^{\text{o}}$C and add the data points to the graph.

b. Repeat part (a) using the form of Eq. (3) and using $\{T^{-1/2}, T^{-3/2}\}$ as basis functions.

c. Perform the fitting using the built-in function **nlinfit** by defining the fitting function directly from Eq. (1):

$$f(c, T) = \frac{T^{3/2}}{c_1 T + c_2}, \qquad c = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}$$

Determine the parameter vector c and make a similar plot as in parts (a,b). You may use the parameters obtained in part (a) or (b) as the initial search vector for the **nlinfit** function.

d. Using the function $f(c, T)$ that you defined in part (c), define an anonymous Matlab function $J(c)$ that represents the sum of the absolute values of the model error, that is,

$$J = \sum_i \big| \nu_i - f(c, T_i) \big| = (L_1 \text{ criterion})$$

and find the parameter vector c that minimizes J using the function **fminsearch**, as discussed in week-11 powerpoints. Make a similar plot as in part (c). You may use the parameters obtained in part (c) as the initial search vector for the **fminsearch** function.

e. Collect the results of the four methods and with at most three **fprintf** commands, print a table exactly as shown below:

```
   a         b        method
-----------------------------------
 6.5443   871.2282   polyfit
 6.7469   795.1066   basis functions
 6.5472   870.5470   nlinfit
 6.4626   910.2021   L1
```

f. Repeat parts (a-e) by introducing an outlier in the data, as follows:

T_{Ci}	-20	0	20	40	70	100	200	300	400	500
ν_i	1.63	1.71	1.82	1.87	2.03	2.17	3.50	2.98	3.32	3.64

Generate similar graphs as above, and a table like the following, and discuss which of the four methods is best in the presence of such outliers.

```
     a           b        method
------------------------------------
  6.3968     822.5254   polyfit
  5.8576    1032.3636   basis functions
  6.3179     821.0588   nlinfit
  6.5484     851.3076   L1
```

2. Data on the viscosity of water as a function of temperature are given below, where T is the temperature in degrees Celsius and ν is the viscosity in mPa·sec:

T_i	10	20	30	40	50	60	70	80	90
ν_i	1.31	1.00	0.80	0.65	0.55	0.47	0.40	0.36	0.32

It is desired to determine which of the following two models is appropriate for the above data:

$$\begin{aligned} &\text{model 1:} \quad \nu(T) = \exp\big(c_0 + c_1\,T\big) \\ &\text{model 2:} \quad \nu(T) = \exp\big(c_0 + c_1\,T + c_2\,T^2\big) \end{aligned} \tag{4}$$

a. Using the *basis-functions* method, fit both models to the given data and determine the c_i coefficients. Then, determine the model that fits the data better.

b. Repeat the fitting using the **polyfit** method, and verify that you get identical results as in part (a).

c. To verify the correctness of your fitted values, plot ν versus T from Eq. (4) using your best model evaluated at 100 equally-spaced points over the interval $0 \le T \le 100$, and add the data points $\{T_i, \nu_i\}$ to the graph, plotted with dots.

d. Using at most four **fprintf** commands, print the c_i coefficients from the two models as in the following table:

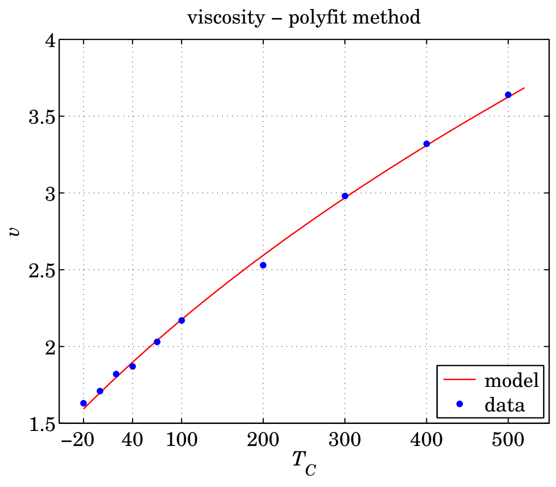
```
               c0        c1       c2
---------------------------------
model-1:   0.3329   -0.0174
model-2:   0.5260   -0.0279   0.0001
```

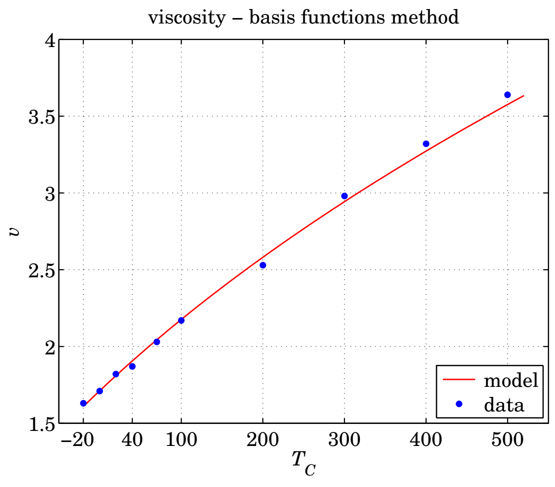
3. The following table gives the values of the density ρ, pressure P, and absolute temperature T of the atmosphere versus height (in km) above sea level,

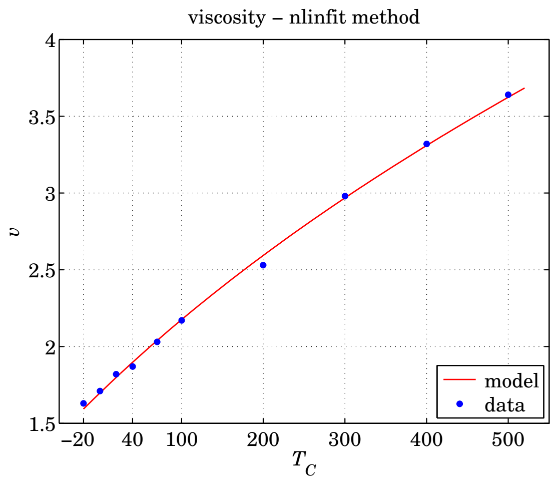
```
   h       rho          P          T
 (km)  (kg/m^3)       (Pa)        (K)
----------------------------------
   0     1.2250   101325.00   288.15
   8     0.5258    35651.62   236.22
  16     0.1665    10352.83   216.65
  24     0.0469     2971.75   220.56
  32     0.0136      889.06   228.49
  40     0.0040      287.14   250.35
```

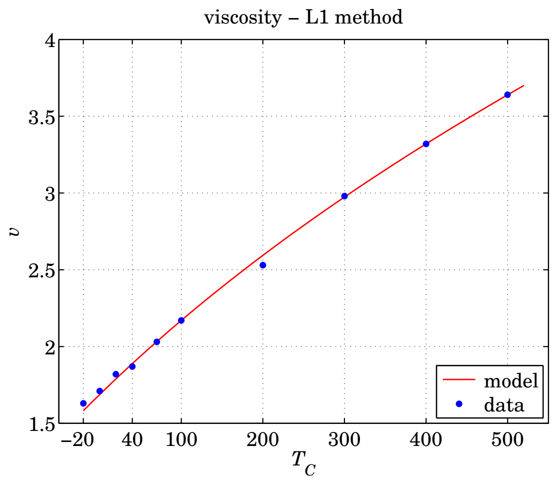
For convenience, the data can be loaded from the file **rec11b.dat**. Perform linear interpolation simultaneously on the three columns ρ, P, T that enlarges the table to the following one, and use at most four **fprintf** commands to print it as shown below, where the starred entries must be replaced by the computed interpolated values,

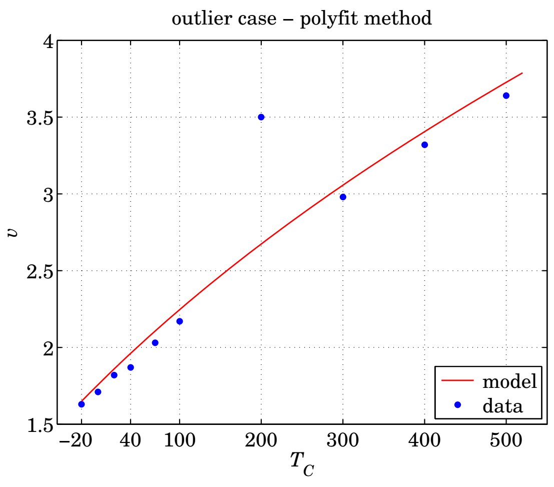
```
  h       rho          P         T
(km)   (kg/m^3)      (Pa)       (K)
----------------------------------
  0     1.2250   101325.00   288.15
  4     *.****    *****.**   ***.**
  8     0.5258    35651.62   236.22
 12     *.****    *****.**   ***.**
 16     0.1665    10352.83   216.65
 20     *.****     ****.**   ***.**
 24     0.0469     2971.75   220.56
 28     *.****     ****.**   ***.**
 32     0.0136      889.06   228.49
 36     *.****      ***.**   ***.**
 40     0.0040      287.14   250.35
```

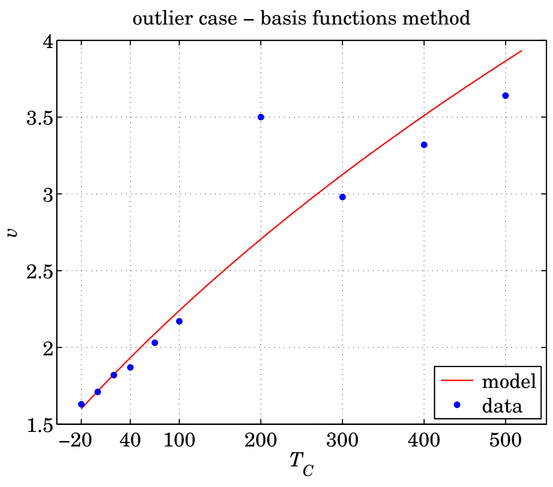


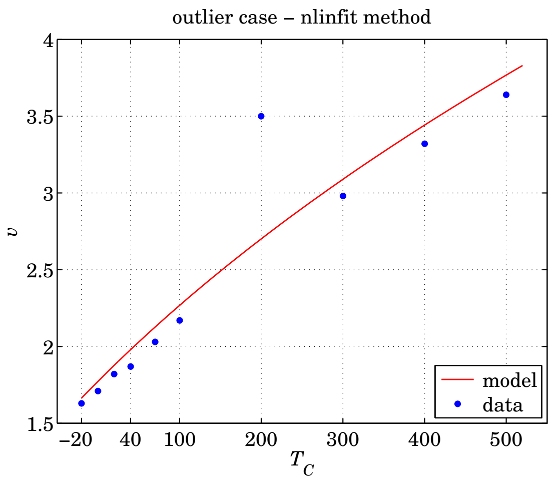


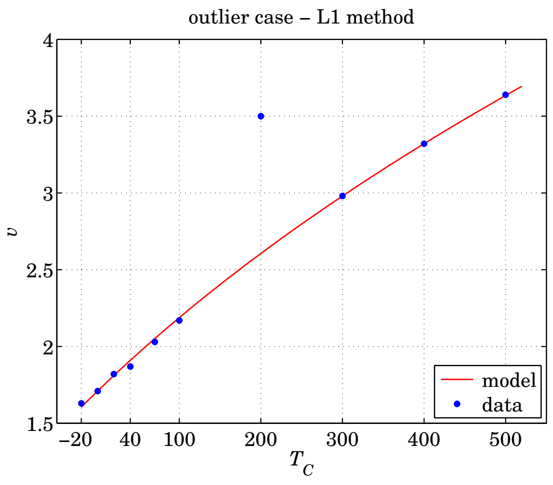


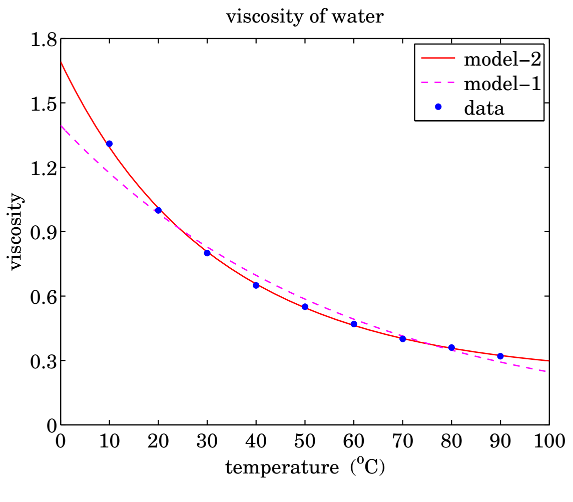



outlier case – polyfit method
model
data
v
T_C


outlier case – basis functions method
model
data
v
T_C


outlier case – nlinfit method
model
data
v
T_C


outlier case – L1 method
model
data
v
T_C


viscosity of water
model–2
model–1
data
viscosity
temperature (°C)

Recitation Exercises – Week 12

440:127 – Spring 2015 – S. J. Orfanidis

Please do the following problems during your recitation session.

1. The following table shows the temperatures of New York City for certain years and certain months. For convenience, the incomplete data can be loaded from the file **NYCinc.dat** (you may see the complete table in the data file **NYCtemp.dat** under week-11 resources on sakai.)

```
year  jan   feb   mar   apr   may   jun   jul   aug   sep   oct   nov   dec
----------------------------------------------------------------------------
1971  27.0        40.1        61.4        77.8        71.6              40.8
1972
1973  35.5        46.4        59.5        77.4        69.5              39.0
1974
1975  37.3        40.2        65.8        75.8        64.2              35.9
```

   a. Use the two-dimensional interpolation function **interp2** to fill the missing entries in the table. Use both linear and spline interplation and compare the results. Then, print them exactly as shown below (shown is the linear case), using at most three **fprintf** commands,

```
year  jan   feb   mar   apr   may   jun   jul   aug   sep   oct   nov   dec
----------------------------------------------------------------------------
1971  27.0  33.5  40.1  50.8  61.4  69.6  77.8  74.7  71.6  61.3  51.1  40.8
1972  31.3  37.3  43.3  51.9  60.5  69.0  77.6  74.1  70.5  60.3  50.1  39.9
1973  35.5  41.0  46.4  53.0  59.5  68.5  77.4  73.5  69.5  59.3  49.2  39.0
1974  36.4  39.8  43.3  53.0  62.6  69.6  76.6  71.7  66.8  57.1  47.2  37.5
1975  37.3  38.8  40.2  53.0  65.8  70.8  75.8  70.0  64.2  54.8  45.3  35.9
```

   b. Then, repeat the linear interpolation case by using two calls to the one-dimensional interpolation function **interp1**, by first filling the vertical gaps by interpolating each month at the new years and producing the following intermediate result, which you must also print with three **fprintf** commands:

```
year  jan   feb   mar   apr   may   jun   jul   aug   sep   oct   nov   dec
----------------------------------------------------------------------------
1971  27.0        40.1        61.4        77.8        71.6              40.8
1972  31.3        43.3        60.5        77.6        70.5              39.9
1973  35.5        46.4        59.5        77.4        69.5              39.0
1974  36.4        43.3        62.6        76.6        66.8              37.5
1975  37.3        40.2        65.8        75.8        64.2              35.9
```

   and then filling the horizontal gaps by interpolating the values between columns by working on the transposed version of the above part. Finally, compare the interpolated values from the **interp2** and the double **interp1** methods.

2. It is desired to fit the following data to the nonlinear model given by Eq. (1)

x_i	y_i
1.0	0.18
1.4	0.21
1.9	0.21
2.1	0.20
2.7	0.20
3.0	0.19
3.3	0.18
3.9	0.16
4.2	0.16

$$y = \frac{1}{\dfrac{a}{x} + b\,x} \tag{1}$$

Perform the data fitting using the following three methods and plot the fitted model over the interval $1 \le x \le 4.2$ and place the data points on the graphs. Compare the fitted coefficients a, b from the three methods.

a. Using the *basis function method* after transforming the model to the form:

$$\frac{1}{y} = \frac{a}{x} + b\,x$$

Do not use **polyfit** here.

b. Using the *basis function method* after transforming the model to the form:

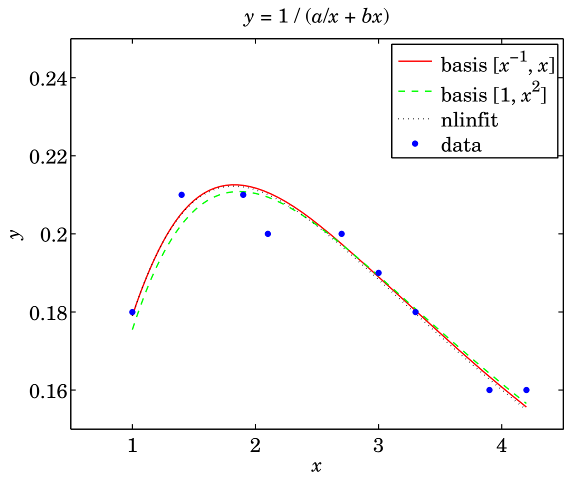
$$\frac{x}{y} = a + b\,x^2$$

Do not use **polyfit** here.

c. Using the built-in function **nlinfit** applied directly to the model of Eq. (1).

d. Collect together the values of a, b for the three cases, and using at most three **fprintf** commands print the results in a table like the one below:

```
   method            a         b
----------------------------------
basis {1/x, x}     4.3012   1.2858
basis {1, x^2}     4.4293   1.2696
nlinfit            4.2925   1.2933
```



3. The *Antoine Equation* gives the vapor pressure of a substance as a function of temperature:

$$\log_{10}(P) = A - \frac{B}{C+T} \tag{2}$$

where the pressure P is in Pascals, and T in Kelvins. For example, the known values of the parameters for Ethanol are as follows, see `http://en.wikipedia.org/wiki/Antoine_equation`,

$$\begin{bmatrix} A \\ B \\ C \end{bmatrix} = \begin{bmatrix} 10.33 \\ 1642.89 \\ -42.85 \end{bmatrix} \tag{3}$$

Consider the following 11 measurements of the vapor pressure of Ethanol at the indicated temperatures:

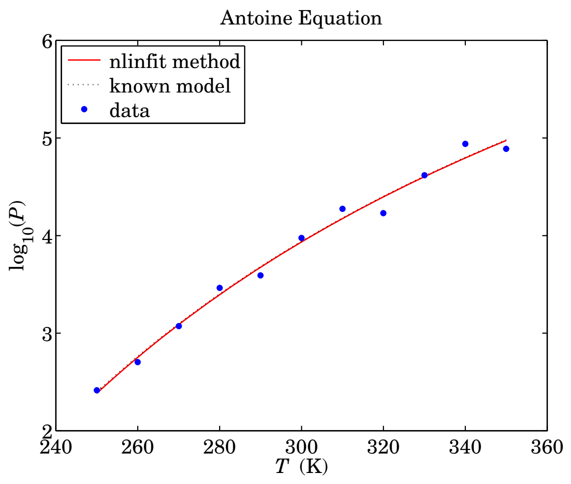
T_i	250	260	270	280	290	300	310	320	330	340	350	(K)
P_i	260	504	1184	2910	3913	9462	18841	16952	41529	86903	77537	(Pa)

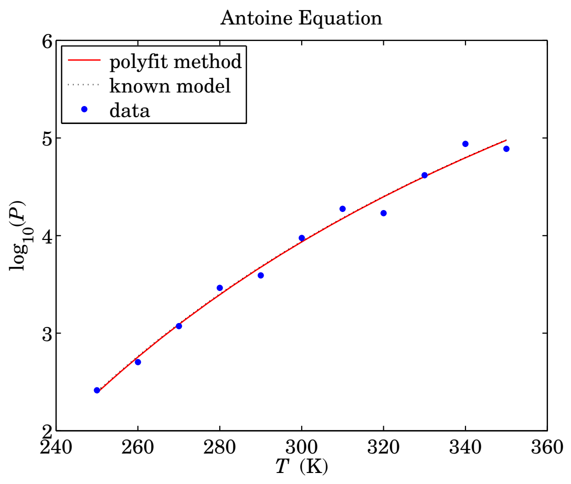
a. Even though the model of Eq. (2) is simple, it cannot be put in the standard basis-functions framework when all three parameters A, B, C are unknown. Start with an initial vector, for example,

$$\begin{bmatrix} A_0 \\ B_0 \\ C_0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1000 \\ -1 \end{bmatrix}$$

then, use the function **nlinfit** to fit the above data to the model of Eq. (2). Compare the known and the fitted coefficients A, B, C, and make a plot of $\log_{10}(P)$ versus T using both the fitted and the known coefficients, and also add the data points to the graph (see example graph at the end).

b. If the coefficient A is considered known from Eq. (3), then B, C can be fitted using the basis function method or **polyfit**. After transforming Eq. (2) appropriately, use **polyfit** to determine B, C and make a similar plot as in part (a).


Antoine Equation
nlinfit method
known model
data
log10(P)
T (K)


Antoine Equation
polyfit method
known model
data
log10(P)
T (K)