Aalto-yliopisto

Credit Exercises

If you wish to obtain the credits for the course, choose 4 of the problems below and prepare a published .pdf-file. Send the pdf-file and the source code by email to Juha (juha.kuortti 'at' aalto.fi).
Please: Use **semicolons(;)** to avoid long outputs!
Deadline for submission will be discussed in class.

Exercise 1: Write functions F2C and C2F that convert Fahrenheit scale to Celsius, and Celsius to Fahrenheit. Be sure to have them work for vectors as well as scalars.

$$T_C = \frac{5}{9}(T_F - 32).$$

(F: Fahrenheit, C: Celsius)
Test your functions, especially: (40,40) and (0C,32F).
Make a table of values and plot.

Exercise 2:

1. Evaluate $y = x^2$ for `x=-4:0.1:4`.
2. Add random noise to these samples. Use `randn`. Plot the noisy signal with markers (.) (period).
3. Fit a 2^{nd} degree polynomial to the noisy data.
4. Plot the fitted polynomial on the same plot, using same x-values and a red line.

Exercise 3: This has lots of quite detailed instructions, looks long but you just need to follow the instructions.
Below are all the steps you need. You should also add your own meaningful comments. When you are done and have “green square” on top-right of the editor, you should publish your work : >> `publish('throwBall.m','pdf')`, or use the editor's built in publishing tools
Study the trajectory of a thrown ball.

a) In the process use double-percents to divide your file into small blocks you can exctute (`CTR-ENTER`) to test your proceeding.

b) At the top of your file, define some constants

i) Initial height h at release $= 1.5m$

ii) Gravitational acceleration $g = 9.8m/s^2$

iii) Velocity of ball v at release $= 4m/s$

iv) Angle of velocity vector at release $= \theta$ on 45^o

c) Make a time vector with 1000 linearly spaced values on the (closed) interval $[0, 1]$.

d) If $x(t)$ is distance and $y(t)$ is height at time t, the equations below describe the trajectory of the ball:

$$\begin{cases} x(t) = v\cos\left(\theta\frac{\pi}{180}\right)t \\ y(t) = h + v\sin\left(\theta\frac{\pi}{180}\right)t - \frac{1}{2}gt^2. \end{cases}$$

(You can also use the recently introduced functions `sind, cosd` [d for "degree"].)

Define vectors `x` and `y` based on these formulas.
Note: The length of these vectors is 1000, so **don't forget the semicolons (;)**.
(You can look at the the first 10 values, say, by `x(1:10), y(1:10)` .)

e) Approximate the time when the ball hits the ground.

i Find the index, when the height first becomes negative (use **find**).

ii The distance at which the ball hits the ground is value of vector `x` at that index.

iii Display the words: 'The ball hits the ground at distance xhit meters', where `xhit` is the value you found above. (Use `disp` and `num2str` in the form `disp(['The ball ... ',num2str(xhit),' meters'])`)

f) Plot the ball's trajectory.

i Just use `plot` in its very basic form, plot also gridlines (**grid on**)

ii) Label the axes meaningfully and give the figure a title, use **xlabel, ylabel, title**

iii) use **hold on**

iv) Plot the ground as a black dashed line.
Hint: One way would be: `plot(x,zeros(size(x)),'--k')` or `plot(x,0*x,'--k')`.
In fact, all one needs for a line, is two points, so the most "economic" way would be just to take first and last elements of the vectors x and y for the plot-function. Do this!

Exercise 4: This assignment focuses on the connection of matrices, images and singular values.
The *singular value decomposition* of a matrix is

$$\mathbf{A} = \mathbf{USV}^T,$$

where matrix S is a diagonal, and U and V are orthogonal square matrices; if this all means nothing to you, don't worry – this assignment requires essentially no mathematical framework.
The information contained in matrix be compressed (in a sense) by dropping off parts of the singular value decomposition, and conveniently the size of singular value tells the importance of the associated singular vectors; even more conveniently they are already sorted inside the S; or in MATLABese, `U(:,1:k)*S(1:k,1:k)*V(:,1:k)'` is the best possible approximation of A using $2k$ vectors.
Any image can be thought of as an $m \times n$ matrix, where element i, j describes the color value of the pixel at the location. We will now see, how singular values can be used to compress images.
Read in any image you wish using MATLAB's `imread`-command. It will (usually, but little depending on the image format). $m \times n \times 3$ array. Each of the three layers of the array corresponds to color intensities of red, green and blue respectively. First, convert the image to grayscale with command `rgb2gray`, or pick one of the layers: `im = A(:,:,1)`. After this, do the singular value decomposition with command `[u,s,v] = svd(im)`.
Now you can test with which k the commands

```
>> M = u(:,1:k)*s(1:k,1:k)*v(:,1:k)';
>> image(M)
```

produces results that are distinguishable. The large components of the image should also start appearing first, which makes singular value decomposition a useful tool in pattern recognition.
In order to do this properly for colored images, you should do the above process separately for each of the three layers of the array; also, for fun, you could try different values of k for each layer, suppressing certain colors more than others.

Exercise 5: Here are 25 observations y_k, taken at equally spaced values of t.

```
t = 1:25
y =[5.0291    6.5099    5.3666    4.1272    4.2948
    6.1261   12.5140   10.0502    9.1614    7.5677
    7.2920   10.0357   11.0708   13.4045   12.8415
   11.9666   11.0765   11.7774   14.5701   17.0440
   17.0398   15.9069   15.4850   15.5112   17.6572];
y = y';
y = y(:)
```

a) Fit the data with a straight line $y(t) = \beta_1 + \beta_2 t$, and plot the residuals, $y(t_k) - y_k$. You should observe that one of the data points has a much larger (in absolute value) residual than the others. This is probably an *outlier*.

b) Discard the outlier, and fit the data again by a straight line. Plot the residuals again. Do you see any pattern in the residuals?

c) Fit the data, with the outlier excluded, by a model of the form

$$y(t) = \beta_1 + \beta_2 t + \beta_3 \sin(t)$$

d) Evaluate the third fit on a finer grid over the interval `[0,26]`. Plot the fitted curve, using line style `'-'` , together with the data, using line style `'o'` . Include the outlier, using a different marker, `'*'` .

Exercise 6: • Plot the unit square : $-1 \le x \le 1, -1 \le y \le 1$ and the unit circle inside it. (i.e. a circle with radius one)

a) Generate random points, uniformly distributed on the unit square $-1 < x <, 1 < y < 1$. Find an approximation for π by counting the ratio of the number of points inside the circle with the total number of points generated.

• Write a function that approximates the area of the ellipse:

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$$

```
[Aappr,Atrue]=function Aellipse(a,b,N)
```

b) Make plots: color points inside and points outside differently. If you include a plot in your function, test for N to avoid plotting for too large value.

Hint: Generate two random vectors x and y. Find a suitable condition for logical indexing to pick the “inside-poins”.

Note: You can use arithmetic (sum) to a logical vector. Try to avoid loops.