Aalto SCI-it **Heikki Apiola, Juha Kuortti**

Matlab 1, basics

Exercises 2 - 9.11.2016

If you need credit for the course, choose 5 problems denoted "Credit", prepare a published .pdf-file, include .m-file/files also. Send by mail to both Heikki and Juha. Minimum requirement: 3 correct, well documented solutions.

Please: Use **semicolons(;)** to avoid long outputs!

Deadline for submission will be discussed in class.

Note: The `publish`-command has to be given in command window, not in the editor window, the latter would be calling itself indefinitely!

If you create .mlx-files, please export them to pdf (send the mlx also).

---

1. First experience on writing little functions: Write functions F2C and C2F. Be sure to have them work for vectors.

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

   (F: Fahrenheit, C: Celsius)

   Test your functions, especially: (40,40) and (0C,32F).

   Make a table of values and plot.

2. mlDi013.tex
   Study Mathwork's example
   http://se.mathworks.com/examples/matlab/mw/graphics-ex24497781-determine-minimum-and-maximum-points-and-add-text

   Let $f(x) = \arcsin(2x\sqrt{1-x^2})$ on $[-1, 1]$.

   (a) Find its min and max points and value and use similar graphical presentation as in the above example.

   (b) Try also `fminsearch`.

   You can very well stop here, there's quite a number of exercises left.

   (c) Same with zeros: first use the basic "find" etc. technique, then use `fzero`.

   (d) Study also this function:

   $$f(x) = (x-1)\sin\left(\frac{12x}{x^2 + 0.4x + 0.1}\right) \quad x \in [-4, 4]$$

   Find out, how many zeros it has on the interval, and find the smallest on $[-0.2, -0.1]$

Hint:
Use `linspace, plot, zoom, ylim` to get an approximation from the plot. Then use `find` for more accuracy.

3. Credit

   Newton's method for finding a zero of a function is based on using the zero of the tangent-line at the starting point (current point) as an approximation of the next point in iteration. This leads to the iteration step:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}.$$

   The followin code does it

```
k=0;xprevious=-Inf;   % Fancy way to get started
while abs(x-xprevious) > eps*abs(x)
    xprevious=x;
    x=x-f(x)/fder(x);
    k=k+1;
end
```

   Complete this code into a function in the file `myNewton.m`.

   Header line: `function z=myNewton(f,fder,x,maxiter)` Here `f,fder` are function handles.

   In the above code the counter `k` serves only as a means to avoid an infinite (or too long) loop, it's not needed for the iteration.
   In your function, take `maxiter` as an optional argument, which could have a suitable default value (eg. 40). Test it in the while-condition (or add an `if,break,end`-block with the test. Use `nargin` to test if `maxiter` was given.

   Write a suitable help-text in the first comment-lines. You might include one of the obvious tests as a help-text: `myNewton(@sin,@cos,2)`.

   Solve the following problems with your "myNewton" (choosing intial points with the help of the graph):

   a) $x \cos x = \sin x + 1, \quad 0 < x < 2\pi$

   b) One of the functions in the previous (mlDi013)

   Try also the function `fzero`

   Optional:
   Consider returning the whole iteration sequence or sequences if you give several starting points. In the latter case it might be natural to use a cell-array, because the sequences have different lengths.

4. CREDIT

   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 $\mathbf{S}$ is a diagonal, and $\mathbf{U}$ and $\mathbf{V}$ 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 $\mathbf{S}$; or in MATLABese, `U(:,1:k)*S(1:k,1:k)*V(:,1:k)'` is the best possible approximation of $\mathbf{A}$ 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.

5. Credit

   (a) Plot the unit square : $-1 \leq x \leq 1, -1 \leq y \leq 1$ and the unit circle inside it.

   (b) 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.

   (c) 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)
```

(d) 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.

Note: Monte Carlo simulations are "embarassingly parallel", thus suitable in Matlab 2.

6. Credit

   Write a function that returns the dot product and cross product of two vectors

   Write a script that tests at least with coordinate unit vectors.

   Using random vectors, test the "triple cross product" formula:

$$(u \times v) \times w = (u \cdot w)v - (v \cdot w)u.$$

   Hint: Use the formula of developing according to first row of a 3×3 matrix and Matlab-function `det`.

   Learn: Picking submatrices and working with their determinants etc.

7. The *bisection* algoritm is a well-known "safe-and-slow" method of finding zeros (or at least initial brackets) of a continuos function.

   Why not subdivide for instance into 100 or more instead of 2 parts at a time. Vector operations give good support to such an idea. That could also lead to a search for initial brackets for finding all zeros on a global interval "simultaneously".

   Here is an idea of how one could proceed:

```
a=-1; b=1
x=linspace(a,b);y=sin(x);I=find(sign(y(1)) ~= sign(y));
I1=I(1);a=x(I1-1),b=x(I1)
```

   Or maybe better:

```
x=linspace(-pi,pi,24); % 100 parts
y=sin(4*x);            % Example function
sy=sign(y);
d=diff(sy);
signchange=d~=0;
leftendpts=x(1:end-1);
rightendpts=x(2:end);
aaok=xalkup(d~=0);   % accepted left endpts.
bbok=xloppup(d~=0)   % accepted right endpts.
cc=(aaok+bbok)/2    % Midpoints of bracket intervals
sin(4*ceet)
```

Write a function *nsection* based on this or similar ideas. Test it with some of the above examples.

Next project is to combine this with vectorized Newton's method. (Vectorization of Newton only means using array-operations with dots (.).)

It is enough to do *nsection* this time, but if you are interested, continue with *Newton* by any means.

We will return to this in Matlab 2-course, because this is a naturally parallelizable problem. At the same context it will be interesting to study, which initial points lead to convergence, and which not. There will probably emerge a fractal structure.

ODE, Euler, ode45, dirfields

One problem from this area [Monday 7.11]

8. Credit

   mlBas018throwBall
   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').`

   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**)

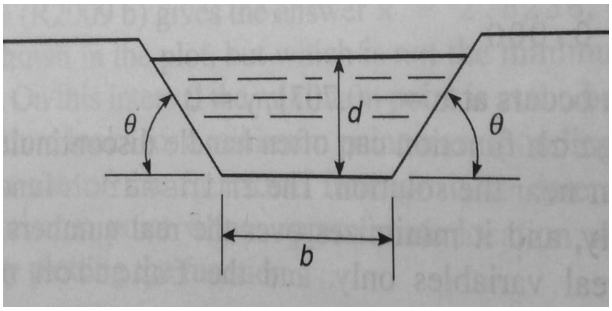
(ii) Label the axes meaningfully and give the figure a title, use **xlabel, ylabel, title**

(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!

---

9. Credit **optim in 2 vars** Figure 3.2–3 [W.J. Palm p. 128] shows the cross section of an irrigation channel. (Finnish:"kastelukanava")



A preliminary analysis has shown that the cross-sectional area of the channel should be $100m^2$. [1] to carry the desired water flow rate. To minimize the cost of concrete used to line the channel, we want to minimize the length of the channel's perimeter. Find the values of d, b, and Θ that minimize this length.

The perimeter length L can be written in terms of the base b ,depth d ,and angle Θ as follows:

$$L = b + \frac{2d}{\sin\Theta}.$$

[1] Misprint in previous version (10)

The area of the trapezoidal cross section is

$$100 = db + \frac{d^2}{\tan\Theta}$$

The variables to be selected are b, d , and Θ . We can reduce the number of variables by solving the latter equation for b to obtain

$$b = \frac{1}{d}(100 - \frac{d^2}{\tan\Theta}).$$

Substitute this expression into the equation for L . The result is

$$L = \frac{100}{d} - \frac{d}{\tan\Theta} + \frac{2d}{\sin\Theta}.$$

We must now find the values of d and Θ to minimize L .

First define L as a (vectorized) function of $[d, \Theta]$

Use mesgrid and plot/surf/contour on a physically reasonable area, like $0 < d < 30, 0 < \Theta < \frac{\pi}{2}$. You can of course use min for the "height matrix" Z similarly as in 1-dim. case. This way you can at least get starting values for `fminbnd`. For instance, $\Theta = 0.1, d = 1$ should lead to a reasonable result. Can you decide, whether this is the global minimum ?

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

    Hint: `polyfit,polyval`

11. (Basic interpolation) mlCF02
    Construct the interpolating polynomial for data that is obtained by evaluating $f(x) = \cos(1+x^2)$ at 7 uniformly spaced points. on $[0, 3]$. Plot data and interpolating polynomial in the same figure.

    Check
    Polynomial should go through all datapoints.

12. Credit
    Data fitting with outlier

    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, `'*'` .