Numerical Geology

WS 2017/2018

BSc class on how to do useful stuff in geosciences, using MATLAB/OCTAVE

Johannes-Gutenberg University, Mainz

(These notes are prepared by E. Moulas, Y. Podladchikov, L. Raess and S. Omlin (University of Lausanne) and B. Kaus, G. Reuber, A. Bauville, Adina Püsök, JGU Mainz)

last update 15.10.2017

1. Introduction

In a few words we can say that MATLAB[1] is a big calculator. MATLAB can do everything that any calculator can do (addition, multiplication etc) and in addition it can plot the solutions. The advantage of MATLAB is that, it can store values into the computer memory and it can be programmed. The easiest way to learn it, is to try it. By starting MATLAB we see a window with the symbol ">>" (Fig. 1), this is called the "**command window**". In the command window we can write like in a normal calculator.

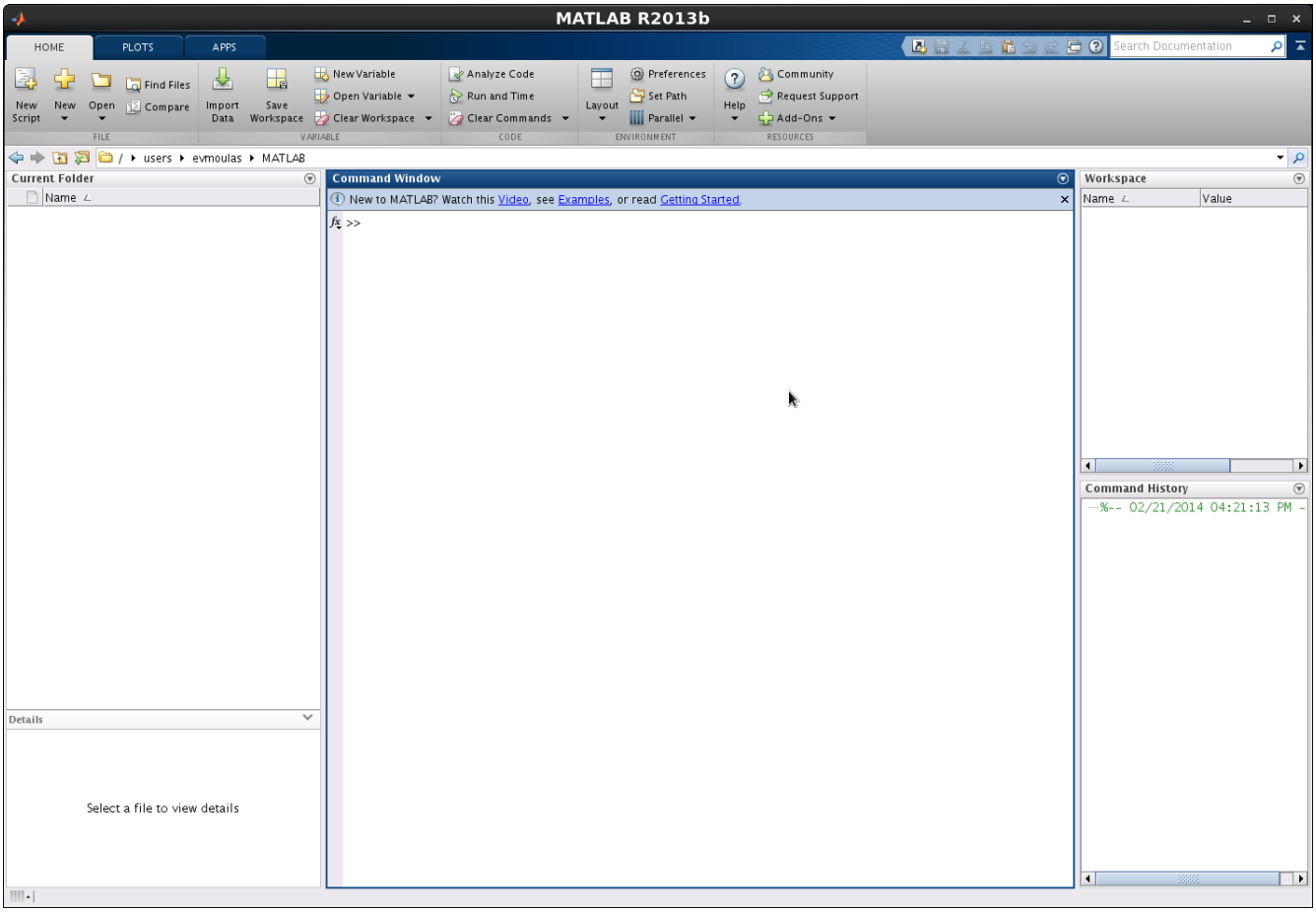


Figure 1. The MATLAB window.

For example we can write "1+1" and the answer that we get is "2". In addition, we can store one value in the memory by typing "M=1". This operation will save the value 1 under the name M, therefore

1 Other open-source alternatives are SciLab (http://www.scilab.org/) and Octave (https://www.gnu.org/software/octave/) Octave has a syntax that is nearly identical to that of MATLAB, and the latest version, 4.2.1 has a graphical user interface (GUI), that is very similar to that of MATLAB. You can download a binary for windows; on mac you will have to install the version of macports which works at least for me. Most of the things we discuss in this class, with the exception of building GUIs, can be done with Octave as well.

every time in the future we want to use this value we can type "1+M" or "M+M" and the result should be the same. Another advantage of storing values into the memory is that they can be updated. For example if we type:

```
>> M = M+2
```

we get

```
M =

     3
```

This means that the computer took the old value of M (which was 1) and added 2 so the new value of M is 3. At this point, we note that MATLAB also recognizes the functions *sin, cos, tan* etc, but we must be careful because by default MATLAB evaluates these functions in **radians**. To evaluate these functions in degrees use the functions *sind, cosd, tand* instead. Constants like "pi" or "e" are also included. You can type "**help** function" for details (e.g. help cosd, help exp etc).

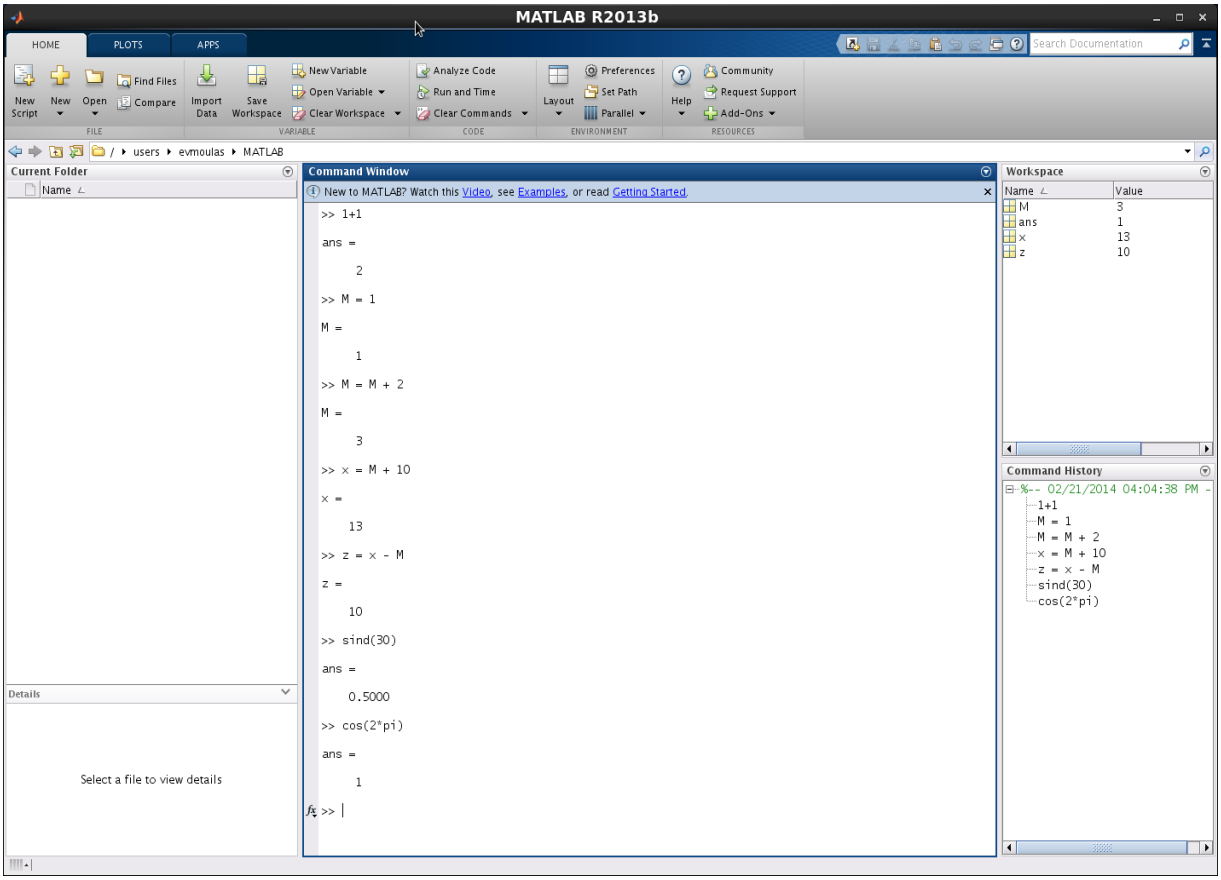


Figure 2. Examples of MATLAB inputs and outputs.

2. The editor

Instead of writing every command on the command window, it is preferable to write in the **editor**. The editor is a place where we can write our commands so that we can modify them, and save them for future use. For example, type "edit" in the command window. Then another window will open (Fig. 3) and there you can write all the commands you want.

By using the editor you can save your current program, open an older version of your program and run your program. The files that are saved in the editor have the ending "**.m**". Apart from the "run" button of the editor, the programs can be executed by typing the name of the program (without the .m) at the command window. While doing this, you need to make sure that you are in the same directory as where the file is saved.

Now lets try to write a program in the editor:

```
clear
M = 100
M = M+1
M = M+1
M = M+1
```

and we can save it as "my_first_example.m"[2]

Then MATLAB will return:

```
M =
  100
M =
  101
M =
  102
M =
```

2 Note that MATLAB will not recognize program names that start with a number and contain spaces or special symbols

103

This shows that MATLAB returns sequentially the output of every command that it was written in the editor. The word “clear” in the beginning is used to clear the memory from previous applications.

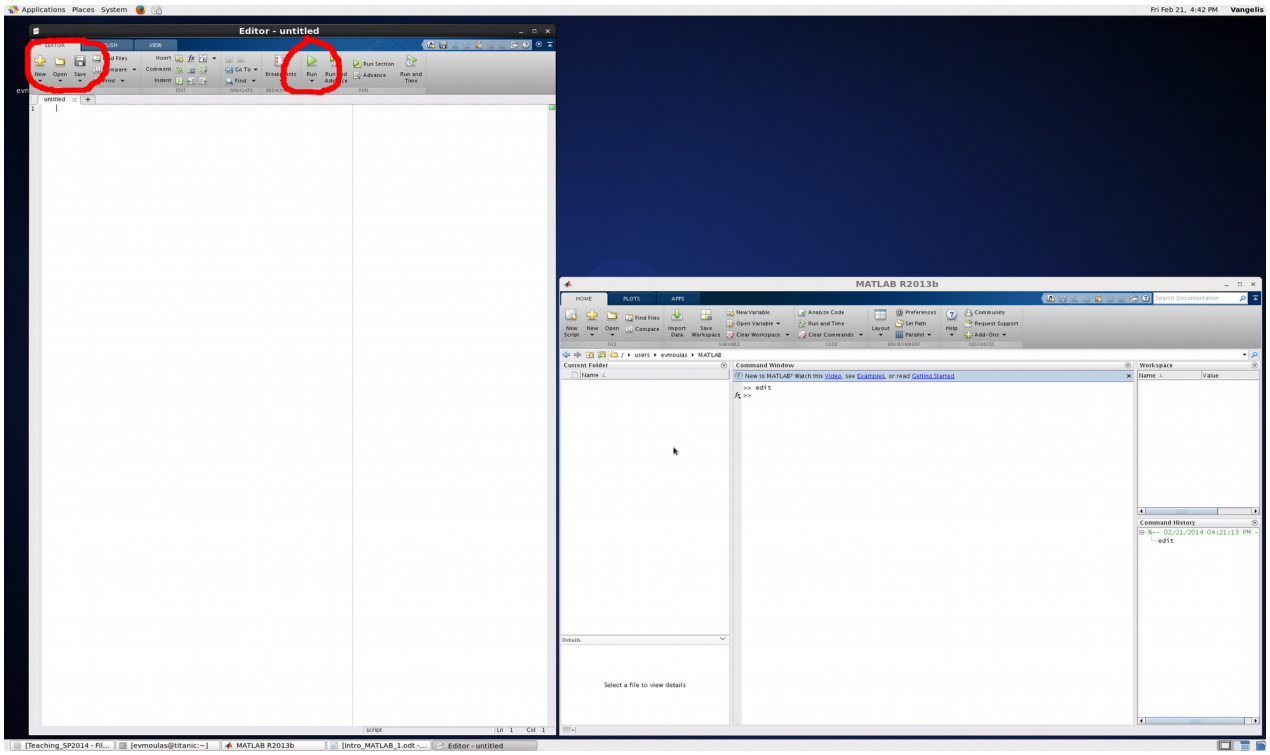


Figure 3. The editor window on the left. The buttons to create, open, save and run a program are highlighted with red.

If we want to check what is the value of “M” at the end of the calculations, we can type “display(M)” or just “M”. The editor is similar to a word-document editor in the sense that we can “create new” files, and “open”, “save” or “save as” our progress (Fig. 3). If our program is not running, **we should first look at the command window** where MATLAB reports possible errors in red.

3.1 Loops

In the previous example we had to use the variable "M" three times in arithmetic calculations. In order to avoid repetition we can use the concept of loops. There are two basic ways to write a loop, the easiest way is to use the "for" loop as shown below

```
clear                %Clear the memory
M  = 100             %Initial value of M is 100
nt =  3              %3 times of repetition
for i = 1:nt         %Start Loop
      M = M + 1      %Every time take previous M and add 1
end                  %End Loop
```

this program will produce the same output as in the one above. You can see that it uses a variable i (**called index**) which increases by 1 and it does the same thing "nt" times (nt=3 in this case). By writing the program like that, we can change the variable "nt" as we want. The symbol "%" is used to add a comment in the MATLAB program. **Anything that it is written after "%" in the same line will be ignored by MATLAB**. Therefore we can use it to write our comments that will help us (or other people) understand the code. The word "end" is used to show where the part of the loop ends. As we will see in a later section, the importance of "end" becomes obvious when more than one loops are used.

Another way to write the same program is to use the "while" loop as shown below:

```
clear                %Clear the memory
M  = 100             %Initial value of M
nt =  3              %times of repetition
i  =  0              %Initial value of i is 0
while i < nt         %For as long as i is less than nt – Start Loop
      M = M + 1      %Read M, add 1, and update M
      i = i+1;       %Read i, add 1, and update i
end                  %End the loop
```

this program will produce the same result as above. The symbol ";" (called semicolon) is used to

suppress the printing of the variable i on the command window. It is important to be sure that "i" will eventually have a value that will make the loop to stop otherwise the computer will run forever. **A way to stop the program in the case of a mistake is by typing "Ctrl+C" on the command window. Sometimes is useful to type "Ctrl+C" more than once.**

Consider the following example which is similar to the previous one:

```
clear                %Clear the memory
M  = 100             %Initial value of M
nt =  3              %times of repetition
i  =  0              %Initial value of i is 0
while i < nt         %For as long as i is less than nt – Start Loop
      M = M + 1      %Read M, add 1, and update M
      i = i-1;       %Read M, substract 1, and update i
end                  %End the loop
```

Question: Will the program run forever? Can find out why?

3.2 The "if" statement

Another important tool in programming is the "if" statement. The use of the "if" statement allows us to program different commands for different cases. In the following example the computer will read a variable A and then it will decide if its smaller than 5 (A<5), equal to 5 (A==5) or larger than 5. Then the computer can perform different operations based on what is the value of A.

For example:

```
clear                   %clear the memory
A = 5;                  % By using the semicolon we suppress the print of A
if A<5                  % Start of "if" statement
        A = A+1;        % If A<5 then Add 1
elseif A==5
        A= A+2;         % If A=5 then Add 2
else                    % For any other case (here A>5)
        A = A+3;        % add 3
end                     %End of the if statement
display(A)              %This command is used to display the value of A
```

Then, the computer output will be:

```
A =

        7
```

Note that for the case that we want to check equality we use the "==" instead of "=" because the latter is used to assign values (i.e. A = 5 will assign the value 5 on A). Other symbols that can be used are "<=" for "less or equal than" or ">=" for "greater or equal than"[3].

In the following section we will see a set of examples that will help you understand the basic concepts described above.

3 type "help <=" or "help >=" in the command window for more details

Example 1 (my_first_program.m)

1a) Money versus time

Lets assume that we have an initial amount of money in the bank (M=1000) and every year we save some money (dM=100).

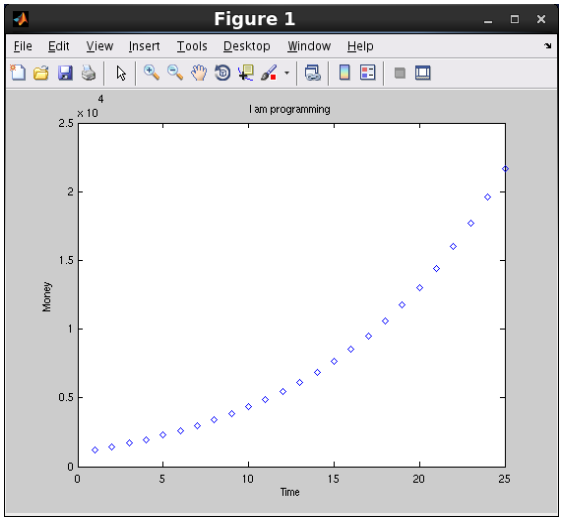
- What will be the amount of money after 25 years? Write a small MATLAB program to do this (called my_first_program.m), and always start this program with "clear, clf"

- Create a plot of the total amount of money that you have (M) versus time in years. Put "time" and "money" on the x- and y-axis and plot the money with blue circles (hint: you can use "help plot") to find out how to do this. Hint: the command "hold on" is useful for this.

1b) Money with an interest rate

Now we can make it a bit more complex. Let's assume that the bank pays you an interest of 10% at the end of every year. Save your earlier code under a new name (Example_1b.m, for example) and modify it such that you plot the amount of money you have versus time.

The result should look like below:



1c) Use a different line-style and color for the plot

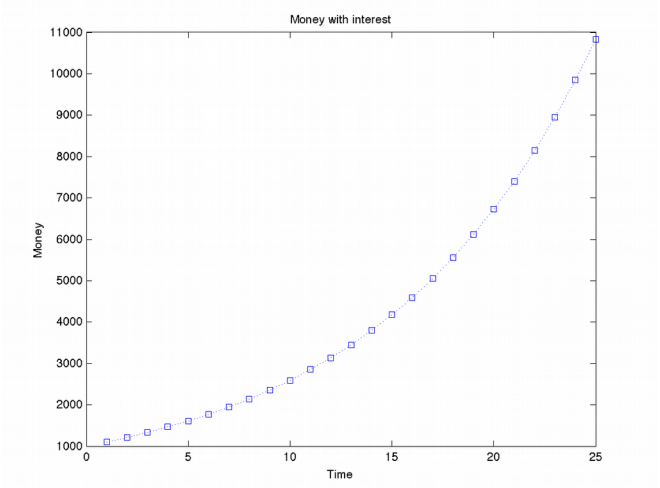
It would often be nice to be able to plot the above data not only as circles, but as a line with symbols. In order to do that we have to store the data in an array (more about that tomorrow). In practice this works as follows:

```
for i = 1:10                %Start Loop
        Time_vector(i) = i;
        Data_vector(i) = 10+i*i;
end                         %End Loop
```

At the end, `Time_vector` will hold the data for the x-axis and `Data_vector` the data to be plotted on the y-axis. Once this is done, you can plot the data with:

```
plot(Time_vector, Data_vector,'ro-')
```

Modify the code of 1b and plot the data as a blue dotted curve with square symbols, such that it looks like shown below. Try to change the linewidth as well.



1d) Fit an equation through these data-points

It would be nice to have an equation that fits this data, that depends in the initial money and time. The ones among you that remember mathematics will maybe be able to derive this. Yet, most of you might have forgotten about that. For those ones, there is good news: MATLAB has a build-in curve-fitting tool that is part of every figure window. In order to use it, simply go to the figure window and select "Tools->Basic Fitting". In there you can select a linear or a quadratic curve fit (which will automatically be plotted together with the data). Matlab will also give you the equation and how well it fits the data.

– Use the plot of example 1c, and fit an equation through the data points.

– What is the equation that fits these data well over the given range?

Example 2 (Traveling car)

2a) Going from Mainz to Berlin

Lets assume that we have a car that is traveling in a straight road from Mainz to Berlin at a constant speed of 119km/h. What is he distance from Mainz after the first 2 hours? Plot the distance as a function of time for 10-minute time intervals?

INFO BOX

Note that we know from physics that dD/dt = V, where D is distance, t is time and V is velocity. In the previous example we approximate dD as D^{new}-D^{old} and therefore the above formula was written as

$(D^{new}-D^{old})/dt = V$

which can be rewritten as:

$D^{new} = D^{old} + dt \cdot V$

In this manner, we can solve numerically similar equations where only the rate-of-change and an initial condition are given.

2b) Returning halfway

Lets assume now that we are at Kassel (ca 200km NW of Mainz) and we decide to come back keeping the same speed[4]. In order to solve such a problem we have to instruct MATLAB to decide that if we covered the distance of 200km and we use the opposite velocity in order to return to the starting point. To do this, we will use the "if statement" discussed earlier in this chapter. Modify the code of example 2a to do this (save it under a new name!), plot distance versus time and tell us when we will be back in Mainz.

Example 3 (Car traveling with constant acceleration)

In the previous example, we assumed that the velocity V was constant (which is given in units of m/s or km/hour). Now we assume that the acceleration (not the velocity) is constant in the above example (Example 2).

The only thing we have to change is the update rule for velocity and then use this velocity to update the

4 Speed is the absolute magnitude of velocity

position as above. Remember that α=dV/dt, where α is the acceleration, V is velocity and t is time.

For this example assume α = 60 km/hour^2 and that the initial velocity is zero. Modify the code of example 2a, to compute distance as a function of time. Create a plot of distance versus time, and create a second plot of velocity versus time.

Example 4 (Car traveling in two dimensions)

If we would like to investigate a more realistic problem we would have to consider traveling in more than one dimensions. The good news are that we can calculate that by applying minimum modifications to the one-dimensional case. Lets now take Example 2 (car traveling at constant velocity) and assume that the initial velocity is 80km/hour towards the North and 90km/hour towards the West. The code now looks as follows:

```
clear,clf,clc
V_North     =     80;   %Constant car velocity
V_West      =     90;   %-//-
D           =      0;   %Initial distance from Mainz at t=0
D_N         =      0;   %Distance ? direction North
D_W         =      0;   %Distance ? direction West
t           =      0;   %Initial time = 0
t_time      =      2;   %Total time needed for calculation 2 hours
dt          =    1/6;   %Evaluate solution every 1/6 of an hour
nt          = t_time/dt; %Two hours divided by dt
for i = 1:nt
    t   = t + dt;              %Time
    D_N = D_N + V_North*dt;    %Distance in direction North
    D_W = D_W + V_West*dt;     %Distance in direction West
    D   = sqrt(D_N.^2+D_W.^2); %Total distance (Pythagoras' formula)
    plot(t,D,'o'),hold on
end
xlabel('time (hours)')
ylabel('Distance (km)')
title('I am going to Paris')
display(D)
```

Exercise: plot the position of the car in the x-y plane (in map view)? Beware of sign conventions.

4. Afternoon problem sets

So far, we looked at a number of topics in which we slightly increased the complexity. By now, you should have sufficient background knowledge to solve the following problems independently. We will discuss the solutions at the end of day 1.

Problem 1 (Varying interest rates)

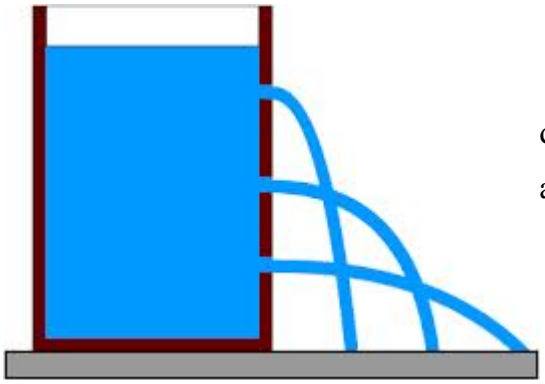
Assume that a bank has one customer that has initially 1000 EUR in a bank account. At the end of every year there is an interest on the amount of money. The interest varies as shown below:

until the end of first year	5%
until the end of second year	7%
after the end of the second year	8%

Can you calculate the amount of money that the customers have after 10 years? Plot the results.
Note that the interest is **added at the end of the year**.

Problem 2 (Rate of Change)

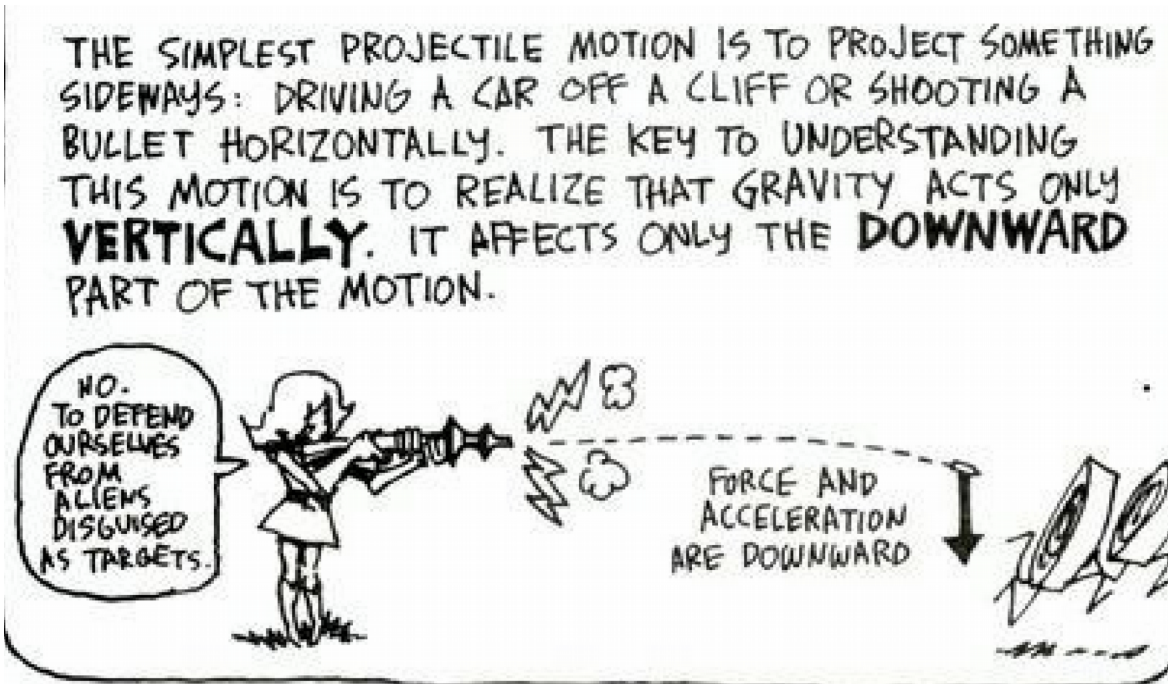
Assume that we have a barrel filled with water. At the bottom of the barrel there is a hole. If the height of the water is h, and dh/dt=-c*h (c is a positive coefficient), can you estimate the evolution of h for the first 2 seconds? (assume that c = 1[sec^{-1}], h = 100[cm] at t=0 and that dt = 0.01[sec]). Write a code to do this.



Note: The rate at which the water leaves the barrel depends on the height of the water. In this example we assume only one hole at the bottom.

Can you calculate the solution for different coefficients?

Problem 3 (Projectile motion)[5]



We now look at how a projectile is deviated under the influence of gravity. Read the cartoon above.

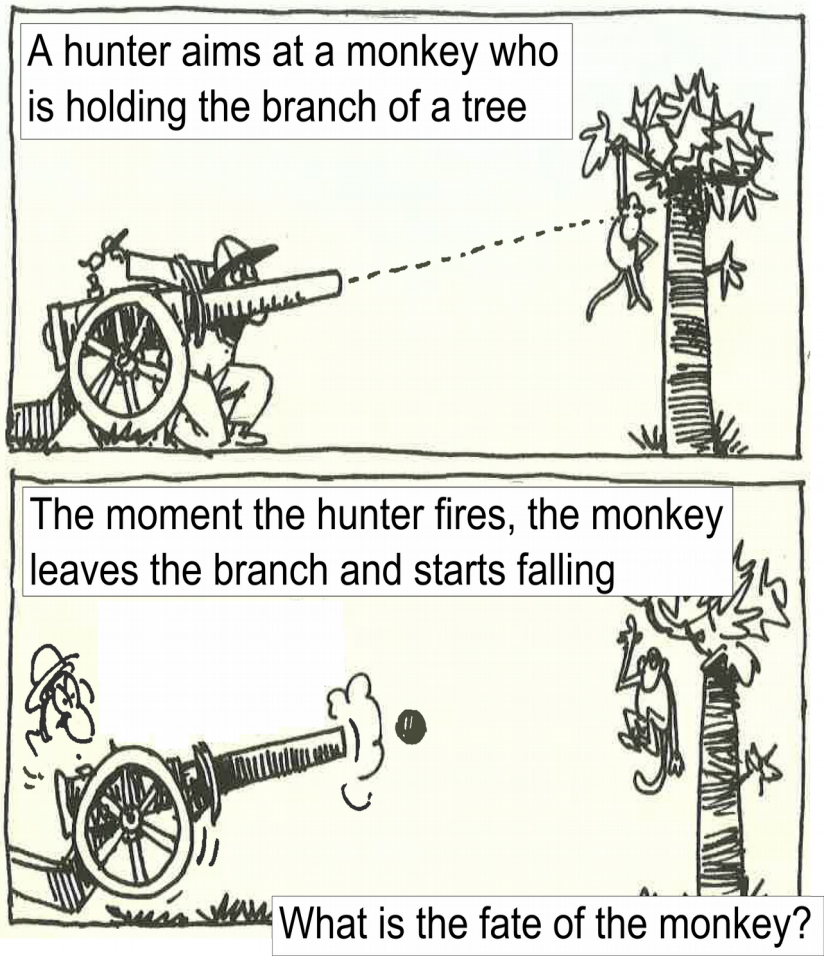
3a) Assume that the resistance from the air is negligible. Calculate the trajectory (x,y) of the bullet for different initial velocities (1000m/s , 500 m/s, 250 m/s) ($g = 10 m/sec^2$, dt=0.01sec, the gun is located 1.5m above the ground).

3b) (optional) Use a "for – loop" to iterate through the different initial velocities. (Hint: Look at "help for").

5 From "Gonick, L. & Huffman, The cartoon guide to Physics"

Problem 4 (Projectile motion 2)[6]

The hunter in the following figure aims at a monkey. When the hunter fires, the monkey starts falling. The monkey is 50m away from the hunter (horizontal distance). Assume that the resistance of the air is negligible. Will the hunter hit the monkey? Plot the trajectories of the bullet and the monkey.



Given:

- $g = 10 m/sec^2$,
- initial velocity of bullet = 60m/sec.
- Angle of the cannon = 10° from the horizontal
- the canon is located 1m above the ground. (optional: Check whether the monkey gets hit by an "if" condition).

(use a timestep of dt = 0.01sec in your code)

(Optional for the ones that are really fast with these exercises):

Find a monkey picture in google, load it into matlab and animate the falling monkey together with the bullet. Plot the time on the same plot (hint: use "title" and "num2str"), and create a *.jpg picture of every snapshot (hint: "print"). Put the whole thing together into a movie and load it on youtube.

6 Modified after "Gonick, L. & Huffman, The cartoon guide to Physics"