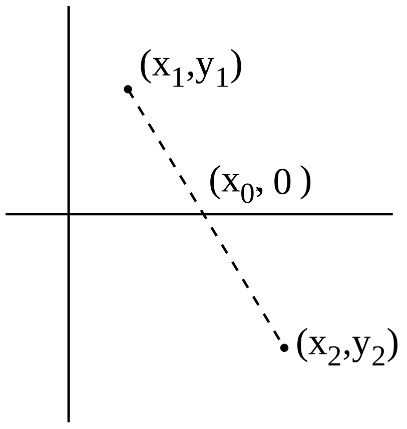
Homework II : Introductory `Octave` and motion in two dimensions

Guidelines for Homework II : Please read carefully!

1. Homework II is due Thursday, 25/10 by 20:00.
2. Homework will be submitted via email to `ustunel@metu.edu.tr`.
3. **VERY IMPORTANT!** The subject line of your email MUST read Phys343 Homework – nothing more, nothing less.
4. **VERY IMPORTANT!** Homework turned in on Thursday, 25/10 between 20:00 on 25/10 and 8:00 on 26/10 will only receive 50% of the full credit. Homework turned in later than that will NOT be accepted.
5. Please include the **a separate file** for each of the questions as attachments to your email. **Please send a single email**. There are four questions in this homework each asking you to write one `Octave` script or function, so I'm expecting four files from you. Before sending me your scripts or functions, make sure that they work.
6. Each question is worth 25 points.

Goal of the assignment : The goal of this assignment is two-fold : you will practice your skills at writing a simple function in `Octave` and you will expand the cyclist and projectile problems to include more effects due to air friction. This will allow you to practice not only writing code but also interpreting the results of a code.
Files to download : You can download the `cannon_nodrag.m` code that we wrote in lab or you can use your own version.

Question I : Write an `Octave` function called `interpolate`, which takes the x and y coordinates of two points in the xy-plane, interpolates them using a straight line and returns the x-intercept, which corresponds to $y = 0$. Schematically, you are asked to find x_0 on the following plot given the pairs (x_1, y_1) and (x_2, y_2).



Thus your function should

- take in as input four numbers (x_1, y_1, x_2, y_2)
- return a single number x_0

Question II : Download the cannon ball problem that we coded up in lab (`cannon_nodrag.m`) from the Web site. You can also use your own version if you still have it. As you will remember, we developed the program to be a script and not a function (Remember the difference?). In the code, we ran a `while` loop until the y coordinate of the trajectory became less than zero. Having run this script, you will see that the last element of the `y`-array `y(Nsteps)` is negative and the second to last `y(Nsteps-1)` is positive. The real range of the trajectory (which is the x-coordinate corresponding to $y = 0$) is then somewhere between the corresponding elements of the `x`-array, namely between elements `x(Nsteps-1)` and `x(Nsteps)`. Do the following

1. Using your `interpolate` function from Question 1, modify your script `cannon_nodrag.m` to approximate the x-coordinate of the point where the cannon ball hits the ground, in other words, calculate the real range of the trajectory.

2. Using the elements `vx(Nsteps-1)` and `vy(Nsteps-1)`, calculate the angle with which the cannon ball strikes the ground. In the absence of friction, you should find an angle similar to the launch angle.

3. Convert the script, `cannon_nodrag` to a function that takes in the initial velocity and the launch angle and returns the real (interpolated) range of the projectile and the angle with which the cannon ball strikes the ground. The units on the outputs should be km's and degrees respectively. The frame of your function should therefore look like this :

```
function [range,theta_final] = cannon_nodrag(v0,theta_init)
     ⋮
endfunction
```

   It should still display the graph.

Hint : For calculating the strike angle, look into the `Octave` functions `atan` and `atan2`.

Question III : Make a copy of the function that you have created in Question 2 and call it `cannon_drag.m`. In this problem, you are going to explore how air friction affects the range of the cannon ball. As we discussed in class, the drag force for large objects and high speeds is approximately

$$F_{drag} = -\frac{1}{2}\frac{CA\rho v^2}{m}$$

where C is the drag coefficient, A is the cross section area of the projectile, ρ is the density of air and m is the mass of the projectile. Follow these steps to modify `cannon_drag.m` to include air friction.

1. Add to the declaration section in the beginning the new constants `C`, `A`, `rho` and `m`. You can use the following values for the constants

`C`	0.03
`A`	0.02 m^2
`rho`	1.3 kg/m^3
`m`	10 kg

2. Add the friction term to the lines where the velocities are modified. Beware that the friction term is proportional to the square of the *norm* of the velocity and its direction is always opposite the *direction* of the velocity. You should thus find the x and y components of the drag force for each step in the loop.

3. Your function should still return the interpolated, real range and the angle with which the cannon ball strikes the ground.

Question 4 : Write a script `max_range.m` that will run the function from Question 4, i.e. `cannon_drag` for the array of launch angles `theta=25:1:55`, collect the output ranges (the interpolated x values) in an array and produce a plot of range vs angle. This plot is a good way to demonstrate how air friction affects the motion. There are a few noticeable differences :

1. How much shorter the range has become.

2. The nonparabolicity of the trajectories.

3. The new angle which produces the maximum range.