In this assignment you will create and use multiple python files that depend on each other: you'll define your own functions (Section 6.1) and modules (Section 6.4). You will make a simulation using Euler integration (Chapter 9), and you'll plot your results using matplotlib (Chapter 8).

AE1205 Assignment 4: Mars lander

In this assignment you will help design the final landing phase of a Martian lander.

The preliminary design is to skip the parachute phase to keep the design simple. Instead we arrive at Mars with a huge speed towards the surface, use aero-braking first and then we simply burn our retro rockets to brake just before the landing to land with a landing speed of less than 3 m/s (so we use a target speed of 2 m/s).

As many Mars landers have been lost in this final landing phase, it is quite a sensitive issue. Hence, the team needs all the help they can get. This is where your programming skills come in handy. You offer to help them by modelling this phase and determining the altitude where the Thruster should start burning ($\mathbf{h_T}$) and the initial fuel weight ($\mathbf{m_{fuel}}$).

Step 1: Calculate the Martian atmosphere

In this step you will make the first of two python files: marsatm.py. It will contain two functions; marsinit() and marsatm(). In your second python file you will import this first file (as you would import any other python module, see reader sec. 6.4). The data on the Martian atmosphere is given in the data file marsatm.txt on Brightspace. Make two functions in the file marsatm.py:

```python
def marsinit():
    # read and parse the marsatm.txt file
    ...
    # returns a variable called marstable to be used for interpolation
    return marstable
```

```python
def marsatm(h, marstable):
    # use linear interpolation of the marstable to get
    # pressure p, density rho, temperature and speed of sound c at a
    # given altitude h in meters (use R = 191.84 J/kg K)
    return p, rho, temp, c
```

Load the marsatm.txt file using open() (sec. 7.2). Have a look at the contents of this file to figure out how to load it into table data. It contains comment lines, so you'll have to take this into account.

Linear interpolation can be implemented as follows: For a given value of x, which in your interpolation table falls in between values x_1 and x_2, $x_1 < x < x_2$:

$$\begin{aligned} k &= \frac{x - x_1}{x_2 - x_1} \\ y(x) &= (1-k)\cdot y_1 + k \cdot y_2 \end{aligned}$$

You can test marsatm(h), e.g., by using altitudes at the levels specified in the table, or values exactly in between two table altitudes. In your main program Marslander.py you can import the two functions you've created, and add the gravity acceleration g_0 as a constant with the value $g_0 = 3.711 m/s^2$ to your program.

Extra: Using globals instead of returning

Note that you can also store the tabular data from the `marsatm.txt` file as a global variable in the namespace of your `marsatm.py` file. This way, the `marsinit()` function would not return anything, and the `marsatm()` function accesses the table data from the global variables. Because `marsinit()` and `marsatm()` are both defined in the same file (`marsatm.py`) they share the same global namespace.

Step 2: Simulate the landing phase

Assume that the lander uses aero braking to slow down to a speed 262 m/s, and take this speed as your initial condition. The attitude control is such that the lander always points its thruster in the direction of the speed. This results in a thrust force that is always in the same direction as the drag. Furthermore, assume a flat Mars surface, so that the height h is equal to the y-coordinate. Based on these assumptions, make a two dimensional simulation (x, y) using Backward Euler integration (see Chapter 9).

Use the following control law for the thrust (we ignore dynamic engine effects for now):

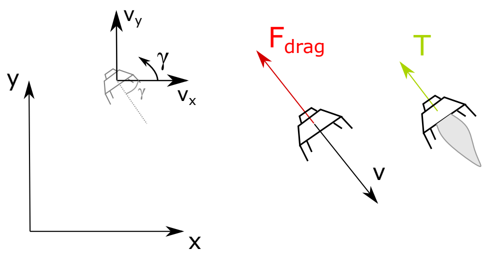
```
For h < hT:
$\dot{m} = \frac{m \cdot g_0}{v_e} + k_v \cdot \Delta v_y$    (but maximum $\dot{m}$ the rocket engines can give is 5 kg/s!)

$\Delta v_y = v_{y_{ref}} - v_y$    (as long as this number is positive)

$v_{y_{ref}} = -2.0$ m/s    (to make sure we stay with some margin under 3 m/s!)
Note: Below an altitude of 30 cm: do not use the thrusters anymore!
```

For the thruster, we use the specific impulse data, expressed as an effective exhaust speed v_e. We only know the zero fuel weight (m_{zfw}). The thrust is controlled by the mass flow, which during the burn reduces the total mass of the lander during this phase due to the fuel consumption. For your controller, use a gain of $k_v = 0.05$.

$$\begin{aligned} F_{drag} &= C_D \frac{1}{2} \rho V^2 S \\ T &= \dot{m} \cdot v_e \end{aligned}$$

Here, $C_D S = 4.92 m^2$, $v_e = 4400$ m/s, $m_{zfw} = 699.0$ kg.

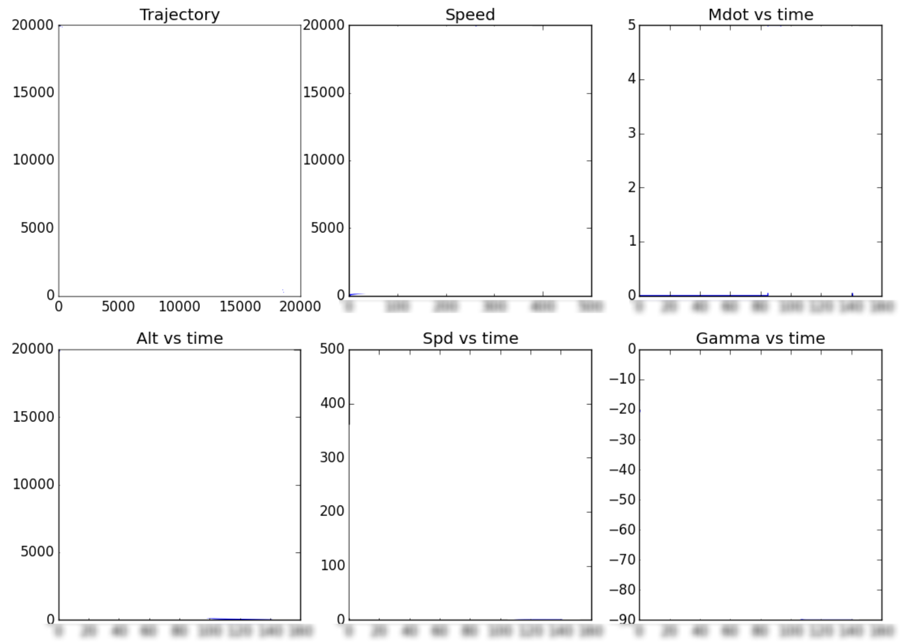
The initial condition for the landing phase is: $V = 262$ m/s, $h = 20$ km, and $\gamma = -20°$, where γ indicates the flight-path angle. The figure below shows the coordinate frames used (with arrows pointing in the positive directions) and the thrust and drag forces.



Simulate the landing phase including the drag and gravity forces, first without firing the thruster. Do this by using v_x and v_y speed components. Only for the drag force you need the updated absolute value of the speed, which you convert to x- and y-components with the flight path angle gamma for the drag force calculation.

Step 3: plot the results of your simulation

Make plots to show the different parameters (use `subplot(231)`, `subplot(232)` etc., see Chapter 8): y vs x to see trajectory, speed V vs y, mass flow, and then speed, altitude and flight path angle in degrees as a function of the time, so the figures should be drawn in a window looking like the window on the second page.



Step 4: Optimise thruster altitude and fuel weight

Now use your simulation to determine good values of the parameters below. First try your lander with the thruster off. Check the final impact speed and based on this decide a first estimate at which altitude h_T we need to start firing the thruster and how much fuel m_{fuel} (in kg) we need. You can only use the thruster when the fuel weight is still larger than zero. Constantly update the total mass as zero-fuel weight + fuel weight ($m = m_{zfw} + m_f(t)$). When you have found values for h_T and m_{fuel}, try to minimize the fuel mass so that it still results in a soft enough landing. You can also try different, more complex control laws for the thruster, taking into account the altitude, the acceleration etc. or even by using parachutes.

$$\begin{aligned} h_T &= \cdots ? \cdots m \\ m_{fuel} &= \cdots ? \cdots kg \end{aligned}$$