The following people have participated in creating these solutions: Nicolaas E. Groeneboom, Magnus Pedersen Lohne, Karl R. Leikanger
NOTE: There might be errors in the solution. If you find something which doens't look right, please let me know

Partial solutions to problems: Lecture 1-2

NOTE: The solution to numerical problems are given in Python code, but you are free to choose programming language for solving the problems.

The solutions to problems 1-6 are already given in the problem text or can be easily found on the web.

Problem 7

1. **NB: The numerical solutions given here are NOT the most efficient ways to solve the problem, try without looking at the solution first and you might find a much quicker and easier way to do it!.** Here you should obtain an eccentric elliptical orbit. Perihelion is about 298 km (you will probably not get exactly this number, it will depend on numerical precision) above the surface of Mars. Here is an outline of the code needed to solve the problem:

```
from scitools.all import *

#Function that calculates the gravitational force on the bodies
#for given position
def grav_force(x1,x2,y1,y2):
    r = ...                         #Distance between the objects
    F = ...                         #Grav. force (abs-value) on each body
    a = abs(x2 - x1)                #Displacement x-direction
    b = abs(y2 - y1)                #Displacement y-direction
    theta = arctan(float(b)/a)      #Angle betwen x-axis and r-direction

    #Give the components the right positive/negative-sign
    if x1<x2:
        Fx_1 = F*cos(theta)
        Fx_2 = -F*cos(theta)
    else:
        Fx_1 = ...
        Fx_2 = ...
    if y1<y2:
        Fy_1 = ...
        ...
```

```
    else:
        ...
        ...
    return Fx_1, Fx_2, Fy_1, Fy_2


#____
#MAIN

#Constants
m1 = ...                 #Mass of Mars
m2 = ...                 #Mass of the spaceship
dt = ...                 #Timestep
n = ...                  #Number of calculations
...

#Declare arrays of data (position, velocity ...)
x_1 = zeros(n, float)
y_1 = zeros(n, float)
...
...
vx_1 = zeros(n, float)
vy_1 = zeros(n, float)
...
...

#Initial values
x_1[0] = ...
y_1[0] = ...
...
...

teller = range(n-1)
#Calculations
for i in teller:
    #Calculate the gravitational force
    Fx_1, Fx_2, Fy_1, Fy_2 = grav_force(x_1[i], x_2[i], y_1[i], y_2[i])

    #Calculate the new velocity by Euler's method
    vx_1[i+1] = vx_1[i] + (Fx_1/m1)*dt
    vy_1[i+1] = ...
    ...

    #Calculate the new position by standard kinematics, use the
    #velocity from this timestep (v[i+1]).
    x_1[i+1] = ...
    y_1[i+1] = ...
    ...
```

```
#Plot-commands
...
```

2. The lander takes two full revolutions about Mars until it lands in the equatorial area. Here is an outline of the code needed to solve the problem:

```
from scitools.all import *


def grav_force(x1,x2,y1,y2):
    ...
    ...


#Function that calculates the frictionforce on the spaceship for given
#set of velocity-components
def fric_force(vx_2, vy_2):
    v = sqrt(...)                          #Velocity spaceship
    f = -k*v                               #Force of friction
    theta = arctan(...)                 #Angle between x-axis and v
    f_x = abs(...)                 #x-component
    f_y = abs(...)                 #y-component

    #Give the components the correct positive/negative-sign
    if vx_2 > 0:
        f_x = ...
    if vy_2 > 0:
        f_y = ...
    return f_x, f_y



#____
#MAIN

#Constants
...

#Declare LISTS and give initial values (convert to arrays later,
#you do not know how many elements you need)
x_1 = [0]
x_2 = [..]
...
...
vx_1 = ...
vx_2 = ...
...
...
```

```
i = 0             #While-variable
land = 'no'       #Variable which have value 'no' is eclipting
while land=='no' and i<(n-2):
    #Calculate the gravitational force-components
    ...

    #Calculate the frictionforce-components
    f_x, f_y = fric_force(vx_2[i], vy_2[i])

    #Calculate the new velocity by Euler's method (use APPEND-command)
    vx_1.append(vx_1[i] + (Fx_1/m1)*dt)
    vx_2.append(...)
    ...
    ...

    #Calculate the new position by standard kinematics
    ...
    ...

    #Check if the spaceship has landed
    if sqrt(...)<r:
        land = 'yes'

    i = i + 1

#Make arrays
x_1 = array(x_1)
...
...


#Plotcommands
...
...
```

3. The lander lands close to the area where it separated from Mars Express, in the equatorial area.

4. The orbit will take the complicated shape shown in figure 1. Here is an outline of the code needed to solve the problem:

```
from scitools.all import *


def grav_force(x1,x2,y1,y2,mass1,mass2):
    ...
    ...
```

```
#____
#MAIN

#Constants
...

#Declare arrays of data
...

#Initial values
...

#Calculations
for i in teller:
    #Grav. force planet - small star
    Fx_1_1, Fx_2_1, Fy_1_1, Fy_2_1 = grav_force(x_1[i], x_2[i], y_1[i], y_2[i], m1, m
    #Grav. force planet  - large star
    ...
    #Grav. force small star - large star
    ...

    #Sum up
    Fx_1 = Fx_1_1 + Fx_1_2
    Fx_2 = ...
    Fx_3 = ...
    Fy_1 = ...
    ...

    #Calculate the new velocity by Euler's method
    ...
    ...

    #Calculate the new position by standard kinematics
    ...
    ...

#Plot-commands
...
```

5. The distance from the two stars will vary a lot making huge changes in temperature on the planet. Some periods when the planet is far away from the two stars, the temperatures will be very cold, other periods the planet will be heated to very high temperatures by the two stars. It is unlikely that life can withstand such high changes in temperature.

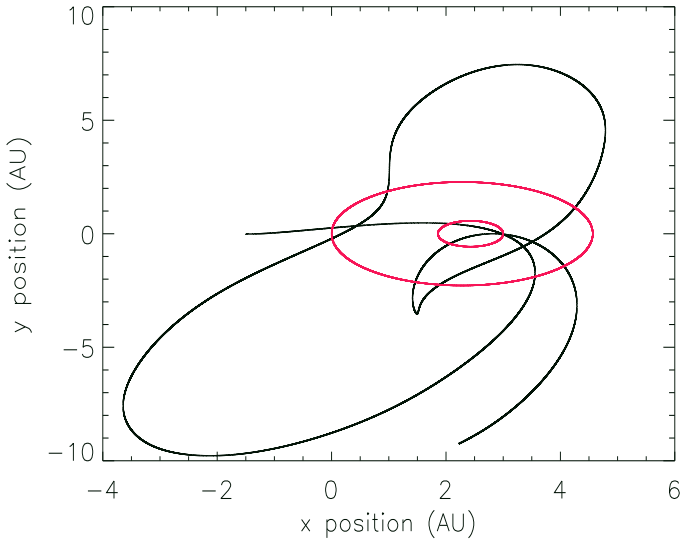

Figure 1: Trajectory of a planet between two stars. The two red ellipses are the orbits of the stars about the common center of mass.