VRIJE
UNIVERSITEIT
BRUSSEL

First assignment: root-finding methods

Basic techniques for computer simulations WPO | 11 March 2025

1 Instructions

In order to solve the assignment, you have to answer all questions included in the problem set.

Provide your answers as **one Python file** called `wpo1.py` that has to print all solutions to the questions and produce all plots requested in the problem set. The file has to be uploaded on Canvas before the deadline of the assignment.

Solutions to the questions have to be defined in the file as variables named `sol_x` where `x` is the number of the question. The solution file has to comply with the following format:

```

import numpy as np

def main():
    """ Your names """
    sol_1 = square(2)
    print(sol_1)
    sol_2a, sol_2b = logarithm(1000)
    print(sol_2a, sol_2b)

def square(x):
    """ Ex1 : Square of x """
    return x**2

def logarithm(x):
    """ Ex2 : Calculate log base ten and log base e """
    return np.log10(x), np.log(x)

if __name__=='__main__':
    main()
```

Listing 1: Example solution file format

2 Problem sets

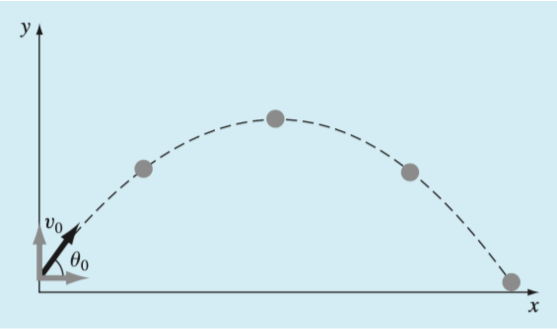

Figure 1: Ball trajectory.

In baseball, the trajectory of a ball thrown by a first player is defined by the (x,y) coordinates, as displayed in Fig. 1. This trajectory can be modeled as:

$$y = (\tan\theta_0)x - \frac{g}{2u_0^2\cos^2\theta_0}x^2 + y_0,$$

where g: downward acceleration of gravity (it can be assumed constant and equal to 9.81 m/s^2).
If the ball is thrown at an initial angle $\theta_0 = 30^o$ and initial speed $u_0 = 30$ m/s, find the distance that a second player should have from the first player, in order to catch the ball. The ball leaves the first player's hand at an elevation of 1.8 m and the second player receives it at an elevation of 1 m.
Find and print the solution by writing your own Python functions for:

1. the *bisection* method;
2. the *regula falsi* method. Print also the number of iterations that the specific method needed to converge to the solution;
3. the *Newton-Raphson* method;
4. the *secant* method.

Note 1: Use the same tolerance for all methods.
Note 2: Print distances in meters, with two digits after the decimal point.