VRIJE UNIVERSITEIT BRUSSEL

Fourth assignment: Interpolation and Extrapolation

Basic Techniques for Computer Simuations WPO | 1 April 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 `wpo4.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:

```
1
2 import numpy as np
3
4 def main():
5     """ Your names """
6     sol_1 = square(2)
7     print(sol_1)
8     sol_2a, sol_2b = logarithm(1000)
9     print(sol_2a, sol_2b)
10
11 def square(x):
12     """ Ex1 : Square of x """
13     return x**2
14
15 def logarithm(x):
16     """ Ex2 : Calculate log base ten and log base e """
17     return np.log10(x), np.log(x)
18
19 if __name__=='__main__':
20     main()
```

Listing 1: Example solution file format

2 Problem sets

The distance d required for the braking components to stop an automobile is a function of its speed u. The following experimental data were collected to quantify this relationship:

$u(km/h)$	30	45	60	75	90	120
$d(m)$	5.0	12.3	21.0	32.9	47.6	84.7

Estimate the braking distance d when the car is traveling at 80 km/h (interpolation) and when the car is traveling at 150 km/h (extrapolation) by implementing:

1. the simplest polynomial interpolation technique that consists of solving the linear system given by the n equations obtained by imposing the passage of the $(n-1)$th order polynomial, $P_{n-1}(x) = a_0 + a_1 x + \cdots + a_{n-1}x^{n-1}$, for each of the n points of the dataset. Print the polynomial equation and generate a plot showing the interpolating polynomial, along with the points of the dataset and the interpolated value;

2. the spline interpolation with `scipy.interpolate.interp1d`, using the three different methods: `nearest neighbour`, `linear`, and `cubic`. Then, generate a plot showing all three interpolating polynomials along with the points of the dataset and the interpolated value;

3. the *least-squares fit* with `numpy.polyfit`, using a first-order (linear) and a second-order polynomial. Then, produce a plot showing both polynomials along with the points of the dataset and the interpolated value;

Note 1: For every question that asks for a plot, use a separate figure (2 figures in total).
Note 2: Every plot should include a title, axes labels and a legend.