Exercise #3

04. September 2023

Exercises marked with a (J) should be handed together with a Jupyter notebook.

Problems marked with 4N or 4D must be submitted only by the respective course, while unmarked problems must be submitted by both courses.

Optional exercises will not be corrected.

Problem 1. (Trapezoidal rule)

In this exercise we will study the trapezoidal rule in more detail.

a) Use the trapezoidal rule to approximate the integral

$$I = \int_{\frac{\pi}{6}}^{\frac{\pi}{3}} \sin(x)\cos(2x)\,dx.$$

where I is the exact integral and I_h is the approximate one.

b) Compute the error in your approximation, i. e.

$$E = |I - I_h|.$$

c) Recall the error bound for the trapezoidal rule,

$$\left|I - I_h\right| \leq \frac{(b-a)^3}{12} \max_{\xi\in[a,b]} |f''(\xi)|,$$

where $f(x) \in C^{n+1}[a,b]$.

Use the expression above to find an upper bound for the error E, and verify that the actual error is less than the result you get. You can use python to find $\max_{\xi\in[a,b]} |f^{(n+1)}(\xi)|$.

Problem 2. (Gauß–Legendre quadrature)

The Gauß–Legendre quadrature with 4 nodes on the interval $[-1,1]$ is given by the nodes

$$x_0 = -\frac{1}{35}\sqrt{525+70\sqrt{30}}, \quad x_1 = -\frac{1}{35}\sqrt{525-70\sqrt{30}}, \quad x_2 = \frac{1}{35}\sqrt{525-70\sqrt{30}}, \quad x_2 = \frac{1}{35}\sqrt{525+70\sqrt{30}}$$

and the weights

$$w_0 = \frac{1}{36}(18-\sqrt{30}), \qquad w_1 = \frac{1}{36}(18+\sqrt{30}), \qquad w_2 = \frac{1}{36}(18+\sqrt{30}), \qquad w_3 = \frac{1}{36}(18-\sqrt{30}).$$

a) Transfer this quadrature rule to the interval $(-3,3)$ to obtain an approximation $G_h \approx \int_{-3}^{3} e^x \, dx$ (use python or else for the actual computations, but no need to submit a jupyter notebook).

b) For the Gauß–Legendre quadrature of a function f using n points on the interval $[a,b]$, the error is

$$E = \frac{(b-a)^{2n+1}(n!)^4}{(2n+1)[(2n)!]^3} f^{(2n)}(\xi) \tag{1}$$

for some $\xi \in (a,b)$, provided the function f is $2n$-times continuously differentiable. This estimate is given and you don't have to prove it. Based on the expression (1), what error estimates E_m would you expect for the composite Gauß–Legendre rule G_m? Give a brief explanation considering the interval (a,b) is divided in m subintervals each of length $h = \frac{b-a}{m}$.

c) Assume now that $f(x) = \frac{x^8}{8!}$. In what follows you don't have to compute the actual numbers but you can leave the number n as variable.

- use expression (1) to compute the error E_1 of the approximation $G_1 = \int_{-3}^{3} f(x)\, dx$ considering the whole interval $[a,b]$ and the Gauß–Legendre quadrature above
- Imagine to split the interval in the intervals $(-3,0)$ and $(0,3)$ and compute the error E_2 for the resulting composite Gauß–Legendre quadrature

- Compute E_2/E_1

Problem 3. (Composite Simpson's rule - **J**)

In this exercise we will arrive at a composite formula for the Simpson's rule and study its error. We recall that Simpson's rule applied to a single interval $[a, b]$ is

$$\int_a^b f(x)\mathrm{d}x = \frac{b-a}{6}\Big(f(a) + 4f\big(\frac{a+b}{2}\big) + f(b)\Big)$$

We now divide the interval $[a, b]$ into m segments with m an even number, each of length $h = 1/m$, and set $x_i = ih, i = 0, \dots, m$.

a) Show that the composite Simpson's rule defined on the divided interval takes the form

$$S_n = \frac{1}{3}h[f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + \ldots + 2f(x_{m-2}) + 4f(x_{m-1}) + f(x_m)].$$

Note: use only the points given!

b) Implement the composite Simpson's rule with python (see attached jupyter notebook).

c) Given that the error bound for the composite Simpson's rule is

$$E = \frac{h^4}{180}(b-a)\max_{\xi\in[a,b]}\left|f^{(4)}(\xi)\right|$$

Find an m such that $E \leqslant 10^{-3}$ for the integral

$$\int_0^1 \mathrm{e}^{-x}dx.$$

and verify computationally that the number m found actually satisfies the requirement (you can compute the exact solution and then compare it to the actual error on the jupyter notebook).

Problem 4. (Debug)

Consider the following Python code for a certain numerical method:

```
        import numpy as np

        def f(x):
            return x**2

        def Method(f, a, b, m):
            xs = np.linspace(a, b, m+1)
            ys = [f(x) for x in xs]
            s = ys[0] + ys[-1] + 2* sum(ys[1:-1])
            return s
```

If the method had been implemented correctly, running `Method(f,0,1,m)` should return an output equal to $1/3$ regardless of the input m. However, there is a mistake on one line of the code that prevents this. In fact, running `Method(f,0,1,2)`, `Method(f,0,1,4)` and `Method(f,0,1,10)`, for example, will return 1.5, 2.75 and 6.7, respectively.

a) Find the mistake and rewrite the incorrect line so as to have the correct implementation.

b) Once the mistake is fixed, what numerical method will be effectively implemented?