Introduction to computational thinking and programming for CFD (13251)

Dr. rer. nat. Marten Klein

Chair of Numerical Fluid and Gas Dynamics, BTU Cottbus-Senftenberg

Sheet 2

Goals

- Taylor series
- Functions, recursion
- Loops, branches
- Increment operator
- Algorithm for the sum

Tasks

1. Algorithm for the sum.

   (a) Develop and implement an algorithm that computes the following sum:

   $$\sum_{n=0}^{N} n \cdot d \qquad \text{for} \quad N = 100, \quad d = 2.0 \cdot 10^{-4}.$$

   (b) Compare the result with Gauss' product $0.5 \cdot d \cdot N \cdot (N+1)$.

   (c) Print out `repr(x)` for the result of case (a) and (b), respectively. Are there differences? Why or why not?

2. What is the definition of the Taylor series of a function $f(x)$ around a point x_0?

3. We consider the function $f(x) = \exp(-2x - 1)$ over the interval $x \in [-1, 4]$.

   (a) Expand the Taylor series $T_N(x; x_0)$ of $f(x)$ around $x_0 = -0.5$ up to of 4th order, that is, give $T_4(x; -0.5)$ explicitly.

   (b) Implement $T_4(x; -0.5)$ in a Python function.

   (c) Plot the `numpy`-based reference function $f(x)$ together with your approximation $T_4(x; -0.5)$. Where does the largest and where the smallest error occur?

Please turn the page!

4. Now consider the general case for arbitrary order N.
   (a) Determine analytically the Taylor series of $T_N(x; -0.5)$.
   (b) Implement a recursive function for the factorial $n! = n \cdot (n-1) \cdot \ldots \cdot 1$.
   (c) Implement $T_N(x; -0.5)$ in a Python function, passing N as the second parameter of the function.
   (d) Plot the reference function $f(x)$, the approximation $T_4(x; -0.5)$, and $T_N(x; -0.5)$ for various integer values of N. What do you observe for increasing N?
5. ($*$) Determine numerically the order N for which the error between $f(x)$ and $T_N(x; -0.5)$ at $x = 4$ is less than 10^{-8}.

Hints and remarks

- Standard libraries

```
import numpy as np
import matplotlib.pyplot as pl
```

- Define a function

```
def myfunc(x):
    val = 3*(x-1.)**(1./3.)
return val
```

- Call a function

```
x = myfunc(1.0)
print( x )
```

- Save / show a plot

```
# generate data
x = np.linspace(1., 2., 4)
y = myfunc(x)

# plot
pl.plot(x, y)
pl.savefig('myfig.png') # save figure
pl.show()               # display figure
```