Computer Methods in Engineering
Exercise 2

The approximation of derivatives is done by evaluating the function in a few points around the point where the derivative should be evaluated. This procedure is collected into an operator denoted as the numerical differential operator. In practice, when a derivative is to be found, the analytical differential operator is replaced by the numerical equivalent. An important fact about the numerical operator is that it is only an *approximation* of the analytical operator; hence when using numerical operators, errors are introduced in the results produced. In applied sciences it is important to know as much as possible about the error, such that it is possible to know how far the numerical derivative is from the true derivative, and how large the error becomes when doing changes to the setup of the problem.

When working with numerical differentiation it is convenient to introduce the operator notation. Using the operator notation, the first order derivative of a smooth function $f(x)$ may be approximated with the following four different operators

$$D_+ f(x) = \frac{f(x+\Delta x) - f(x)}{\Delta x}, \tag{1}$$

$$D_- f(x) = \frac{f(x) - f(x-\Delta x)}{\Delta x}, \tag{2}$$

$$D_0 f(x) = \frac{f(x+\Delta x) - f(x-\Delta x)}{2\Delta x}, \tag{3}$$

$$D_3 f(x) = \frac{2f(x+\Delta x) + 3f(x) - 6f(x-\Delta x) + f(x-2\Delta x)}{6\Delta x}, \tag{4}$$

where Δx is a small distance from the point x. Similarly, the second order derivative of $f(x)$ may be given as

$$D^2 f(x) = \frac{f(x-\Delta x) - 2f(x) + f(x+\Delta x)}{\Delta x^2}. \tag{5}$$

This formula is known as the central difference for the second order derivative. The above operators are just examples of how first and second order derivatives may be approximated by numerical differential operators.

Problem 1

a) An important property of the above operators are that they are linear operators, i.e. that

$$D_i(af(x) + bg(x)) = aD_i f(x) + bD_i g(x), \quad i = +, -, 0, 3,$$

for two functions $f(x)$ and $g(x)$ and two constants a and b. Thus it is possible to play around with the operators to construct new operators. This will become clear in the following problems.

Prove that D_+ and D_0 are linear operators.

b) Show that

$$D_0 f(x) = \frac{1}{2}(D_+ f(x) + D_- f(x)).$$

c) Prove, by a Taylor expansion of the involved terms, that the forward difference in (1) is an approximation for the first order derivative. Find the truncation error (the remaining part of the Taylor expansion). What happens with the truncation error when $\Delta x \to 0$?

d) Prove that the central difference in (3) is an approximation of the first derivative. Find the truncation error (the remaining part of the Taylor expansion).

e) Based on the results in c) and d), which of D_+ and D_0 is the best approximation for the first order derivative? Why?

f) Find the truncation error for $D_3 f(x)$.

g) Show that

$$D^2 f(x) = D_+ D_- f(x).$$

h) Find the truncation error for $D^2 f(x)$.

Solution:

a) *Using the definition of a linear operator given in the text results in*

$$\begin{aligned} D_+(af(x) + bg(x)) &= \frac{af(x+\Delta x) - af(x) + bg(x+\Delta x) - bg(x)}{\Delta x} \\ &= a\,\frac{f(x+\Delta x) - f(x)}{\Delta x} + b\,\frac{g(x+\Delta x) - g(x)}{\Delta x} \\ &= aD_+ f(x) + bD_+ g(x), \end{aligned}$$

and

$$\begin{aligned} D_0(af(x) + bg(x)) &= \frac{af(x+\Delta x) - af(x-\Delta x) + bg(x+\Delta x) - bg(x-\Delta x)}{2\Delta x} \\ &= a\,\frac{f(x+\Delta x) - f(x-\Delta x)}{2\Delta x} + b\,\frac{g(x+\Delta x) - g(x-\Delta x)}{2\Delta x} \\ &= aD_0 f(x) + bD_0 g(x). \end{aligned}$$

b) *Using the definition given in* (1) *and* (2) *yields*

$$\frac{1}{2}(D_+ f(x) + D_- f(x)) = \frac{1}{2}\left(\frac{f(x+\Delta x) - f(x) - f(x) + f(x-\Delta x)}{\Delta x}\right) = D_0 f(x).$$

c) *Using the well-known Taylor series expansion of* (1) *yields*

$$\begin{aligned} D_+ f(x) &= \frac{f(x+\Delta x) - f(x)}{\Delta x} \\ &= \frac{1}{\Delta x}(f(x) + \Delta x f'(x) + \frac{\Delta x^2}{2} f''(x) + \cdots - f(x)) \\ &= f'(x) + \frac{\Delta x}{2} f''(x) + \cdots, \end{aligned}$$

from which we observe that D_+ is an approximation of the first order derivative, with truncation error

$$\tau_{D_+}(x) = \frac{\Delta x}{2} f''(x) = \mathcal{O}(\Delta x).$$

Since the truncation error goes with the power of one *in Δx we say that D_+ is a* first order approximation. *When $\Delta x \to 0$ we see that $\tau_{D_+} \to 0$ which means that the numerical approximation approaches the true differential operator.*
***d)** Following the same procedure as in c) gives*

$$\begin{aligned} D_0 f(x) &= \frac{f(x+\Delta x) - f(x-\Delta x)}{2\Delta x} \\ &= \frac{1}{2\Delta x}\Big(f(x) + \Delta x f'(x) + \frac{\Delta x^2}{2} f''(x) + \frac{\Delta x^3}{6} f^{(3)}(x) + \cdots \\ &\quad - f(x) + \Delta x f'(x) - \frac{\Delta x^2}{2} f''(x) + \frac{\Delta x^3}{6} f^{(3)}(x) + \cdots \Big) \\ &= f'(x) + \frac{\Delta x^2}{6} f^{(3)}(x) + \cdots, \end{aligned}$$

and hence D_0 is an approximation for the first order derivative with truncation error

$$\tau_{D_0}(x) = \frac{\Delta x^2}{6} f^{(3)}(x) = \mathcal{O}(\Delta x^2).$$

Therefore D_0 is a second order approximation.
***e)** From the above truncation error calculations we conclude that since D_0 is a second order approximation, this is a better approximation to $f'(x)$ compared to D_+.*
***f)** Following the same procedure as in c) and d) gives the truncation error*

$$\tau_{D_3}(x) = \frac{\Delta x^3}{12} f^{(4)}(x) = \mathcal{O}(\Delta x^3),$$

and hence is D_3 an example of a third order approximation to the first order derivative.
***g)** Using the definition of D_+ and D_- yields*

$$\begin{aligned} D_+ D_- f(x) &= \frac{1}{\Delta x}\left(D_- f(x+\Delta x) - D_- f(x)\right) \\ &= \frac{1}{\Delta x}\left(\frac{f(x+\Delta x) - f(x)}{\Delta x} - \frac{f(x) - f(x-\Delta x)}{\Delta x}\right) = D^2 f(x). \end{aligned}$$

***h)** Following the same procedure as in c), d), and e) gives the truncation error*

$$\tau_{D^2}(x) = \frac{\Delta x^2}{12} f^{(4)}(x) = \mathcal{O}(\Delta x^2),$$

and hence D^2 is a second order approximation.

Problem 2

In this problem you will use python to numerically prove the order of the truncation errors for D_+ and D_0, given in (1) and (3), respectively, that you found analytically in Problem 1. To do this, use the function

$$f(x) = \sin(x),$$

evaluated in the point $x = 1$. The truncation error between the numerical differential operator D is given as

$$e(x) = |Df(x) - f'(x)|,$$

where $f'(x)$ in this case is trivial to find. Using the results gives the error

$$e(x) = |Df(x) - f'(x)| = C\Delta x^p,$$

for some integer p and constant C. The integer p is the order of the numerical operator, and is on the same form as you found in Problem 1. It is the integer p you want to find numerically in this problem. Taking the logarithm on both sides of the above equation gives

$$\log(e(x)) = \log|C| + p\log(\Delta x).$$

From this you may observe that by using a log-log scale for the errors and grid sampling Δx, the error behaves linearly with a slope that is equal to p. Thus, it is possible to find p using standard regression tools known from statistics, if the values are considered on the log-log scales.

a) Write a function `dplus(f,dx)` that takes $f(x)$ and Δx as input, and returns $D_+f(x)$. Test the operator on a function which you know the derivative of.

b) Write a function `d0(f,dx)` that takes $f(x)$ and Δx as input, and returns $D_0f(x)$. Test the operator on a function which you know the derivative of.

c) Write a program that evaluates $e(x)$ at $x = 1$ for different values of Δx. Use $\Delta x = 0.1, 0.05, 0.01, 0.005, 0.001$. Make a log-log plot of the $e(x)$ for the different values of $\Delta(x)$. *Hint:* Use the command `loglog` for the plotting.

d) Find the integer p for D_+ and D_0 using the numpy regression command `polyfit`. How does your numerical value for p fit the theoretical value you found in Problem 1?

Solution:

a)

```
'''
 Forward derivative

'''

import numpy as np

def dplus(x,dx):
  n = x.shape[0]
  d = np.zeros(n)
  for i in range(0,n-2):
    d[i] = (x[i+1] - x[i])/(dx)
  d[n-1] = d[n-2]
  return(d)
```

b)

```
'''
  Central derivative
```

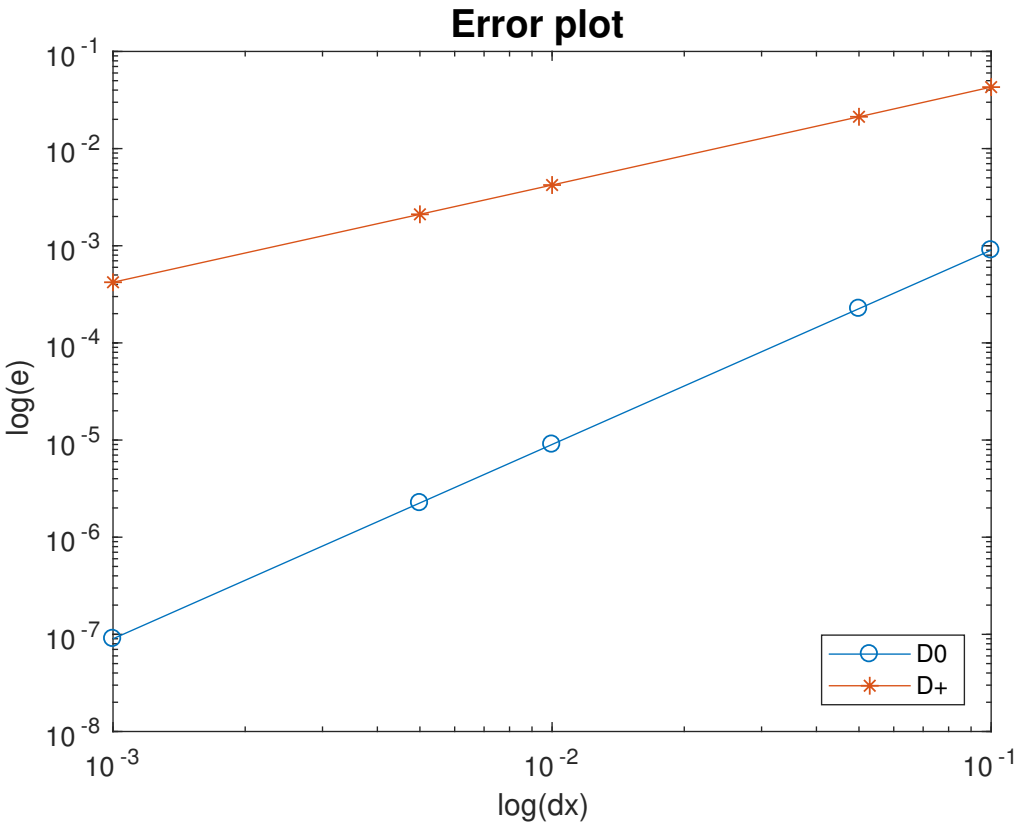
```
'''

import numpy as np

def d0(x,dx) :
  n = x.shape[0]
  d = np.zeros(n)
  for i in range (1,n-2):
    d[i] = (x[i+1] - x[i-1])/(2*dx)
  d[0] = (x[2]-x[1])/dx
  d[n-1] = d[n-2]

  return(d)
```

c) *Using the code below produces the plot for* $e(x)$ *at* $x = 1$.

```
'''
  Running script for exercise 5, TPG4155

'''
from math import *
import d0
import dplus
import numpy as np
import matplotlib.pyplot as pl

# PROBLEM 2
dx=np.array([0.1,0.05,0.01,0.005,0.001])  # The grid sampling

xstart=0   # Interval start point
xend=2     # Interval end point

# Creating error vectors
error0 = np.zeros(dx.shape[0])
errorp = np.zeros(dx.shape[0])

# Creating x-axes for different dx
x1 = np.arange(xstart,xend,dx[0])
x2 = np.arange(xstart,xend,dx[1])
x3 = np.arange(xstart,xend,dx[2])
x4 = np.arange(xstart,xend,dx[3])
x5 = np.arange(xstart,xend,dx[4])

# Finding the derivatives using the two operators
D0f1 = d0.d0(np.exp(x1),dx[0])
D0f2 = d0.d0(np.exp(x2),dx[1])
D0f3 = d0.d0(np.exp(x3),dx[2])
D0f4 = d0.d0(np.exp(x4),dx[3])
D0f5 = d0.d0(np.exp(x5),dx[4])

Dpf1 = dplus.dplus(np.exp(x1),dx[0])
Dpf2 = dplus.dplus(np.exp(x2),dx[1])
Dpf3 = dplus.dplus(np.exp(x3),dx[2])
Dpf4 = dplus.dplus(np.exp(x4),dx[3])
Dpf5 = dplus.dplus(np.exp(x5),dx[4])

# Finding errors for different dx
```

```
error0[0] = abs(exp(1)-D0f1[10])
error0[1] = abs(exp(1)-D0f2[20])
error0[2] = abs(exp(1)-D0f3[100])
error0[3] = abs(exp(1)-D0f4[200])
error0[4] = abs(exp(1)-D0f5[1000])

errorp[0] = abs(exp(1)-Dpf1[10])
errorp[1] = abs(exp(1)-Dpf2[20])
errorp[2] = abs(exp(1)-Dpf3[100])
errorp[3] = abs(exp(1)-Dpf4[200])
errorp[4] = abs(exp(1)-Dpf5[1000])

print("error 0:", error0)
print("error p:", errorp)

# Plotting
pl.loglog(dx,error0,'-o', dx,errorp,'-*');
pl.title('Error plot');
pl.xlabel('log(dx)');
pl.ylabel('log(e)');
#pl.legend('D0','D+','Location','SouthEast');
pl.legend(['D0','D+']);

# Finding the integers n
coeff1=np.polyfit(np.log(dx),np.log(error0),1)
print("p for d0: ", coeff1[0])
coeff2=np.polyfit(np.log(dx),np.log(errorp),1)
print("p for dplus: ", coeff2[0])

pl.show()
```

The log-log plot is given below.



We observe that D_0 is much more accurate than D_+, which is what we expect from the results in Problem 1.

d) *The regression give the following numerical truncation errors*

$$p_{D_+} = 1.0068,$$
$$p_{D_0} = 2.0001,$$

which fits very well with the analytical results ($p_{D_+} = 1, p_{D_0} = 2$).