AERO-222: Introduction to Aerospace Computation
Fall 2021. Homework #4. **Due date: November 15, 2021**

Show all work and justify your answers!

Instructions

- *This homework contains both handwritten and coding problems and shall be submitted according to the following guidelines.*
- *Hardcopy:*
  - *Due on CANVAS at 11:59 PM on the day of the deadline.*
  - *Shall include screenshots of any hand-written work.*
  - *For coding problems, the hardcopy shall include any relevant derivations and emphasize the final results (i.e. boxed, highlighted, etc.). INCLUDE ALL CODING RESULTS (including plots, final values) IN THE HARDCOPY.*
  - *Shall be submitted as a single file according to the provided template with the following naming scheme:* "LastnameHW#.pdf"
- *Coding Submission:*
  - *Due on CANVAS at 11:59 PM on the day of the deadline.*
  - *Shall be submitted as a single file according to the provided template with the following naming scheme:* "LastnameHW#.py"
  - *The script shall print out all outputs asked for in the problem.*
- *Late submissions will be accepted with a 10 point deduction per day late.*

1. **Numerical Differentiation (20 pts) Code.** The derivative of the function,

$$f(x) = 5\,\cos(10\,x) + x^3 - 2\,x^2 - 6\,x + 10$$

is the function, $f'(x) = -50\,\sin(10\,x) + 3\,x^2 - 4\,x - 6$. Discretize the function $f(x)$ using $N = 100$ points, $[x_k, f_k]$, that are uniformly distributed in $x \in [0, 4]$. Evaluate the first numerical derivative using the 5-point difference formula. Plot the absolute error between true and numerical derivatives. Pay attention to the two extremes: you cannot always use the 5-point **central** difference formula.

Solution:
The plots should look very similar to this.

2. **Richardson Extrapolation (20 pts) By hand.** Using the function given in problem #1, compute the accuracy gain using Richardson extrapolation to compute the first derivative using the central seven points formula at $x = 2$, using $h_1 = 0.02$ and $h_2 = 0.03$. Compute also the absolute error with respect to the true solution.

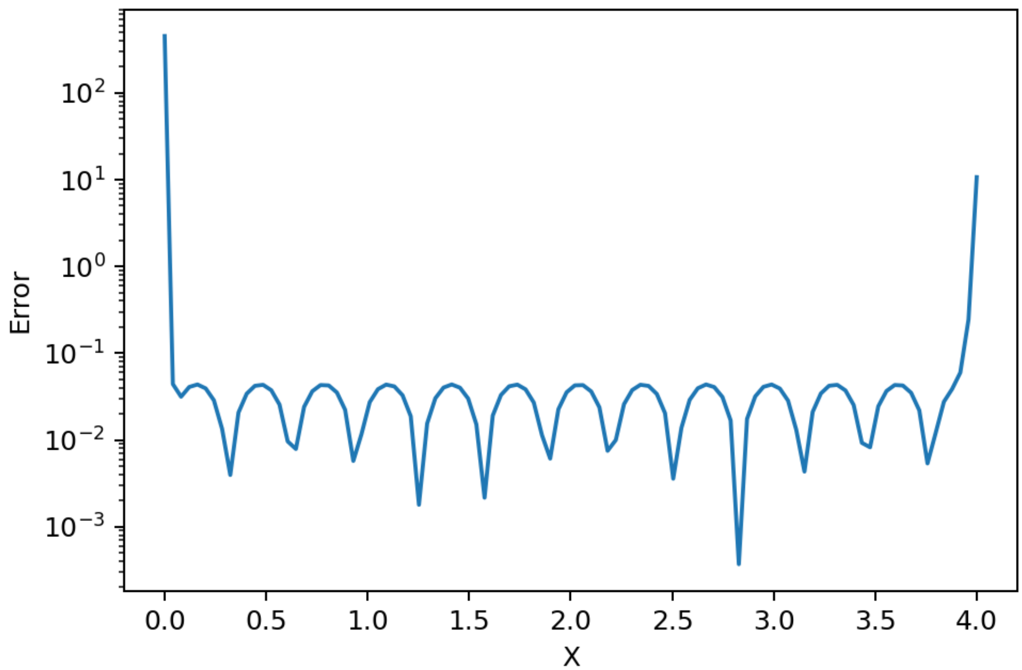

Figure 1: Caption

Solution:

The central seven point first derivative has the expression,

$$f_i' \approx \frac{-f_{i-3} + 9f_{i-2} - 45f_{i-1} + 45f_{i+1} - 9f_{i+2} + f_{i+3}}{60\,h} + \mathcal{O}\left(h^6\right)$$

The points are in the table,

h	f_{i-3}	f_{i-2}	f_{i-1}	f_{i+1}	f_{i+2}	f_{i+3}
$h_1 = 0.02$	2.3956	1.7433	0.9482	-0.9455	-1.9718	-2.9988
$h_2 = 0.03$	3.0557	2.3956	1.3618	-1.4561	-2.9988	-4.4542

Using the central seven point first derivative for h_1 and h_2 we obtain,

$$f'(h_1) \approx -47.6472 \qquad \text{and} \qquad f'(h_2) \approx -47.6470$$

Richardson extrapolation gives

$$f'_{\text{true}} \approx f'_{\text{RE}} = f'(h_1) + c\,h_1^6 = f'(h_2) + c\,h_2^6 \qquad \rightarrow \qquad c = \frac{f'(h_1) - f'(h_2)}{h_2^6 - h_1^6} \approx -3.2009 \cdot 10^5$$

and the absolute errors are

$$|f'_{\text{true}} - f'(h_1)| \approx 2.0705 \cdot 10^{-5}, \quad |f'_{\text{true}} - f'(h_2)| \approx 2.3357 \cdot 10^{-4}, \quad |f'_{\text{true}} - f'_{\text{RE}}| \approx 2.1966 \cdot 10^{-7}$$

3. **Numerical Differentiation (20 pts) By hand** Show all steps to derive the second derivative, f_i'', using using all of the following points

$$f_{i-4}, \qquad f_{i-3}, \qquad f_{i-2}, \qquad f_{i-1}, \qquad f_{i+1},$$

via matrix inversion.

Solution:

$$f_{i-4} = f_i - 4hf_i' + 16\frac{h^2 f_i''}{2!} - 64\frac{h^3 f_i'''}{3!} + 256\frac{h^4 f_i^{(4)}}{4!} - 1024\frac{h^5 f_i^{(5)}}{5!} + \mathcal{O}(h^6)$$
$$f_{i-3} = f_i - 3hf_i' + 9\frac{h^2 f_i''}{2!} - 27\frac{h^3 f_i'''}{3!} + 81\frac{h^4 f_i^{(4)}}{4!} - 243\frac{h^5 f_i^{(5)}}{5!} + \mathcal{O}(h^6)$$
$$f_{i-2} = f_i - 2hf_i' + 4\frac{h^2 f_i''}{2!} - 8\frac{h^3 f_i'''}{3!} + 16\frac{h^4 f_i^{(4)}}{4!} - 32\frac{h^5 f_i^{(5)}}{5!} + \mathcal{O}(h^6)$$
$$f_{i-1} = f_i - hf_i' + \frac{h^2 f_i''}{2!} - \frac{h^3 f_i'''}{3!} + \frac{h^4 f_i^{(4)}}{4!} - \frac{h^5 f_i^{(5)}}{5!} + \mathcal{O}(h^6)$$
$$f_{i+1} = f_i + hf_i' + \frac{h^2 f_i''}{2!} + \frac{h^3 f_i'''}{3!} + \frac{h^4 f_i^{(4)}}{4!} + \frac{h^5 f_i^{(5)}}{5!} + \mathcal{O}(h^6)$$

$$\begin{bmatrix} -4 & -3 & -2 & -1 & 1 \\ 16 & 9 & 4 & 1 & 1 \\ -64 & -27 & -8 & -1 & 1 \\ 256 & 81 & 16 & 1 & 1 \\ -1024 & -243 & -32 & -1 & 1 \end{bmatrix} \begin{Bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{Bmatrix}$$

whose exact rational solutions are,

$$c_1 = \frac{1}{24}, \qquad c_2 = -\frac{1}{4}, \qquad c_3 = \frac{7}{12}, \qquad c_4 = -\frac{1}{6}, \qquad c_5 = \frac{5}{12}$$

or, approximated,

$$c_1 \approx 0.0417, \qquad c_2 = -0.25, \qquad c_3 \approx 0.5833, \qquad c_4 \approx -0.1667, \qquad c_5 \approx 0.4167$$

4. **Numerical Differentiation (20 pts) Code.** Consider the function $f(x) = x + \sin x$ and a step size $h = \pi/8$.

   (a) Compare the three-point forward, backward, and central finite-difference approximations of the second derivative of $f(x)$ with respect to the exact solution at $x = 1$. Provide a table of values.

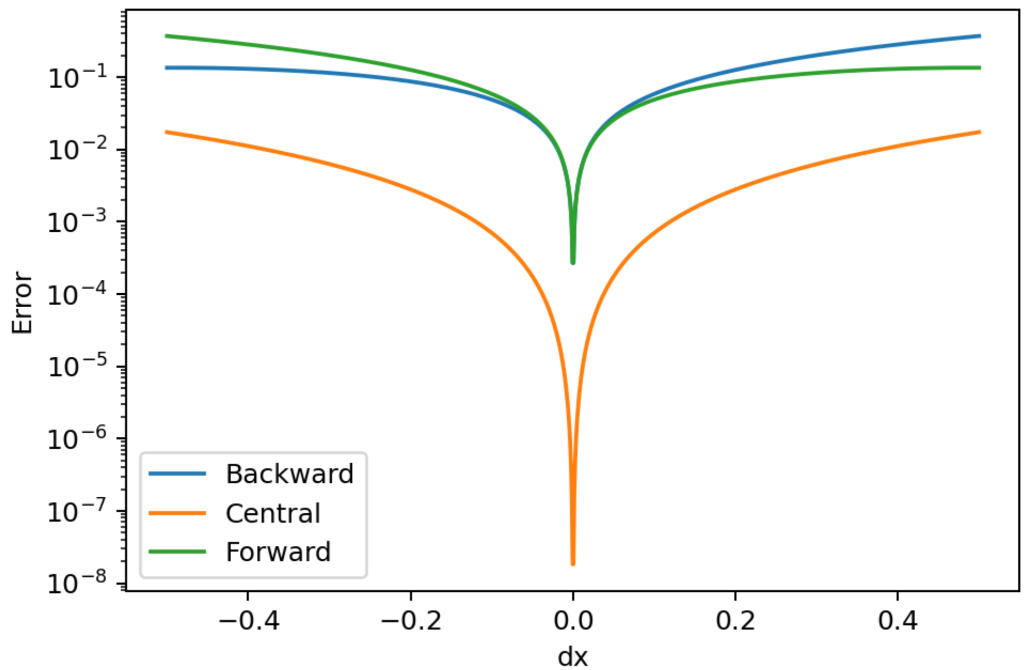

Figure 2: Central Finite Differences should have the best accuracy, as should smaller magnitudes of dx

(b) Compute the absolute error of the 3-point forward, backward, and central approximations for a step size range $\Delta x \in [-0.5, +0.5]$ broken into $1,000$ values. Provide a single semi-logarithmic plot of the absolute errors versus the given x-range. Include proper axes labels and a reference legend.

Solution:

Backward	Central	Forward
0.278	0.0108	0.130

5. An Aerospace Application (20 pts) Code. An airfoil is placed in a wind tunnel and heated to 80°C before lowering the surface temperature to 20.7°C using convective cooling. A thermocouple is placed on the surface of the airfoil and measures the surface temperature at discrete time steps. The following temperatures are recorded,

t (sec)	0	5	10	15	20	25
T (°C)	80	44.5	30	24.1	21.7	20.7

Use numerical differentiation to compute the first derivative $T'(t_k)$ at each time step given in the table above. Specifically, apply the three-point forward, backward or central finite-difference method wherever appropriate in order to estimate the airfoil's temperature gradient, $T'(t_k)$. Provide a table of values.

Solution:

t (sec)	0	5	10	15	20	25
T (°C)	80	44.5	30	24.1	21.7	20.7
T' (°C)	-9.2	-5.0	-2.04	-0.83	-0.34	-0.06

Reference Python Code

```python
import numpy as np
import matplotlib.pyplot as plt
from tabulate import tabulate

### Problem 1
def f1(x): return 5*np.cos(10*x)+x**3-2*x**2-6*x+10
def df1(x): return -50*np.sin(10*x) + 3*x**2 - 4*x - 6

def forw1(x,f,i,h=np.pi/8): return (1*f[i+0]-2*f[i+1]+1*f[i+2])/(1*1.0*
def forw2(x,f,i,h=np.pi/8): return (-3*f[i-1]-10*f[i+0]+18*f[i+1]-6*f[i
def cent1(x,f,i,h=np.pi/8): return (1*f[i-2]-8*f[i-1]+0*f[i+0]+8*f[i+1]
def back1(x,f,i,h=np.pi/8): return (-1*f[i-3]+6*f[i-2]-18*f[i-1]+10*f[i
def back2(x,f,i,h=np.pi/8): return (3*f[i-4]-16*f[i-3]+36*f[i-2]-48*f[i

xList1 = np.linspace(0,4,100)
fx =  [f1(val) for val in xList1]
dfx = [df1(val) for val in xList1]
df1 = []
xList = xList1
for i in range(len(xList)):
        if i < 1:
                val = forw1(xList[i],fx,i,h=xList[1]-xList[0])
        elif i < 2:
                val = forw2(xList[i],fx,i,h=xList[1]-xList[0])
        elif i < len(xList)-3:
                val = cent1(xList[i],fx,i,h=xList[1]-xList[0])
        elif i < len(xList)-2:
                val = back1(xList[i],fx,i,h=xList[1]-xList[0])
        elif i < len(xList)-1:
                val = back2(xList[i],fx,i,h=xList[1]-xList[0])
        df1.append(val)
error1 = [abs(df1[i]-dfx[i]) for i in range(len(df1))]
plt.semilogy(xList,error1)
plt.xlabel('X')
plt.ylabel('Error')
plt.show()
plt.close()

### Problem 4
def f(x):   return x+np.sin(x)
def df(x):  return 1+np.cos(x)
def ddf(x): return -np.sin(x)
```

```
xList = np.linspace(0,4,100)
hList = np.linspace(-.5,.5,1000)
ddfx = [ddf(val) for val in xList]

def back4(x,f,h=np.pi/8): return (f(x-2*h)-2*f(x-h)+f(x))/h**2
def forw4(x,f,h=np.pi/8): return (f(x-0*h)-2*f(x+1*h)+f(x+2*h))/h**2
def cent4(x,f,h=np.pi/8): return (f(x-1*h)-2*f(x-0*h)+f(x+1*h))/h**2

ddf1 = ddf(1.0)
b4 = back4(1.0,f)
f4 = forw4(1.0,f)
c4 = cent4(1.0,f)
table = [["Backwards","Central","Forwards"],
                 [abs(b4-ddf1),abs(c4-ddf1),abs(f4-ddf1)]]
print(tabulate(table))
backddf = [back4(1.0,f,h=i) for i in hList]
centddf = [cent4(1.0,f,h=i) for i in hList]
forwddf = [forw4(1.0,f,h=i) for i in hList]
errorB = abs(backddf-ddf1)
errorC = abs(centddf-ddf1)
errorF = abs(forwddf-ddf1)
plt.semilogy(hList,errorB,label='Backward')
plt.semilogy(hList,errorC,label='Central')
plt.semilogy(hList,errorF,label='Forward')
plt.xlabel('dx')
plt.ylabel('Error')
plt.legend()
plt.show()
plt.close()

### Problem 5
T = [80,44.5, 30, 24.1, 21.7, 20.7]
t = [0,5,10,15,20,25]
h = 5
der = []
for i in range(len(T)):
        if i <1:
                f_x = (-3*T[i+0]+4*T[i+1]-1*T[i+2])/(2*1.0*h**1)
        if 0 < i and i < 5:
                f_x = (-1*T[i-1]+0*T[i+0]+1*T[i+1])/(2*1.0*h**1)
        if i > 4:
                f_x = (1*T[i-2]-4*T[i-1]+3*T[i+0])/(2*1.0*h**1)
        der.append(f_x)
table = [["t",t],["T",T],["T'",der]]
print(tabulate(table))
```