Assignment 1.

Jithin D. George, No. 1622555

Due Friday, Jan. 12.

Reading: Ch 1. Secs. 2.1-2.9.

1. Use MATLAB to evaluate the second order accurate approximation

$$u''(x) \approx \frac{u(x+h) + u(x-h) - 2u(x)}{h^2}$$

for $u(x) = \sin x$ and $x = \pi/6$. Try $h = 10^{-1}, 10^{-2}, \ldots, 10^{-16}$, and make a table of values of h, the computed finite difference quotient, and the error. Explain your results.

Solution:

h	Computed Value	Error
0.1	-0.499583	0.000416528
0.01	-0.499996	4.16665e-06
0.001	-0.5	4.15635e-08
0.0001	-0.5	3.03874e-09
1e-05	-0.500001	-1.15159e-06
1e-06	-0.499933	6.6572e-05
1e-07	-0.4996	0.000399639
1e-08	-1.11022	-0.610223
1e-09	111.022	111.522
1e-10	0	0.5
1e-11	0	0.5
1e-12	1.11022e+08	1.11022e+08
1e-13	1.11022e+10	1.11022e+10
1e-14	-1.11022e+12	-1.11022e+12
1e-15	0	0.5
1e-16	-1.11022e+16	-1.11022e+16

```
import numpy as np
from tabulate import tabulate

def central_diff(f,x,h):
    return (f(x+h)+f(x-h)-2*f(x))/h**2
j=[]
hlist =[]
for i in range(1,17):
    hlist.append(10**-i)
for h in hlist:
```

```
    j.append([h,central_diff(np.sin,np.pi/6,h),central_diff(np.sin,np.pi/6

print(tabulate(j,headers=['h', 'Computed Value', 'Error'],tablefmt="latex"
```

Truncation error($\frac{\sqrt{3}h^2}{48}$) and rounding error($\frac{10^{-16}}{h^2}$) are comparable when $h^4 = 27 * 10^{-16}$ i.e $h \approx 10^{-4}$. At 10^{-5}, we see the first loss of accuracy as step size decreases. This would be where the rounding error begins to dominate truncation error. Then, at 10^{-8}, h^2 reaches machine precision and computed value gets very wrong because of underflow (the value is too small) or overflow (the denominator is too small leading to a large value overall)

2. Use the formula in the previous exercise with $h = 0.2$, $h = 0.1$, and $h = 0.05$ to approximate $u''(x)$, where $u(x) = \sin x$ and $x = \pi/6$. Use one step of Richardson extrapolation, combining the results from $h = 0.2$ and $h = 0.1$, to obtain a higher order accurate approximation. Do the same with the results from $h = 0.1$ and $h = 0.05$. Finally do a second step of Richardson extrapolation, combining the two previously extrapolated values, to obtain a still higher order accurate approximation. Make a table of the computed results and their errors. What do you think is the order of accuracy after one step of Richardson extrapolation? How about after two?

   Solution:

$$u(x+h) = u(x) + hu'(x) + \frac{1}{2}h^2u''(x) + O(h^3)$$

$$u''(x) \approx \frac{u(x+h) + u(x-h) - 2u(x)}{h^2} + \frac{1}{12}h^2u''''(x) + C_1h^4u''''''(x) + O(h^6)$$

When $h_1 = \mathrm{h}/2$,

$$u''(x) \approx 4\frac{u(x+h_1) + u(x-h_1) - 2u(x)}{h^2} + \frac{1}{48}h^2u''''(x) + \frac{1}{4}C_1h^4u''''''(x) + O(h_1^6)$$

The error is $\mathrm{O}(h^2)$ right now. After the first iteration, we could get rid of one term and make it $\mathrm{O}(h^4)$ and after a second iteration, make it $\mathrm{O}(h^6)$.

```
r=[]
s=[]
def richardson(A,f,x,h,step):
    k=2
    if step ==1:
        return (2**k *A(f,x,h/2)-A(f,x,h))/(2**k -1)
    else:
        step-=1
        k+=2
        return (2**k *richardson(A,f,x,h/2,step)-richardson(A,f,x,h,step))
for h in [0.2,0.1]:
    r.append([h,richardson(central_diff,np.sin,np.pi/6,h,1),richardson(cen
```

```
for h in [0.2,0.1]:
    s.append([h,richardson(central_diff,np.sin,np.pi/6,h,2),richardson(cent
print(tabulate(r,headers=['h', 'Computed Value', 'Error'],tablefmt="latex"
print(tabulate(s,headers=['h', 'Computed Value', 'Error'],tablefmt="latex"
```

h	Computed Value	Error
0.2	-0.499999	5.5506e-07
0.1	-0.5	3.47145e-08

Dividing the two errors givs us $15.99 \approx 16 = 2^4$, agreeing with our $\mathrm{O}(h^4)$ estimate.

h	Computed Value	Error
0.2	-0.5	2.48054e-11
0.1	-0.5	4.70901e-13

Dividing the two errors givs us $53 \approx 2^6$, agreeing with our $\mathrm{O}(h^6)$ estimate.

3. Using Taylor series, derive the error term for the approximation

$$u'(x) \approx \frac{1}{2h}[-3u(x) + 4u(x+h) - u(x+2h)].$$

Solution:

$$4u(x+h) = 4u(x) + 4hu'(x) + 4\frac{1}{2}h^2u''(x) + 4\frac{1}{6}h^3u'''(x) + O(h^4)$$

$$u(x+2h) = u(x) + 2hu'(x) + 4\frac{1}{2}h^2u''(x) + 8\frac{1}{6}h^3u'''(x) + O(h^4)$$

Subtracting the two equations, we get

$$u'(x) = \frac{1}{2h}[-3u(x) + 4u(x+h) - u(x+2h)] + 4\frac{1}{6}h^3u'''(x) + O(h^4)]$$

$$u'(x) = \frac{1}{2h}[-3u(x) + 4u(x+h) - u(x+2h)] + \frac{1}{3}h^2u'''(x) + O(h^3)$$

Thus, the error term is $\frac{1}{3}h^2u'''(x) + O(h^3)$ which is second order.

4. Consider a forward difference approximation for the second derivative of the form

$$u''(x) \approx Au(x) + Bu(x+h) + Cu(x+2h).$$

Use Taylor's theorem to determine the coefficients A, B, and C that give the maximal order of accuracy and determine what this order is.

Solution:

$$Bu(x+h) = Bu(x) + Bhu'(x) + B\frac{1}{2}h^2u''(x) + B\frac{1}{6}h^3u'''(x) + BO(h^4)$$

$$Cu(x+2h) = Cu(x) + 2Chu'(x) + 4C\frac{1}{2}h^2u''(x) + 8C\frac{1}{6}h^3u'''(x) + CO(h^4)$$

Thus, we can easily see that to get rid of u(x) and u'(x), we need

$$A = -B - C, B = -2C$$

But we cannot get rid of u"'(x) term.

$$u''(x) \approx Ch^2u''(x) + Ch^3u'''(x) + (B + C)O(h^4)$$

Thus, C = $\frac{1}{h^2}$

So,

$$u''(x) \approx u''(x) + hu'''(x) + O(h^2)$$

This is first order.

$$C = \frac{1}{h^2}, B = -\frac{2}{h^2}, A = \frac{1}{h^2}$$

5. Consider the two-point boundary value problem

$$u'' + 2xu' - x^2u = x^2, \quad u(0) = 1, \quad u(1) = 0.$$

Let $h = 1/4$ and explicitly write out the difference equations, using centered differences for all derivatives.

Solution:

Since h = 0.25, there are 3 interior points u_1, u_2, u_3 and the boundaries are u_0 and u_4.

$$\frac{u_2 - 2u_1 + u_0}{(0.25)^2} + 2 * 0.25\frac{u_2 - u_0}{2 * 0.25} - (0.25)^2u_1 = (0.25)^2$$

$$\frac{u_3 - 2u_2 + u_1}{(0.25)^2} + 2 * 0.5\frac{u_3 - u_1}{2 * 0.25} - (0.5)^2u_2 = (0.5)^2$$

$$\frac{u_4 - 2u_3 + u_2}{(0.25)^2} + 2 * 0.75\frac{u_4 - u_2}{2 * 0.25} - (0.75)^2u_3 = (0.75)^2$$

Plugging in the boundary values, this becomes

$$\frac{u_2 - 2u_1 + 1}{(0.25)^2} + 2 * 0.25\frac{u_2 - 1}{2 * 0.25} - (0.25)^2u_1 = (0.25)^2$$

$$\frac{u_3 - 2u_2 + u_1}{(0.25)^2} + 2 * 0.5\frac{u_3 - u_1}{2 * 0.25} - (0.5)^2u_2 = (0.5)^2$$

$$\frac{-2u_3 + u_2}{(0.25)^2} + 2 * 0.75\frac{-u_2}{2 * 0.25} - (0.75)^2u_3 = (0.75)^2$$

6. A rod of length 1 meter has a heat source applied to it and it eventually reaches a steady-state where the temperature is not changing. The conductivity of the rod is a function of position x and is given by $c(x) = 1 + x^2$. The left end of the rod is held at a constant temperature of 1 degree. The right end of the rod is insulated so that

no heat flows in or out from that end of the rod. This problem is described by the boundary value problem:

$$\frac{d}{dx}\left((1+x^2)\frac{du}{dx}\right) = f(x), \quad 0 \le x \le 1,$$

$$u(0) = 1, \quad u'(1) = 0.$$

(a) Write down a set of difference equations for this problem. Be sure to show how you do the differencing at the endpoints. [Note: It is better **not** to rewrite $\frac{d}{dx}((1+x^2)\frac{du}{dx})$ as $(1+x^2)u''(x) + 2xu'(x)$; leave the equation in the form above.]

(b) Write a MATLAB code to solve the difference equations. You can test your code on a problem where you know the solution by choosing a function $u(x)$ that satisfies the boundary conditions and determining what $f(x)$ must be in order for $u(x)$ to solve the problem. Try $u(x) = (1-x)^2$. Then $f(x) = 2(3x^2 - 2x + 1)$.

(c) Try several different values for the mesh size h. Based on your results, what would you say is the order of accuracy of your method?

Solution:

(a) For the interior points:

$$\frac{(1+x_{j+1/2}^2)\frac{du}{dx}|_{j+1/2} - (1+x_{j-1/2}^2)\frac{du}{dx}|_{j-1/2}}{h} = f(x_j)$$

$$\frac{(1+x_{j+1/2}^2)(u_{j+1}-u_j) - (1+x_{j-1/2}^2)(u_j - u_{j-1})}{h^2} = f(x_j)$$

For the end points:

$$u_0 = 1$$

And

$$u'(1) = 0$$

So, using a second order backward scheme

$$3u_{end} - 4u_{end-1} + u_{end-2} = 0$$

Since all the differencing is second order, the local error is expected to be $O(h^2)$. The global error would be slightly bigger.

(b)

```
def rho(x):
    return 1+x**2

def tridiag(a, b, c, k1=-1, k2=0, k3=1):
    return np.diag(a, k1) + np.diag(b, k2) + np.diag(c, k3)

def u(x):
```

```
    return (1-x)**2
U=[]
def f(x):
    return 6*x**2 - 4*x + 2
wer=[]
error=[]
for N in [10,100,1000,10000]:
    a=[]
    b=[]
    c=[]
    B=[]
    h=1/N
    U=[]

    for i in range(N-1):
        a.append(rho((i+0.5)*h)/(h**2))
        b.append((-rho((i+0.5)*h)-rho((i+1.5)*h))/(h**2))
        c.append(rho((i+1.5)*h)/(h**2))
        B.append(f((i+1)*h))
        U.append(u((i+1)*h))
    #b=-a-c
    a=a+[1]
    c=[0]+c
    b=[1]+b+[-1]
    B=[1]+B+[0]
    U=[1]+U+[0]
    A = tridiag(a, b, c)

    A[-1,-1]=3
    A[-1,-2]=-4
    A[-1,-3]=1

    Y= np.linalg.solve(A, B)
    eee=U-Y
    wer.append([h,np.linalg.norm(eee)])
    error+=[np.linalg.norm(eee)]
print(tabulate(wer,headers=['h', 'Error'],tablefmt="latex"))
```

(c)

h	Error
0.1	0.00555442
0.01	0.000167169
0.001	5.26522e-06
0.0001	1.58201e-07

Dividing the errors givs us a jump of $33 \approx 10^{1.5}$ showing that the error is $O(h^{1.5})$ in between O(h) and $O(h^2)$.