

```

1  # -*- coding: utf-8 -*-
2  """
3  Created on Tue Oct  1 17:34:07 2019
4
5  @author: Alexm
6  """
7
8  print("Alexandra Mulholland 17336557")
9  import numpy as np
10 import matplotlib.pyplot as plt
11
12 #Aim: To compute the second derivative of  $f(x)=xe^x$  using Richardson extrapolation
13 #Defining the function  $f(x)=xe^x$  to be differentiated
14
15 #PART 1
16
17 def function(x):
18     f=x*np.exp(x)
19     return f
20 #First derivative, analytic using the product rule
21 def analytic(x):
22     """analytic first derivative"""
23     f= (x+1)*np.exp(x)
24     return f
25 #Defining the analytic second derivative, using product rule again
26 def analytic2(x):
27     """analytic second derivative"""
28     f= (x+2)*np.exp(x)
29     return f
30
31 #plot of function
32 x=np.arange(0.0,2.0,0.001)
33 plt.figure(1)
34 plt.xlabel('x')
35 plt.ylabel('f(x)')
36 plt.title('Plot of  $f(x)=xe^x$ ')
37 plt.plot(x,function(x), color = 'purple')
38 plt.grid(True)
39 plt.show()
40
41 #Plotting analytical second derivative
42 x=np.arange(0.0,2.0,0.0001)
43 plt.figure(3)
44 plt.xlabel('x')
45 plt.ylabel('f''(x)')
46 plt.title('Plot of second derivative  $f''(x)=(x+2)e^x$ ')
47 plt.plot(x,analytic2(x), color = 'pink')
48 plt.grid(True)
49 plt.show()
50
51 #PART 2
52
53 h=0.4
54 #Defining the central difference approximaton for step size h
55 def D_1_h(x,h):
56     f= (function(x+h)-2*function(x)+function(x-h))/h**2
57     return f
58 #Doubling the step size
59 def D_1_2h(x,h):
60     f= (function(x+2*h)-2*function(x)+function(x-2*h))/(4*h**2)
61     return f
62 #Minimising error
63 def D_2_h(x,h):
64     f= (4*D_1_h(x,h)-D_1_2h(x,h))/3
65     return f
66
67 def D_3_h(x,h):
68     f = (16*D_2_h(x,h)-D_2_h(x,2*h))/15
69     return f
70
71 def D_4_h(x,h):
72     f = (64*D_3_h(x,h)-D_3_h(x,2*h))/63

```

```

73     return f
74     #Above is wrong
75
76     #Have to do for decreasing h
77     #For d1 d2 d3 d4, plotting a matrix M using a loop
78     x=2
79     h=0.4
80     #top row is producing the h values
81     #analytic value is 29.5562243957226
82     print (analytic2(x))
83     print ("" )
84     print(D_1_h(x,h))
85     print ("" )
86     M = [[h,0,0,0],
87           [D_1_h(x,h),0,0,0],
88           [0,0,0,0],
89           [0,0,0,0],
90           [0,0,0,0]]
91     #j is column no., i is row no.
92     #With each iteration, halving h
93     #j ranging from 1 to 4, so that would exclude 1st column (column 0)
94     #M[row][column]
95     for j in range(1,4):
96         h=h/2
97         M[0][j]=h
98         M[1][j]=D_1_h(x,h)
99     #first row- h, h/2,h/4 etc
100    #second row, CD with decreasing h by half
101    #for i=1,M[0,1] --- d1(h/2)
102
103    #j now ranging from 0 to 3 i.e. including first column and excluding first row
104    #Genral formula applied
105    #where i takes value of n
106    for j in range(0,3):
107        for i in range(1,4):
108             $M[i+1][j+1]=((2^{2*(i)}) * M[i][j+1] - M[i][j]) / (2^{2*(i)} - 1)$ 
109    #In order to prevent random values, as code reads the blank spaces
110    #in the table in the notes as zeros
111    #so for this range of i and j, if i is greater than j, the value is made to be
112    #zero in this cell of the matrix
113    for j in range(1,5):
114        for i in range(0,5):
115            if i-1>j:
116                M[i][j]=0
117    #printing the matrix
118    print(M)
119    print ("" )
120    print ("" )
121
122    #PART 3
123
124    #Second derivative at x=2 to highest accuracy
125    #now defining a second matrix, which goes to higher values of Dn in order to
126    #find most accurate value
127    #now including i and j up to 8 (excluding h row) and renaming them y and z respectively
128    #Definig a costant x
129    #new matrix to be called A
130    x=2
131    h=0.4
132    #past D8, rounding errors occur
133    print ("" )
134    A = [[h,0,0,0,0,0,0,0],
135          [D_1_h(x,h),0,0,0,0,0,0,0],
136          [0,0,0,0,0,0,0,0],
137          [0,0,0,0,0,0,0,0],
138          [0,0,0,0,0,0,0,0],
139          [0,0,0,0,0,0,0,0],
140          [0,0,0,0,0,0,0,0],
141          [0,0,0,0,0,0,0,0],
142          [0,0,0,0,0,0,0,0]]
143    for z in range(1,8):
144        h=h/2

```

```

145     A[0][z]=h
146     A[1][z]=D_1_h(x,h)
147     for z in range(0,7):
148         for y in range(1,8):
149             A[y+1][z+1]=((2**(2*(y)))*A[y][z+1]-A[y][z])/(2**(2*(y))-1)
150     #Again, redefining the cells the code will see as having a zero value
151     for z in range(1,9):
152         for y in range(0,9):
153             if y-1>z:
154                 A[y][z]=0
155     print(A)
156     print ("")
157     print ("")
158     #Creating an empty list
159     #In range of i values 1 to 8 appending A[i] to [i-1]
160     #so i is i and j is equal to i-1
161     #Therefore, goes in a diagonal motion
162     # i is n in the general formula
163     Dn=[]
164     for y in range(1,9):
165         Dn.append(A[y][y-1])
166     errorrel=abs((analytic2(x)-Dn)/analytic2(x))
167     for y in range(0,7):
168         if (errorrel[y])-errorrel[y+1] < 0:
169             print("\nMost accurate estimation for initial value of h=0.4 is: D",y+1,"\nGiving a relative error of:",e
rorrel[y])
170             break
171     #D5 values
172     print ("")
173     print (29.556224395715102-analytic2(x))
174     print (29.55622439574263-analytic2(x))
175     print (29.5562243957235-analytic2(x))
176
177     #D6 Values
178     print ("")
179     print (29.556224395742646-analytic2(x))
180     print (29.556224395715073-analytic2(x))
181
182     #D7 value
183     print ("")
184     print (29.556224395715066-analytic2(x))
185
186
187     #PART 4 and PART 5
188
189     #change in accuracy as h decreases
190     #plotting D1 and D2 versus h and finding the relationship.
191     plt.plot(A[0],abs((A[1]-analytic2(x))/analytic2(x)), color = 'green')
192     plt.ylabel("Relative error")
193     plt.xlabel("step size, h")
194     plt.title("Variation of relative error with step size, h")
195     plt.show()
196
197     #Defining the improved estimations
198     def D_5_h(x,h):
199         f = ((256*D_4_h(x,h))-(D_4_h(x,2*h)))/255
200         return f
201
202     def D_6_h(x,h):
203         f = ((1024*D_5_h(x,h))-(D_5_h(x,2*h)))/1023
204         return f
205
206     def D_7_h(x,h):
207         f = ((4096*D_6_h(x,h))-(D_6_h(x,2*h)))/4095
208         return f
209
210     def D_8_h(x,h):
211         f = ((16384*D_7_h(x,h))-(D_7_h(x,2*h)))/16383
212         return f
213
214     #Defining the relative error for each estimate
215     x=2

```

```

216 def error_D1h(x,h):
217     f = np.abs(((analytic2(x)-D_1_h(x,h)))/analytic2(x))
218     return f
219
220 def error_D2h(x,h):
221     f = np.abs(((analytic2(x)-D_2_h(x,h)))/analytic2(x))
222     return f
223
224 def error_D3h(x,h):
225     f = np.abs(((analytic2(x)-D_3_h(x,h)))/analytic2(x))
226     return f
227
228 def error_D4h(x,h):
229     f = np.abs(((analytic2(x)-D_4_h(x,h)))/analytic2(x))
230     return f
231
232 def error_D5h(x,h):
233     f = np.abs(((analytic2(x)-D_5_h(x,h)))/analytic2(x))
234     return f
235
236 def error_D6h(x,h):
237     f = np.abs(((analytic2(x)-D_6_h(x,h)))/analytic2(x))
238     return f
239
240 def error_D7h(x,h):
241     f = np.abs(((analytic2(x)-D_7_h(x,h)))/analytic2(x))
242     return f
243
244 def error_D8h(x,h):
245     f = np.abs(((analytic2(x)-D_8_h(x,h)))/analytic2(x))
246     return f
247
248 #Starting with h=0.4
249 #plots show the rounding error with decreasing initial h
250 h = np.arange(0,0.4,0.001)
251
252 plt.figure(3)
253 plt.xlabel('step size, h')
254 plt.ylabel('f"(x)')
255 plt.title('Relative error of each estimation versus h, initialised at 0.4')
256 plt.loglog(h,error_D1h(x,h), label = 'D1(h)',color = 'red')
257 plt.loglog(h,error_D2h(x,h), label = 'D2(h)', color = 'orange')
258 plt.loglog(h,error_D3h(x,h), label = 'D3(h)', color = 'yellow')
259 plt.loglog(h,error_D4h(x,h), label = 'D4(h)', color = 'green')
260 plt.loglog(h,error_D5h(x,h),label = 'D5(h)', color = 'blue')
261 plt.loglog(h,error_D6h(x,h), label = 'D6(h)', color = 'purple')
262 plt.loglog(h,error_D7h(x,h), label = 'D7(h)', color = 'pink')
263 plt.loglog(h,error_D8h(x,h), label = 'D8(h)', color = 'brown')
264 plt.legend()
265 plt.show()
266
267 #starting with h=0.2
268 h = np.arange(0,0.2,0.001)
269
270 plt.figure(4)
271 plt.xlabel('step size, h')
272 plt.ylabel('f"(x)')
273 plt.title('Relative error of each estimation versus h, initialised at 0.2')
274 plt.loglog(h,error_D1h(x,h), label = 'D1(h)', color = 'red')
275 plt.loglog(h,error_D2h(x,h), label = 'D2(h)', color = 'orange')
276 plt.loglog(h,error_D3h(x,h), label = 'D3(h)', color = 'yellow')
277 plt.loglog(h,error_D4h(x,h), label = 'D4(h)', color = 'green')
278 plt.loglog(h,error_D5h(x,h),label = 'D5(h)', color = 'blue')
279 plt.loglog(h,error_D6h(x,h), label = 'D6(h)', color = 'purple')
280 plt.loglog(h,error_D7h(x,h), label = 'D7(h)', color = 'pink')
281 plt.loglog(h,error_D8h(x,h), label = 'D8(h)', color = 'brown')
282 plt.legend()
283 plt.show()
284
285
286 #h=0.1
287 h = np.arange(0,0.1,0.001)

```

```

288 plt.figure(5)
289 plt.xlabel('step size, h')
290 plt.ylabel('f"(x)')
291 plt.title('Relative error of each estimation versus h, initialised at 0.1')
292 plt.loglog(h,error_D1h(x,h), label = 'D1(h)', color = 'red')
293 plt.loglog(h,error_D2h(x,h), label = 'D2(h)', color = 'orange')
294 plt.loglog(h,error_D3h(x,h), label = 'D3(h)', color = 'yellow')
295 plt.loglog(h,error_D4h(x,h), label = 'D4(h)', color = 'green')
296 plt.loglog(h,error_D5h(x,h),label = 'D5(h)', color = 'blue')
297 plt.loglog(h,error_D6h(x,h), label = 'D6(h)', color = 'purple')
298 plt.loglog(h,error_D7h(x,h), label = 'D7(h)', color = 'pink')
299 plt.loglog(h,error_D8h(x,h), label = 'D8(h)', color = 'brown')
300 plt.legend()
301 plt.show()
302
303
304 #h=0.05
305 h = np.arange(0,0.05,0.001)
306
307 plt.figure(6)
308 plt.xlabel('step size, h')
309 plt.ylabel('f"(x)')
310 plt.title('Relative error of each estimation versus h, initialised at 0.05')
311 plt.loglog(h,error_D1h(x,h), label = 'D1(h)', color = 'red')
312 plt.loglog(h,error_D2h(x,h), label = 'D2(h)', color = 'orange')
313 plt.loglog(h,error_D3h(x,h), label = 'D3(h)', color = 'yellow')
314 plt.loglog(h,error_D4h(x,h), label = 'D4(h)', color = 'green')
315 plt.loglog(h,error_D5h(x,h),label = 'D5(h)', color = 'blue')
316 plt.loglog(h,error_D6h(x,h), label = 'D6(h)', color = 'purple')
317 plt.loglog(h,error_D7h(x,h), label = 'D7(h)', color = 'pink')
318 plt.loglog(h,error_D8h(x,h), label = 'D8(h)', color = 'brown')
319 plt.legend()
320 plt.show()

```