"""

The questions provided by the RandomizeQuestionsByID: 8, 16, 17, 19, 21, 32

Main git link: https://github.com/Benny902/Numeric-Analysis-Hackaton

Eden Dahan 318641222

Ruth Avivi 208981555

Ron Mansharof 208839787

Benny Shalom 203500780

"""

Question 19: https://github.com/Benny902/Numeric-Analysis-Hackaton/tree/main/q19

By LU Method:

||A||= 1.8333333333333333 , ||A1||= 408.0000000000026

Cond ||A||*||A^-1||= 748.0000000000048

J1: [[1, 0, 0], [-0.5, 1, 0], [-0.3333333333333333, 0, 1]]

J2: [[1, 0, 0], [0, 1, 0], [0, -0.9999999999999997, 1]]

L= $J1^{-1} * J2^{-1}$ = [[1.0, 0.0, 0.0], [0.5, 1.0, 0.0], [0.3333333333333333, 0.9999999999999997, 1.0]]

U= J2*J1*A = [[1.0, 0.5, 0.3333333333333333], [0.0, 0.08333333333333334, 0.0888888888888889], [0.0, 0.0, 0.005555555555555536]]

X : { 10.00000000000131936, -38.00000000000131936, 30.00000000000131936, }

גורם ההצגה גדול מידיי, לכן נבחר להשתמש בשיטה השנייה, זיידל.

By seidel Method:

$$\vec{x}_{r+1} = -(L+D)^{-1}U\vec{x}_r + (L+D)^{-1}\vec{b}$$

Count	var1	var2	var3
0	0.0	0.0	0.0
1	1.0	-1.5	0.20833333333333343
2	1.6805555555555556	-2.6770833333333335	0.5454282407407413
3	2.1567322530864197	-3.6441695601851856	0.9606581950874493
4	2.5018653817301093	-4.473291718910751	1.42183901242159
5	2.7626995219815123	-5.2104285422884615	1.9085364745580569
6	2.969035446291545	-5.88495552535586	2.4078019962089168
7	3.1398770972749577	-6.515667143069124	2.911455433378143
8	3.2873484270751807	-7.1146142156463785	3.4143537244326727
9	3.419189199678965	-7.689549092842952	3.913287699922082
10	3.5403453131141154	-8.245483744612734	4.406279158909059
11	3.6539821526700145	-8.785682598186815	4.892132993283496

12 3.7621303013322427 -9.312295196960985 5.37015182731416

13 3.8660969893757726 -9.82675935454928 5.839954210893645

14 3.966728273643425 -10.330058068635372 6.301358796388508

15 4.064576102188184 -10.822883250573657 6.754310559570098

16 4.160004772096796 -11.305740077822767 7.198833810450465

17 4.253258768761229 -11.779013510979691 7.6350022741225665

18 4.344505997448991 -12.243010701765412 8.062920048125113

19 4.433865334841002 -12.697988038355339 8.482709489875836

20 4.521424189219058 -13.144168401235465 8.894503519512568

21 4.607249694113543 -13.581752180804742 9.298440735816689

22 4.691395845130142 -14.01092431955773 9.69466232423026

23 4.773908051702112 -14.431858820725864 10.083310106403811

24 4.854826041561662 -14.844721642145352 10.464525316745586

25 4.934185715490814 -15.249672560795412 10.838447841842909

26 5.012020333116736 -15.646866381057286 11.205215754460381

.

.

.

1635 8.999999999999911 -35.999999999999545 29.99999999999958

1636 8.999999999999913 -35.99999999999955 29.999999999999584

1637 8.999999999999915 -35.99999999999957 29.9999999999996

1638 8.999999999999918 -35.999999999999574 29.999999999999602

1639 8.99999999999992 -35.99999999999959 29.999999999999616

1640 8.999999999999922 -35.999999999999595 29.999999999999627

Solution={ var1= 9.00000000000131936 var2= -36.00000000000131936 var3= 30.00000000000131936 }

רמת הדיוק נקבעת ע"י האפסילון של המחשב שהיא 2 בחזקת מינוס 52

השתמשנו בשיטה זו כי היא יותר מדויקת

Question 21: https://github.com/Benny902/Numeric-Analysis-Hackaton/tree/main/q21

By LU Method:

||A||= 19 , ||A1||= 0.48347107438016534

Cond ||A||*||A^-1||= 9.185950413223141

J1: [[1, 0, 0], [-0.4, 1, 0], [-0.5, 0, 1]]

J2: [[1, 0, 0], [0, 1, 0], [0, 0.4411764705882353, 1]]

L= $J1^{-1}$* $J2^{-1}$ = [[1.0, 0.0, 0.0], [0.4, 1.0, 0.0], [0.5, -0.4411764705882353, 1.0]]

U= J2*J1*A = [[10.0, 8.0, 1.0], [0.0, 6.8, -5.4], [0.0, 0.0, 7.117647058823529]]

X : { -2.00000000000132016, 1.50000000000132016, 1.00000000000132016, }

גורם ההצגה גדול מידיי, לכן נבחר להשתמש בשיטה השנייה, ייעקובי.

By jaacobian Method:

$$\vec{x}_{r+1} = -D^{-1}(L+U)\vec{x}_r + D^{-1}\vec{b}$$

L= [[0, 0, 0], [4, 0, 0], [5, 1, 0]]

D= [[10, 0, 0], [0, 10, 0], [0, 0, 10]]

U= [[0, 8, 1], [0, 0, -5], [0, 0, 0]]

||G||= 0.9

Converge

Count	var1	var2	var3
0	0.00000	0.00000	0.00000
1	-0.70000	0.20000	0.15000
2	-0.87500	0.55500	0.48000
3	-1.19200	0.79000	0.53200
4	-1.38520	0.94280	0.66700
5	-1.52094	1.08758	0.74832
6	-1.64490	1.18254	0.80171
7	-1.72620	1.25881	0.85419
8	-1.79247	1.31758	0.88722
9	-1.84278	1.36060	0.91448

10	-1.87993	1.39435	0.93533
11	-1.90902	1.41964	0.95053
12	-1.93076	1.43887	0.96254
13	-1.94735	1.45358	0.97149
14	-1.96001	1.46469	0.97832
15	-1.96958	1.47316	0.98354
16	-1.97688	1.47960	0.98747
17	-1.98243	1.48449	0.99048
18	-1.98664	1.48821	0.99276
19	-1.98985	1.49104	0.99450
20	-1.99228	1.49319	0.99582
21	-1.99413	1.49482	0.99682
22	-1.99554	1.49606	0.99758
.			
.			
.			
33	-1.99978	1.49981	0.99988
34	-1.99983	1.49985	0.99991
35	-1.99987	1.49989	0.99993
36	-1.99990	1.49992	0.99995
37	-1.99993	1.49994	0.99996
38	-1.99994	1.49995	0.99997
39	-1.99996	1.49996	0.99998
40	-1.99997	1.49997	0.99998

Solution : { var1= -2.000000000132016, var2= 1.500000000132016, var3= 1.000000000132016, }

רמת הדיוק נקבעת ע"י האפסילון של המחשב שהיא 2 בחזקת מינוס 52

השתמשנו בשיטה זו כי היא יותר מדויקת

Question 32: https://github.com/Benny902/Numeric-Analysis-Hackaton/tree/main/q32

l0(0.65)= ((0.65-0.35)/(0.2-0.35))* ((0.65-0.45)/(0.2-0.45))* ((0.65-0.6)/(0.2-0.6))* ((0.65-0.75)/(0.2-0.75))* ((0.65-0.85)/(0.2-0.85))* ((0.65-0.9)/(0.2-0.9))= -0.003996003996003995

l1(0.65)= ((0.65-0.2)/(0.35-0.2))* ((0.65-0.45)/(0.35-0.45))* ((0.65-0.6)/(0.35-0.6))* ((0.65-0.75)/(0.35-0.75))* ((0.65-0.85)/(0.35-0.85))* ((0.65-0.9)/(0.35-0.9))= 0.054545454545454564

l2(0.65)= ((0.65-0.2)/(0.45-0.2))* ((0.65-0.35)/(0.45-0.35))* ((0.65-0.6)/(0.45-0.6))* ((0.65-0.75)/(0.45-0.75))* ((0.65-0.85)/(0.45-0.85))* ((0.65-0.9)/(0.45-0.9))= -0.16666666666666677

l3(0.65)= ((0.65-0.2)/(0.6-0.2))* ((0.65-0.35)/(0.6-0.35))* ((0.65-0.45)/(0.6-0.45))* ((0.65-0.75)/(0.6-0.75))* ((0.65-0.85)/(0.6-0.85))* ((0.65-0.9)/(0.6-0.9))= 0.8

l4(0.65)= ((0.65-0.2)/(0.75-0.2))* ((0.65-0.35)/(0.75-0.35))* ((0.65-0.45)/(0.75-0.45))* ((0.65-0.6)/(0.75-0.6))* ((0.65-0.85)/(0.75-0.85))* ((0.65-0.9)/(0.75-0.9))= 0.45454545454545486

l5(0.65)= ((0.65-0.2)/(0.85-0.2))* ((0.65-0.35)/(0.85-0.35))* ((0.65-0.45)/(0.85-0.45))* ((0.65-0.6)/(0.85-0.6))* ((0.65-0.75)/(0.85-0.75))* ((0.65-0.9)/(0.85-0.9))= -0.20769230769230781

l6(0.65)= ((0.65-0.2)/(0.9-0.2))* ((0.65-0.35)/(0.9-0.35))* ((0.65-0.45)/(0.9-0.45))* ((0.65-0.6)/(0.9-0.6))* ((0.65-0.75)/(0.9-0.75))* ((0.65-0.85)/(0.9-0.85))= 0.06926406926406924

P6(0.65) = l0(0.65)*y0 + l1(0.65)*y1 + l2(0.65)*y2 + l3(0.65)*y3 + l4(0.65)*y4 + l5(0.65)*y5 + l6(0.65)*y6

lagrange formula - sigma(from i=1 to n)*Li(x)*Yi

lagrange sol = 13.90225949050949200000132132

$f(x)=((y_1-y_2)/(x_1-x_2))*point + (y_2x_1 - y_1x_2)/(x_1-x_2)$

f(x)=((13.9776-13.7241)/(0.6-0.75))*0.65 + (13.7241 * 0.6 - 13.9776 * 0.75)/(0.6-0.75)

f(0.65) = 13.893099999999993

linear sol = 13.89309999999999300000132132

Process finished with exit code 0

תמיד נעדיף להשתמש בקירוב ע"פי לאגראנז', בגלל שקירוב לינארי הוא קירוב מסדר ראשון , לעומת קירוב לאגרנאז' שהוא קירוב מסדר גודל הנקודות פחות אחד , במקרה שלנו הוא ממעלה שישית

Question 8a: https://github.com/Benny902/Numeric-Analysis-Hackaton/tree/main/q8

Mehthod 1 : Bisection:

interval (-1, -0.9):

----

root : -0.9189385414123535, iterations: 21

----

interval (-0.9, -0.8):

guess -0.8 is not an approximated root

interval (-0.8, -0.7):

----

root : -0.7750974893569947, iterations: 23

----

interval (-0.7, -0.6):

guess -0.6000000000000001 is not an approximated root

interval (-0.6, -0.5):

----

root : -0.5723649501800536, iterations: 22

----

interval (-0.5, -0.4):

guess -0.4 is not an approximated root

interval (-0.4, -0.3):

guess -0.30000000000000004 is not an approximated root

interval (-0.3, -0.2):

----

root : -0.22579135894775387, iterations: 20

----

interval (-0.2, -0.1):

guess -0.1 is not an approximated root

interval (-0.1, 0.0):

guess -6.223015277861142e-62 is not an approximated root

interval (0.0, 0.1):

guess 0.1 is not an approximated root

interval (0.1, 0.2):

guess 0.2 is not an approximated root

interval (0.2, 0.3):

guess 0.29999999999999993 is not an approximated root

interval (0.3, 0.4):

guess 0.4 is not an approximated root

interval (0.4, 0.5):

guess 0.5 is not an approximated root

interval (0.5, 0.6):

guess 0.5999999999999999 is not an approximated root

interval (0.6, 0.7):

guess 0.7 is not an approximated root

interval (0.7, 0.8):

guess 0.8 is not an approximated root

interval (0.8, 0.9):

guess 0.8999999999999999 is not an approximated root

interval (0.9, 1.0):

guess 1.0 is not an approximated root

interval (1.0, 1.1):

guess 1.1 is not an approximated root

interval (1.1, 1.2):

guess 1.1999999999999997 is not an approximated root

interval (1.2, 1.3):

guess 1.2999999999999998 is not an approximated root

interval (1.3, 1.4):

guess 1.4 is not an approximated root

interval (1.4, 1.5):

guess 1.5 is not an approximated root

interval (1.5, 1.6):

guess 1.6 is not an approximated root

interval (1.6, 1.7):

guess 1.6999999999999997 is not an approximated root

interval (1.7, 1.8):

guess 1.7999999999999998 is not an approximated root

interval (1.8, 1.9):

guess 1.9 is not an approximated root

interval (1.9, 2.0):

guess 2.0 is not an approximated root

[-0.9189385414123535, -0.7750974893569947, -0.5723649501800536, -0.22579135894775387]

Process finished with exit code 0

Method 2 : Newthon raphson

interval (-1, -0.9):

----

root : -0.9189385332046592, iterations: 5

----

interval (-0.9, -0.8):

interval (-0.8, -0.7):

----

root : -0.7750974982919796, iterations: 4

----

interval (-0.7, -0.6):

----

root : -0.5723649435387966, iterations: 5

----

interval (-0.6, -0.5):

interval (-0.5, -0.4):

interval (-0.4, -0.3):

----

root : -0.22579138626602957, iterations: 4

----

interval (-0.3, -0.2):

interval (-0.2, -0.1):

interval (-0.1, 0.0):

interval (0.0, 0.1):

interval (0.1, 0.2):

interval (0.2, 0.3):

interval (0.3, 0.4):

interval (0.4, 0.5):

interval (0.5, 0.6):

interval (0.6, 0.7):

interval (0.7, 0.8):

interval (0.8, 0.9):

interval (0.9, 1.0):

interval (1.0, 1.1):

interval (1.1, 1.2):

interval (1.2, 1.3):

interval (1.3, 1.4):

interval (1.4, 1.5):

interval (1.5, 1.6):

interval (1.6, 1.7):

interval (1.7, 1.8):

interval (1.8, 1.9):

interval (1.9, 2.0):

[-0.9189385332046592, -0.7750974982919796, -0.5723649435387966, -0.22579138626602957]

Process finished with exit code 0

Question 8b: https://github.com/Benny902/Numeric-Analysis-Hackaton/tree/main/q8

C:\Users\Bennysh\PycharmProjects\pythonProjectzzz\venv\Scripts\python.exe C:/Users/Bennysh/PycharmProjects/numericAnalysis2.py/main.py

Simpson method is going by the formula - (h/3)*(f(a)+2*sigma(from j=1 to last even)*f(X2j)+4*sigma(from j=1 to last odd)*f(X2j-1)+f(b))

h = 0.08

a, b = -0.4, 0.4

Adding f(start) + f(end): -0.23222430688752257 + 0.09589362245627364

Iteration 1, Last sum += 4 * (odd index value):

-0.13633068443124893 += -0.5386245207225094

Iteration 2, Last sum += 2 * (even index value):

-0.6749552051537584 += -0.03723338075379197

Iteration 3, Last sum += 4 * (odd index value):

-0.7121885859075504 += 0.28913361359938233

Iteration 4, Last sum += 2 * (even index value):

-0.4230549723081681 += 0.2545530305228372

Iteration 5, Last sum += 4 * (odd index value):

-0.1685019417853309 += 0.6061982845504544

Iteration 6, Last sum += 2 * (even index value):

0.43769634276512354 += 0.3094103603585617

Iteration 7, Last sum += 4 * (odd index value):

0.7471067031236852 += 0.5819194173484812

Iteration 8, Last sum += 2 * (even index value):

1.3290261204721663 += 0.2604141591743404

Iteration 9, Last sum += 4 * (odd index value):

1.5894402796465066 += 0.45164903838037757

Result = h/3 * 2.0410893180268843 = 0.05442904848071692

h = 0.04

a, b = -0.4, 0.4

Adding f(start) + f(end): -0.23222430688752257 + 0.09589362245627364

Iteration 1, Last sum += 4 * (odd index value):

-0.13633068443124893 += -0.7606628898656949

Iteration 2, Last sum += 2 * (even index value):

-0.8969935742969438 += -0.2693122603612547

Iteration 3, Last sum += 4 * (odd index value):

-1.1663058346581985 += -0.30099388237072233

Iteration 4, Last sum += 2 * (even index value):

-1.4672997170289208 += -0.03723338075379197

Iteration 5, Last sum += 4 * (odd index value):

-1.5045330977827127 += 0.12483126949136411

Iteration 6, Last sum += 2 * (even index value):

-1.3797018282913487 += 0.14456680679969117

Iteration 7, Last sum += 4 * (odd index value):

-1.2351350214916574 += 0.41658893260263197

Iteration 8, Last sum += 2 * (even index value):

-0.8185460888890255 += 0.2545530305228372

Iteration 9, Last sum += 4 * (odd index value):

-0.5639930583661883 += 0.5706786264961138

Iteration 10, Last sum += 2 * (even index value):

0.00668556812992549 += 0.3030991422752272

Iteration 11, Last sum += 4 * (odd index value):

0.3097847104051527 += 0.6206827993788409

Iteration 12, Last sum += 2 * (even index value):

0.9304675097839936 += 0.3094103603585617

Iteration 13, Last sum += 4 * (odd index value):

1.2398778701425552 += 0.6047393418534166

Iteration 14, Last sum += 2 * (even index value):

1.8446172119959718 += 0.2909597086742406

Iteration 15, Last sum += 4 * (odd index value):

2.1355769206702124 += 0.5531992983618675

Iteration 16, Last sum += 2 * (even index value):

2.68877621903208 += 0.2604141591743404

Iteration 17, Last sum += 4 * (odd index value):

2.94919037820642 += 0.48654291582384995

Iteration 18, Last sum += 2 * (even index value):

3.4357332940302703 += 0.22582451919018878

Iteration 19, Last sum += 4 * (odd index value):

3.661557813220459 += 0.4171013633847828

Result = h/3 * 4.078659176605242 = 0.05438212235473656

0.0543821223547365600000132315

Process finished with exit code 0

Question 16a: https://github.com/Benny902/Numeric-Analysis-Hackaton/tree/main/q16

Method 1 : Bisection

interval (0, 0.1):

----

root : 0, iterations: 1

----

interval (0.1, 0.2):

guess 0.2 is not an approximated root

interval (0.2, 0.3):

guess 0.29999999999999993 is not an approximated root

interval (0.3, 0.4):

guess 0.4 is not an approximated root

interval (0.4, 0.5):

guess 0.5 is not an approximated root

interval (0.5, 0.6):

guess 0.5999999999999999 is not an approximated root

interval (0.6, 0.7):

guess 0.7 is not an approximated root

interval (0.7, 0.8):

guess 0.8 is not an approximated root

interval (0.8, 0.9):

guess 0.8999999999999999 is not an approximated root

interval (0.9, 1.0):

guess 1.0 is not an approximated root

interval (1.0, 1.1):

guess 1.1 is not an approximated root

interval (1.1, 1.2):

guess 1.1999999999999997 is not an approximated root

interval (1.2, 1.3):

guess 1.2999999999999998 is not an approximated root

interval (1.3, 1.4):

guess 1.4 is not an approximated root

interval (1.4, 1.5):

guess 1.5 is not an approximated root

interval (1.5, 1.6):

guess 1.6 is not an approximated root

interval (1.6, 1.7):

----

root : 1.6666666671633719, iterations: 27

----

interval (1.7, 1.8):

guess 1.7999999999999998 is not an approximated root

interval (1.8, 1.9):

guess 1.9 is not an approximated root

interval (1.9, 2.0):

guess 2.0 is not an approximated root

interval (2.0, 2.1):

guess 2.0999999999999996 is not an approximated root

interval (2.1, 2.2):

guess 2.2 is not an approximated root

interval (2.2, 2.3):

guess 2.3 is not an approximated root

interval (2.3, 2.4):

guess 2.3999999999999995 is not an approximated root

interval (2.4, 2.5):

guess 2.5 is not an approximated root

interval (2.5, 2.6):

guess 2.5999999999999996 is not an approximated root

interval (2.6, 2.7):

guess 2.7 is not an approximated root

interval (2.7, 2.8):

guess 2.8 is not an approximated root

interval (2.8, 2.9):

guess 2.8999999999999995 is not an approximated root

interval (2.9, 3.0):

guess 3.0 is not an approximated root

[0, 1.6666666671633719]

Process finished with exit code 0

Method 3: Secant

interval (0, 0.1):

----

root : 0.0005032325142576226, iterations: 9

----

interval (0.1, 0.2):

interval (0.2, 0.3):

interval (0.3, 0.4):

interval (0.4, 0.5):

interval (0.5, 0.6):

interval (0.6, 0.7):

interval (0.7, 0.8):

interval (0.8, 0.9):

interval (0.9, 1.0):

interval (1.0, 1.1):

interval (1.1, 1.2):

interval (1.2, 1.3):

interval (1.3, 1.4):

interval (1.4, 1.5):

interval (1.5, 1.6):

interval (1.6, 1.7):

interval (1.7, 1.8):

interval (1.8, 1.9):

interval (1.9, 2.0):

guess -0.36038487401635777 is not an approximated root

interval (2.0, 2.1):

interval (2.1, 2.2):

----

root : 1.666666666666711, iterations: 11

----

interval (2.2, 2.3):

interval (2.3, 2.4):

interval (2.4, 2.5):

interval (2.5, 2.6):

interval (2.6, 2.7):

interval (2.7, 2.8):

interval (2.8, 2.9):

interval (2.9, 3.0):

[0.0005032325142576226, 1.666666666666711]

Process finished with exit code 0

Question 16b: https://github.com/Benny902/Numeric-Analysis-Hackaton/tree/main/q16

Romberg method is using trapezoid method -

The formula of Trapezoidal is - sigma(from i=1 to N)*(h/2)*(f(Xi-h)+f(Xi))

The formula of Romberg Method is - R(n,m)=1/(4^m-1)*(4^m*R(n,m-1)-R*(n-1,m-1))

number iteration of trapezoid method: 1

Approximation: -0.0010319931668386325

number iteration of romberg method: 1

Approximation: -0.000636967392349573

The result is -

-0.00059664378937282070000132319

Simpson method is going by the formula - (h/3)*(f(a)+2*sigma(from j=1 to last even)*f(X2j)+4*sigma(from j=1 to last odd)*f(X2j-1)+f(b))

h = 0.05

a, b = 0.5, 1

Adding f(start) + f(end): -0.00278493319694596 + -0.00024681960817335923

Iteration 1, Last sum += 4 * (odd index value):

-0.003031752805119319 += -0.009533729501806004

Iteration 2, Last sum += 2 * (even index value):

-0.012565482306925323 += -0.003984460904118892

Iteration 3, Last sum += 4 * (odd index value):

-0.016549943211044214 += -0.0065216368727756035

Iteration 4, Last sum += 2 * (even index value):

-0.023071580083819817 += -0.0026176142430043743

Iteration 5, Last sum += 4 * (odd index value):

-0.02568919432682419 += -0.00412797266735453

Iteration 6, Last sum += 2 * (even index value):

-0.02981716699417872 += -0.0016001970884502903

Iteration 7, Last sum += 4 * (odd index value):

-0.03141736408262901 += -0.0024414687517919135

Iteration 8, Last sum += 2 * (even index value):

-0.03385883283442093 += -0.0009167601055671794

Iteration 9, Last sum += 4 * (odd index value):

-0.03477559293998811 += -0.0013558531585052085

Result = h/3 * -0.036131446098493315 = -0.0006021907683082219

h = 0.025

a, b = 0.5, 1

Adding f(start) + f(end): -0.00278493319694596 + -0.00024681960817335923

Iteration 1, Last sum += 4 * (odd index value):

-0.003031752805119319 += -0.010337851674141833

Iteration 2, Last sum += 2 * (even index value):

-0.013369604479261152 += -0.004766864750903002

Iteration 3, Last sum += 4 * (odd index value):

-0.018136469230164154 += -0.008740540525864197

Iteration 4, Last sum += 2 * (even index value):

-0.02687700975602835 += -0.003984460904118892

Iteration 5, Last sum += 4 * (odd index value):

-0.030861470660147244 += -0.007227199847366735

Iteration 6, Last sum += 2 * (even index value):

-0.03808867050751398 += -0.0032608184363878017

Iteration 7, Last sum += 4 * (odd index value):

-0.04134948894390178 += -0.005856683588648083

Iteration 8, Last sum += 2 * (even index value):

-0.04720617253254986 += -0.0026176142430043743

Iteration 9, Last sum += 4 * (odd index value):

-0.04982378677555424 += -0.004658836581969017

Iteration 10, Last sum += 2 * (even index value):

-0.054482623357523255 += -0.002063986333677265

Iteration 11, Last sum += 4 * (odd index value):

-0.05654660969120052 += -0.003642205924089868

Iteration 12, Last sum += 2 * (even index value):

-0.06018881561529039 += -0.0016001970884502903

Iteration 13, Last sum += 4 * (odd index value):

-0.06178901270374068 += -0.0028008471291111545

Iteration 14, Last sum += 2 * (even index value):

-0.06458985983285184 += -0.0012207343758959567

Iteration 15, Last sum += 4 * (odd index value):

-0.0658105942087478 += -0.002119879595169322

Iteration 16, Last sum += 2 * (even index value):

-0.06793047380391712 += -0.0009167601055671794

Iteration 17, Last sum += 4 * (odd index value):

-0.0688472339094843 += -0.001579737150538552

Iteration 18, Last sum += 2 * (even index value):

-0.07042697106002285 += -0.0006779265792526042

Iteration 19, Last sum += 4 * (odd index value):

-0.07110489763927545 += -0.0011592232604991958

Result = h/3 * -0.07226412089977464 = -0.000602201007498122

-0.000602201007498122000000132319

Process finished with exit code 0

Question 17a: https://github.com/Benny902/Numeric-Analysis-Hackaton/tree/main/q17

Method 1 : Bisection

interval (0, 0.1):

----

root : 0, iterations: 1

----

interval (0.1, 0.2):

guess 0.2 is not an approximated root

interval (0.2, 0.3):

guess 0.29999999999999993 is not an approximated root

interval (0.3, 0.4):

----

root : 0.33333435058593747, iterations: 16

----

interval (0.4, 0.5):

guess 0.5 is not an approximated root

interval (0.5, 0.6):

guess 0.5999999999999999 is not an approximated root

interval (0.6, 0.7):

guess 0.7 is not an approximated root

interval (0.7, 0.8):

guess 0.8 is not an approximated root

interval (0.8, 0.9):

guess 0.8999999999999999 is not an approximated root

interval (0.9, 1.0):

guess 1.0 is not an approximated root

interval (1.0, 1.1):

guess 1.1 is not an approximated root

interval (1.1, 1.2):

guess 1.1999999999999997 is not an approximated root

interval (1.2, 1.3):

guess 1.2999999999999998 is not an approximated root

interval (1.3, 1.4):

guess 1.4 is not an approximated root

interval (1.4, 1.5):

guess 1.5 is not an approximated root

[0, 0.33333435058593747]

Process finished with exit code 0

Method 3 : Secant :

interval (0, 0.1):

----

root : 0.0011948633229897886, iterations: 7

----

interval (0.1, 0.2):

interval (0.2, 0.3):

interval (0.3, 0.4):

----

root : 0.3333334441522644, iterations: 8

----

interval (0.4, 0.5):

interval (0.5, 0.6):

interval (0.6, 0.7):

interval (0.7, 0.8):

interval (0.8, 0.9):

interval (0.9, 1.0):

interval (1.0, 1.1):

interval (1.1, 1.2):

interval (1.2, 1.3):

interval (1.3, 1.4):

interval (1.4, 1.5):

[0.0011948633229897886, 0.3333334441522644]

Process finished with exit code 0

Question 17b: https://github.com/Benny902/Numeric-Analysis-Hackaton/tree/main/q17

Romberg method is using trapezoid method -

The formula of Trapezoidal is - sigma(from i=1 to N)*(h/2)*(f(Xi-h)+f(Xi))

The formula of Romberg Method is - R(n,m)=1/(4^m-1)*(4^m*R(n,m-1)-R*(n-1,m-1))

number iteration of trapezoid method: 1

Approximation: 0.00046908780310846935

number iteration of romberg method: 1

Approximation: 0.00019785535174425081

The result is -

0.00021008486834757877000000132321

Simpson method is going by the formula - (h/3)*(f(a)+2*sigma(from j=1 to last even)*f(X2j)+4*sigma(from j=1 to last odd)*f(X2j-1)+f(b))

h = 0.05

a, b = 0.5, 1

Adding f(start) + f(end): 0.0003978475995637085 + 0.00024681960817335923

Iteration 1, Last sum += 4 * (odd index value):

0.0006446672077370678 += 0.0018498281122907177

Iteration 2, Last sum += 2 * (even index value):

0.0024944953200277857 += 0.0009961152260297226

Iteration 3, Last sum += 4 * (odd index value):

0.003490610546057508 += 0.0020313295177497785

Iteration 4, Last sum += 2 * (even index value):

0.005521940063807287 += 0.0009928881611395899

Iteration 5, Last sum += 4 * (odd index value):

0.006514828224946877 += 0.0018763512124338774

Iteration 6, Last sum += 2 * (even index value):

0.008391179437380755 += 0.0008616445860886181

Iteration 7, Last sum += 4 * (odd index value):

0.009252824023469374 += 0.0015446026797050877

Iteration 8, Last sum += 2 * (even index value):

0.010797426703174462 += 0.0006776052954192198

Iteration 9, Last sum += 4 * (odd index value):

0.011475031998593682 += 0.0011666643456905276

Result = h/3 * 0.01264169634428421 = 0.0002106949390714035

h = 0.025

a, b = 0.5, 1

Adding f(start) + f(end): 0.0003978475995637085 + 0.00024681960817335923

Iteration 1, Last sum += 4 * (odd index value):

0.0006446672077370678 += 0.001735551740914323

Iteration 2, Last sum += 2 * (even index value):

0.002380218948651391 += 0.0009249140561453589

Iteration 3, Last sum += 4 * (odd index value):

0.00330513300479675 += 0.0019349288187027608

Iteration 4, Last sum += 2 * (even index value):

0.005240061823499511 += 0.0009961152260297226

Iteration 5, Last sum += 4 * (odd index value):

0.006236177049529233 += 0.002023615957262686

Iteration 6, Last sum += 2 * (even index value):

0.008259793006791919 += 0.0010156647588748893

Iteration 7, Last sum += 4 * (odd index value):

0.009275457765666809 += 0.002017848967517408

Iteration 8, Last sum += 2 * (even index value):

0.011293306733184216 += 0.0009928881611395899

Iteration 9, Last sum += 4 * (odd index value):

0.012286194894323806 += 0.001937746188960564

Iteration 10, Last sum += 2 * (even index value):

0.01422394108328437 += 0.0009381756062169387

Iteration 11, Last sum += 4 * (odd index value):

0.015162116689501308 += 0.001804083308194047

Iteration 12, Last sum += 2 * (even index value):

0.016966199997695355 += 0.0008616445860886181

Iteration 13, Last sum += 4 * (odd index value):

0.01782784458378397 += 0.0016361384219560203

Iteration 14, Last sum += 2 * (even index value):

0.019463983005739992 += 0.0007723013398525439

Iteration 15, Last sum += 4 * (odd index value):

0.020236284345592537 += 0.0014504439335369045

Iteration 16, Last sum += 2 * (even index value):

0.02168672827912944 += 0.0006776052954192198

Iteration 17, Last sum += 4 * (odd index value):

0.02236433357454866 += 0.0012602397493060363

Iteration 18, Last sum += 2 * (even index value):

0.023624573323854696 += 0.0005833321728452638

Iteration 19, Last sum += 4 * (odd index value):

0.02420790549669996 += 0.0010754239886558803

Result = h/3 * 0.025283329485355843 = 0.00021069441237796534

0.00021069441237796534000000132321

Process finished with exit code 0