**Introduction to Numerical Analysis Assignment 3**

**שאלה 1:** יהי

1. הבא הערכה עבור עבור בעזרת שיטת הטרפזים, עם קטעים באינטרוול .

נחלק את המקטע ל- קטעים. כלומר עבור הקטע ה- הוא מהצורה: , כאשר הוא המרחק בין כל שני מקטעים. וכעת:

כאשר:

נציב במקרה שלנו את

1. תחילה נמצא ביטוי עבור השגיאה: , נקח את הערך המתאים מהטבלה תוך שימוש בנוסחת השגיאה עבור אי זוגי:

*כעת נרצה לחפש את הערך המקסימלי של הפונק׳ הזו בקטע . תחילה נבדוק קצוות:*

*נגזור ונבדוק האם הפונ׳ עולה או יורדת בקטע:*

*כעת קל לראות כי הפונקצייה מתאפסת רק ב , ועבור אז היא שלילית ולכן יורדת. לכן הפונ׳ מקבלת ערך מקסימלי ב-.*

*כעת נציב ונקבל:*

רוצים

לכן ה-m המינימלי הוא 13.

מריצה עם פייתון נקבל כי האינטגרל עם m=13 יהיה:

0.21702861600351894.

הקוד מצורף למטה

1. ראינו בסעיף ב׳ שהנגזרת השנייה , כלומר הפונקציה שלנו היא קמורה בקטע.

לכן, מתכונות של קמירות, כל ישר שנעביר בין 2 נק׳ על הפונקציה בקטע יהיה מעליה ולכן עבור שיטת הטרפזים המורכבת תחשב קירוב לאינטגרל שיהיה גדול משטח האינטגרל האמיתי בקטע (מכיוון שסכום הטרפזים יהיה עם החלקים שמעל הפונקציה בקטע).

לכן, מתקיים כי לכל .

**שאלה 2:**

1. תחילה נריץ את האלגוריתמים עבור שיטת אוילר, שיטת RK2 והפונקציה המדוייקת:

שיטת אוילר:

Text

Description automatically generated with medium confidence

שיטת RK2:

Text

Description automatically generated

פונקציה מדוייקת:

A black background with white text

Description automatically generated with medium confidence

כעת עבור השגיאה נחסר בין התוצאה של הפונקציה המדויקת לתוצאות עבור אוילר וRK2 בכל שלב:

Text, table

Description automatically generated

1. כעת נחסר את התוצאות של RK2 מהתוצאות של אוילר:

Text, calendar

Description automatically generated

1. ניתן לראות את כמות האיווליואציות בתמונות לעיל
2. קוראים ל-f 20 פעמים עם h=0.05:

A black screen with white text

Description automatically generated with low confidence

1. השוואת שגיאות עבור אותן כמות איווליואציות:

Text

Description automatically generated with medium confidence

קוד הפייתון:

import math  
  
  
def calculate\_integral\_num\_with\_m(m):  
 ln2 = math.log(2) # base for math.log is e  
 two\_m = 2 \* m  
 ln2\_div\_2m = ln2 / two\_m  
 result = ln2\_div\_2m  
 acc = 0  
 for i in range(1, m - 1):  
 i\_div\_m = i / m  
 i\_div\_m = i\_div\_m \*\* 2  
 acc += math.log(1 + i\_div\_m)  
 result += acc / m  
 return result  
  
  
def euler\_method(f, h, base\_case, num\_of\_iterations):  
 *'''* ***:param*** *num\_of\_iterations:* ***:param*** *f:* ***:param*** *h:* ***:param*** *base\_case: tuple such that y(base\_case[0]) = base\_case[1]* ***:return****:  
 '''* x = base\_case[0] # 0  
 y = base\_case[1] # 1  
 evaluation = 0  
 for \_ in range(num\_of\_iterations):  
 evaluation += 1  
 y = y + h \* f(x, y)  
 x += h  
 return x, y, evaluation  
  
  
def runge\_kutta\_2(f, lamda, h, base\_case, num\_of\_iterations):  
 *'''* ***:param*** *f:* ***:param*** *lamda:* ***:param*** *h:* ***:param*** *base\_case:* ***:param*** *num\_of\_iterations:* ***:return****:  
 '''* x = base\_case[0]  
 y = base\_case[1]  
 two\_times\_lamda = 2 \* lamda  
 half\_time\_lamda = (1 / two\_times\_lamda)  
 alpha2 = half\_time\_lamda  
 alpha1 = 1 - alpha2  
 h\_times\_lamda = h \* lamda  
 evaluation = 0  
 for \_ in range(num\_of\_iterations):  
 evaluation += 2  
 k1 = f(x, y)  
 k2 = f(x + h\_times\_lamda, y + h\_times\_lamda \* k1)  
 y = y + h \* (alpha1 \* k1 + alpha2 \* k2)  
 x += h  
  
 return x, y, evaluation  
  
  
def exact\_function(f, x):  
 return f(x)  
  
  
def find\_minimal\_for\_epsilon(f, epsilon):  
 i = 0  
 curr = float('inf')  
 while curr > epsilon:  
 i += 1  
 curr = f(i)  
 return i  
  
  
def print\_rk2\_euler():  
 def f(x, y):  
 return x + y  
  
 def exact(x):  
 e = math.e  
 return x, 2 \* (e \*\* x) - x - 1  
  
 euler\_results = []  
 rk\_results = []  
 exact\_results = []  
 euler\_better\_results = []  
 h = 0.1  
 base\_case = (0, 1)  
 lamda = 2 / 3  
 print("Eulers Method h=0.1:")  
 for i in range(11):  
 x, y, evalutaion = euler\_method(f, h, base\_case, i)  
 euler\_results.append(y)  
 print("x={:.1f},\t\ty={:.8f},\t\tnum of evaluations={}".format(x, y, evalutaion))  
  
 print("Eulers Method h=0.05:")  
 for i in range(21):  
 x, y, evalutaion = euler\_method(f, 0.05, base\_case, i)  
 if i % 2 == 0:  
 euler\_better\_results.append(y)  
 print("x={:.1f},\t\ty={:.8f},\t\tnum of evaluations={}".format(x, y, evalutaion))  
  
 print("RK2 method:")  
 for i in range(11):  
 x, y, evalutaion = runge\_kutta\_2(f, lamda, h, base\_case, i)  
 rk\_results.append(y)  
 print("x={:.1f},\t\ty={:.8f},\t\tnum of evaluations={}".format(x, y, evalutaion))  
  
 print("Exact function 2\*e\*\*x - x - 1")  
 for i in range(11):  
 x, y = exact\_function(exact, h \* i)  
 exact\_results.append(y)  
 print("x={:.1f},\t\ty={:.8f}".format(x, y))  
  
 print("\t\teuler error\t\t\trk2 error")  
 for i in range(len(exact\_results)):  
 euler = euler\_results[i]  
 rk = rk\_results[i]  
 exact\_result = exact\_results[i]  
 print("x={:.1f}\t{:.8f}\t\t\t{:.8f}".format(0.1 \* i, exact\_result - euler, exact\_result - rk))  
  
 print("\t\tRK2 - euler")  
 for i in range(len(exact\_results)):  
 euler = euler\_results[i]  
 rk = rk\_results[i]  
 print("x={:.1f}\t{:.8f}".format(0.1 \* i, rk - euler))  
  
 print("\t\teuler h=0.05 error\trk2 error")  
 for i in range(len(exact\_results)):  
 euler = euler\_better\_results[i]  
 rk = rk\_results[i]  
 exact\_result = exact\_results[i]  
 print("x={:.1f}\t{:.8f}\t\t\t{:.8f}".format(0.1 \* i, exact\_result - euler, exact\_result - rk))  
  
  
def print\_integral\_m():  
 m = 13  
 print(calculate\_integral\_num\_with\_m(m))  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 print\_integral\_m()  
 print\_rk2\_euler()