**Introduction to Numerical Analysis Assignment 2**

**שאלה 2:**

נגדיר כאשר הוא שורש מכיוון ש:

כעת נבדוק את סדר ההתכנסות ע״י מציאת מינימלי כך ש- .

תחילה נסדר בצורה נוחה יותר את הביטוי:

כעת נגזור פעם ראשונה:

נציב את :

נגזור פעם שנייה:

נציב :

נגזור פעם שלישית:

נציב :

ולכן סדר ההתכנסות הוא .

**שאלה 3:**

1. נתונה הפונקציה .

נכתוב את איטרציית ניוטון עבור פונקציה זו.

נחשב את הנגזרת כדי לבנות את הפונקצייה לאירטציית ניוטון:

כפי שלמדנו בכיתה איטרציית ניוטון היא:

במקרה שלנו:

1. ע״מ למצוא את תחום ההתכנסות נגדיר את פונקציית האיטרצייה לפי מה שמצאנו בסעיף א׳:

נבדוק את התנאים הדרושים:

1. פונקציית כיווץ:

כלומר כרגע התחום הוא .

1. כעת נבדוק כי לכל בתחום, גם בתחום.

יהי :

נשים לב כי זה מתקיים תמיד מכיוון שאין ל- אפשרות להיות מכיוון שבדטרמיננטה של המשוואה הריבועית אז אין פתרון למשוואה הריבועית עבור שוויון ל0, וגם היא רציפה וחיובית בנקודה : אזי הביטוי לכל בתחום.

יהי :

נשים לב כי זה מתקיים תמיד מכיוון שאין ל- אפשרות להיות מכיוון שבדטרמיננטה של המשוואה הריבועית אז אין פתרון למשוואה הריבועית עבור שוויון ל0, וגם היא רציפה וחיובית בנקודה : אזי הביטוי לכל בתחום.

1. רציפות: קל לראות כי רציפה.

סה״כ הראנו כי לכל מתקיימים תנאי משפט 4 ולכן שיטת ניוטון תתכנס לשורש כנדרש.

שאלה 4:

1. תהי פונ׳ גזירה וקמורה ממש ב- ו- לכל x. נראה כי ל- שורש יחיד.

נניח בשלילה כי קיימים שני שורשים: . אזי . נניח בה״כ ולכן גם . הפונ׳ קמורה ממש ולכן מההגדרה:

סתירה!

1. יהי שורש ו- ניחוש התחלתי של איטרציית ניוטון רפסון.

אם סיימנו.

אחרת, נראה כי סדרת הערכים של איטרציית ניוטון רפסון מונוטונית וחסומה, ולכן נסיק כי היא מתכנסת. לאחר מכן נראה כי ההתכנסות היא לשורש.

יהי סדרת הנקודות המתקבלת ע״י איטרציית ניוטון.

נראה כי ואז נסיק כי הסדרה היא מונוט

חסימות: נראה כי :

מהגדרת פונ׳ קמורה ממש:

נציב

הראנו כי לכל לכן לכל .

מונוטוניות:נראה כי לכל .

נסמן את האיבר הבא באיטרציית ניוטון:

הראנו חסימות מלמטה ע״י לכל ו- מונ׳ עולה לכן .

כיוון שהנגזרת חיובית נקבל כי .

לכן סה״כ:

כנדרש.

התכנסות לשורש:

הראנו כי לכן סדרת הנקודות מונוטונית יורדת. כמו כן הראנו כי לכל לכן הסדרה חסומה מלמטה לכן מתכנסת. נסתכל על גבול הסדרה:

כעת נציב את הגבול בהגדת :

לכן הסדרה מתכנסת לשורש הפונקציה כנדרש.

1. עבור היינו מסיקים כי מונוטונית יורדת.

* בהוכחת חסימות היינו מצליחים לחסום את מלמעלה ע״י השורש (יהיה שינוי כיוון באי שוויון בעת החלוקה במספר שלילי במקום חיובי).
* בהוכחת מונוטוניות תשתנה המסקנה כי מחסירים את , אזי המכנה שלילי (כי הפונקציה מונוטונית יורדת), והמונה חיובי מכיוון שבחסימות נראה כי וגם הפונ׳ היא מונוטונית יורדת אזי כלומר המונה חיובי, אזי בכל איטרציה מחסירים מ- איבר שלילי כלומר מגדילים אותו. לכן יתקיים .
* בהוכחת התכנסות לשורש אין שינוי.

1. אם היה נתון כי קעורה ממש בכל התחום אז השינוי היחיד היה בהוכחת החסימות, ושם היינו מוכיחים כי לכל מכיוון שהיה מתשנה הכיוון של האי שוויון שהתבססנו עליו.

שאלה 1:

from \_\_future\_\_ import absolute\_import, print\_function, division  
import matplotlib.pyplot as plt  
import math  
  
  
def bisection\_method(f, a, b, tolerance):  
 *"""* ***:param*** *f: function to determine the find the root of it* ***:param*** *a: left limit of the domain* ***:param*** *b: right limit of the domain* ***:param*** *tolerance: the tolerance which should suffice |z-a'|<tolerance where a' is the root and z is the returned value* ***:return****: z s.t |z-a'|<tolerance, # of iterations to find the root  
 """* i = 0  
 fb = f(b)  
 fa = f(a)  
 if fa \* fb > 0:  
 raise ValueError("f(a)\*f(b) should be negative but is {0}".format(fa \* fb))  
 elif fa == 0:  
 return a, i  
 elif fb == 0:  
 return b, i  
  
 limit = 2 \* tolerance  
 while math.fabs(a - b) >= limit:  
 i += 1  
 z = (a + b) / 2  
 fz = f(z)  
 if fz == 0:  
 return z, i  
 fa = f(a)  
 if fa \* fz < 0:  
 b = z  
 else:  
 a = z  
  
 return (a + b) / 2, i  
  
  
def regula\_falsi(f, x0, x1, tolerance, goal):  
 *"""* ***:param*** *f: function to determine the find the root of it* ***:param*** *x0: initial guess s.t f(x0)\*f(x1) < 0* ***:param*** *x1: initital guess s.t f(x0)\*f(x1) < 0* ***:param*** *tolerance: the tolerance to be clos to the root* ***:param*** *goal: the goal to return* ***:return****: x s.t |f(x)| < tolerance  
 """* fx0 = f(x0)  
 fx1 = f(x1)  
 if fx0 == 0:  
 return x0, 0  
 elif fx1 == 0:  
 return x1, 0  
 elif fx0 \* fx1 > 0:  
 raise ValueError("f(x0)\*f(x1) should be negative but is {0}".format(fx0 \* fx1))  
 i = 0  
 xi\_minus\_1 = x1  
 xi\_minus\_2 = x0  
 xi = float('inf')  
 while math.fabs(xi - goal) >= tolerance:  
 i += 1  
 fxi\_minus\_1 = f(xi\_minus\_1)  
 delta\_f = fxi\_minus\_1 - f(xi\_minus\_2)  
 delta\_xi = xi\_minus\_1 - xi\_minus\_2  
 xi = xi\_minus\_1 - fxi\_minus\_1 \* delta\_xi / delta\_f  
 fxi = f(xi)  
 if fxi == 0:  
 return xi, i  
 if fxi \* fxi\_minus\_1 < 0:  
 xi\_minus\_2 = xi  
 else:  
 xi\_minus\_1 = xi  
  
 return xi, i  
  
  
def f(x):  
 return x \*\* 2 - 0.2 \* x - 3  
  
  
def main():  
 #start a  
 x0 = -1  
 x1 = 4  
 tolerance = 10 \*\* -8  
 bisection\_result, bisection\_iter\_num = bisection\_method(f, x0, x1, tolerance)  
 print("got result {0} in bisection method with {1} iterations.".format(bisection\_result, bisection\_iter\_num))  
 #finish a  
 #start b  
 tolerance\_axis = []  
 bisection\_iterations\_axis = []  
 falsi\_iterations\_axis = []  
 for d in range(1, 6):  
 d = -1 \* d  
 tolerance\_axis.append(d)  
 tolerance = 10 \*\* d  
 bisection, bisection\_iter = bisection\_method(f, x0, x1, tolerance)  
 bisection\_iterations\_axis.append(bisection\_iter)  
 falsi\_result, falsi\_iter = regula\_falsi(f, x0, x1, tolerance, bisection\_result)  
 falsi\_iterations\_axis.append(falsi\_iter)  
 plt.plot(tolerance\_axis, bisection\_iterations\_axis, color='orange', marker="o", label='Bisection Method')  
 plt.plot(tolerance\_axis, falsi\_iterations\_axis, color='blue', marker="o", label='Regula Falsi')  
 plt.xlabel("d")  
 plt.ylabel("num of iterations")  
 plt.title("Regula Falsi and Bisection methods iterations per 10\*\*d comparison")  
 plt.legend()  
 plt.show()  
 #finish b  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()