דוח סיכום אנליזה נומרית

קבוצה מס': 3

חברי הקבוצה הנוכחים: איתי כלף ,בר-אילן קימברובסקי ,שי מנשרוב ,אריאל גואטה,יואב יונג.

1**.הגדרת הבעיה** –אחד השימושים העיקריים של ניטור קרינה באוויר הוא במיפוי.

בשל מגבלות מתמטיות ,ניטור אווירי(ניתוח איכות האוויר) של שדה רדיו אקטיבי אינו יכול לספק התפלגות תלת-ממדית של הזיהום בענן הרדיואקטיבי. עם זאת ניתן להשיג הערכה מדויקת יותר של התפלגות זיהום דו ממדית על הקרקע ,ניתן להשיג זאת ע"י מדידת שדה רדיואקטיבי מגובה קבוע של 100-500 מטר. יש להשתמש בשתי גישות שתלויות בעצמן על מנת להשיג את הפתרון לבעיה זו:

* המטריצה האלמנטרית
* יישום המבוסס על משוואת שטיין

את הנתונים אנחנו אוספים ע"י מסוק בעל גלאי קרינת גמא. המסוק נמצא בגובה קבוע של 200 מטר מעל פני הקרקע ,המסוק מרחף תמיד מעל לנקודת האמצע של כל ריבוע ושם הוא מבצע את המדידה.

עוצמת הקרינה היא: 

מקרא:

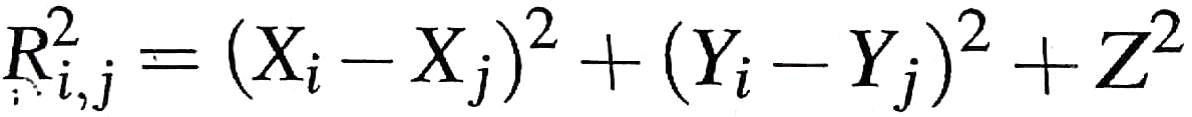
* C הינו מקדם פרופורציה של המוניטור
* k פקטור הקרינה באוויר
* μ מקדם דילול הקרינה שבאוויר
* R מרחק במטרים

האזור של הדגימה הוא ריבועי N\*N ,המסוק (הגלאי) נמצא בגובה H מעל פני הקרקע(אזור הדגימה).

המדידה אשר מתקבלת ע"י הגלאי היא סכום התרומות מכל נקודות השטח המזוהמות , מעוצבים בצורת 2 N

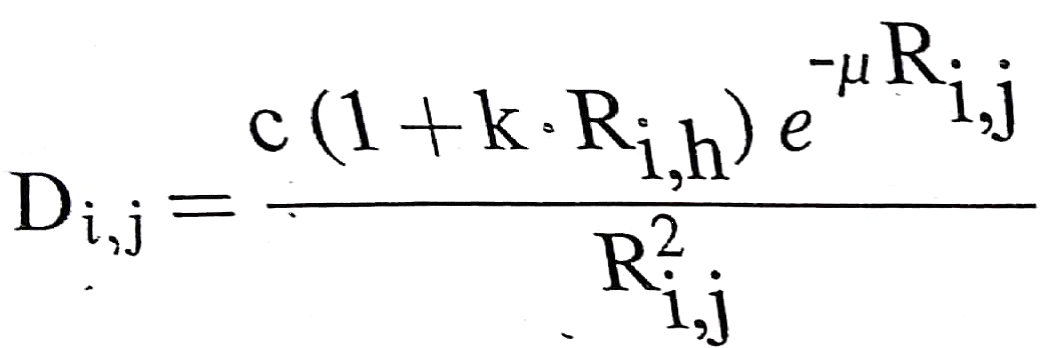
אשר הינם מקורות של נקודה אחת שממוקמת במרכז של רשת התאים.

המרחק מהגלאי(המסוק) מעל נקודה i לכל נקודה מותנית j על הרשת ניתן ע"י הנוסחה הבאה:



X,Y,Z הינם הקורדינטות הקרטזיות בנוסחה הנ"ל ,אשר נמדדות במטרים.

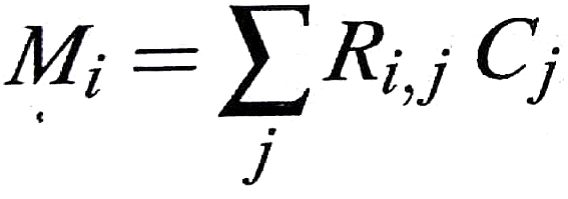
ניקח את משוואה 2 ונכניס אותה אל תוך המשוואה ה-1 ונקבל:



אשר מספקת לנו את פונקציית התגובה של הגלאי מתא עם אינדקס j כאשר החיישן ממוקם בגובה h בדיוק מעל התא באינדקס i .

אנחנו מציינים את כמות הזיהום המרוכז בנקודה J כ-Cj , ליהיות שווה ערך לזיהום הכולל בתוך ריבוע אחד.

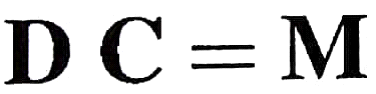
הגלאי קורא מעל תא i אשר מתואר באמצעות המשוואה הבאה:



j הינו סכימה של כל נקודות הרשת.

אשר מפיקה 2 N משוואות קוויות שמחברות את הזיהום הלא ידוע בתא j (Cj ) עם המדידה של Mi .

בסימון המטריצה:



הפתרון שלה הוא:



מקרא:

D - הינו מקדם פרופורציה של המוניטור

C - וקטור לא ידוע של עוצמות

M – וקטור הערכים הנמדדים

פתרון המטריצה ממשואה 1.6 מספק שיטה כללית ועקבית לחישוב התפלגות שדה הזיהום ממדידות קרינה אשר נעשות ע"י המסוק. ישנה בעיתיות עם שיטה זו ,היא ניתנת לפתרון רק עבור פרמטרים מסוימים היא קיימת כאשר המטריצה ההפוכה ל-D קיימת. כך שהיחס המוצג במשוואה 7 מסופק.



הערכים המספרים של מרכיבי המטריצה D (משוואה 1.3( משתנים על פני טווח של מספר סדרי גודל בתוך המטריצה , כך שגם אם ההפיכה של מטריצה D קיימת זה עלול להיות בלתי מותנה. מה שהופך את המטריצה ההופכית לבלתי משתנת ומעקבת את פתרון משוואה 1.6 . ניתן לראות ממשואה 1.3 וממשואה 1.4 כאשר המטריצה תלויה בשלושה פרמטרים:

תגובת הגלאי ,גובה המדידות R ואת גודל הרשת הלא תלותית של רמת הזיהום של השדה.

חישוב מטריצה ההפיכה D מאפשר לנו למפות את המרחב של הפרמטרים הדו מימדיים ולמצוא את האזור אשר מספקת משוואה 1.6 . אנו יכולים למצוא את הפתרון של כל רמת זיהום על אזור זה. חישוב שדה הזיהום דורש דיסקרטיזציה של התפלגות הזיהום המתמשך למספר סופי של מקורות בנקודה. כאשר זיהום של כל ריבוע מתרכז בנקודה אחת במרכזו המדויק. אלו הנעלמים של משוואה 1.6 ויש להם את אותו מספר נקודות כמו הנקודות שנמדדו. הם מחושבים ע"י מדידות של שדה מתמשך שנלקח מלמעלה בכל נקודה של הרשת , והם נפתרים באמצעות משוואה 1.6 . בפועל קריאות שנלקחו ממסוק עשויות לשבש את התפלגות החלקיקים המזוהמים באזור ,כאשר הם מתבצעים בטווח מרחק מסוים כלומר זרם האוויר שנוצר מלהבי המסוק העלול לגרום להשהיה רדיואקטיבית של החלקיקים אם המסוק מגיע קרוב מידי. אם המסוק מרחף בגובה גדול מידי ,קריאות בודדות מן התאים יגרום להתכנסות וביטול מה שיגרום להבדל רמות מקור הפעילות. בנוסף הגדלת ממדי הרשת כדי לשפר את הדיוק יגרום לקריאות מוגברות בשל ההשפעה המצטברת של מקורות הצבע.

שיקולים בפיתוח התוכנה:

אחת הבעיות העיקריות בפיתוח תוכנה זה המוטיבציה לחדד את הפיתוח של החלק המדעי של התוכנה.

בעיקר את תהליך ההדמיה מורכב, גדול, מבלבל רגיש מאוד לשינויים בפרמטרים ויקר.

התוכנה שצריכים לפתח במטרה למצוא ריכוז זיהום דו ממדי מבוסס על קריאת נתונים של מסוק,

זהו פרמטר מאוד רגיש לשינויים, בנוסף העובדה שטעויות עלולות לצוץ ממגוון של מקורות: חוסר דיוק להבין את התהליך, פגמים במודל המתמטי בהפשטה של התהליך, קשיים בפירוש המודל המתמטי לצורה הניתנת לחישוב, בעיה בתוכנה, פגמים לוגים וחוסר דיוק בחישוב אריתמטי ובכך אתה תקבל מערכת מאוד רגישה ללא הוכחת תהליך.

הבעיה עצמה:

הבעיה העיקרית היא בניטור של קרינת שדה רדיו אקטיבי כאשר הנתונים בתמונה התלת מימדית לא ברורים ויותר נוח לתת תוצאה בתמונה דו מימדית, אך עם זאת עדיין קיימים כשלים במתן התוצאה והניטור. מכיוון שהניטור מתבצע במסוק מהאוויר (גבהים של 100 עד 500 מטרים מעל פני הקרקע) יש חוסר דיוק בחישוב, בעיות מתמטיות בתוכנה, כשלים לוגים ובנוסף מחסור בתקציב.

**2.השיטה והצגת הכלים לפתרון סעיף ב':**

**שיטת המיתר:**

def secant(f,a,b,n):  
 for i in range(0,n): # ריצה בלולאה N איארציות  
 if f(a)-f(b) == 0: # תנאי עצירה הכרחי  
 return a  
 res = (a - (f(a)\*(b-a)\*1.0)/(f(b)-f(a))) # הערך הבא באיטרציה  
 a = b  
 b = res  
 print(b)  
 return b  
  
print(secant(lambda x: 16\*x\*\*3 - 16\*x\*\*2 + 1, -0.226, 0, 10))  
print(secant(lambda x: 16\*x\*\*3 - 16\*x\*\*2 + 1, 0.298, 0, 10))  
print(secant(lambda x: 16\*x\*\*3 - 16\*x\*\*2 + 1, 0.927, 0, 10))

לפתרון הבעיה השתמשנו בשיטת המיתר בקוד בפייטון אשר חברי הקבוצה כתבו יחדיו.

לבדיקה לקחנו שני ערכים התחלתיים שקרובים יחסית לשורש , בנינו ישר שעובר דרך הנקודות ובכל

איטרציה חדשה קידמנו אותו להיות החיתוך של הישר.

ככל שנחשב יותר איטרציות כך נקבל ערך קרוב יותר לערך השורש האמיתי.

הקוד בנוי כך ששלחנו לפונקציה שלנו (פונקציה, 2 נקודות על הפונקציה , ומספר איטרציות לריצה),

רצנו על הפונקציה מ-0 ועד מספר האיטרציות לריצה שרצינו ובדקנו שעם הנקודה הראשונה

בפונקציה פחות הנקודה השנייה בפונקציה יתנו לנו 0 הפונקציה תחזיר לנו את הנקודה הראשונה.

במקרה שלא, ניישם את התוצאה במשתנה שיצרנו ע"י הנוסחא של שיטת היתר, ובכל איטרציה

נחליף את הנקודה הראשונה להיות השנייה ואת הנקודה השנייה להיות התוצאה של המשתנה

שיצרנו ונדפיס בכל פעם את התוצאה.

וכך נרוץ עד שנגיע למספר האיטרציות שקיבלנו בפונקציה, ובסופו של דבר שנסיים את כל

האיטרציות נחזיר את הנקודה השנייה.

לבדיקת תוצאות הקוד לקחנו את התוצאות שקיבלנו והצבנו אותם בפונקציה הנתונה וראינו

שהתוצאות הם בקירוב ל0.

**שיטת החצייה :**

import math  
  
def splithalf\_method(a, b,func, tol):#function gets a and b is intervals , polynom and tolerance  
 i=0  
 if func(a) \* func(b) <= 0:#If it is a continuous function in the interval, we will print an error  
 print("error")  
 middle = a #turn the middle to a value  
 while math.fabs(a-b) >= tol:#turn a-b to absolute number and check if its bigger or equal to tolerance  
 middle = (a + b) / 2  
 if func(middle) == 0:  
 return middle  
 if func(a) \* func(middle) < 0:  
 b = middle  
 else:  
 a = middle  
 i+=1  
 print("Operation number:",i,"\n","Middle:",middle)  
 return middle  
  
  
y = lambda x:16\*(x\*\*3)-16(x\*\*2)+1 #create polynom function  
print(splithalf\_method(-0.226, 0, y, 0.01))  
print(splithalf\_method(0.298, 0, y, 0.01))  
print(splithalf\_method(0.927, 0, y, 0.01))

לפתרון הבעיה השתמשנו בשיטת החצייה בקוד בפייטון אשר חברי הקבוצה כתבו יחדיו.

שיטת החצייה היא אלגוריתם למציאת שורש של פונקציה שעושה שימוש בחלוקת המרווח לשורש

ובוחר מרווח קטן יותר שהשורש נמצא שם.

הפונקציה שלנו מקבלת קטע לבדיקה בתחום של הפונקציה הנדרשת לבדיקה, רמת דיוק נדרשת,

ואת הפונקציה עצמה.

נבדוק אם התנאי מתקיים שכאשר הפונקציה בנקודה א' כפול הפונקציה בנקודה ב' קטנה מ0

אם לא הפונקציה תסיים ותחזיר שגיאה.

אם התנאי מתקיים נמשיך ונבדוק את ערך הפונקציה בנקודה חדשה שניצור בין נקודה א' וב'.

אם הנקודה החדשה קטנה או שווה לרמת הדיוק הנדרשת הפונקציה תסיים ותחזיר תוצאה.

אם לא נחצה את הפונקציה לשתי חלקים ונצור נקודות חדשות לאמצע כל קטע ונחזור על הבדיקה

חלילה.

לבדיקת תוצאות הקוד לקחנו את התוצאות שקיבלנו והצבנו אותם בפונקציה הנתונה וראינו

שהתוצאות הם בקירוב ל0.

**שיטת גאוס-זיידל :**

def gaussSeidel(A, b, x, N, tol):  
 max = tol  
 for i in range(max):  
 for j in range(N):  
 summ = 0.0  
 for k in range(N):  
 if (k != j):  
 summ = summ + A[j][k] \* x[k]  
 x[j] = (b[j] - summ) / A[j][j]  
 print("[", end="")  
 for j in range(N):  
 print(x[j], ",", end="")  
 print("]\n")  
  
  
gaussSeidel([[1, 1, 1], [1, 3, 9], [1, 5, 25]], [10.5, 6.1, 3.5], [0.0, 0.0, 0.0], 3, 30)

למציאת וקטור המקדמים השתמשנו בשיטת גאוס זיידל בקוד בפייטון אשר נכתב ע"י חברי הקבוצה.

הסבר הקוד :

פרמטרים : מטריצה אשר כוללת אלכסון דומיננטי.

וקטור תוצאה.

וקטור המקדמים שמתקבל לאחר ביצוע הקוד.

גודל המטריצה (ריבועי).

רמת דיוק (מספר איטרציות).

שיטת גאוס-זיידל היא שיטה איטרטיבית לפתרון מערכת משוואות ליניאריות ודומה לשיטת יעקובי.

השיטה הנ"ל היא טכניקה לפתרון מטריצה ריבועית.

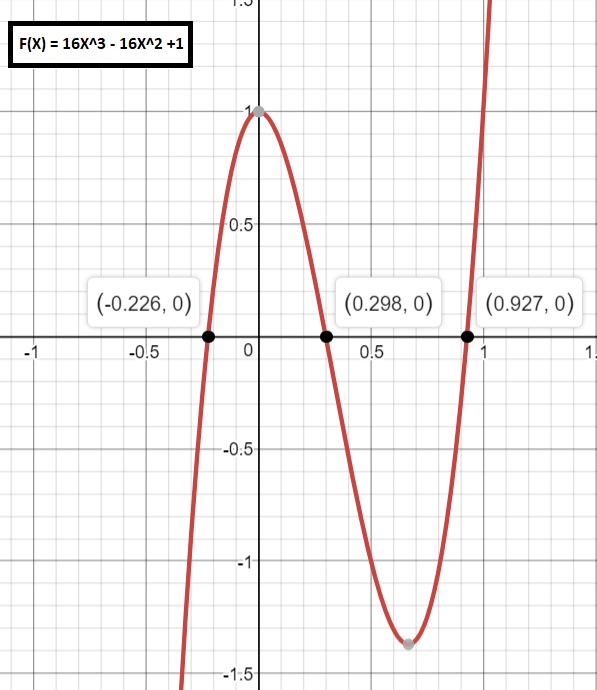
שיטה איטרטיבית כזו או אחרת תתכנס לתוצאה אם המטריצה היא בעלת אלכסון דומיננטי.

לאחר שקיבלנו את וקטור המקדמים, נציב בנוסחא :

Ax^2 + bx + c

**3. הצגת נתונים – סעיף ג':**

גרף הפונקציה :



.(-0.226,0) , (0.298,0) , (0.927,0) קיבלנו 3 שורשים –

**הפתרונות שקיבלנו עבור השורשים:**

**(-0.226,0)עבור השורש הראשון**

-0.22556987974418571

-0.22607931589982802

-0.2258028027739527

-0.2258029813409571

-0.22580298147788838

-0.22580298147788833

-0.22580298147788833

-0.22580298147788833

)**0.298,0)עבור השורש השני**

0.2987628826555001

0.29832419708785524

0.2984841274975575

0.298484141619375

0.2984841416186576

0.2984841416186576

0.2984841416186576

)**0.927,0)עבור השורש השלישי**

0.9235861742844047

0.8855858133823167

0.9277375638638898

0.9272722755591011

0.9273187919785912

0.9273188398647102

0.9273188398592308

0.9273188398592307

0.9273188398592307

0.9273188398592307

**המומצע החשבוני של השורשים החיוביים של המשוואה:**

בשורש הראשון התוצאות יהיו שליליות לכן ניקח, את שני השורשים השניים ונחשב ביניהם.

נעזרנו בקוד הקודם וחישבנו את הממוצע.

def secant(f,a,b,n):  
 sum = 0  
 times = 0  
 for i in range(0,n): # ריצה בלולאה N איארציות  
 if f(a)-f(b) == 0: # תנאי עצירה הכרחי  
 return a  
 res = (a - (f(a)\*(b-a)\*1.0)/(f(b)-f(a))) # הערך הבא באיטרציה  
 a = b  
 b = res  
 if (b > 0):  
 sum = sum + b  
 times += 1  
 return sum/times  
  
  
print(secant(lambda x: 16\*x\*\*3 - 16\*x\*\*2 + 1, 0.298, 0, 10))  
print(secant(lambda x: 16\*x\*\*3 - 16\*x\*\*2 + 1, 0.927, 0, 10))

קיבלנו ממוצע בשורש הראשון ובשורש השני :

0.2984841416186576

0.9273188398592307

ומכאן אחרי שנעזרנו בפונקציה לקחנו את 2 התוצאות שיצאנו לנו ועשינו ממוצא ביניהם והתוצאה

שנקבל עבור כל השורשים החיוביים הוא : 0.6129014907389442

חישבנו את התוצאה ומכאן נקבל כי :

С = 0.6129014907389442

**קירוב הפתרונות לפי שיטת גאוס-זיידל :**

[10.5 ,0.0 ,0.0 ,]

[10.5 ,-1.4666666666666668 ,0.013333333333333357 ,]

[11.953333333333333 ,-1.9911111111111115 ,0.060088888888888976 ,]

[12.431022222222222 ,-2.2906074074074074 ,0.1008805925925926 ,]

[12.689726814814815 ,-2.4992173827160493 ,0.13225440395061724 ,]

[12.866962978765432 ,-2.6524175381069957 ,0.15580498847078184 ,]

[12.996612549636215 ,-2.766285815291084 ,0.1733926610727682 ,]

[13.092893154218316 ,-2.8511423679577437 ,0.18651274742281607 ,]

[13.164629620534928 ,-2.9144147824467574 ,0.19629777166795434 ,]

[13.218117010778803 ,-2.961598985263464 ,0.2035951166215407 ,]

[13.258003868641923 ,-2.99678663941193 ,0.20903717313670903 ,]

[13.287749466275221 ,-3.0230280081685343 ,0.21309562298269802 ,]

[13.309932385185837 ,-3.04259766401004 ,0.21612223739457456 ,]

[13.326475426615465 ,-3.0571918543888787 ,0.21837935381315718 ,]

[13.338812500575722 ,-3.0680755616313786 ,0.22006261230324684 ,]

[13.348012949328131 ,-3.076192153352451 ,0.22131791269736495 ,]

[13.354874240655086 ,-3.08224515164379 ,0.22225406070255452 ,]

[13.359991090941236 ,-3.086759212421409 ,0.2229521988466324 ,]

[13.363807013574776 ,-3.090125601064823 ,0.22347283966997353 ,]

[13.36665276139485 ,-3.092636106141537 ,0.22386111077251342 ,]

[13.368774995369023 ,-3.094508330773881 ,0.22415066634001526 ,]

[13.370357664433865 ,-3.095904553831334 ,0.22436660418891222 ,]

[13.371537949642422 ,-3.0969457957808775 ,0.22452764117047863 ,]

[13.372418154610399 ,-3.097722308381569 ,0.2246477354918978 ,]

[13.37307457288967 ,-3.098301397438917 ,0.22473729657219657 ,]

[13.37356410086672 ,-3.0987332566721633 ,0.2248040872997639 ,]

[13.3739291693724 ,-3.099055318356758 ,0.22485389689645566 ,]

[13.374201421460302 ,-3.0992954978428013 ,0.22489104271014818 ,]

[13.374404455132654 ,-3.0994746131746624 ,0.22491874442962634 ,]

[13.374555868745036 ,-3.0996081895372245 ,0.2249394031576435 ,]

[13.37466878637958 ,-3.0997078049327906 ,0.22495480953137495 ,]

A = 0.2249548095313749

B = -3.099707804932790

C = 13.37466878637958

כעת נציב את המקדמים בנוסחא :

Ax^2 + bx + c 🡺 13.37466878637958\*x^2 -3.099707804932790\*x + 0.2249548095313749 = f(4.74)

חישבנו את התוצאה ומכאן נקבל :

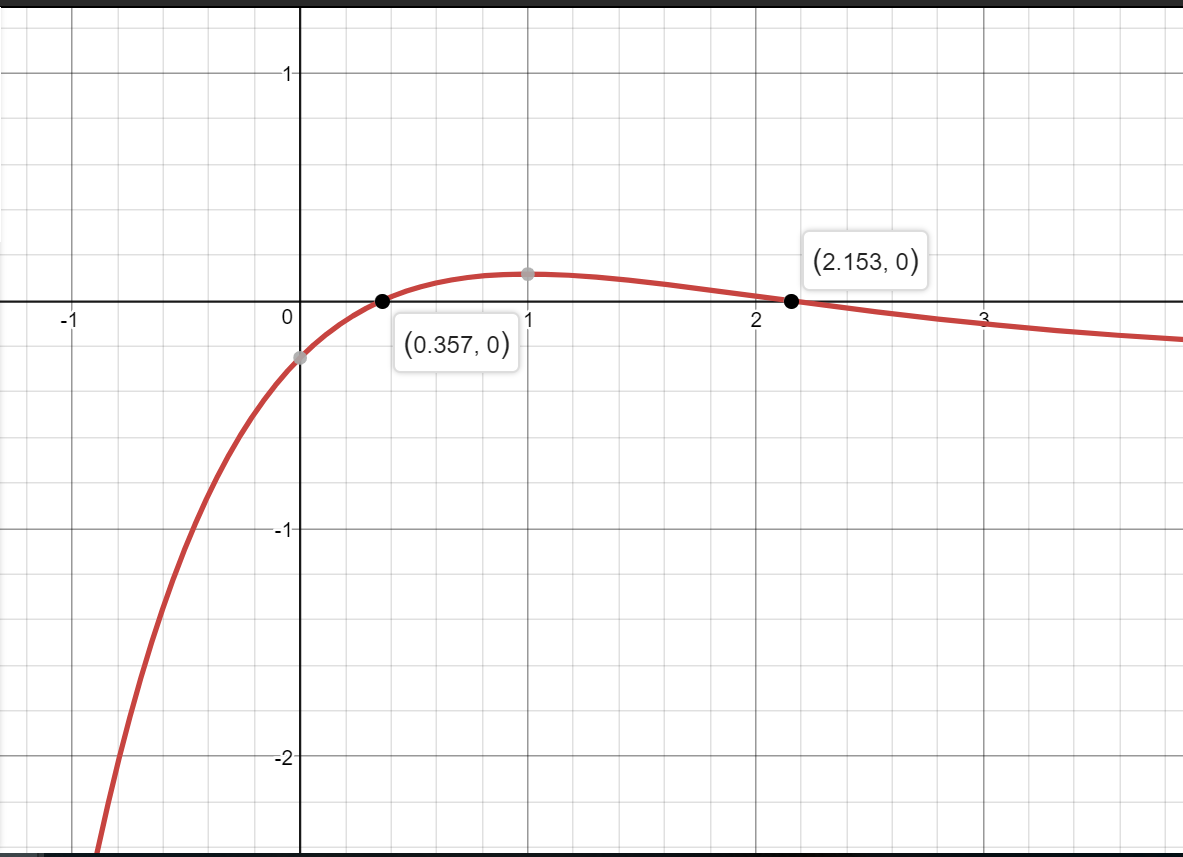
K = 3.736263537

כעת נפתור את המשוואה הנתונה:

F(X) = xe^-x – 0.25

נשתמש בשיטת המיתר ושיטת החצייה ונירא שהתוצאות שנקבל יאפסו את המשוואה.

גרף הפונקצייה:



שני השיטות מיתר ושיטת החצייה יתכנסו לנו לאותם התוצאות אשר מאפסות את המשוואה הנתונה.

מכיוון שקיבלנו תוצאות זהות בשני השיטות, נציג תוצאות עבור שיטת המיתר:

:[0,1] עבור הקטע

קיבלנו את התוצאות הנ"ל :

0.679570457115

-0.6107771764233012

0.5966471106222617

0.5313989715160998

0.2802840685423097

0.3757285787323032

0.3591319798900189

0.35736184874833227

0.3574030471366775

0.35740295618623724

0.35740295618623724

:[2,3] עבור הקטע

קיבלנו את התוצאות הנ"ל :

2.1703954768177947

2.1511010076853063

2.153294925268731

2.153292364480372

2.1532923641089377

2.1532923641089377

2.1532923641089377

אנו צריכים את השורש הקטן לכן ניקח את השורש :

ומכיוון שאנו צריכים את מאית התוצאה נחלק ב 100. 0.35740295618623724

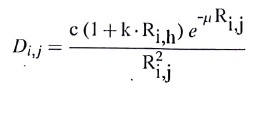
u = 0.0035740295618623724

חישוב סופי:

נתונות לנו הפתרונות הסופיים שהמסור מחשב שהם:

(900,950,1000,1100)

הרכבנו את מטריצה על פי הנוסחא הנתונה:



לפי הנוסחא הנתונה קיבלנו 3 תוצאות שהם:

D1 = 0.004124692756  
D2 = 0.003184809929  
D3 = 0.005909750541

לפי תוצאות אלו בנינו את המטריצה והשתמשנו לפיתרון הסופי בשיטת גאוס-זיידל.

def gaussSeidel(A, b, x, N, tol):  
 max = tol  
 for i in range(max):  
 for j in range(N):  
 summ = 0.0  
 for k in range(N):  
 if (k != j):  
 summ = summ + A[j][k] \* x[k]  
 x[j] = (b[j] - summ) / A[j][j]  
 print("[", end="")  
 for j in range(N):  
 print(x[j], ",", end="")  
 print("]\n")  
  
  
D1 = 0.004124692756  
D2 = 0.003184809929  
D3 = 0.005909750541  
  
gaussSeidel([[D3, D1, D1, D2],  
 [D1, D3, D2, D1],  
 [D1, D2, D3, D1],  
 [D2, D1, D1, D3]],  
  
 [900, 950, 1000, 1100],  
 [0.0, 0.0, 0.0, 0.0], 4, 30)

בסיום הפעולות על המטריצה קיבלנו את 4 התשובות שלנו שהם:

Z1 = 35028.21167307371

Z2 = 32971.6858664142

Z3 = 51320.809563496085

Z4 = 108424.400308975