אנליזה נומרית תשע"ט

דוח סיכום האקטון

קבוצה מס': 26

חברי הקבוצה הנוכחים: ניצן ראש, עומרי אלמלם, שירי אבודרם, יניב בן צבי, לירון גליקמן, בוריס לבייקין.

1. סעיף א' – הגדרת הבעיה –

המאמר עוסק בחישוב זיהום אוויר בצורה דו ממדית המבוסס על מידע מניטור האוויר, עקב בעיות מתמטיות, ניטור אזור מזוהם רדיואקטיבית לא יכולים לספק התפלגות תלת מימדית של זיהום באזור אשר נמצא בתוך ענן הרדיואקטיבי.

עקרונות חישוב:

1.1.1 עושים מיפוי של האזור המזוהם ע"י איזוטופים רדיואקטיביים. ישנה רשת "שנפרסת" על האזור המרובע הזה, המחלקת אותו לאזור N על N המסוק הנושא גלאי קרינה, כל תא (ריבוע) בגודל N מהרשת הזו שנוצרה נחשב כתא בעל פעילות רדיואקטיבית הומוגנית. בכל תא יכול להיות אזור הנקרא "מרכז הזיהום", אליו נתייחס בחישובים שלנו כאל נקודת המקור הממוקמת במרכז התא. היחס בין אזור הקרינה המדודה לבין האזור המזוהם ניתן על ידי סט משוואות ליניאריות.

1.1.2 אחת הבעיות המרכזיות בפיתוח תוכנה הוא המוטיבציה לשיפור פיתוח הצד המדעי של התוכנה. לרוב, תהליך הסימולציה מורכב, גדול, מבלבל, מאוד רגיש לשינויים בפרמטרים ומאוד יקר. ולכן מאמרים רבים מתרכזים בהבנה של הגורמים אשר משפיעים על פיתוח תוכנה מדעית, למרות זאת, רובם מתרכזים בתהליך הפקה ומחזור החיים של התוכנה. התוכנה אשר צריכה להיווצר על מנת למצוא את הרמה האמתית של האזור המזוהם בדו מימד על ידי מסוק וגלאי רגיש מאוד לפרמטרים שמקבל מהשטח ותהליך זה אינו מוכח. המערכת שלנו שייכת לקטגוריה של אמצעי מחשוב בזמן אמת, שבה הפרמטרים המשפיעים על המערכת נגזרים מפעולת האיתור ותלויים בה בצורה ישירה. במאמר זה מציגים מקרה למידה של אזור המזוהם רדיואקטיבית, המקרה דורש פתרון של מערכת משוואות ליניאריות אשר המטריצה הנוצרת יכולה להיות מאוד לא מדויקת, ישנן מספר דרכים זמינות לפתור סוג כזה של בעיות במאמר זה אנו רוצים לפתור בעיה זו מזווית של מהנדסי תוכנה אשר צריכים להבטיח למשתמש שהתוצאות שהתקבלו אכן אמינות ותקינות.

1.2 הגדלים אותם צריך לספק ליישום - פונקציית התגובה של גלאי D לדחיסת יחידות קרינה הנפלטת מנקודת מקור מגלאי במרחק R ניתנת ע"י

$$D = C(1 + kR)e^{-\mu R}/R^2 \qquad (1.1)$$

נתייחס ל: כאשר

C - מקדם פרופורציה $\frac{m^2}{\text{gamma}}$

K מקדם בניית הקרינה באוויר - m^{-1}

- מקדם ספיגת הקרינה באווירμ

R - מרחק בין המקור לגלאי [m]

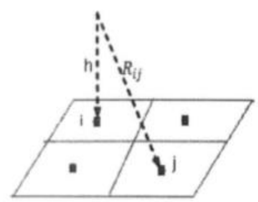


Fig. 1 Sensor location and distances with respect to cells i and j.

כמו שרואים בתמונה הנ"ל, מרחב הדגימה מחולק לרשת מעל אזור הבדיקה. המדידה h והגלאי זז בגובה NxN מקור נקודות יחידותN^2 המושגת על ידי הגלאי היא סכום כל המסות מהאזורים המזוהמים, ממודלת על ידי הממוקמות במרכזו של כל תא ברשת.

כולל על הרשת ניתנת על ידי.j עד i כאשר הגובה מהגלאי מעל הנקודות

$$R_{i,j}^2 = (X_i - X_j)^2 + (Y_i - Y_j)^2 + Z^2 \qquad (1.2)$$

כאשר הנקודות x,y,z קואורדינטות קרטזיות.
חילוף של משוואות 1.1 ו1.2 מספקת לנו פונקציות של תגובת גלאי מתא j כאשר הגלאי ממוקם בגובה j בדיוק מעל תא i.

$$D_{i,j} = \frac{c\,(1 + k \cdot R_{i,h})\, e^{-\mu R_{i,j}}}{R_{i,j}^2} \qquad (1.3)$$

אנו מציינים את כמות הזיהום המרוכזת בנקודה j כ Cj כאשר זו שווה בערך לכמות הזיהום הכללית בהתאמה לתא הרלוונטי. המידע המתקבל מהגלאי אשר נמצא מעלה תא i ניתנת ע"י:

$$M_i = \sum_j R_{i,j}\, C_j \qquad (1.4)$$

כאשר j הוא סכום כל הנקודות ברשת. זה מפיק לנו N^2 משוואות ליניאריות אשר מקשרת את כל הזיהומים הלא ידועים בתא Cj j המדידות Mi בתוך סימון מטריצה.

$$\mathbf{D\,C = M} \qquad (1.5)$$

כאשר הפתרון הוא:

$$\mathbf{C = D^{-1}M} \qquad (1.6)$$

כאשר:

D - מקדם המטריצה ממשוואה 1.3
C - וקטור עוצמות לא ידוע
M - וקטור הערכים המדודים
פתרון המטריצה של המשוואה 1.6 יספק לנו שיטה כללית ועקבית לחישוב שדה התפלגות הזיהום ממדידות הקרינה ע"י המסוק.
ישנן מספר בעיות העולות כאשר השיטה מיושמת, משוואה 1.6 פתירה אך ורק עבור פרמטרים מסוימים, לדוגמא, כאשר המטריצה ההפוכה D^{-1} קיימת, אז היחס המוצג במשוואה 1.7 מתקיים.

$$\mathbf{D^{-1}D = I} \qquad (1.7)$$

על מנת לפתור את המשוואה 1.6 עבור גלאי מסוים, אנו חייבים למצוא את הערכים האופטימליים (או טווח ערכים אופטימליים) עבור גובה של מדידות R, את מספר התאים N בריבוע, אשר גורמים למטריצה D להיות בעלת ערכים תקינים או לא לצרוך הרבה מזיכרון המחשב על מנת לחשב את המטריצה ההופכית D^{-1}.
כאשר ערכים אלה נמצאים, מטריצה 1.6 יכולה להיות פתירה על כל שיטה אלמנטרית.

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

על מנת לפתור את המטלה השתמשנו במס' שיטות:
2.1. בשיטת החצייה
2.2. בשיטת המיתר
2.3 קירוב פולומינלי

ע"מ למצוא את הקבועים C ו μ נעזרנו בשתי שיטות ע"מ לבצע ולידציה לנתונים ולאמת את החישוב ע"י שתי שיטות.

ע"מ למצוא את הקובע K השתמשנו בקירוב פולומינלי בכדי לחשב את הערך בנק' 4.74 של פונקצית הערכים ע"י הטבלה.

הקודים של השיטות הנ"ל נלקחו מהאינטרנט ונכתבו עבורם בדיקות(תחת תיקיית test) :

<u>שיטת החצייה:</u>

הקוד נלקח מGitHub ונמצא בקישור הנ"ל:

```
https://github.com/TheAlgorithms/Python/blob/master/arithmetic_analysis/bisection.py
```

<u>שיטת המיתר:</u>

הקוד נלקח מGitHub ונמצא בקישור הנ"ל:

```
https://www.math.ubc.ca/~pwalls/math-python/roots-optimization/secant/
```

כדי לוודא שהפונקציות שהשתמשנו בהם עובדות ומחזירות לנו את הנתונים שאנו מצפים לקבל, בנינו להן קובץ בדיקה בכדי לבדוק שהפונקציה תקינה ועובדת.

השתמשנו בשיטת קירוב פולמוניאלי שמבוסס על קוד של חבר לכיתה.

3. סעיף ג' – הצגת הנתונים –

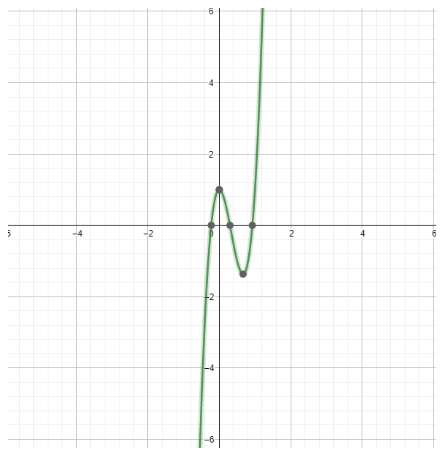
נתייחס לכל שלבי הפתרון בסדר כרונולוגי:

שלב 1 – חישוב הקבוע C (מקדם פרופורציה) :

חישבנו את C ע"י מציאת הממוצע החשבוני של השורשים החיובים של הפונקציה:

$$f(x) = 16x^3 - 16x^2 + 1$$

התנהגות הפונקציה לפי צילומי מסך מאתר GeoGabra:



חישבנו את הנתון ע"י שתי שיטות, שיטת החצייה ושיטת המיתר וביצענו השוואה ע"מ לאמת תוצאה, האימות בוצע בהשוואה של עד 5 ספרות אחרי הנק'.

```python
def calc_c(f):
    ## calc f root by one method ScantMethod
    print("#################### scant method print ####################")
    scant_method_result = [ScantMethod.secant(f, 0, 0.5, 100),
                           ScantMethod.secant(f, 0.6, 1.2, 100),
                           ScantMethod.secant(f, -1.5, 0, 100)]
    scant_method_result = filter_negative_number(scant_method_result)
    print("#################### end scant method print ####################")

    ## calc f root by one method bisection
    print("#################### bisection method print ####################")
    bisection_method_result = [bisection(f, 0, 0.5),
                               bisection(f, 0.6, 1.2),
                               bisection(f, -1.5, 0)]
    bisection_method_result = filter_negative_number(bisection_method_result)
    print("#################### end bisection method print ####################")
    print(scant_method_result)
    print(bisection_method_result)
    if round_list(scant_method_result) == round_list(bisection_method_result):
        print("C: the method solution result are equal")
        return sum(scant_method_result) / len(scant_method_result)
    else:
        print("the result are not equal")
        exit()
```

צילומי מסך של ההדפסות של השיטה לחישוב C כולל האיטרציות של הפונק ' עזר (חצייה ומיתר):

```
###################### scant method print #####################
current a: 0.25 b: 0.25
current a: 0.3 b: -0.008000000000000007
current a: 0.29844961240310075 b: 0.0001821409621981518
current a: 0.29848412526348583 b: 8.627416780981889e-08
current a: 0.29848414161091347 b: 4.0850656191082635e-11
current a: 0.29848414161865394 b: 1.9317880628477724e-14
current a: 0.2984841416186576 b: 0.0
Found exact solution.
```

```
current a: -0.22580203434087376 b: 9.161699282111968e-06
current a: -0.2258021655073612 b: 7.892922774810351e-06
current a: -0.2258022785089817 b: 6.799854710637376e-06
current a: -0.2258023758613268 b: 5.858162007488232e-06
current a: -0.22580245973161484 b: 5.046881314418883e-06
current a: -0.2258025319869339 b: 4.347952384020637e-06
current a: -0.2258025942358064 b: 3.745816033795535e-06
current a: -0.22580264786399584 b: 3.2270678002044306e-06
current a: -0.22580269406535636 b: 2.7801595410270608e-06
current a: -0.2258027338684092 b: 2.39514236599625e-06
current a: -0.22580276815923805 b: 2.0634451765033646e-06
current a: -0.22580279770121647 b: 1.7776838437955078e-06
current a: -0.22580282315200018 b: 1.5314968458968181e-06
###################### end scant method print #####################
```

```
current a: 0.9273188398304505 b: -3.339071241725833e-10
current a: 0.9273188398466863 b: -1.4554046856574132e-10
current a: 0.927318839853763 b: -6.343547909182234e-11
current a: 0.9273188398568474 b: -2.76507705621043e-11
current a: 0.9273188398581919 b: -1.20543575121701e-11
current a: 0.9273188398587779 b: -5.252687174106541e-12
current a: 0.9273188398590333 b: -2.291500322826323e-12
current a: 0.9273188398591448 b: -9.965361869035405e-13
current a: 0.9273188398591932 b: -4.369837824924616e-13
current a: 0.9273188398592144 b: -1.900701818158268e-13
current a: 0.9273188398592236 b: -8.348877145181177e-14
current a: 0.9273188398592277 b: -3.375077994860476e-14
current a: 0.9273188398592294 b: -1.5987211554602254e-14
current a: 0.9273188398592301 b: -7.105427357601002e-15
current a: 0.9273188398592305 b: -3.552713678800501e-15
current a: 0.9273188398592307 b: 0.0
Found exact solution.
```

```
0.5
this is middle status:
0.25
this is [mid=a,b] status:
0.25
this is [a,mid=b] status:
0.375
this is [a,mid=b] status:
0.3125
this is [mid=a,b] status:
0.28125
this is [mid=a,b] status:
0.296875
this is [a,mid=b] status:
0.3046875
this is [a,mid=b] status:
0.30078125
this is [a,mid=b] status:
0.298828125
this is [mid=a,b] status:
0.2978515625
this is [mid=a,b] status:
0.29833984375
this is [a,mid=b] status:
0.298583984375
this is [mid=a,b] status:
0.2984619140625
this is [a,mid=b] status:
0.29852294921875
this is [a,mid=b] status:
0.298492431640625
this is [mid=a,b] status:
0.2984771728515625
this is [a,mid=b] status:
0.29848480224609375
this is [mid=a,b] status:
0.2984809875488281
this is [mid=a,b] status:
0.29848289489746094
this is [mid=a,b] status:
0.29848384857177734
this is [a,mid=b] status:
0.29848432540893555
this is [mid=a,b] status:
0.29848408699035645
this is [a,mid=b] status:
0.298484206199646
```

```
this is middle status:
0.8999999999999999
this is [mid=a,b] status:
0.8999999999999999
this is [a,mid=b] status:
1.0499999999999998
this is [a,mid=b] status:
0.9749999999999999
this is [a,mid=b] status:
0.9374999999999999
this is [mid=a,b] status:
0.91875
this is [a,mid=b] status:
0.9281249999999999
this is [mid=a,b] status:
0.9234374999999999
this is [mid=a,b] status:
0.9257812499999999
this is [mid=a,b] status:
0.9269531249999998
this is [a,mid=b] status:
0.9275390624999998
this is [mid=a,b] status:
0.9272460937499998
this is [a,mid=b] status:
0.9273925781249999
this is [a,mid=b] status:
0.9273193359374998
this is [mid=a,b] status:
0.9272827148437498
this is [mid=a,b] status:
0.9273010253906249
this is [mid=a,b] status:
0.9273101806640623
this is [mid=a,b] status:
0.927314758300781
this is [mid=a,b] status:
0.9273170471191404
this is [mid=a,b] status:
0.9273181915283202
this is [mid=a,b] status:
0.92731876373291
this is [a,mid=b] status:
0.9273190498352049
this is [a,mid=b] status:
0.9273189067840575
```

```
this is middle status:
-0.75
this is [mid=a,b] status:
-0.75
this is [mid=a,b] status:
-0.375
this is [a,mid=b] status:
-0.1875
this is [mid=a,b] status:
-0.28125
this is [mid=a,b] status:
-0.234375
this is [a,mid=b] status:
-0.2109375
this is [a,mid=b] status:
-0.22265625
this is [mid=a,b] status:
-0.228515625
this is [a,mid=b] status:
-0.2255859375
this is [mid=a,b] status:
-0.22705078125
this is [mid=a,b] status:
-0.226318359375
this is [mid=a,b] status:
-0.2259521484375
this is [a,mid=b] status:
-0.22576904296875
this is [mid=a,b] status:
-0.225860595703125
this is [mid=a,b] status:
-0.2258148193359375
this is [a,mid=b] status:
-0.22579193115234375
this is [mid=a,b] status:
-0.22580337524414062
this is [a,mid=b] status:
-0.2257976531982422
this is [a,mid=b] status:
-0.2258005142211914
this is [a,mid=b] status:
-0.22580194473266602
this is [a,mid=b] status:
-0.22580265998840332
this is [mid=a,b] status:
-0.22580301761627197
this is [a,mid=b] status:
-0.22580283880233765
```

```
###################### end bisection method print #####################
[0.2984841416186576, 0.9273188398592307]
[0.2984841465950012, 0.9273188352584838]
C: the method solution result are equal.
```

שלב 3 – חישוב הקבוע K (מקדם הקרינה באוויר) :

השתמשנו בשיטה לחישוב הערך בנק' 4.74, בעזרת פולינום אינטרפולציה.

```python
def calc_k(x, y, k_const):
    k = polynomialAproxMethod.polynomialAproxMethod(x, y)
    print(k)
    return round_digit((lambda x: k[0] + k[1] * x + k[2] * (x ** 2))(k_const))
```

```python
class polynomialAproxMethod:
    @staticmethod
    def polynomialAproxMethod(X, Y):
        matX = np.vander(X, len(X), True)
        invMatX = inv(matX)
        res = np.dot(invMatX, Y)
        print("Polynom: ", end='')
        for i in range(len(res) - 1):
            print("{}x^{} + ".format(res[i], i), end='')
        print("{}x^{}".format(res[len(res) - 1], len(res) - 1))
        return res
```

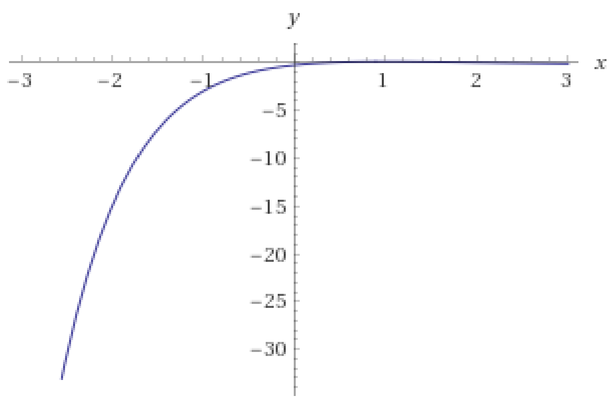
```
Polynom: 13.375x^0 + -3.1000000000000014x^1 + 0.2250000000000001x^2
[13.375 -3.1     0.225]
```

שלב 3 – חישוב הקבוע µ (מקדם פרופורציה) :

קיבלנו את µ ע"י מציאת הממוצע החשבוני של הפונקציה:

$$f(x) = \mathrm{x}e^{\mathrm{x}} - 0.25$$

התנהגות הפונקציה לפי צילומי מסך מאתר GeoGabra:



חישבנו את הנתון ע"י שתי שיטות, שיטת החצייה ושיטת המיתר וביצענו השוואה ע"מ לאמת תוצאה, האימות בוצע בהשוואה של עד 5 ספרות אחרי הנק'.

```
def calc_mu(f):
    ## calc f root by one method ScantMethod
    scant_method_result = [ScantMethod.secant(f, 0.2, 1, 100),
                           ScantMethod.secant(f, 1, 3, 100)]
    scant_method_result = filter_negative_number(scant_method_result)

    ## calc f root by one method bisection
    bisection_method_result = [bisection(f, 0.2, 1),
                               bisection(f, 1, 3)]
    bisection_method_result = filter_negative_number(bisection_method_result)
    print(scant_method_result)
    print(bisection_method_result)
    arr = np.array(scant_method_result)
    if round_list(scant_method_result) == round_list(bisection_method_result):
        print("MU: the method solution result are equal")
        return (arr.min() / 100)
    else:
        print("the result are not equal")
        exit()
```

צילומי מסך של ההדפסות של השיטה לחישוב C כולל האיטרציות של הפונק ' עזר (חצייה ומיתר):

```
current a: 0.5380295262944645 b: 0.06415418924980076
current a: 0.39384833492439136 b: 0.015633506470715652
current a: 0.3641044165260452 b: 0.002986535656584244
current a: 0.35861246698848004 b: 0.0005428238236379612
current a: 0.3576205093174646 b: 9.776089753454897e-05
current a: 0.3574420630509686 b: 1.7577290790438127e-05
current a: 0.35740998516572264 b: 3.1594338395501964e-06
current a: 0.35740421953066726 b: 5.678626790461827e-07
current a: 0.35740318324771914 b: 1.0206414613467274e-07
current a: 0.357402996992814 b: 1.8344350316557012e-08
current a: 0.35740296351656825 b: 3.297093986720512e-09
current a: 0.35740295749976614 b: 5.925981372278954e-10
current a: 0.357402956418345 b: 1.0650974546777547e-10
current a: 0.357402956223977 b: 1.914335356900665e-11
current a: 0.35740295618904355 b: 3.440692175615822e-12
current a: 0.357402956182764 b: 6.183942247162122e-13
current a: 0.357402956181636 b: 1.1118883591620943e-13
current a: 0.35740295618143325 b: 1.992850329202156e-14
current a: 0.357402956181396 b: 3.608224830031759e-15
current a: 0.357402956181390 b: 6.106226635438361e-16
current a: 0.357402956181389 b: 1.1102230246251565e-16
current a: 0.35740295618138895 b: 5.551115123125783e-17
current a: 0.35740295618138884 b: -2.7755575615628914e-17
current a: 0.357402956181388 b: 0.0
Found exact solution.
```

```
current a: 2.0788979747652774 b: 0.010003537762442838
current a: 2.1621778430676653 b: -0.001189040250081791
current a: 2.153330630627221 b: -5.123824590808068e-06
current a: 2.1532925256449365 b: -2.162927412174831e-08
current a: 2.153292364792177 b: -9.129577649424903e-11
current a: 2.1532923641132276 b: -3.853306562717762e-13
current a: 2.153292364110362 b: -1.609823385706477e-15
current a: 2.15329236411035 b: 0.0
Found exact solution.
```

```
this is middle status:
0.6
this is [a,mid=b] status:
0.6
this is [a,mid=b] status:
0.4
this is [mid=a,b] status:
0.30000000000000004
this is [mid=a,b] status:
0.35000000000000003
this is [a,mid=b] status:
0.375
this is [a,mid=b] status:
0.36250000000000004
this is [mid=a,b] status:
0.35625000000000007
this is [a,mid=b] status:
0.35937500000000006
this is [a,mid=b] status:
0.3578125000000001
this is [mid=a,b] status:
0.3570312500000001
this is [a,mid=b] status:
0.3574218750000001
this is [mid=a,b] status:
0.35722656250000007
this is [mid=a,b] status:
0.3573242187500001
this is [mid=a,b] status:
0.3573730468750001
this is [mid=a,b] status:
0.35739746093750013
this is [a,mid=b] status:
0.3574096679687501
this is [a,mid=b] status:
0.35740356445312516
this is [mid=a,b] status:
0.35740051269531264
this is [mid=a,b] status:
0.3574020385742189
this is [mid=a,b] status:
0.35740280151367204
this is [a,mid=b] status:
0.35740318298339857
this is [a,mid=b] status:
0.35740299224853533
```

```
this is [a,mid=b] status:
2.25
this is [mid=a,b] status:
2.125
this is [a,mid=b] status:
2.1875
this is [a,mid=b] status:
2.15625
this is [mid=a,b] status:
2.140625
this is [mid=a,b] status:
2.1484375
this is [mid=a,b] status:
2.15234375
this is [a,mid=b] status:
2.154296875
this is [a,mid=b] status:
2.1533203125
this is [mid=a,b] status:
2.15283203125
this is [mid=a,b] status:
2.153076171875
this is [mid=a,b] status:
2.1531982421875
this is [mid=a,b] status:
2.15325927734375
this is [mid=a,b] status:
2.153289794921875
this is [a,mid=b] status:
2.1533050537109375
this is [a,mid=b] status:
2.1532974243164062
this is [a,mid=b] status:
2.1532936096191406
this is [mid=a,b] status:
2.153291702270508
this is [a,mid=b] status:
2.153292655944824
this is [mid=a,b] status:
2.153292179107666
this is [a,mid=b] status:
2.153292417526245
this is [mid=a,b] status:
2.1532922983169556
```

```
[0.3574029561813889, 2.15329236411035]
[0.3574028968811037, 2.1532923579216003]
MU: the method solution result are equal
```

שלב 4 – חישוב מטריצת המקדמים D:

השטח שלנו הוא שטח של 2X2 וכל שטח חולק ל4 אזורי מדידה המשפיעים זה על זה.

החישוב מתבצע ע"י מספר פונקציות

Cal_d_mat 4X4 מטריצת המקדמים D – ומחשבת כל ערך במטריצה בעזרת הפונקציה calc_d_pos.

בנוסף יש שימוש בפונקציה calc_r לחישוב המרחקים מנק' לנק'.

```python
def calc_d_pos(c, k, mu, i, j, h):
    r = calc_r(i, j, h)
    return c * (1 + k * h * (e ** (mu * r))) / (r ** 2)


def calc_d_mat(c, k, mu, h):
    d_mat = np.zeros(shape=(4, 4))
    for i in range(4):
        for j in range(4):
            d_mat[i][j] = calc_d_pos(c, k, mu, i + 1, j + 1, h)
            print("d_mat[{0}][{1}] = {2}".format(i, j, d_mat[i][j]))
    # d_mat[0][0] = calc_d_pos(c, k, mu, 1, 1, 200)
    return d_mat
```

```python
def calc_r(i, j, h):
    if i == j:
        return h
    elif i != j:
        if (i, j) in ((1, 4), (4, 1), (2, 3), (3, 2)):
            return math.sqrt(
                h ** 2 +
                100 ** 2 + 100 ** 2
            )
        else:
            return math.sqrt(100 ** 2 + h ** 2)
```

תוצאות החישוב:

```
d_mat[0][0] = 0.023415926226099587
d_mat[0][1] = 0.02038075950812037
d_mat[0][2] = 0.02038075950812037
d_mat[0][3] = 0.01832934147824878
d_mat[1][0] = 0.02038075950812037
d_mat[1][1] = 0.023415926226099587
d_mat[1][2] = 0.01832934147824878
d_mat[1][3] = 0.02038075950812037
d_mat[2][0] = 0.02038075950812037
d_mat[2][1] = 0.01832934147824878
d_mat[2][2] = 0.023415926226099587
d_mat[2][3] = 0.02038075950812037
d_mat[3][0] = 0.01832934147824878
d_mat[3][1] = 0.02038075950812037
d_mat[3][2] = 0.02038075950812037
d_mat[3][3] = 0.023415926226099587
matrix d solution:
[[0.02341593 0.02038076 0.02038076 0.01832934]
 [0.02038076 0.02341593 0.01832934 0.02038076]
 [0.02038076 0.01832934 0.02341593 0.02038076]
 [0.01832934 0.02038076 0.02038076 0.02341593]]
```

שלב 5 חישוב וקטור C :

השתמשנו בשתי שיטות שונות לחישוב C.

שיטה 1: שימוש במטריצה הפיכה ע"מ למצוא את C.

$$C = D^{-1} \times M$$

השיטה בעיתית מכיוון שלא לכל מטריצה קיימת מטריצה הפיכה, ובמידה ויש, ע"מ שהמטריצה תתכנס לערכים הרצויים המטריצה צריכה להיות עם ערכים תקינים המאפשרים למחשב לחשב את ההופכית של המטריצה.

```
#################### c calculate 1 ######################
print("final solution calc c = (D^-1) X M :")
c = np.matmul(d_inverse, m)
print(c)
print("check result (D^-1) X D = I")
print(np.matmul(d, d_inverse))

#################### c calculate 2 ######################
c = gauss(d, m, [1, 1, 1, 1], 1000)
print("gauss method solution:")
print(c)
```

```
final solution calc c = (D^-1) X M :
[ 5015.65318168 -5652.67397879  4177.10424738 44334.76608637]
```

בדיקת המטריצה שהתקבל בשיטה הקודמת, ניתן לראות שהמטריצה שמתקבלת אינה מטריצת היחידה והיא איננה מתכנסת לערכים הנדרשים, ובכך מוכיחה לנו הבדיקה את הבעיה בשיטה זו.

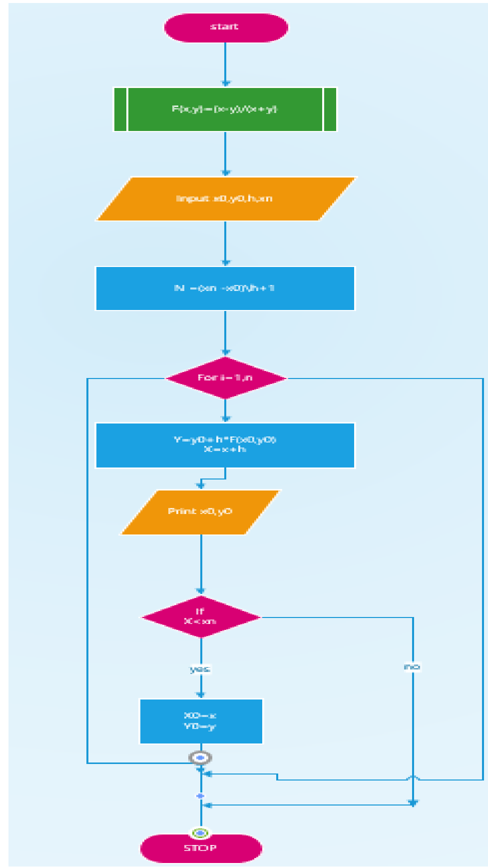
```
check result (D^-1) X D = I
[[ 1.00000000e+00  0.00000000e+00 -8.88178420e-16  8.88178420e-16]
 [ 8.88178420e-16  1.00000000e+00  8.88178420e-16 -8.88178420e-16]
 [ 8.88178420e-16  8.88178420e-16  1.00000000e+00 -8.88178420e-16]
 [ 1.77635684e-15  0.00000000e+00 -8.88178420e-16  1.00000000e+00]]
```

שיטה 2: שימוש בשיטת גאוס זידל:

שיטה איטרטיבית לפתרון משוואות ליניאריות אשר מתבססת על פתרון מטריצות אלכסוניות דומיננטיות או סימטריות חיוביות.

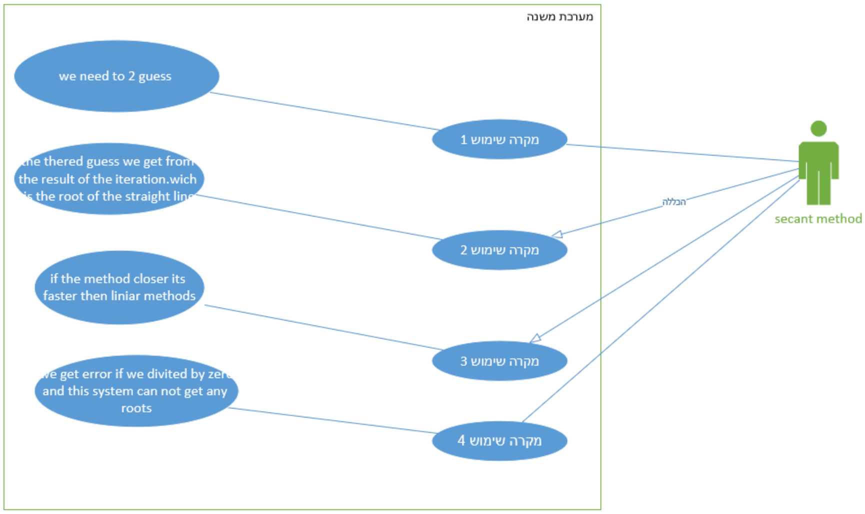
```
gauss method solution:
[ 5015.65318168 -5652.67397879  4177.10424738 44334.76608637]
```

לשיטת החצייה: DFD

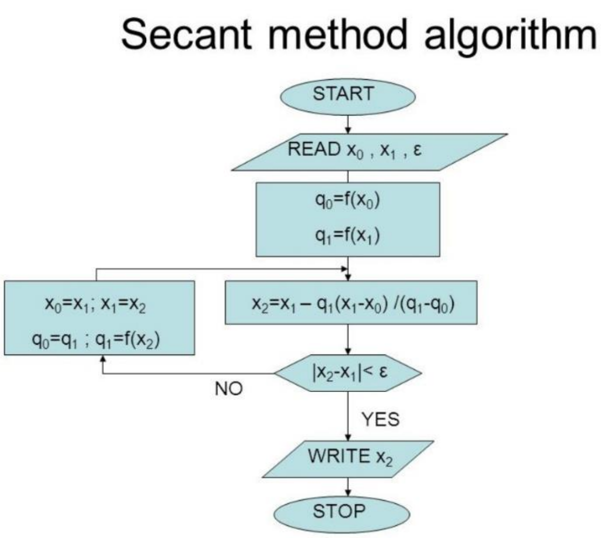


Use Case לשיטת החצייה:

:Use Case לשיטת ההמיתר



:Use Case לשיטת המיתר



4. סעיף ד' – תוצאות –

לאחר החישובים, התוצאות שקיבלנו לכל המשתנים הם:

```
Equation mu element : 0.003574029561813889

Equation c element : 0.6129014907389442

Equation k element : 3.7362

matrix d solution:
[[0.02341593 0.02038076 0.02038076 0.01832934]
 [0.02038076 0.02341593 0.01832934 0.02038076]
 [0.02038076 0.01832934 0.02341593 0.02038076]
 [0.01832934 0.02038076 0.02038076 0.02341593]]


matrix d inverse solution:
[[ 355.45778103 -251.09989123 -251.09989123  158.8622165 ]
 [-251.09989123  355.45778103  158.8622165  -251.09989123]
 [-251.09989123  158.8622165   355.45778103 -251.09989123]
 [ 158.8622165  -251.09989123 -251.09989123  355.45778103]]

final solution - c = (D^-1) X M c vector:
[ 5015.65318168 -5652.67397879  4177.10424738 44334.76608637]


gauss method solution:
[ 5015.65318168 -5652.67397879  4177.10424738 44334.76608637]


check result (D^-1) X D = I
[[ 1.00000000e+00  0.00000000e+00 -8.88178420e-16  8.88178420e-16]
 [ 8.88178420e-16  1.00000000e+00  8.88178420e-16 -8.88178420e-16]
 [ 8.88178420e-16  8.88178420e-16  1.00000000e+00 -8.88178420e-16]
 [ 1.77635684e-15  0.00000000e+00 -8.88178420e-16  1.00000000e+00]]
```

5. סעיף ה' – תוצאות –

בהתחלה החלטנו שאנחנו מתחלקים לקבוצות בתוך הקבוצה: חלק עבדו על חישוב חלק על הכנסת חישובים לפונקציה וחלק על הקובץ הסיכום, מציאת הנתונים נעשתה בעזרת פונקציות שבנינו ומצאנו באינטרנט.

לכל שאלה בדקנו את התשובה עם 2 מתודות כדי לוודא תקינות של התשובה, ורק לאחר שווידאנו את התשובה המשכנו הלאה.

המתודות שהשתמשנו בהן הן: שיטת המיתר, שיטת החצייה , קירוב פולינומיאלי וגאוס זידל.

היה קושי בבדיקת התוצאה מכיוון שלא ניתן היה למצוא מטריצה הופכית לD בשיטות שאנחנו מכירים ולכן הבדיקה הסופית של נכונות התשובה לא היתה אפשרית.

6. נספח א' –

```python
class ScantMethod:
    @staticmethod
    def secant(func, a, b, iterations):
        if func(a) * func(b) >= 0:
            print("Secant method fails.")
            return None
        x0 = a
        x1 = b
        for n in range(1, iterations + 1):
            m_n = x0 - func(x0) * (x1 - x0) / (func(x1) - func(x0))
            f_m_n = func(m_n)
            print("current a: {0} b: {1}".format(m_n, f_m_n))
            if func(x0) * f_m_n < 0:
                x0 = x0
                x1 = m_n
            elif func(x1) * f_m_n < 0:
                x0 = m_n
                x1 = x1
            elif f_m_n == 0:
                print("Found exact solution.")
                return m_n
            else:
                print("Secant method fails.")
                return None

        return x0 - func(x0) * (x1 - x0) / (func(x1) - func(x0))
```

שיטה: שיטת המיתר

```python
def bisection(function, a, b):  # finds where the function becomes 0 in [a,b] using bolzano
    start = a
    end = b
    print(start)
    print(end)
    if function(a) == 0:  # one of the a or b is a root for the function
        print(a)
        return a
    elif function(b) == 0:
        print(b)
        return b
    elif function(a) * function(b) > 0:  # if none of these are root and they are both positive or negative,
        # then his algorithm can't find the root
        print(str(a * b) + ">0")
        print("couldn't find root in [a,b]")
        return
    else:
        mid = (start + end) / 2
        print("this is middle status:")
        print(mid)
        while abs(start - mid) > 10 ** -7:  # until we achieve precise equals to 10^-7
            if function(mid) == 0:
                print("this is middle status:")
                print(mid)
                return mid
            elif function(mid) * function(start) < 0:  # find new middel number
                end = mid
                print("this is [a,mid=b] status:")
                print(end)
            else:
                start = mid
                print("this is [mid=a,b] status:")
                print(start)
            mid = (start + end) / 2  # finding new middel number
        return mid
```

שיטה: שיטת החצייה

```python
import numpy as np
from numpy.linalg import inv


class polynomialAproxMethod:
    @staticmethod
    def polynomialAproxMethod(X, Y):
        matX = np.vander(X, len(X), True)
        invMatX = inv(matX)
        res = np.dot(invMatX, Y)
        print("Polynom: ", end='')
        for i in range(len(res) - 1):
            print("{}x^{} + ".format(res[i], i), end='')
        print("{}x^{}".format(res[len(res) - 1], len(res) - 1))
        return res
```

שיטה : שיטה פולומינליאלית

```python
def gauss(A, b, x, n):

    L = np.tril(A)
    U = A - L
    for i in range(n):
        x = np.dot(np.linalg.inv(L), b - np.dot(U, x))
        # print( str(i).zfill(3)),
        # print(x)
    return x
```

שיטה : גאוס זידל