בס"ד

המכללה האקדמית להנדסה ע"ש סמי שמעון – SCE

המחלקה להנדסת תוכנה

**פרויקט גמר בקורס אנליזה נומרית**

בנימין יעקובי, 323492835 – שנה ב'

הקדמה...............................................................................................................................................2

היסטוריה של השיטה...........................................................................................................................3

Regula-Falsi באנליזה נומרית..............................................................................................................4

המתמטיקה שמאחורי שיטת Regula-Falsi............................................................................................5

קוד שיטת Regula-Falsi......................................................................................................................6

מדריך למשתמש...................................................................................................................................7

תוצאות האלגוריתם עבור פונקציות נבחרות.......................................................................................8-11

נכונות האלגוריתם..............................................................................................................................12

נספח 1..............................................................................................................................................13

רשימת מקורות..................................................................................................................................14

**הקדמה**

1. במתמטיקה, שיטת Regula-Falsi או בשמה הנרדף שיטת False Position היא שיטה ישנה מאוד לפתרון משוואות עם נעלם אחד. במילים פשוטות, השיטה היא טכניקת ניסוי וטעייה של שימוש בערכי בדיקה עבור המשתנה ולאחר מכן התאמת ערך הבדיקה בהתאם לתוצאה. במקורות מסוימים השיטה נקראת גם בתור *guess & check*. גרסאות מודרניות של הטכניקה משתמשות בשיטות סיסטמתיות לבחירת ערכי בדיקה חדשים ועוסקות בשאלה האם ניתן לקבל קירוב לפתרון ואם כן, כמה מהר ניתן למצוא את הקירוב.
2. ניתן להבחין בין שני סוגים בסיסיים של Regula-Falsi מבחינה היסטורית:
3. Regula-Falsi פשוט – Simple False Position המשמש כשיטה לפתרון בעיות הניתנות לפתרון באופן ישיר, כגון בעיות שניתן לכתוב אותן כך: a \* X = b = F(x)
4. Regula-Falsi כפול – Double False Position המשמש כשיטה לפתרון בעיות מורכבות יותר, כגון בעיות שניתן לכתוב אותן כך: F(x) = a \* X + c = 0
5. בעבודה זו נעסוק בשלושה חלקים עיקריים:
   1. תחילה נעסוק בשיטה בפן ההיסטורי ואופן ההתפתחות שלה וע"י מי היא פותחה, נעסוק בשימושיות השיטה בתחום האנליזה הנומרית וברקע המתמטי מאחורי השיטה.
   2. בשלב השני נציג את הקוד שנכתב עבור השיטה בשפת Python 3.x, נציג מדריך מורחב לשימוש בקוד עם הסבר פשוט על האלגוריתם הממומש בקוד הנ"ל.
   3. לבסוף, נציג ארבעה דוגמאות לבעיות מתמטיות שנחפש וננסה למצוא את שורשיהן באמצעות הקוד שנכתב עבור השיטה. בנוסף לפלט מהאלגוריתם, נציג את התוצאות בצירוף גרפים וטבלאות ונשווה בין התוצאות שאנו מקבלים ובין שני מקורות מוסמכים כאשר העיקרית בינם היא ספר הקורס:

Numerical Analysis, R. Burden and J. Faires, Ninth Edition

**היסטוריה של השיטה**

טכניקת Regula-Falsi הפשוטה הומצאה במצריים העתיקה.

Double False Position הומצא בשלהי העת העתיקה כאלגוריתם אריתמטי בלבד. בטקסט המתמטי הסיני העתיק הנקרא "תשעת הפרקים על האמנות המתמטית", מתאריך 200 לפנה"ס עד 100 לספירה, רוב פרק 7 הוקדש לאלגוריתם זה.

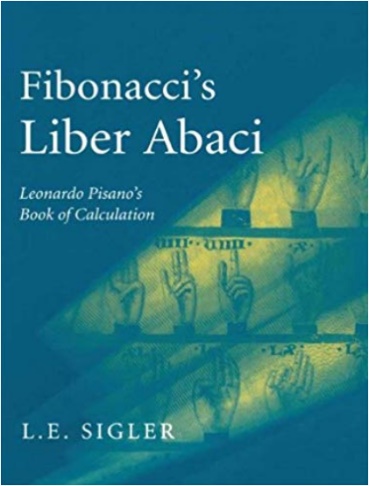
בין המאות התשיעי לעשירי כתב המתמטיקאי המצרי, אבו כאמיל, מאמר שאבד על השימוש בשיטה, הידוע בשם "ספר שתי השגיאות".

הכתבה הישנה ביותר ששרדה על Double False Position מהמזרח התיכון היא זו של קוסטא אבן לוקה (המאה העשירי), מתמטיקאי ערבי מהעיר בעל-בכ שבלבנון. הוא הצדיק את הטכניקה בהוכחה גיאומטרית רשמית בסגנון אוקלידי.

פיבונאצ'י הקדיש את פרק 13 לספרו ספר החשבונייה (liber abaci, 1202 לספירה) כדי להסביר ולהוכיח את השימושים של Double False Position.

בשנת 1494, לוקה פאצ'ולי מתמטיקאי איטלקי השתמש במונח *cataym el* בספרו *summa de arithmetica*, שכנראה לקח את המונח מפיבונאצ'י. סופרים אירופיים אחרים היו עוקבים אחר פאצ'ולי ולפעמים סיפקו תרגום ללטינית או לשפות מקומיות.

המונח שבו השתמש פאצ'ולי כמעט נעלם בגרסאות של המאה השש עשרה באירופה והטכניקה נקראה בשמות שונים כגון *"Rule of false", "rule of position"* ו- *"rule of False Position"*. השם *Regula-Falsi* הופיע כגרסה הלטינית של rule of false כבר בשנת 1690.



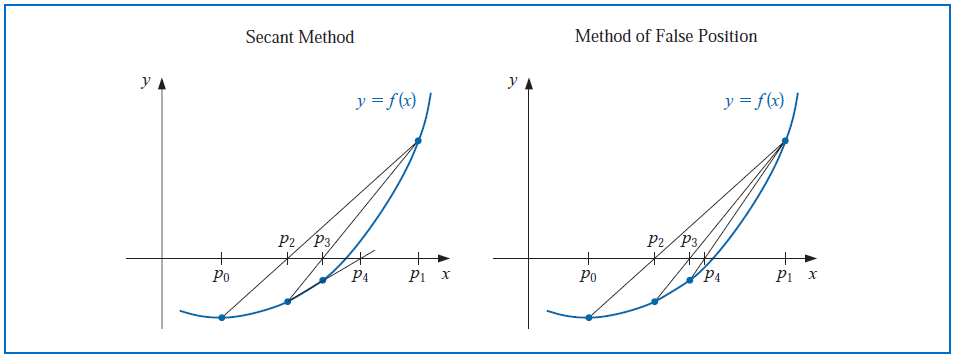
**Regula-Falsi באנליזה נומרית**

שיטת Regula-Falsi מספק פתרון מדויק עבור פונקציות ליניאריות, אבל טכניקות אלגבריות החליפו את השימוש שלה עבור פונקציות אלה. עם זאת, באנליזה נומרית, Double False Position הפך לאלגוריתם למציאת שורש ומשמש בטכניקות קירוב איטרטיביות.

משוואות רבות, כולל משוואות מורכבות, ניתנים לפתרון רק על ידי קירוב מספרי איטרטיבי. זה מורכב מניסוי וטעייה, שבו מנסים להציב ערכים שונים. ניסוי וטעייה אלה יכולים להיות מודרכים על ידי חישוב, בכל שלב של ההליך נעשה הערכה חדשה לפתרון. ישנן דרכים רבות להגיע לאומדן מחושב ו-Regula-Falsi מספק אחד מדרכים אלו.

בהינתן משוואה, יש להזיז את כל החלקים שלה לצד אחד כך שיש לה את הצורה f (x) = 0, כאשר f היא פונקציה מסוימת של המשתנה הלא ידוע x. ערך c שפותר את המשוואה הזו, כלומר: f (c) = 0, נקרא שורש של הפונקציה f והוא הפתרון של המשוואה המקורית. אם f היא פונקציה רציפה וישנן שתי נקודות a0 ו- b0 כך ש- f (a0) ו- f (b0) הם סימנים מנוגדים, אזי לפי משפט הערך הביניים , לפונקציה f יש שורש במרווח (a0, b0).

ישנם אלגוריתמים רבים למציאת שורש אשר ניתן להשתמש בהם כדי להשיג קירובים לשורש. אחת הנפוצים ביותר היא השיטה של ניוטון, אבל השיטה יכולה להיכשל בלמצוא את השורש בנסיבות מסוימות וזה עלול להיות יקר מבחינה חישובית שכן זה דורש חישוב של נגזרת הפונקציה.

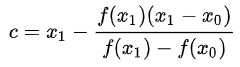
השיטה Regula-Falsi מייצרת קירובים באופן זהה לשיטת Secant אך היא כוללת בדיקה על מנת להבטיח שהשורש נמצא תמיד בין שני איטרציות עוקבות.

**המתמטיקה מאחורי שיטת Regula-Falsi**

שיטה זו היא ואריאציה של שיטת המיתר המשתנה אשר פועלת לפי שיקולים של שיטת החציה, כך שאם בתחום הניחוש קיים שורש אחד, התכנסות השיטה מובטחת. אם הפונקציה קרובה לקו ישר באזור השורש, שיטה זו עדיפה על פני שיטת החצייה ביעילותה.

בדומה לשיטת החציה , נתחיל עם קטע [x0, x1] כך ש f(x) רציפה בקטע ומתקיים f(x0) \* f(x1) < 0.

כעת אם נעביר מיתר, ונניח שהמיתר חותך את ציר x בנקודה c. נשתמש בשיטת "*המיתר המשתנה"* (נספח 1):



בהנחה שהפונקציה לא מתנהגת בצורה מוזרה, אנחנו נצפה שנקודת החיתוך של המיתר עם ציר ה-x יהיה די קורב לנקודת החיתוך של הפונקציה עם ציר ה-x.

נתחיל את החישוב עם קטע כלשהו [x0, x1] כך ש: f(x0) \* f(x1) < 0

* אם f(x0) \* f(c) > 0 , אזי נמשיך לחפש עם הקטע [c, x1] עד לתוצאה נוספת.
* אם f(x0) \* f(c) < 0 , אזי נמשיך לחפש עם הקטע [x0, c] עד לתוצאה נוספת.
* אם f(x0) \* f(c) = 0 , אזי c = a וסיימנו את חישוב השורש.

*האלגוריתם עליו מבוסס Regula-Falsi*

שיטה זו דומה ברעיונה לשיטות איטרטיביות אחרות, כגון שיטת ניוטון-רפסון, המיתר וכדומה. אולם, בשיטה זו (Regula-Falsi) אנו לא נדרשים להשתמש בנגזרת הפונקציה. כאשר  הנגזרת היא מסובכת, ניתן להשתמש במיתרים במקום במשיקים. בדומה לניוטון-רפסון, גם כאן ההתכנסות תלויה בבחירת הנקודות ההתחלתיות.

* התחל עם קטע כלשהו [x0, x1] כך ש: f(x0) \* f(x1) < 0
* חשב את x2 עפ"י הנוסחה שחישבנו למעלה.
  + אם f(x2) = 0 אז סיימנו, כלומר מצאנו את הדרוש.
  + אם f(x2) \* f(x0) < 0 , אזי נמשיך לחפש עם הקטע [x0, x2]
  + אחרת, נמשיך עם הקטע [x1, x2] עד למציאה או אי מציאת הדרוש.

**קוד שיטת Regula-Falsi**

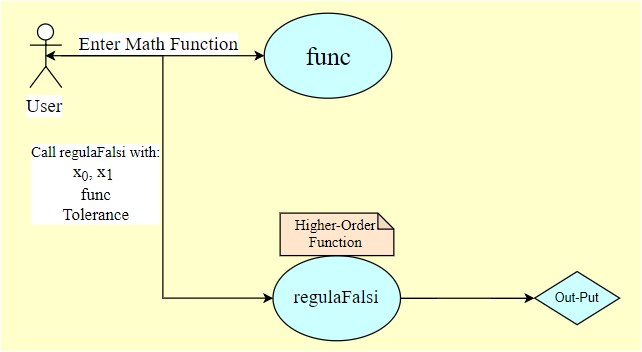
import math  
import numpy as np  
  
def func(x):  
 f = math.pow(x,3) + 2\*math.pow(x,2) - 3\*x - 1  
 return f  
  
print("Sample input: regulaFalsi(1,2,10\*\*-4, 100,func)\n")  
def regulaFalsi(a, b, TOL, N, f):  
 i = 1  
 FA = f(a)  
 print("%-20s %-20s %-20s %-20s %-20s" % ("n", "a\_n", "b\_n", "p\_n", "f(p\_n)"))  
 while (i <= N):  
 p = (a \* f(b) - b \* f(a)) / (f(b) - f(a))  
 FP = f(p)  
 if (FP == 0 or np.abs(f(p)) < TOL):  
 break  
 else:  
 print("%-20.8g %-20.8g %-20.8g %-20.8g %-20.8g\n" % (i, a, b, p, f(p)))  
 i = i + 1  
 if (FA \* FP > 0):  
 a = p  
 else:  
 b = p  
 return

regulaFalsi(1, 2, 10\*\*-4, 100, func)

*הדוגמה מיושמת על הפונקציה f(x )= x3 + 2x2 - 3x - 1 רק לשם הדגמה בלבד!*

**מדריך למשתמש**

1. הקוד כתוב בשפת Python 3.x ועל כן לא יפעל בגרסאות קודמות של השפה.
2. בתפעול הקוד אנו נעזרים בספריות מובנות: Numpy & Math ולפיכך יש לייבא אותן.
3. הקוד מורכב משתי פונקציות מרכזיות (יחסי הגומלין ביניהן כפי שמובא בשרטוט):
   * Func – פונקציית עזר (Helper Function)
     1. פונקציה מסדר ראשון המחזירה למשתמש את *הפונקציה המתמטית* הנדרשת.
     2. פונקציה זו מכילה את *הפונקציה המתמטית* שאת שורשיה אנחנו מחפשים.
     3. יש להכניס ידנית את *הפונקציה המתמטית* לפני הרצת הקוד.
   * regulaFalsi – פונקציה ראשית למימוש האלגוריתם.
     1. פונקציה מסדר גבוה המקבלת ארבעה משתנים:
        + a – הגבול הימני בתחום הראשון שמתחילים לחפש בו.
        + b – הגבול הימני בתחום הראשון שמתחילים לחפש בו.
        + TOL – רמת הדיוק הנדרשת (בקוד שבמסמך נקבע על 10 בחזקת 4-(.
        + N – סך כל האיטרציות שאנו רוצים לאפשר לאלגוריתם לרוץ עד לקבלת תוצאה.
        + func – פונקציית העזר func



**תוצאות האלגוריתם עבור פונקציות נבחרות – דוגמה 1**

עבור הפונקציה: f(x )= x3 + 2x2 - 3x - 1 כאשר a = 1 , b = 2

*פתרון הפונקציה ע"י האלגוריתם המובא בעמ' 6*

Sample input: regulaFalsi (1, 2, 10\*\*-4, 100, func)

f(p\_n) p\_n b\_n a\_n iter

1 1 2 1.1 -0.549

2 1.1 2 1.1517436 -0.27440072

3 1.1517436 2 1.1768409 -0.13074253

4 1.1768409 2 1.1886277 -0.060875863

5 1.1886277 2 1.1940789 -0.028040938

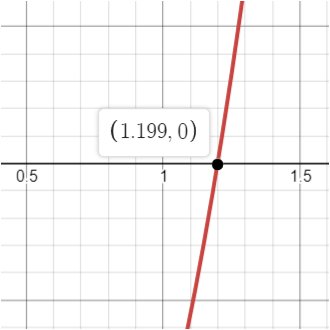
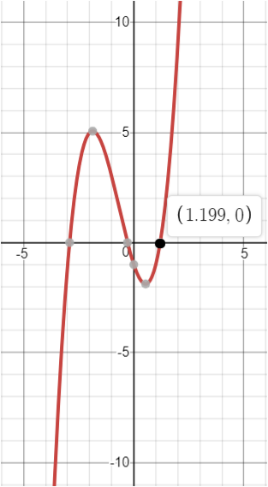
6 1.1940789 2 1.1965821 -0.01285224

7 1.1965821 2 1.1977278 -0.0058772415

8 1.1984904 2 1.1985996 -0.0005596125

9 1.198599 2 1.1986494 -0.00025543669

**10 1.1986494 2 1.1986721 -0.0001165895**

*תצוגת הגרף התואם כפי שמופיע באתר Desmos.com*

**תוצאות האלגוריתם עבור פונקציות נבחרות – דוגמה 2**

עבור הפונקציה: f(x) = cos(x) - x כאשר a = 0.5 , b = pi/4

*פתרון הפונקציה ע"י האלגוריתם המובא בעמ' 6*

Sample input: regulaFalsi(1,2,10\*\*-4, 100,func)

f(p\_n) p\_n b\_n a\_n iter

1 0.5 7.8539816 0.83732777 -0.16787748

2 0.5 0.83732777 0.73350764 0.0093230497

3 0.73350764 0.83732777 0.73896992 0.0001928095

4 0.73896992 0.83732777 0.73908276 3.971221e-06

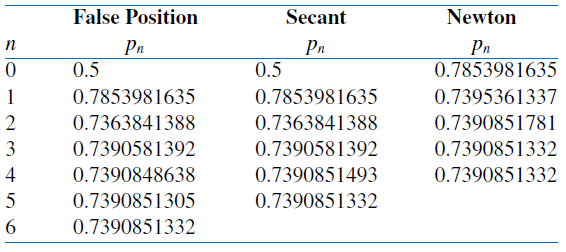
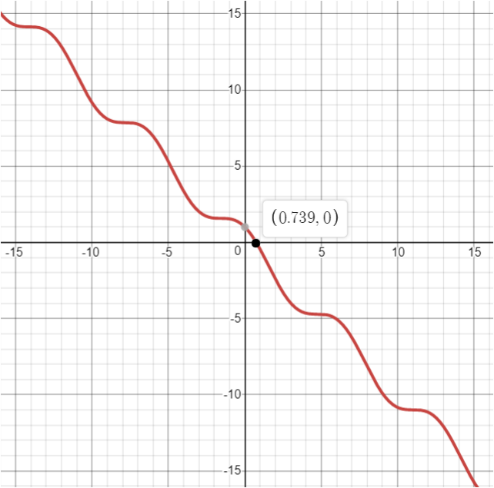
5 0.73908276 0.83732777 0.73908508 8.1787e-08

6 0.73908508 0.83732777 0.73908513 1.684392e-09

**7 0.73908513 0.83732777 0.73908513 3.468907e-11**

*בצד ימין – טבלת הפתרונות כפי שמופיע בספר Numerical Analysis (Ninth Edition)*

*בצד שמאל – תצוגת הגרף התואם כפי שמופיע באתר Desmos.com*

כפי שניתן לראות, בהשוואה לשיטת המיתר (Secant Method) ובהשוואה לשיטת ניוטון אנו מצליחים לקבל באמצעות האלגוריתם את התוצאה הנכונה בתור שורש המשוואה הנ"ל.

**תוצאות האלגוריתם עבור פונקציות נבחרות – דוגמה 3**

עבור הפונקציה: f(x) = x \* cos[x/(x-2)] כאשר a = 1 , b = 1.5

*פתרון הפונקציה ע"י האלגוריתם המובא בעמ' 6*

Sample input:regulaFalsi(1,1.5,10\*\*-5, 100,func)

iter a\_n b\_n p\_n f(p\_n)

1 1 1.5 1.1333888 0.29460897

2 1.1333888 1.5 1.1940806 0.10632161

3 1.1940806 1.5 1.2145202 0.029852038

4 1.2145202 1.5 1.220146 0.0075815344

5 1.220146 1.5 1.2215675 0.0018694797

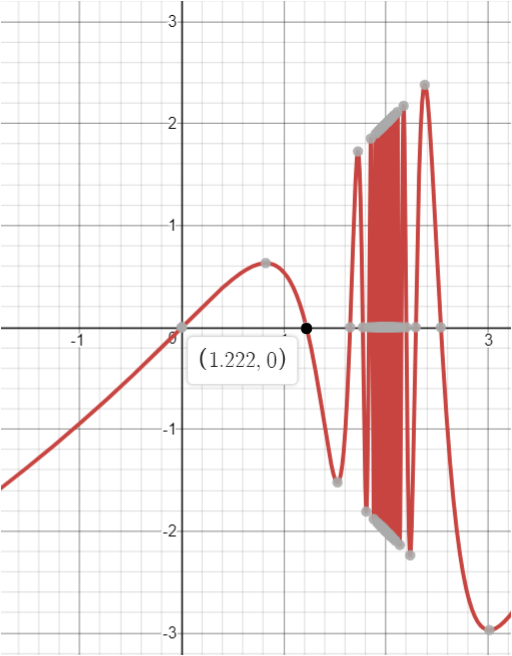
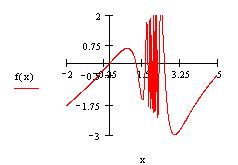
6 1.2215675 1.5 1.2219176 0.00045749013

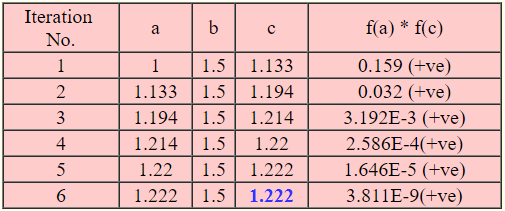
7 1.2219176 1.5 1.2220033 0.00011174428

**8 1.2220033 1.5 1.2220242 2.7281524e-05**

*בצד שמאל – תצוגת הגרף התואם כפי שמופיע באתר Desmos.com*

*בצד ימין – תצוגת גרף + טבלת הפתרונות כפי שמופיע באתר האקדמי mat.iitm.ac.in*





**תוצאות האלגוריתם עבור פונקציות נבחרות – דוגמה 4**

עבור הפונקציה: f(x) = exp(x) – 3x2 כאשר a = 3 , b = 4

*פתרון הפונקציה ע"י האלגוריתם המובא בעמ' 6*

Sample input:regulaFalsi(3,4,10\*\*-4, 100,func)

iter a\_n b\_n p\_n f(p\_n)

1 3 4 3.5117044 -3.4908782

2 3.5117044 4 3.6806583 -0.96923545

3 3.6806583 4 3.7215597 -0.22121453

4 3.7215597 4 3.7305921 -0.048158256

5 3.7305921 4 3.7325442 -0.010375262

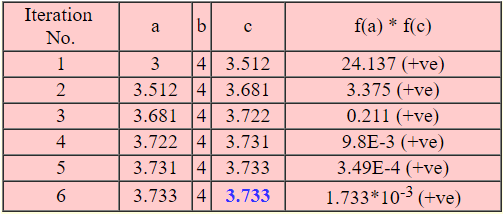
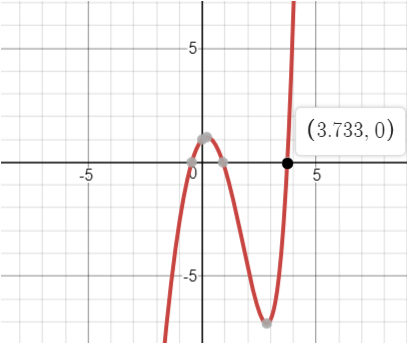
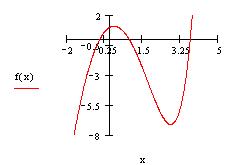
6 3.7325442 4 3.7329641 -0.002230226

7 3.7329641 4 3.7330543 -0.00047916842

**8 3.7330543 4 3.7330737 -0.00010293955**

*בצד שמאל – תצוגת הגרף התואם כפי שמופיע באתר Desmos.com*

*בצד ימין – תצוגת גרף + טבלת הפתרונות כפי שמופיע באתר האקדמי mat.iitm.ac.in*



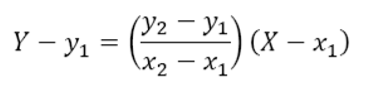
**הוכחה לנכונות האלגוריתם**

1. בארבעת הדוגמאות המובאות בעמודים 8-11 תחת הכותרת *תוצאות האלגוריתם עבור פונקציות נבחרות* ניתן לראות כי בהשוואה למקורות המוסמכים שבחרנו להשוות למולם, התוצאות אשר מתקבלות מהרצת האלגוריתם (עמוד 6) זהות לתוצאות המופיעות במקורות אלו. שני המקורות הבאים נבחרו להשוואת הפתרונות של תוצאות האלגוריתם:
2. math.iitm.ac.in – משמש את המחלקה למתמטיקה של המוסד האקדמי: Indian Institute Of Technology Madras, Chennai
3. ספר הקורס אשר מופיע בסילבוס ושעליו מתבססת מרבית עבודה גמר זו: Numerical Analysis (R. Burden and J. Faires, Ninth Edition)
4. יש לציין כי בתהליך העבודה על האלגוריתם, ניתן היה לשים לב כי קיימים הבדלים קטנים בין כמות האיטרציות המבוצעות ע"י האלגוריתם המופיע במסמך זה ובין כמות האיטרציות המופיעות בטבלאות למולם התבצעה השוואה.
5. כמו כן, יש לציין כי הבדלים אלו מינוריים אשר לא משפיעים כלל על התוצאה הסופית אליו מתכנס האלגוריתם. קרוב לוודאי הוא שהבדלים אלו נובעים מהבדלים ברמת הדיוק הגנרית אשר הוגדרה על ידינו (10 בחזקת 4-, כלומר אלפית) ובין רמות הדיוק במקורות הנ"ל אשר באמצעותם הגיעו להתכנסות הסופית.
6. נוסף על כך, נעזרנו במנוע Desmos.com על מנת להציג את גרף הפונקציות אשר נבדקו. זאת על מנת להציג בצורה נראית לעין את החיתוך בפועל עם ציר ה-x, כדרך נוספת להראות שהתוצאות אליה התכנס האלגוריתם אכן נכונות ומשקפות את המציאות.
7. לסיכום, על סמך התוצאות אשר התקבלו מהאלגוריתם (כפי שהוצגו בפירוט קודם לכן עמודים 8-11) ועל סמך ההשוואה עם המקורות המוסמכים, בדגש על ספר הקורס, ניתן לומר כי התוצאות אליהן מתכנס האלגוריתם עבור פונקציות נתונות, הן לינאריות והן פולינומיאליות כאחד, אמינות.

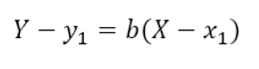
**נספח 1 – אופן מציאת ערך c עבור שיטת "*המיתר המשתנה"***

הנוסחאות למציאת משוואת קו-ישר:

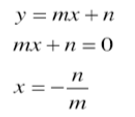
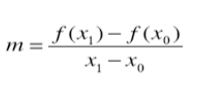
1. כאשר ידועות 2 נקודות על גבי הקו, נסמן את הנקודות ב –(x1, y1), (x2, y2) :



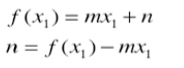
1. כאשר ידוע השיפועb ונקודה אחת על הקו (x1, y1):



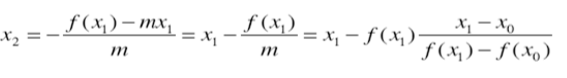
נחשב את x2  (כזכור, נקודת החיתוך של המיתר עם ציר ה-x) באופן הבא:



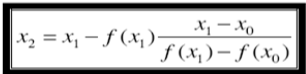
משוואת המיתר: כאשר:



כדי לחשב את n:



ונקבל :



שזה בעצם הנוסחה הנדרשת עבור c (מסומן כאן ב-x2):

**רשימת מקורות**

<https://dlmf.nist.gov/>

<http://math-wiki.com>

https://math.iitm.ac.in/

<https://he.wikibooks.org>

<https://scholar.google.co.il>

https://www.wolframalpha.com/

<https://people.xiph.org/~greg/opus/p250-le.pdf>

<http://www2.lv.psu.edu/ojj/courses/cmpsc-201/numerical/regula.html>

<http://mathfaculty.fullerton.edu/mathews/n2003/RegulaFalsiMod.html>

<http://m-hikari.com/ams/ams-2012/ams-25-28-2012/hassanAMS25-28-2012.pdf>

<https://pdfs.semanticscholar.org/1ef4/ee58a159dc7e437e190ec2839fb9a654596c.pdf>

<https://ece.uwaterloo.ca/~dwharder/NumericalAnalysis/10RootFinding/falseposition/>

<https://mat.iitm.ac.in/home/sryedida/public_html/caimna/transcendental/bracketing%20methods/regula-falsi/regula-falsi.html>

R. Burden and J. Faires, Numerical Analysis, Brooks Cole, 2011. (Ninth Edition)