**מציאת שורשים למשוואת לא לינאריות**

 הנושא שעומד לפנינו הוא, בהינתן פונקציה לא לינארית f(x) כמו

***f(x)= 8x7 - 3sinx x8-6ex***

איך מוצאים את נקודה (או נקודות) x\*  שבו (בהם) מתקים

***f(x\*) = 0***

בשלב זה נגביל את הדיון לפונקציות ממשיות בעלי משתנה אחד.

מציאת שורשים למשוואות לא לינאריות הינו אחד התחומים האופיניים ביותר של האנליזה הנומרית. מצד אחד למציאת שורשים של פונקציות כלליות מן הסוג שלעיל אין שום פתרון נוסחתי (או ישיר או סגור) גם כאשר הוא מורכב מפונקציות אלמטריות ידועות. אפילו לפולינומים פשוטים ממעלה חמישית ואילך אין נוסחה – זה עובדה מתמטית מוכחת. אולם בהינתן פונקציה ספציפית זו או אחרת ניתן יהיה, בדרך כלל, למצוא את השורשים שלה. אם למשל נשרטט את הפונקציה על פני תחום רחב וצפוף של נקודות אנחנו נדע פחות או יותר היכן הפונקציה חותכת את ציר ה-x. אבל זהו דרך לא יעילה ולא מדויקת ולא תמיד מעשית לפתור את הבעיה הזו.

מציאת שורשים של משוואות היא בעיה טבעית. פונקציות לעיתים קרובות מוגדרות לפי הצורך של מציאת השורשים שלהם, למשל נקודת אופטימום של פונקציה היא שורש של פונקצית הנגזרת, וערכים של קבועים חשובים מוגדרים שורשים של פונקציות.

מאחר ולבעיה אין פתרונות אלגבריים (מלבד משוואות עד מעלה 4) הפתרון לבעיה נתונה יהיה כמעט תמיד איטרטיבי. שיטות איטרטיביות מהוות סכמה של שיפור חוזר ונשנה של הערכה לפתרון. מלבד העובדה שאי אפשר לדעת כמה חישובים אנחנו צריכים לעשות, יש בעיות עקרוניות נוספות: רוב השיטות בנויות למציאת שורש בודד כלשהוא של הפונקציה, ואם לפונקציה יש יותר משורש אחד וצריך את כולם או אחד מסוים זו בעיה בפני עצמה. ברוב השיטות הפונקציה הנחקרת צריכה לקיים תנאים מסוימים, וחלק מן השיטות מחייבות תנאי התחלה שמקיימים תנאים מסוימים, ואין דרך כללית למצוא תנאי התחלה כאלה אלא אם הפונקציה מקיימת תנאים נוספים. לעיתים קרובות פתרון משוואה לא לינארית נעשית בשניים או יותר שלבים בכדי לספק לשיטת הפתרון האחרונה את תנאי ההתחלה שהיא צריכה. רוב התוכנות המנסות להתמודד עם בעיות נתונות ללא הכתבת דרישות מהמשתמש בנויות כך. רוב השיטות אינן מקיימות את תכונת "ההתכנסות הגלובלית" כלומר להתכנס מכל הערכה ראשונית לפתרון.

שיטות איטרטיביות מהסוג שאנחנו דנים כאן יעבדו הפונקציות שאנחנו מנסים למצוא את השורשים שלהן חייבות להיות לכל הפחות **רציפות**. תנאים נוספים המקלים על ההתמודדות עם בעיית מציאת השורשים הם אם הפונקציה **מונוטונית** ו/או אם הפונקציה **גזירה** וידוע איך **לחשב את הנגזרת.**

שני משפטים יסודיים שמהווים בסיס לחלק מן האלגוריתם פתרון משוואות הם הבאים:

**משפט 1** : תהי f(x) פונקציה **רציפה**  ויהיו a,b המקיימים a < b ו-f(a)\*f(b) < 0. אזי חייבת להיות לפחות נקודה אחת \*x, a < x\* < b, המקיימת f(x\*) = 0.

**משפט 2**: תהי f(x) פונקציה רציפה **ומונוטונית ממש**. ויהיו a,b המקיימים a < b ו-f(a)\*f(b) < 0. אזי חייבת להיות נקודה **יחידה** \*x, a < x\* < b, המקיימת f(x\*) = 0.

קל להמציא דוגמאות גרפיות לפחות שבה פונקציות לא רציפות אינן עוברות דרך ציר ה-x וכן או שפונקציה רציפה לא מונוטונית עוברת דרך ציר ה-x במספר נקודות מבודדות. אם הפונקציה רק מונוטונית היא יכולה להתלכד על ציר ה-x בקטע. ככלל, פונקציות הנובעות מאפליקציה הנדסית או פיסיקלית שאנחנו מחפשים לו שורש יהיה לכל הפחות רציף, ובדרך כלל מונוטונית, לפחות כאשר מוגבל לקטע. כאשר אנחנו רוצים למצוא שורש של משוואה המשקפת בעיה מעשית, זה בדרך כלל לא הגיוני שלמשוואה יש יותר מפתרון אחד. למשל האם לחוזק אופטימלי של שיטת בניה לא הגיוני בדרך כלל שיהיה לא שתי פתרונות שונים בתכלית.

שיטות איטרטיביות לפתרון משוואות לא לינאריות יש כמה סוגים. צורה קלאסית אחת היא חישוב קירוב חדש על סמך קירוב קודם. באופן כללי שיטות אלה הם מהצורה

**צורה (1):**

בהינתן f(x) ו-x0 חשב סידרה {xn} כאשר

 ***xn+1 = F(xn)***

כאשר למעשה

הפתרון שאנחנו מחפשים הוא הנקודה x\*

***x\* = F(x\*)***

נקודה כזו נקרית **נקודת שבת** fixed point ויש תחום במתמטיקה החוקר תכונות של נקודות כאלו.

שיטות מסוימות משתמשות ביותר מנקודה אחת בכדי לחשב את הנקודה הבאה. הדבר מאפשר לעיתים להפחית מן הדרישות של האלגוריתם.

למשל עבור סכמה של 2 נקודות, נקבל את

**צורה (2):**

בהינתן f(x) x0 ו-x1 חשב סידרה {xn} כאשר

 ***xn+1 = F(xn-1, xn)***

כאשר למעשה

הפתרון שאנחנו מחפשים הוא הנקודה x\*

***x\* = F(x\*, x\* )***

גישה נוספת היא גישת ה**קטע** שבו מתחילים מקטע (a0, b0) שחייבת להחיל את הפתרון ולצמצם אותו עד שההפרש b0 – a0 יהיה קטן מערך קטן רצוי.

**צורה (3):**

בהינתן f(x) וקטע (a0, b0) חשב סידרה { (an, bn) { כאשר

 ***(an+1, bn+1) = F(an, bn)***

אנחנו נראה את ארבע השיטות הכלליות הבסיסיות ביותר לפתרון בעיות מסוג זה:

* שיטת החצייה
* שיטת regula falsi
* שיטת ניוטון
* שיטת המיתר

**שיטת החצייה** היא מצורה 3 שפותרת את הבעיה ע"י קבלת קטע ראשוני (a,b) שבו השורש חייב להימצא ופונקצית המשוואה חייבת להיות בעלת סימנים הפוכים בקצוות. בכל איטרציה השיטה הזו מקטינה את הקטע בחצי. זו שיטה כללית ואמינה בתנאים שלה אולם היא מתכנסת לאט.

**שיטתregula falsi**  היא גם שיטה מצורה 3 אבל מחשבת את הקטע החדש לפי השורש של הקו הישר המחבר את ערכי פונקצית המשוואה בשתי הקצוות.

שיטה זו לא בהכרח מקטינה את הקטע אבל היא מוצאת את השורש יותר מהר ובתנאים שלה היא אמינה.

**שיטת ניוטון** היא שיטה מצורה 1 המניחה שהפונקציה גזירה ונגזרת ידועה או ניתנת לחישוב. שיטה זו מתכנסת כמעט תמיד אם הניחוש הראשוני הוא **מספיק קרוב** לפתרון וכאשר הוא מתכנס הוא בדרך כלל מהרלמדי.

**שיטת המיתר** היא שיטה מסוג 2 היא מעין הפשטה של שיטת ניוטון: היא המוותרת על חישוב הנגזרת ובמקום זאת מסתמכת שני ניחושים בדומה ל- regula falsi מסתמכת על קו ישר המחבר בין הנקודות המוגדרות ע"י שני ערכי הפונקציה בנקודות הללו. תכונות ההתכנסות שלה דומות לאלו של שיטת ניוטון אבל לאט יותר.

אבל לפני כן נעשה כאן ניתוח כללי של שיטות איטרטיביות מצורה 1 אבל ההיגיון תופס גם לצורות האחרות. זה יוביל אותנו לשיטה אמנם לא קונסטרוקטיבית במיוחד של **שיטת נקודות השבת fixed point**. הניתוח הזה ישפוך אור על שיטות איטרטיביות באופן כללי.

**ניתוח שיטות מסוג צורה (1)**

 כאשר אנחנו מנסים לפתח שיטת פתרון מהסוג

בהינתן f(x) ו-x0 חשב סידרה {xn} כאשר

 ***xn+1 = F(xn)***

על מנת להגיע לנקודה x\* המקים

***x\* = F(x\*)***

למעשה

***xn = F(F(F(….(F(x0)))…)))***

החוכמה היא למצוא F כזה המייצר סדרה מתכנסת. לדוגמא, בהינתן f(x) שממנו אנחנו מחפשים x\* המקים f(x\*) = 0, אפשר להגדיר

***F(x) = x + f(x)***

שאכן מקיים

***x\* = F(x\*)***

אך בדרך כלל הוא לא יתכנס.

מקרה נדיר שזה כן יעבוד היא מציאת השורש של המשוואה

***f(x) = cos(x) – x = 0***

באו נבחן מספר דוגמאות.

נניח שאנחנו רוצים לחשב את הפונקציה x√ לכל 0<Mנתון. דרך אחת הוא לחשוב על הבעיה כמציאת שורש חיובי של הפונקציה

***f(x) = x2 – M***

אבל הנוסחה

***xn+1 = F(xn) = xn2 +xn – M***

לא יתכנס בדרך כלל.

לדוגמא, √10= 3.16227766…

אם נתחיל ב-x0 = 4.0 שלא רחוק נקבל

x1 = 10.0

x2 = 100.000000

x3 = 10090.000000

x4 = 101818180.000000

 אם נתחיל מ-x0= 3.0

x1 = 2.000000

x2 = -4.000000

x3 = 2.000000

x4 = -4.000000

אבל גישה אחרת תצליח. ניתן במקרה הזה "לחלץ" את x.

***x2 – M = 0***

***x – M/x = 0***

***f(x) = x – M/x***

***F(x) = f(x) + x = 2x – M/x = x***

***2x = x +M/x***

***x = (x + M/x)/2***

***xn+1 = (xn + M/xn)/2***

מה שבעצם נעשה כאן הוא במקום לקחת את f(x) בתור x2 – M לוקחים את הצורה השקולה

 ***f(x) = x – M/x***

שניהם מתאפסים ב-√M.

***F(x) = x +f(x) = 2x – M/x***

אנחנו מגיעים לנוסחה

***x = 2x – M/x***

גם זה לא נוסחה מתכנסת אבל אפשר "לחלץ" את x לנוסחה אחרת

***x = (x + M/x)/2***

הנוסחה הזו מתכנסת לכל 1 < |x| , x

אנחנו נראה בהמשך שהשיטה הזו מזדהה עם שיטת ניוטון- רפסון.

**קוד C**

 הרוטינה הבאה תחשב √M לכל x0

long double my\_abs(long double x)

{

 if (x < 0)

 return -x;

 else

 return x;

} /\* my\_abs \*/

long double my\_sqrt(long double x0, long double eps)

{

 long double xn;

 if (x0 == 0.0)

 return 0.0;

 xn = x0/2;

 do {

 xn = (xn + x0/xn)/2.0;

 } while( my\_abs(xn\*xn - x0) > eps);

 return xn;

} /\* my\_sqrt \*/

**מתי סכמה של נקודת שבת מתכנסת?**

זהו תחום מחקר מתמטי בפני עצמו. ישנם מספר משפטים חשובים על נקודות שבת. אבל תוצאה יחסית פשוטה ניתן לראות מיד:

תשובה חלקית לשאלה הזו נותן משפט לגרנג' אם נפתח את סביב השורש x\*:

***F(x) = F(x\*) + (x – x\*) F'(x\* + θ (x - x\*))***

אלא ש-

***F(x\*) = x\****

לפיכך אם נציב בתור x את xn ונסמן את x\* + θ (x - x\*) בתור ξ אזי

***F(xn) = x\* + (xn - x\*) F'(ξ)***

אבל F(xn)= xn+1ולפיכך

***xn+1 – x\* = (xn - x\*) F'(ξ)***

אם ניקח ערך מוחלט משני הצדדים של השיויון

***|xn+1 – x\*| = |xn - x\*| |F'(ξ)|***

לפיכך, אם 1 > |F'(ξ)| עבור כל ξ בכל התחום שבין xn ל-x\* אזי xn+1 יהיה יותר קרוב ל- x\* מאשר xn, כלומר הסדרה תתכנס.

לפיכך תנאי מספיק שהסדרה תתכנס הוא שנתחיל או נגיע ל- x0שנמצא בתחום שבו נגזרת F'(x) של הסכמה F(x) מקבל ערכים שבערכם המוחלט קטנים מ-

1.

 למעשה מה שיש לנו בתנאים הללו זה ש-

***|xn+1 – x\*| = |x0 - x\*| |F(ξ1)| |F(ξ2)| …. |F(ξn)|***

 לדוגמא, עבור נוסת השורש

***F(x) = (x + M/x)/2***

***F'(ξ) = (1 – M/ξ2)/2***

כלומר הסדרה מתכנסת

ו-F'(ξ) < 1/2 זאת משום שאם x0> 0 ו- 0 < M אזי xn > 0, לכל 0 < n, ומשום שבהמשך נראה ש-

 ***xn+1 =((xn+M/xn)/2) > √M***

בהנחה שזה האחרון נכון, משתמע ממנו ש- ξ > √M, שכן הוא בין xn+1 לשורש שהוא √M. ולכן

***M/ξ2  < 1***

ולפיכך

***|F'(ξ)| = (1 – M/ξ2)/2 < 1/2 < 1***

בכדי להשתכנע ש-

***((xn+M/xn)/2) > √M***

נסתכל על הפונקציה

***f(x) = (x+M/x)/2***

בתחום 0 < x < ∞. זו פונקציה שיש לה מינימום גלובלי חיובי, שכן f(x) 🡪 ∞ כאשר x 🡪 ∞ ו- x 🡪 0 והנגזרת

***f'(x) = (1 – M/x2)/2***

היא שלילית ליד האפס בשלב מסוים מחליפה סימן ונשארת חיובית. הנגזרת מתאפסת בנקודה

***1 – M/x\*2 = 0***

***x\* = √M***

וערך המינימלי הזה f(x\*)הינו:

 ***f(x\*) = f(√M) = (√M+M/√M)/2 = √M***

לפיכך

***f(x) = (x+M/x)/2 > √M***

לכל 0 < x < ∞ מש"ל.

כלומר השיטה הזו **מתכנסת מכל נקודת התחלה** ואפילו די מהר. שיטה המתכנסת מכל נקודת התחלה הם נקראות **מתכנסות גלובלית** והתכונה הזו נדירה יחסית.

מהסיבה הזו גם ניסיון לפתור את המשוואה cos(x) – x = 0 יצליח משום שבמקרה הזה F(x) = cos(x) ו- F'(x) = -sin(x) ו- 1 >|F'(x)| בערכים שנקבל אם נחשב

***xn = cos(cos(cos( …. cos(x) )))) ...)***

תהליך שבודאי מוכר למי שהשתעשע עם מחשב כיס.

התשובה אגב, תלויה בשיטת הייצוג של הזוית:

ברדיאנים הפתרון היא בערך 0.7390851 ובמעלות 0.9998477.

**שיטות קלאסיות למציאת שורשים למשוואת לא לינאריות**

**שיטת החצייה bisection**

****

 שיטת החצייה הוא אלגוריתם מצורה 3 של מציאת שורש ע"י . שיטת החצייה היא שיטה טבעית ופשוטה: בהינתן פונקציה רציפה f(x) ואינטרוול (a,b) שבו f(a)f(b) < 0, לפי משפט 1 חייב להיות לפחות שורש

 אחד x\* בתחום (a,b) וקל להקטין את הקטע בחצי:

* נחשב x' = (a+b)/2
* אם f(x')| < ε| מצאנו את השורש
* אחרת אם f(x')f(a) < 0 אזי השורש נמצא בתחום (a,x') ולפיכך b = x'
* אחרת אזי השורש נמצא בתחום (x',b) ולפיכך a = x'

**תנאים הכרחיים:** הפונקציה חייבת להיות רציפה ויש צורך לחשב או לקרב אותה בכל נקודה. היא גם צריכה להיות גם שלילית וגם חיובית בסביבות השורש! את השורש של פונקציה **שרק נושקת לציר ה-x לא** ניתן לפתור בשיטה הזו. אולי הקושי העיקרי הוא שצריך לספק לאלגוריתם את הקטע הראשוני. אם רוצים להוריד את הדרישה הזו מהמשתמש יש צורך לשלב את האלגוריתם הזה עם גישה אחרת. עבור פונקציה מונוטוניות לא קשה למצוא קטע ראשוני אבל על פונקציות כלליות יותר הבעיה קשה יותר.

חסרון נוסף של השיטה שהיא מתכנסת יחסית לאט.

 לשיטה יש יתרון משמעותי אחד: בהינתן קטע ראשוני מתאים הוא כמעט תמיד מתכנס, למעט כאשר שגיאות חישוביות מטעות אותו לגבי סימן של ערך הפונקציה.

**קוד C**

long double bisection( long double (\*fun)(long double), long double a, long double b, long double eps)

{

 long double x, f;

 do {

 x = (a+b)/2.0;

 f = (\*fun)(x);

 if (fabsl(f) < eps)

 return x;

 if ( (\*fun)(a)\*f < 0.0)

 b = x;

 else

 a = x;

 } while( fabsl(b-a) > eps);

 return x;

} /\* bisection \*/

**שיטת הנקודה המדומה regula falsi**

 שיטת regula falsi היא שיטה דומה מאד לשיטת החצייה (bisection) למעט שנקודת הביניים לא נבחרת באופן שירותי באמצע הקטע. במקום זאת מותחים קו ישר בין הנקודות (a, f(a)) ו-(b, f(b)) ו-x יהיה נקודת החיתוך של הקו הישר הזה עם ציר ה-x.

נוסחת הקו הישר הזה הינו

***y - f(b) = ((f(b) – f(a))/(b – a))(x – b)***

אנחנו מחפשים את x' שבו y = 0 לפיכך:

***-f(b) = ((f(b) – f(a))/(b – a))(x' – b)***

 או

***x' = b + f(b)(a-b)/(f(b)-f(a)) =***

 ***( bf(b) – bf(a) + f(b)a – f(b)b)/(f(b) – f(a)) =***

 ***(af(b) – bf(a))/(f(b) – f(a))***

לפיכך הנוסחה עבור x' הוא

***x' = (f(b)a – bf(a))/(f(b) – f(a))***

מעבר לחישוב x' ההמשך הוא בדיוק כמו בשיטת החצייה:

* אם f(x')| < ε| מצאנו את השורש
* אחרת אם f(x')f(a) < 0 אזי השורש נמצא בתחום (a,x') ולפיכך b = x'
* אחרת אזי השורש נמצא בתחום (x',b) ולפיכך a = x'

 בשיטה הזו יש מקום לתקווה שהיא תתכנס יותר מהר, משום שהנקודה x' נבחרת על סמך נתונים מהפונקציה הנחקרת. לדוגמא, אם |f(a)| < |f(b)| סביר להניח שהשורש יותר קרוב ל-a מאשר ל-b. השיטה הזו תתחשב בזה ואילו שיטת החצייה לא.

דרך אחרת להסתכל על השיטה הזו הוא מעין משחק בנדמה לי. אנחנו לא יודעים למצוא שורשים של משוואות פונקציונליות כלליות אבל אנחנו יודעים למצוא שורשים של קוים ישירים. אנחנו לוקחים את הקו הישר בתור "קירוב" למשוואה שאותו אנחנו רוצים לפתור, ומוצאים את השורש שלו. הגישה הזו תהיה מדויקת (ותתכנס בצעד אחד) אילו המשוואה הנחקרת היתה מלכתחילה קו ישר. עבור משוואות אחרות זה יהיה קירוב חדש לפתרון. הלוגיקה הזו, חוזרת במספר הקשרים באנליזה נומרית.

בניגוד לשיטת החצייה, איטרציות של שיטת regula falsi לא בהכרח מקטינות בצורה דרסטית את הקטע, כפי שאפשר ללמוד מדוגמאות. בניגוד לשיטת החצייה, הבדיקה אם f(x')| < ε| **היא קריטית**.

**קוד C**

long double regula( long double (\*fun)(long double), long double a,

long double b, long double eps)

{

 long double x, f;

 do {

 x = (a\*(\*fun)(b) - b\*(\*fun)(a))/((\*fun)(b) - (\*fun)(a));

 f = (\*fun)(x);

 if (fabsl(f) < eps)

 return x;

 if ( (\*fun)(a)\*f < 0.0)

 b = x;

 else

 a = x;

 } while( fabsl(b-a) > eps);

 return x;

} /\* regula \*/