**אופטימיזציה**

מחלקה מיוחדת באנליזה נומרית היא האופטימיזציה, שבמידה מסוימת מיוחסת לתחום מיוחד של מתמטיקה שימושית הנקרא חקר ביצועים.

אופטימיזציה מתחלקת למספר לא מבוטל של תת תחומים: תכנון לינארי, תכנון לא לינארי, אופטימיזציה במימד אחד, אופטימיזציה בהרבה מימדים, אופטימיזציה מאולצת ולא מאולצת. מכל זה נדון רק במה שאפשר לקרוא **אופטימיזציה או תכנון לא לינארי לא מאולץ במימד אחד**.

באופן עקרוני מה שאנחנו בוחנים כאן הוא למצוא את נקודת המינימום של פונקציה לא לינארי והתנאי היחיד יהיה (במידה ויהיה תנאי כלשהוא) שהמינימום מוגבל לקטע מסוים.

כלומר הבעיה שלנו הוא

***Min f(x)***

***x***

או

***Min f(x)***

***a<x<b***

מציאת אופטימום של פונקציות כלליות היא בעיה קשה למדי, ולא לחינם מדובר בתחום מחקרי בפני עצמו. עבור פונקציות כלליות קשה אפילו לוודא אופטימליות של נקודת אופטימום אפילו היא נתונה. כפי שנראה, השיטות הבסיסיות למציאת אופטימום מניחות הנחות חזקות על פונקצית המטרה.

אם מניחים שפונקצית המטרה גזירה ופונקצית הנגזרת ידועה (ואלו הנחות מגבילות על פונקצית המטרה), אז ידוע שבנקודת האופטימום הנגזרת מתאפסת. אבל גם שיטות המחפשות אופטימום על בסיס מציאת שורשים של הנגזרת אינן בהכרח פוטרות את הבעיה על פונקציה כללית: שורש של הנגזרת יכולה להיות אופטימום **מקומי** בלבד ופעמים אפילו לא זה: היא יכולה להיות **נקודת עוקף** (saddle point) שבו הפונקציה יכולה גם לעלות וגם לרדת סמוך לנקודת השורש של הנגזרת.

יש תנאים **הכרחיים** לאופטימום מקומי (f'(x\*) = 0) ויש תנאים **מספיקים** לאופטימום מקומי (למשל (f'(x\*) = 0 ו-f''(x\*) >0) אבל **לא** קיימים תנאים **הכרחיים ומספקים** גם יחד. עבור פונקציות כלליות כנראה שאין פתרון גורף לבעיות הללו. מבחינה מעשית שיטות המבוססות על מציאת שורשים של הנגזרת בעצם מניחות שהפונקציה קמורה (convex function) ועבור בעיות הנדסיות זה בדרך כלל המקרה (או שניתן להביא אותו למצב כזה).

אנחנו נצא מתוך הנחה שפונקציה מקיימת תכונה שנקראת strictly unimodal יש לה מינימום יחיד ומינימום לוקלי יהיה גם גלובלי. אנחנו גם נתעלם מנושא נקודות העוקף. שטח האופטימיזציה באופן כללי מנסה להתמודד עם הנושאים הללו, במידה זו או אחרת, אבל לא נכנס לזה כאן. למען האמת מבחינה מעשית אנחנו יוצאים מנקודת הנחה שהפונקציה קמורה לפחות בסביבת האופטימום שאותו אנחנו רוצים למצוא. כאשר אנחנו נדרשים לפתור בעיה מסוג זה בהקשר הנדסי או פיסיקלי, פונקצית המטרה בדרך כלל תקים תנאים מסוג זה.

לכאורה בכדי להתמודד עם הבעיה הזו אין צורך באלגוריתמים חדשים, שכן נקודות אופטימום לוקלי של פונקציות גזירות ניתן לאתר על ידי מציאת שורשים של הנגזרת. ואכן, חלק מהשיטות הנפוצות ביותר במינימיזציה הם להשתמש בשיטות הרגילות (חציה, regula falsi, ניוטון, מיתר) למציאת שורשים של פונקצית הנגזרת. למשל, מציאת הפתרון של הבעיה

***min –sin(x)***

***0<x <л/2***

יכול להפטר ע"י bisection או regula falsi בין 0 ל- л/2, או שיטת ניוטון

לפי הנוסחה

***xk+1 = xk – f'(xk)/f''(xk) = xk + cos(xk)/sin(xk)***

לשימוש בשיטת ניוטון ושיטות מציאת שורשים האחרות לצורך אופטימיזציה – מציאת שורשים של הנגזרת – יש יתרון על מציאת שורשים באופן כללי שכן פונקציות נגזרת חייבות לקיים תנאים מסוימים. זאת משום שלא כל פונקציה יכולה להיות נגזרת של פונקציה אחרת. שיטות החצייה ו-regula falsi מחייבות שפונקצית המטרה תחליף סימן אבל בנקודות אופטימום זה **תמיד** קורה – בניגוד לפונקציות כלליות שיכולות רק להשיק את ציר ה-x בפתרון.

יחד עם זאת חלק מהחסרונות של השיטות הללו באים לידי ביטוי גם כאן: שיטות ניוטון והמיתר לא תמיד מתכנסות.

לשיטת ניוטון ושיטת המיתר לאופטימיזציה יש גם משמעות גאומטרית של **התאמת עקומות**.

**אופטימיזציה ע"י התאמת עקומות**

**שיטת ניוטון**

שיטת ניוטון יכולה להיות מוצגת כך שמעברים פרבולה q(x) (פולינום מסדר שני) מסוימת בנקודה xk, מוצאים את המינימום x\*של הפרבולה ולוקחים x\* xk+1 = הקירוב הבא למינימיום של פונקצית המטרה f(x). התכונות ש-q(x) מקיימת הן ש- q(xk) ערך הפרבולה בנקודה xk, q'(xk) ערך הנגזרת שלה בנקודה xk ו- q''(xk)ערך הנגזרת השניה שלה בנקודה xk מזדהים עם f(xk) ערך הפונקציה בנקודה xk, f'(xk) ערך הנגזרת שלה בנקודה ו- f''(xk) ערך הנגזרת השנייה שלה בנקודה xk. כלומר:

***q(xk) = f(xk)***

***q'(xk) = f'(xk)***

***q''(xk) = f''(xk)***

***q(x) = f(xk) + f'(xk)(x – xk) + (1/2)f''(xk)(x – xk)2***

הוכחה:

את הנוסחה של פרבולה ניתן להציג כ-

***q(x) = c + b(x – xk) + a(x – xk)2***

אזי מ- ***q(xk) = f(xk)*** נובע ***c = f(xk)***,

מ- ***q'(xk) = f'(xk)*** נובע ***b = f'(xk)***,

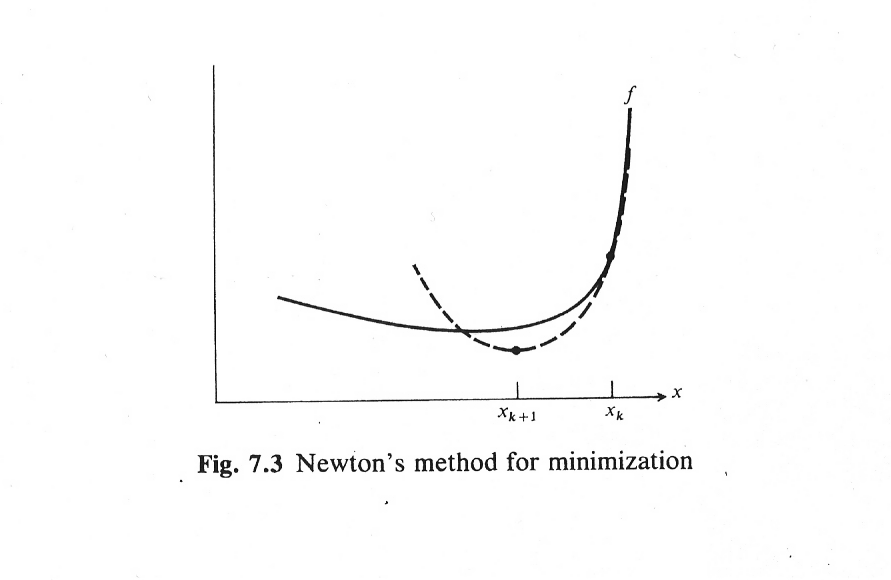
מ- ***q''(xk) = f''(xk)*** נובע ***2a = f''(xk)*** או ***a =1/2 f''(xk)***.

q(x) מקבלת מינימום בנקודה ***x\****

***q'(x\*) = f'(xk) – f''(xk)(x\*-xk) = 0***

***x\* = xk – f'(xk) /f''(xk)***

גראפית השיטה נראית כך:



הרעיון הוא שאנחנו שהעקום הרציף היא הפונקציה "האמיתית" שאותו אנחנו מחפשים את המינימום שלה, והעקום המרוסק היא הפרבולה שהמינימום שלה הוא הקירוב החדש לפתרון. הפרבולה מזדהה עם ערך הפונקציה האמיתית ב-xk והנגזרות (משיקים) שלהן בנקודה xk מזדהות. מאחר והפונקציה האמיתית והפרבולה מזדהה גם בערך הנגזרת השניה בנקודה xk , שתי הפונקציות כמעט זהות סמוך לנקודה xk.

**שיטת המיתר**

שיטת המיתר מתקבלת ע"י 2 נקודות. אנחנו מעברים פרבולה q(x) דרך הנקודה xk כך ש- f(xk)=q(xk) q'(xk)=f'(xk) , ובתור q''(xk)ערך לוקחים את השיפוע של הקו הישר בין שתי הנקודות (xk,f'(xk)) ו- xk-1,f'(xk-1))).

כלומר הפרבולה מתקבלת ממערכת המשוואות:

***q(xk) = f(xk)***

***q'(xk) = f'(xk)***

***q''(xk) = ( f'(xk)- f'(xk-1))/ (xk - xk-1)***

לפיכך, לפי אותו אלגברה כמו בשיטת ניוטון, נקבל ש-

***q(x) = f(xk) + f'(xk)(x – xk) +***

***(1/2)((f'(xk-1) - f'(xk))/(xk-1 – xk)) (x – xk)2***

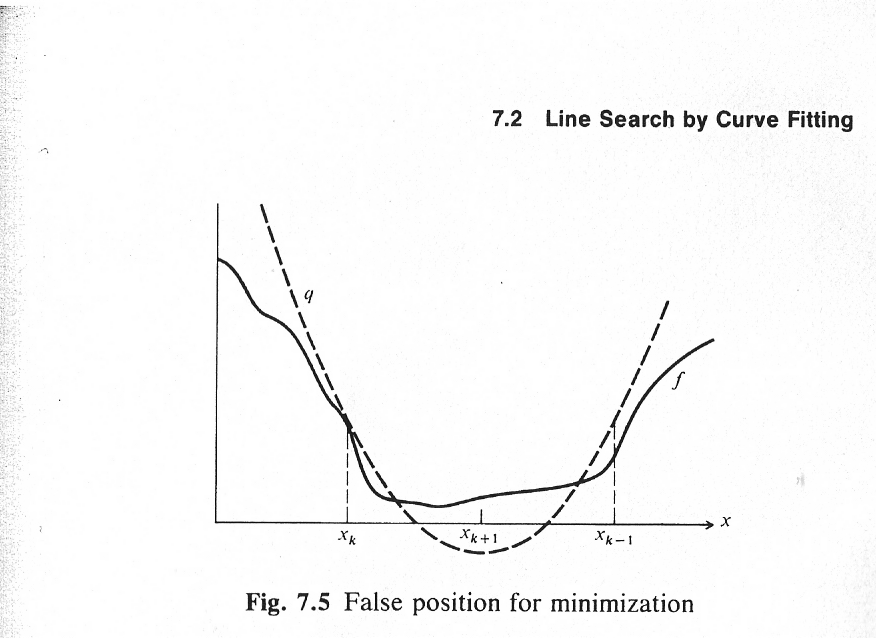
היא מקבלת מינימום בנקודה

***q'(x\*) = f'(xk) – ((f'(xk-1) - f'(xk))/(xk-1 – xk))(x\*-xk) = 0***

***x\* = xk – f'(xk) /((f'(xk-1) - f'(xk))/(xk-1 – xk)) =***

***= xk – f'(xk) (xk-1 – xk) /((f'(xk-1) - f'(xk))***

גראפית השיטה נראית כך:



שוב: בעוד שהעקום הרציף היא הפונקציה האמיתית שאת המינימום שלה אנחנו מחפשים, העקום המרוסק הא הפרבולה שהמינימום שלה משמש לקירוב הבא (xk+1) לפתרון. הפרבולה מזדהה עם ערך הפונקציה האמיתית בנקודה xk והנגזרות (משיקים) שלהן גם כן מזדהות בנקודה xk. הנגזרות השניות שלהן שונות, ולכן ערכי הפונקציה והפרבולה יכולות להפרד כמעט מיד ליד xk.

גישה נוספת להתאמת פרבולה היא שיטת התאמת הריבוע.

**שיטת התאמת ריבוע Quadratic fit**

בגישה הזו מתאימים פרבולה על ערכי הפונקציה בשלוש נקודות. יתרונותיה הם שאין צורך לחשב נגזרות כלשהן בשיטה הזו ועם עומדים בתנאי ההתחלה היא תתכנס בכל מקרה. חסרונה היא שצריך כתנאי התחלה לחסום בקטע את נקודת המינימום, כלומר לדעת חסם עליון וחסם תחתון שלו. בהינתן קטע (x1, x3) שהמינימום חייב להימצא בו,והערכה למינימום x2, x1< x2< x3, אפשר לשפר את ההערכה למינימום על ידי העברת פרבולה שמזדהה עם ערכי הפונקציה בשלוש הנקודות x1, x2, x3, כלומר q(xi) = f(xi), i = 1, 2, 3, ונמצא את המינימום של q(x). כאשר נעשה זאת, נקבל נקודה x4 שאמור להיות (ובדרך כלל יהיה) קירוב יותר טוב למינימום מאשר x2, כלומר .f(x4) < f(x2) באיזה מידה כן או לא תלוי באיזה מידה פונקצית המטרה f(x) מתנהגת כמו פרבולה, אבל נראה שהאלגוריתם יכול להתמודד גם עם זאת.

אם נכתוב את הפרבולה לפי שיטת לגרנז:

***q(x) = f(x1)(x-x2)(x-x3)/((x1-x2)(x1-x3)) +***

***f(x2)(x-x1)(x-x3)/((x2-x1)(x2-x3)) +***

***f(x3)(x-x1)(x-x2)/((x3-x1)(x3-x2))***

נסמן

***fi = f(xi), aij = xi - xj, i, j = 1, 2, 3,***

לפיכך

***q(x) = f1(x-x2)(x-x3)/(a12a13) +***

***f2(x-x1)(x-x3)/(a21a23) +***

***f3(x-x1)(x-x2)/(a31a32)***

אם נגזור את q לפי x ונשווה לאפס נקבל

***q'(x\*) = 0 =***

***f1(x\*-x3)/(a12a13) + f1(x\*-x2)/(a12a13) +***

***f2(x\*-x3)/(a21a23) + f2(x\*-x1)/(a21a23) +***

***f3(x\*-x2)/(a31a32) + f3(x\*-x1)/(a31a32)***

***x\*2( f1/(a12a13) + f2/(a21a23) + f3/(a31a32) ) =***

***f1x3/(a12a13) + f1x2/(a12a13)) +***

***f2x3/(a21a23) + f2x1/(a21a23)) +***

***f3x2/(a31a32) + f3x1/(a31a32))***

נכפיל את שני האגפים ב- a12a23a13כאשר נבוא בחשבון ש-aij= -aji

***x\*2(a23f1 + a31f2 + a12f3) =***

***f1x3a23 +f1x2a23 + f2x3a31 +f2x1a31 + f3x2a12 + f3x1a12 =***

***f1x3a23 +f1x2a23 + f2x3a31 +f2x1a31 + f3x2a12 + f3x1a12 =***

***f1a23(x3+x2) + f2a31(x3+x1) + f3a12(x1+x2)***

אבל

***a23(x3+x2) = (x2-x3) (x2+x3) = x22-x32***

***a31(x3+x1) = (x3-x1) (x3+x1) = x32-x12***

***a12(x1+x2) = (x1-x2) (x1+x2) = x12-x22***

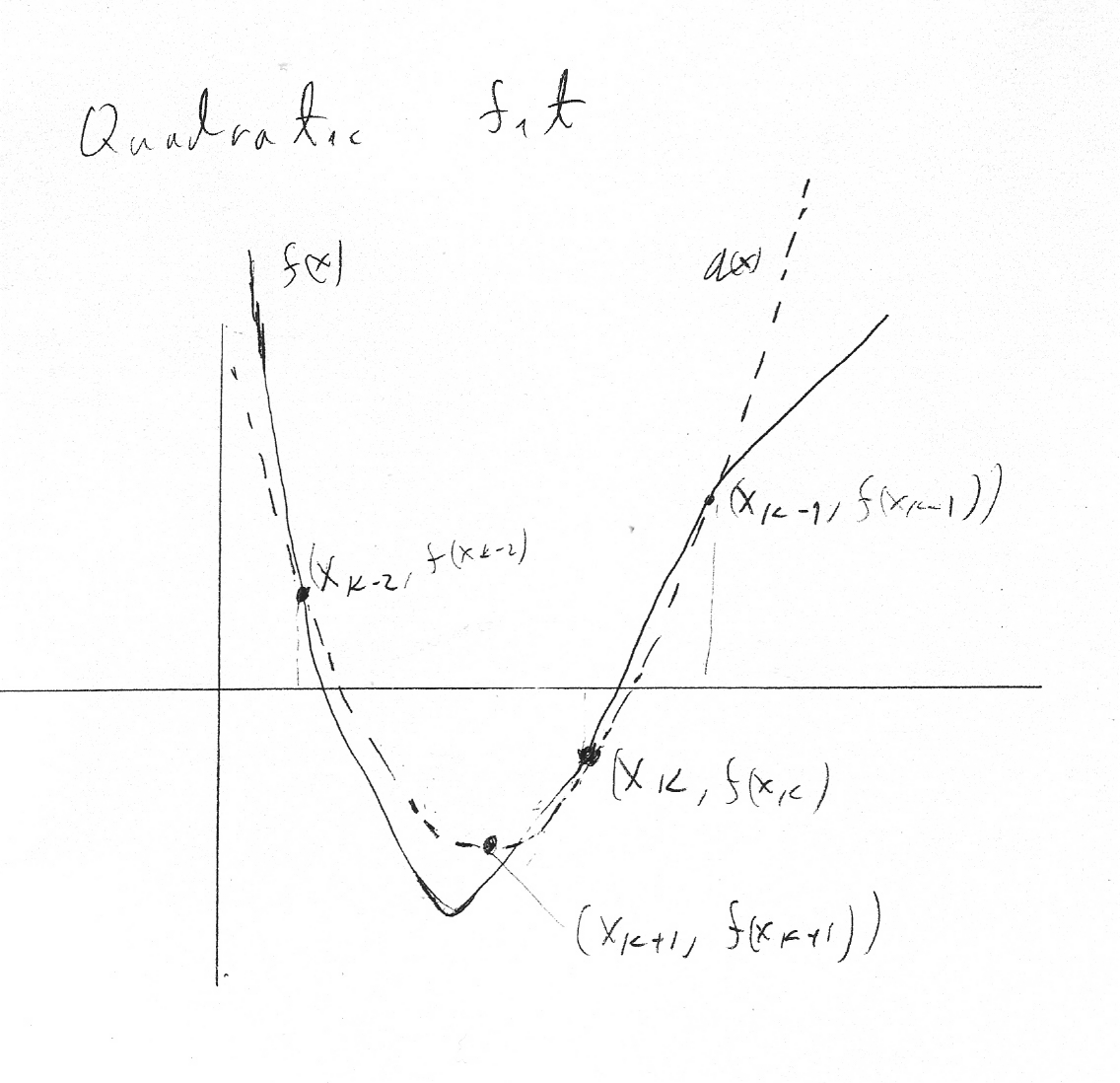
נסמן

***bij = xi2 –xj2***

נקבל

***x\*= 1/2(f1b23 + f2 b31 + f3 b12)/(a23f1 + a31f2 + a12f3)***

גראפית זה נראה כך:



כמו קודם, העקום הרציף הינה הפונקציה האמיתית, העקום המרוסק היא הפרבולה שהמינימום שלה הוא הקירוב הבא (xk+1) לפתרון. ערכי הפרבולה מזדהה עם ערכי הפונקציה האמיתית בנקודות xk-2, xk-1, xk.

**הקטנת הקטע והתכנסות גלובלית**

שיטת ההתאמה הריבועית היא סכמת קטע, שבו מתחילים עם קטע שבו המינימום נמצא ומקטינים אותו. סכמה כזו יכולה בתנאים מסוימים להתכנס תמיד.

אנחנו מתחילים עם שלישיה (x1, x2, x3), x1 ו-x3 הם קצוות הקטע ו-x2 הערכה למינימום.

אנחנו מחשבים הערכה חדשה למינימום x4.

תחת ההנחות ש-f(x3) f(x1) > f(x2) < הנקודה 4xתהיה בתוך הקטע:

x1 < x4 < x3.

אם x1 < x4 < x2:

אם :f(x4) < f(x2)

השלישיה החדשה תהיה (x1, x4, x2)

אחרת (((f(x4) > f(x2 :

השלישיה החדשה תהיה (x4, x2, x3)

אחרת x2 < x4 < x3:

אם :f(x4) < f(x2)

השלישיה החדשה תהיה (x2, x4, x3)

אחרת (((f(x4) > f(x2 :

השלישיה החדשה תהיה (x1, x2, x4)

**קוד C**

long double quad(long double (\*fp)(long double), long double x1, long double x3, long double eps)

{

long double fx1, x2, fx2, fx3, x4, fx4;

long double a12, a31, a23, b31, b23, b12;

long double oldx4;

x2 = (x1 + x3)/2;

fx1 = (\*fp)(x1);

fx2 = (\*fp)(x2);

fx3 = (\*fp)(x3);

x4 = x2;

do {

a12 = x1 - x2;

a23 = x2 - x3;

a31 = x3 - x1;

b12 = x1\*x1 - x2\*x2;

b31 = x3\*x3 - x1\*x1;

b23 = x2\*x2 - x3\*x3;

oldx4 = x4;

x4 = 0.5\*(b23\*fx1 + b31\*fx2 + b12\*fx3)/(a23\*fx1 + a31\*fx2 + a12\*fx3);

fx4 = (\*fp)(x4);

if (x4 < x2)

if (fx4 < fx2 ) /\* (x1, x4, x2) \*/

{

x3 = x2;

x2 = x4;

fx3 = fx2;

fx2 = fx4;

} /\* if \*/

else /\* (x4, x2, x3) \*/

{

x1 = x4;

fx1 = fx4;

} /\* else \*/

else /\* x2 <= x4 \*/

if (fx4 < fx2 ) /\* (x2, x4, x3) \*/

{

x1 = x2;

fx1 = fx2;

x2 = x4;

fx2 = fx4;

} /\* else \*/

else /\* (x1, x2, x4) \*/

{

x3 = x4;

fx3 = fx4;

} /\* else \*/

} while ( fabs(oldx4 - x4) > eps);

return x4;

} /\* quad \*/