**דוגמאות ראשונות**

חישוב ערך של נגזרת (0x)f' של פונקציה נתונה x))f בנקודה נתונה 0x.

תנאים הכרחיים: יכולת לחשב את f(x) בכל נקודה בסביבות x0.

**מוטיבציה**: לעיתים ידוע ערך של פונקציה מתוך מדידות, לוחות או תוכנת מחשב, אבל הוא לא ידוע באופן אנליטי. ערך נגזרת בנקודה נתונה נחוץ לכל מיני אלגוריתמים, כמו למציאת שורש הפונקציה בשיטת ניוטון-רפסון או מציאת מינימום בשיטות דומות. כמו כן זה יכול לעזור בהבנת הפונקציה.

**רקע מתמטי:**

 עבור פונקציה גזירה, ההגדרה של הנגזרת מספקת לנו את הנוסחה לקרב אותו:

 ***f'(x) = lim f(x+h)-f(x-h)***

 ***h🡪0 2h***

 ***f'(x) = lim f(x+h)-f(x)***

 ***h🡪0 h***

 ***f'(x) = lim f(x)-f(x-h)***

 ***h🡪0 h***

לפיכך אפשר לקרב את f'(x) על ידי חישוב כל אחד מהנוסחאות עם ערך h מספיק קטן. לדוגמא מהגבול הראשון אנחנו מקבלים

 ***f'(x) ≈ f(x+h)-f(x-h)***

 ***2h***

השאלה היא רק איזה h לבחור? ככל שנבחר h קטן יותר הקירוב יהיה מדויק יותר עד שניתקל בבעיה של הדיוק המוגבל של ייצוג מספרים במחשב. לדוגמא אם אנחנו עובדים בדיוק של 6 ספרות עשרוניות ו-x=1000.0 ו-h=0.0001 אזי החישוב של x+h ו-x-h יהיה שווה ל-x והחישוב של קירוב ל-f'(x) יהיה אפס.

 בכל מקרה חייבים לבחור את h **כמספר יחסי ל-x** כי "קטן" תלוי ברמת הדיוק וגודל המספרים שאיתם עובדים והשאלה היא רק עד כמה להקטין אותו. דרך אחת להתמודד עם הבעיה היא ולהקטין את h עד שהערכים העוקבים של החישוב הקירוב של f'(x) יהיו בהפרש קטן בערך מוחלט מאפסילון רצוי, שמשמעותו שהערכים המחושבים התחילו להתכנס. גם כאן אפסילון חייב להיות יחסית לערכים של f'(x). ברוטינת C הבאה לחישוב הקירוב נלקח ערך התחלתי h=x/2.0 ועל כל חישוב נוסף חוצים את h בשניים.

double approx\_fderiv(double (\*f)(double), double x, double eps)

{

 double h, fd0, fd1;

 if (x != 0)

 h = x/2.0;

 else

 h = 0.5;

 fd1 = ((\*f)(x+h) - (\*f)(x-h))/(2\*h);

 do {

 fd0 = fd1;

 h = h/2.0;

 fd1 = ((\*f)(x+h) - (\*f)(x-h))/(2\*h);

 } while(fabs(fd0 - fd1) > eps );

 return fd1;

} /\* approx\_deriv \*/

 גישה מתוחכמת קצת יותר לחישוב h הוא, שוב, לבחור ערך יחסית ל-x ולהמשיך להקטין אותו כל עוד הקטנות של h משפרות את הקירוב ולהפסיק כאשר זה כבר לא כך. אפשרות אחת היא לבדוק בין ששלושה חישובים עוקבים של הקירוב אם ההפרש בין הקירוב האחרון ללפני האחרון קטן יותר מהקירוב הלפני אחרון לזה שקדם לו. כל עוד ההפרשים הולכים וקטנים החישובים של הקירוב (יש להניח) הולכים ומתכנסים, ברגע שההפרש נעשה יותר גדול יש להניח שעברנו את ה-h האופטימלי.

להלן פונקציה המממשת את הרעיון האחרון:

double approx\_fderiv(double (\*f)(double), double x)

{

 double h, fd0, fd1, fd2;

 if( x == 0)

 h = 1/2.0;

 else

 h = x/2.0;

 fd1 = ((\*f)(x+h) - (\*f)(x-h))/(2\*h);

 h = h/2.0;

 fd2 = ((\*f)(x+h) - (\*f)(x-h))/(2\*h);

 do {

 fd0 = fd1;

 fd1 = fd2;

 h = h/2.0;

 fd2 = ((\*f)(x+h) - (\*f)(x-h))/(2\*h);

 } while(fabs(fd0 - fd1) > fabs(fd1 - fd2) );

 return fd1;

} /\* approx\_deriv \*/