מעבדה באופטימיזציה

חורף תשפ"ו

תרגיל מספר 1

אחד הדרכים לשפר את הדיוק הנומרי של אלגוריתם הוא ע"י טכניקה הנקראת scaling להעביר חלק מחישובי הביניים לתחום (-1.0, 1.0) שם הצפיפות של המספרים המיוצגים היא מרבית. רק לעיתים אפשר לעשות את הטריק הזה. אחד המקומות שאפשר הוא בחישוב נומרי של ערכי פולינומים בנקודה.

חישוב ערכי פולינומים Pn(x) בנקודה נתונה x0 בדיוק גבוה הוא עניין (באופן מפתיע אולי) בעייתי למדי מבחינה נומרית. יש מספר סכמות, חלקן די מורכבות, להתמודד עם הבעיה. אנחנו נממש סכמה יחסית פשוטה של scaling להתמודדות חלקית עם הבעיה בכדי להמחיש את הנושא:

בהינתן מקדמים של פולינום, מצא את הערך המרבי cmax של הערכים המוחלטים של המקדמים.

אם cmax > 1, חלק את כל המקדמים ב-cmax בכדי לקבל פולינום Pn/cmax(x) . חשב את ערך הפולינום Pn/cmax(x)בנקודה המקורית x0 כלומר Pn/cmax(x0).

החזר את cmax\* Pn/cmax(x0) בתור ערך הפולינום Pn (x0).

לדוגמא, אם n=4 וערכי הפולינום הם

double coeffs[5] = {3, -5, 1, -2, 4};

*אזי* **cmax = 5***נחשב את המקדמים הזמניים*

double temp\_coeffs[5] = {0.6, -1, 0.2, -0.4, 0.8};

*אם נניח* **x0 = 1** *אזי נחשב*

Pn/cmax(1) = 0.2

Pn (1) = 5\*0.2 = 1;

*עליך לממש שתי רוטינות שהכרזתן:*

**double compute\_polynom(double coeffs[], double x, int n)**

d**ouble scaled\_compute\_polynom(double coeffs[], double x, int n)**

*לדוגמא, פלט אפשרי של התוכנית הבאה:*

**int main()**

**{**

**double coeffs[9] = {10.1, 0.0, -20.2, 0.0, 30.3, 0.0, -60.6};**

**double x, val1, val2;**

**x = sqrt(2.0);**

**val1 = compute\_polynom(coeffs, x, 8);**

**val2 = scaled\_compute\_polynom(coeffs, x, 8);**

**printf("val1 = %lg, val2 = %lg\n", val1, val2);**

**return 0;**

**} // main**

*תהיה:*

**val1 = 1.42109e-014, val2 = 1.34559e-014**

הערת אגב:

כנראה שלא בכל מקרה scaling משפר את הדיוק.