הקדמה

אנו יכולים להציג מספרים באמצעות בסיסים שונים. הבסיס המוכר לנו הוא הבסיס העשרוני (המשתמש 10 ספרות 9,...,1,0). המחשב משתמש בבסיס בינארי (2 ספרות בלבד).

1. נעביר בסיס בינארי לבסיס עשרוני:

$$1001_2 = 2^3\cdot 1 + 2^2\cdot 0 + 2^1\cdot 0 + 2^0\cdot 1 = 2^3 + 1 = 9_{10}$$

$2^3\ 2^2\ 2^1\ 2^0$

הערה: בשברים, נפעל באופן זהה כאשר החזקה של 2 קטנה בקפיצות של 1 בכל פעם לצד ימין.

2. נעביר בסיס עשרוני לבסיס בינארי:

מספרים שלמים ⇐ נחלק ב-2

שארית	המס. / 2	המספר
0	28	56
0	14	28
0	7	14
1	3	7
1	1	3
1	0	1

החץ כלפי מעלה כי מחלקים ב-2

$$\Rightarrow 56_{10} = 111000_2 = 38_{Hex}$$

שברים ⇐ נכפיל ב-2

שארית	מספר*2	שלם
.375	0	0
.75	0.750	0
.5	1.5	1
.0	1	1

החץ כלפי מטה כי כופלים ב-2

$$\Rightarrow 0.375_{10} = 0.011_2 = 0.6_{Hex}$$

3. נעביר בסיס **(משולש)** טרנארי לבסיס עשרוני:

$$1001_3 = 3^3\cdot 1 + 3^2\cdot 0 + 3^1\cdot 0 + 3^0\cdot 1 = 3^3 + 1 = 28_{10} = 1C_{Hex}$$

$3^3\ 3^2\ 3^1\ 3^0$

הערה: בשברים, נפעל באופן זהה כאשר החזקה של 3 קטנה בקפיצות של 1 בכל פעם לצד ימין.

4. נעביר בסיס עשרוני לבסיס **(משולש)** טרנארי:

מספרים שלמים ⇐ נחלק ב-3

שארית	המס. / 3	המספר
2	18	56
0	6	18
0	2	6
2	0	2

החץ כלפי מעלה כי מחלקים ב-3

$$\Rightarrow 56_{10} = 2002_3$$

שברים ⇐ נכפול ב-3 ונפרד שבר

שארית	שלם	מספר*3
.375	0	–
.105	1	1.105
.315	0	0.315
.945	0	0.945
.835	2	2.835
.505	2	2.505

החץ כלפי מטה כי כופלים ב-3

$$\Rightarrow 0.375_{10} = 0.10022...{}_3$$

פקודות ב-*Maple* וב-*Matlab*:

Matlab	*Maple*
$>> dec2bin(a)$	$> convert(a, decimal, 2)$
$>> bin2dec(b)$	$> convert(a, binary)$

דוגמה ב-*Maple*	דוגמה ב-*Matlab*	
> r1:=convert(convert(12345.7, binary),decimal,2); > r2:=convert(convert(12345.6, binary),decimal,2); > r:=r1-r2; $r1 := 12336.$ $r2 := 12336.$ $r := 0.$	» 12345.7-12345.6 -0.1 *ans =* *3.6379e-013*	» 12345.7-12345.6 *ans =* *0.1000*

אריתמטיקה של נקודה צפה

כיצד נוכל להציג מספרים בזיכרון המחשב?

תא של אחסון מספר מסוים במחשב *binary single precision (32 bits)*
נראה כך:

סה"כ: 32 מקומות ב**IEEE 754** תקנון בפורמת *binary single precision (32 bits)*

את המספר x נוכל לפענח באופן הבא: $x = (-1)^s \cdot (mantissa) \cdot 2^{\exp-127}$.
כך שלדוגמא:
$36_{10} = 100100_2 = 1.001_2 \cdot 2^{101_2}$ $\Leftarrow$ ($mantissa$ =001) , (סימן s=0), והמעריך הוא:
$(\exp = 127_{10} + 5_{10} = 10000100_2)$ ועז מקבלים הצגת מספר 36.0_{10} במחשב בצורה :

+/-	מעריך=132								שבר (אך בלי סיפרה 1 הראשונה)=001																							
0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

כדי להימנע בשגיאות חישובים, המחשב משתמש באלגברה שונה מזו המוכרת לנו.
ב-*Maple*, לדוגמה, $1/n$ הוא מספר באלגברה רציונאלית, בעוד $1./n$ הוא מספר השייך לאריתמטיקה של נקודה צפה.
ולכן, במחשב <u>אין שגיאות</u>, מכוון שהוא משתמש באלגברה *(FOATING POINT ARITHMETIC)* שונה מהסוג המוכר לנו.

Sign Bit	**Exponent Field**	**Significand (Mantissa)**
Single precision (**32 bits**):		
Bit 31	Bits 30 - 23	Bits 22 – 0
0	10010000	00100000000000000000000
0: + 1: -	Decimal value of exponent field and exponent 132-**127**=5	Decimal value of the significand = 36
Double precision (**64 bits**):		
Bit 63	Bits 62 - 52	Bits 51 – 0
0	10010000000	00100000000000000000000……
0: + 1: -	Decimal value of exponent field and exponent 1028 -**1023**=5	Decimal value of the significand = 36

תא של אחסון מספר מסוים במחשב *binary double precision (64 bits)*
נראה כך:

סה"כ: 64 מקומות ב**IEEE 754** תקנון בפורמת *binary double precision (64 bits)*

את המספר x נוכל לפענח באופן הבא: $x=(-1)^{s}\cdot(mantissa)\cdot 2^{\exp-1023}$.

תורת השגיאות

שגיאה מוחלטת: $\Delta a = \|a-\tilde{a}\|$	שגיאה יחסית: $\delta_a = \frac{\|a-\tilde{a}\|}{\|a\|} = \frac{\|\Delta a\|}{\|a\|}$

מספר התנאי של אלגוריתם: $y = A[x]$	יחס שגיאות יחסיות בין פלט וקלט : $cond(A) = \frac{\delta_y}{\delta_x}$

לדוגמא חישוב π עם דיוק של שתי ספורות:

$$\Delta\pi = |\pi - 3.14| \square 10^{-3}$$

$$\delta_\pi = \frac{|\pi - 3.14|}{\pi} \sim 0.5\times 10^{-4}$$

לעיתים, באלגוריתמים עם קלט ופלט עם נקודות צפה (כמו פתרון משוואות לדוגמא) נקבל שהשגיאה בפלט תהיה גדולה הרבה יותר מהשגיאה בקלט.

לדוגמא חישוב מספר התנאי, ניקח את שתי המשוואות הליניאריות הבאות *Ax=b*:

$$(1)\begin{cases} x+10y=11 \\ 10x+101y=111 \end{cases} \quad (2)\begin{cases} x+10y=11 \\ 10x+101y=111.1 \end{cases}$$

במשוואה הראשונה הפתרון הוא $\begin{pmatrix} x=1 \\ y=1 \end{pmatrix}$, בעוד במשוואה השנייה הפתרון הוא $\begin{pmatrix} x=0 \\ y=1.1 \end{pmatrix}$!

נחשב את השגיאות של הקלט והפלט:

שגיאה בקלט: $\Delta_b = 0.1$, $\delta_b \sim 10^{-3}$ שגיאה בפלט: $\Delta_x = 1.1$, $\delta_x \sim 0.8$

$\Leftarrow$ השגיאה היחסית גדלה כמעט פי אלף!

<u>דוגמא:</u> ב*Matlab* מספר התנאי *(Conditional number)* של המטריצה $\begin{pmatrix} 1 & 10 \\ 10 & 101 \end{pmatrix}$ שווה ל

$$>> A = [1\ 10; 10\ 101];\ cond(A)$$

1.0402e+04

מכאן נווה ששגיא בפלט פי 1000 יותר גדלה משגיא בקלט ואפילו באריתמטיקה ב*MATLAB* – *DOUBLE PRECISION* המספר התנאי די גדול.

<u>משפט:</u> עבור שגיאה מוחלטת יתקיים: $\Delta_{a+b} \le \Delta_a + \Delta_b$, $\Delta_{a-b} \le \Delta_a + \Delta_b$.

נניח שנבחר מספרים $a_1,...,a_m$ כך ש- m=מספר האיברים, ε=השגיאה לכל אחד מן האיברים. אזי: $\Delta_{a_1+...+a_n} \le m\cdot\varepsilon$

<u>משפט:</u> עבור שגיאה יחסית יתקיים: $\delta_{a\times b} \le \delta_a + \delta_b$, $\delta_{a\div b} \le \delta_a + \delta_b$.

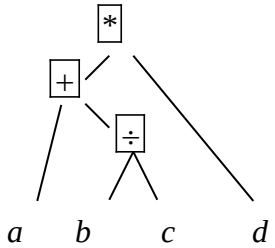
נניח שנבחר מספרים $a_1,...,a_m$ כך ש- m=מספר האיברים, ε=השגיאה לכל אחד מן האיברים.
אזי: $\delta_{a_1\cdot...\cdot a_n} \le m\cdot\varepsilon$

סוגי שגיאות

1. שגיאת עיגול , *ROUNDING ERROR* (כאשר מעגלים את התוצאה). לדוגמא: במדחום דיגיטאלי *t=36.6* אך לא *t=36.584*.
2. שגיאת צמצום, *TRUNCATION ERROR*, לדוגמא כאשר יש זיכרון מוגבל במחשב, $0.56 = 0.1.....$ או $\pi = 3.145...$
3. שגיאת התנאי, *CONDITIONAL ERROR*, בפונקציות. כאשר $\delta_{\text{קלט}} >> \delta_{\text{פלט}}$, אז לאלגוריתם יש <u>תנאי חולה</u> $(ill\ condition)$. לדוגמא במשוואה $ax = b$. אם $a \approx 0$ אז $x \to \infty$ ולכן יש תנאי חולה.

דוגמא לעץ פעולות:

$$\left(a + b/c\right)\cdot d \quad \Rightarrow$$



בהמשך ננסה לברר איך אפשר למדוד שגיאות בחישובים באריתמטיקה של נקודה צפה:

$a = 0.1234567...$

$b = 0.1234562...$

$c = b - a = 0.5...\times 10^{-7}$

כלומר איבדנו את הדיוק בספרות המנטיסה! עכשיו השגיאה גדולה מאוד, והמנטיסה כולה תהיה שגויה.

דוגמא נוספת ב*Matlab*:

$>> s = rand;\ t = rand;$

$>> x = t + s;\ \ q = (t - x) + s;$

$>> q$

נקבל לפעמים $\pm 1.102\times 10^{-16}$ כאשר התשובה אמורה להיות תמיד 0!

ולכן, לפני שמבצעים תרגיל במחשב, יש להיות מודעים ל- (B, p, e), כאשר B הוא הבסיס, p הוא מספר הספרות המקסימאלי במנטיסה, ו- e הוא מספר הספרות המקסימאלי במעריך.
המספר *EPS* - הוא השגיאה המקסימאלית האפשרית בפעלה חיבור\חיסור ושווה ליחידה במקום האחרון במנטיסה *ULP* (*UNIT ON THE LAST PLACE*). לדוגמה *ULP* בפורמת *DOUBLE PRECISION* שווה ל

2^(-52)= 2.220446049250313080847263336181 6 *e-16*

דוגמא:ב*Matlab* מספר *EPS* שווה ל

$>> eps$

$2.204...\times 10^{-16}$

מכאן נווה שאריתמטיקה ב*DOUBLE PRECISION – MATLAB*

חישובי פונקציות

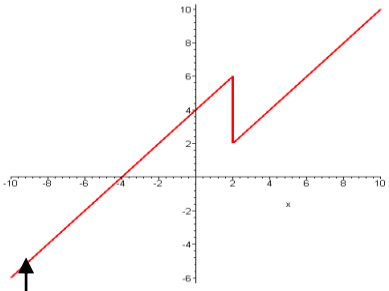
קלט: $x_1, x_2, \ldots, x_n$

פלט: $f\left(x_1, x_2, \ldots, x_n\right)$

$$\Delta_f = \left| f\left(\tilde{x}_1, \ldots, \tilde{x}_n\right) - f\left(x_1, \ldots, x_n\right) \right| = \frac{\partial f}{\partial x_1} \cdot x^*\left(x_1 - \tilde{x}_1\right) + \ldots + \frac{\partial f}{\partial x_n} \cdot x^*\left(x_n - \tilde{x}_n\right) + o\left(x - \tilde{x}\right)$$

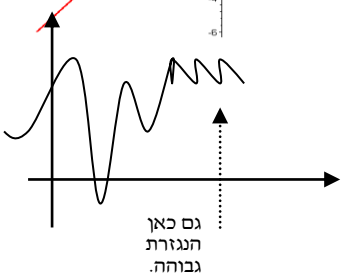
$$\leq \left| \frac{\partial f\left(x^*\right)}{\partial x_1} \right| \cdot \left| x_1 - \tilde{x}_1 \right| + \ldots + \left| \frac{\partial f\left(x^*\right)}{\partial x_n} \right| \cdot \left| x_n - \tilde{x}_n \right| \leq \max_{x_i \in I} \left| \frac{\partial f}{\partial x} \right| \cdot \sum_{i=1}^{n} \left| x_i - \tilde{x}_i \right| = \max_{x_i \in I} \left| \frac{\partial f}{\partial x} \right| \cdot \sum_{i=1}^{n} \Delta x_i$$

עבור $f = x + y$ לדוגמא, נציב בחסם שמצאנו ונקבל $\Delta_f \leq \Delta_x + \Delta_y$. ולכן, כשהנגזרות גבוהות, השגיאה מאוד גבוהה.



ניקח לדוגמא את הפונקציה: $f(x) = \begin{cases} x & x \leq 2 \\ x+4 & x > 2 \end{cases}$

באזור הנקודה $x = 2$ הנגזרת גבוהה מאוד, ולכן גם השגיאה.



$$\delta_f = \frac{\Delta f}{|f|} \leq \sum_{i=1}^{n} \frac{\left| \frac{\partial f\left(x^*\right)}{\partial x_i} \right|}{|f|} \cdot \Delta x_i \underset{\frac{1}{|f|} \cdot \frac{\partial f}{\partial x_i} = \frac{2\ln|f|}{2x_i}}{=}$$

$$= \sum_{i=1}^{n} \left| \frac{2\ln|f|}{2x_i} \right| \cdot |x_i| \cdot \delta_{x_i} \Rightarrow \delta_f \leq M \sum_{i=1}^{n} \delta_{x_i}$$

כאשר $M = \max_{1 \leq i \leq n}$.

ניקח לדוגמא את $f(x, y) = x \cdot y$.

$$\Delta_{x\cdot y} = \underbrace{\max\{x,y\}}_{\frac{\partial f}{\partial x}=y,\frac{\partial f}{\partial y}=x}\cdot\left(\Delta_x+\Delta_y\right)$$

$$\left.\begin{aligned} x\cdot\frac{\partial \ln(x\cdot y)}{\partial x} &= x\cdot\frac{1}{xy}\cdot y = 1 \\ y\cdot\frac{\partial \ln(x\cdot y)}{\partial y} &= y\cdot\frac{1}{xy}\cdot x = 1 \end{aligned}\right\} \Rightarrow \delta_f \le 1\cdot\sum_{i=1}^{n}\delta_{x_i}$$

למשל: $\delta_{e^x} \le \max\{x\}\cdot\delta_x$

שגיאות באלגוריתמים תלויות בכמות הפעלות

בכמה פעולות אנו צריכים להשתמש ע"מ להעביר פונקציה המוצגת כמערך $(a_0,a_1,\ldots,a_n)$ להצגה כפולינום?

1. $p_n(x) = a_0x^n + a_1x^{n-1} + \ldots + a_{n-1}x + a_n$. אולם דרך זו לוקחת הרבה פעולות:

   כפל: $(n+1)+n+(n-1)+(n-2)+\ldots+1$

   חיבור: n פעולות.

   נסכום את הפעולות: $N = \frac{(n+2)(n+1)}{2} + n = \frac{n^2+5n+2}{2}$

2. *Horner scheme*: נציג את הפולינום באופן:

   $$p_n(x) = a_n + x(a_{n-1} + x(a_{n-2} + \ldots + x(a_1 + a_0x)\ldots))$$

3. שקול לכתוב:

$$\begin{cases} s_0 = a_0 \\ s_m = a_m + xs_{m-1}, \quad m = 1,2,\ldots,n \end{cases}$$

ואז מספר הפעולות קטן יותר.

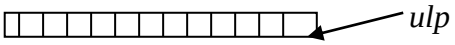
יעילות אלגוריתמים

נניח שיש שתי תוכנות: $(B_1, p_1, e_1) - A$, $(B_2, p_2, e_2) - B$

אם $p_1 > p_2$ ו- $B_1 = B_2$ נעדיף את תוכנה A בכל מקרה! (כלומר, ה-*mantissa* הכי חשובה).

אם $B_1 > B_2$ ו- $p_1 = p_2$ אז נעדיף את תוכנה A בכל מקרה! (גם כאן, המעריך מקבל עדיפות שנייה).

הערה: נגדיר את *ulp* להיות הדיוק של המספר (*unit on last place*). $ulp = B^{-p}$



<u>הגדרות</u>: 1. נגדיר *flops* $N_q \approx$ (כמות הפעולות לביצוע האלגוריתם, את ה- *flops* נגדיר בהמשך)

2. נגדיר Ω להיות "תחום חשוד" (היכן שהשגיאה גדולה). דוגמאות: כאשר הנגזרות החלקיות גדולות או לא קימות. במקרים אלו, כדאי לבצע דיוק מרבי או להחליף אריתמטיקה.