

Algebră Liniară și Optimizare

Curs 3

Anca Ignat & Corina Forăscu

Pagina web: <https://edu.info.uaic.ro/algebra-liniara/>

Discord: <https://discord.gg/eGQYadYH>



Recap. curs 2

1. Produse scalare
2. Norme
 1. Vectoriale
 2. Euclidiene
 3. Matriceale
 1. Naturale (induse)
 4. Spectrale
3. Istoria rezolvării sistemelor liniare

Structură curs 3

1. Reprezentări ale numerelor
 - Numere în format binar
 - Numere în format zecimal
2. Reprezentarea zecimală (trunchiere și rotunjire)
3. Erori în calculele numerice (surse, propagare)
4. Condiționare și stabilitate numerică
 - Exemplu: polinomul Wilkinson

Reprezentări ale numerelor

IEEE Binary Floating Point Arithmetic Standard 754

- IEEE 754-1985 – versiunea originală, prima publicare a standardului binar pentru virgulă mobilă
- IEEE 754-2008 – revizuire majoră care a inclus aproape tot standardul original, plus standardul IEEE 854-1987 (care definea aritmetică în virgulă mobilă independentă de bază) și a extins standardul cu formate zecimale.
- IEEE 754-2019 – versiunea actuală, publicată în iulie 2019, este o revizuire minoră ce aduce clarificări, corecții de erori și operații recomandate noi.

Reprezentări ale numerelor

IEEE Binary Floating Point Arithmetic Standard 754

Fără IEEE 754:

- Programele ar calcula și interpreta numerele în mod diferit pe procesoare sau calculatoare diferite.
- Transferul de date numerice între sisteme ar putea duce la rezultate incorecte sau neașteptate.
- Limbajele de programare și compilatoarele ar trebui să implementeze propriile reguli individuale.

Datorită IEEE 754, astăzi:

- ✓ Majoritatea unităților hardware de calcul (FPUs, procesoare) folosesc acest standard.

Reprezentări ale numerelor

IEEE Standard 754 definește:

1. **Formate aritmetice** – seturi de tipuri de date în formate binare și zecimale, care includ:

- Numere finite
- Zeros („+0” și „-0”)
- Numere subnormale
- „Infinit”
- Valori speciale „NaN” (Not a Number – nu este un număr)

2. **Formate de schimb** (interchange) – codări (șiruri de biți) folosite pentru a transmite date eficient și compact

3. **Reguli de rotunjire** – proprietăți ce trebuie respectate la rotunjirea numerelor în timpul operațiilor sau conversiilor

4. **Operații** – operații aritmetice și funcții (de exemplu trigonometrice) pe formatele definite

5. **Gestionarea excepțiilor** – modul în care se semnalează condiții excepționale (de exemplu: împărțire la zero, depășiri, etc.)

Reprezentări ale numerelor

IEEE 754 definește mai multe formate de reprezentare a numerelor în virgulă mobilă, atât în bază 2 (binare), cât și în bază 10 (zecimale).

Fiecare format specifică:

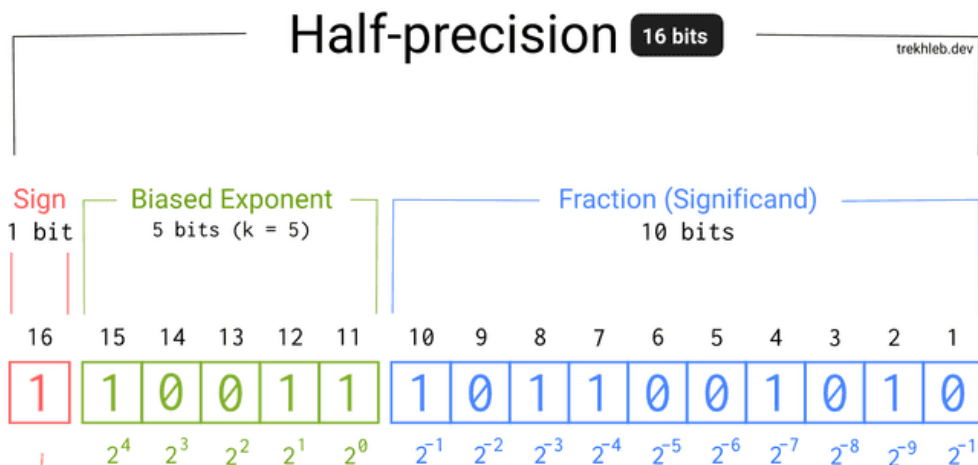
- numărul total de biți
- împărțirea în *semn*, *exponent* și *fracțiune* (mantisă)
- domeniul valorilor reprezentabile
- precizia numerică

Format	Biți	Precizie aproximativă	Utilizare tipică
binary16	16	3–4 cifre	Grafică, AI
binary32	32	~7 cifre	Aplicații generale
binary64	64	~15–16 cifre	Științific, standard implicit
binary128	128	~34 cifre	Cercetare avansată

Reprezentări binare

Format **binary16** (half precision – 16 biți)

- 1 bit semn
- 5 biți exponent
- 10 biți fracțiune
- Bias exponent: 15
- Precizie: ~3–4 cifre zecimale



= -27.15625

exponent = $2^4 + 2^1 + 2^0 = 19$

bias = $2^{k-1} - 1 = 2^{5-1} - 1 = 15$

biased_exponent = exponent - bias = 19 - 15 = 4

fraction = $2^{-1} + 2^{-3} + 2^{-4} + 2^{-7} + 2^{-9} =$

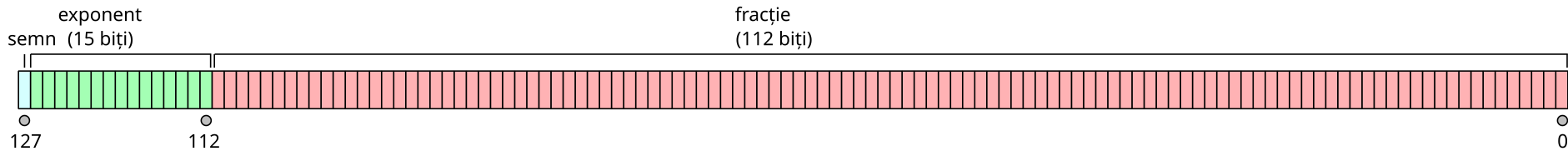
= $0.5 + 0.125 + 0.0625 + 0.0078125 + 0.001953125 =$
 = 0.697265625

$-1^1 \times 2^4 \times (2^0 + 0.697265625) = -27.15625$

Reprezentări binare

Format **binary128** (quadruple precision – 128 biți)

- 1 bit semn
- 15 biți exponent
- 112 biți fracțiune
- Bias exponent: 16383
- Precizie: ~34 cifre zecimale



Reprezentarea numerelor zecimale

Un număr are forma:

$$(-1)^{\text{semn}} \times \text{coeficient} \times 10^{\text{exponent}}$$

Diferența față de formatele binare:

- baza este **10**, nu 2
- coeficientul este un număr zecimal finit
- exponentul este în baza 10

Reprezentarea zecimală, aproximare

Reprezentarea zecimală a unui număr, folosind k cifre este:

$$\pm 0.d_1 d_2 \dots d_k \times 10^n \quad 1 \leq d_1 \leq 9, \quad 0 \leq d_i \leq 9, \quad i = 2, \dots, k$$

Orice număr real y de forma

$$y = 0.d_1 d_2 \dots d_k d_{k+1} d_{k+2} \dots \times 10^n$$

poate fi reprezentat folosind k cifre printr-o simplă **trunchiere**:

$$fl(y) = 0.d_1 d_2 \dots d_k \times 10^n$$

Eroarea de trunchiere este: $y - fl(y)$

Reprezentarea lui y cu k cifre se poate face și prin **rotunjire**:

$$fl(y) = 0.\delta_1 \delta_2 \dots \delta_k \times 10^n$$

$fl(y)$ înseamnă **floating-point representation** (valoarea lui y reprezentată/rotunjită în sistemul mașină)

Reprezentarea zecimală: rotunjire

Orice număr real y de forma

$$y = 0.d_1d_2\dots d_k d_{k+1}d_{k+2}\dots \times 10^n$$

poate fi reprezentat folosind k cifre printr-o **rotunjire**, astfel:

- Dacă $d_{k+1} < 5$, atunci $fl(y) = 0.d_1d_2\dots d_k \times 10^n$ (*round-down*)
- Dacă $d_{k+1} > 5$, atunci $fl(y) = 0.d_1d_2\dots (d_k + 1) \times 10^n$ (*round-up*)
- Dacă $d_{k+1} = 5$, atunci se face:
 - round-up SAU
 - *round-to-even*:
 - Dacă d_k este par, rămâne la fel
 - Dacă d_k este impar, se mărește cu 1

Reprezentarea zecimală: aproximare

Un număr r^* aproximează numărul r cu t cifre exacte dacă t este cel mai mare întreg nenegativ pentru care:

$$\frac{|r - r^*|}{|r|} \leq 5 \times 10^{-t}$$

La trunchiere avem:

$$\left| \frac{y - fl(y)}{y} \right| \leq 10^{-k+1}$$

La rotunjire avem:

$$\left| \frac{y - fl(y)}{y} \right| \leq 0.5 \times 10^{-k+1}$$

Reprezentarea zecimală: operații în calculator

$$\mathbf{x +_c y = fl(fl(x) + fl(y))}$$

$$\mathbf{x -_c y = fl(fl(x) - fl(y))}$$

$$\mathbf{x \times_c y = fl(fl(x) \times fl(y))}$$

$$\mathbf{x \div_c y = fl(fl(x) \div fl(y))}$$

Indicele c arată că:

- lucrăm în aritmetică finită
- apar erori de reprezentare
- apar erori de rotunjire la fiecare operație

Este o notare standard în analiza numerică pentru a diferenția:

- operațiile matematice exacte
- operațiile efectuate pe calculator (aproximative)

Surse de erori în calculele numerice

1. Erori în datele de intrare:

- măsurători afectate de erori sistematice sau perturbații temporare,
- erori de rotunjire: $1/3$, π , $1/7$, ...

2. Erori de modelare:

- erori de discretizare: limita unui șir, suma unei serii, funcții neliniare approximate de funcții liniare, aproximarea derivatei unei funcții, ...
- simplificări în modelul matematic, idealizări, ignorarea unor parametri, ...

3. Erori în timpul calculelor:

- erori de de rotunjire datorate capacității limitate de memorare a datelor, operațiile nu sunt efectuate exact
- erori ale bibliotecilor folosite (*,bug'*)

4. Erori umane (date, algoritmi, înțelegerea problemei)

Eroare absolută, eroare relativă

Fie a valoarea exactă și \tilde{a} valoarea aproximativă.

Definim eroarea absolută prin $a - \tilde{a}$ sau $|a - \tilde{a}|$ sau $\|a - \tilde{a}\|$.

Eroare absolută: $|a - \tilde{a}| \leq \Delta_a$ (deci $a = \tilde{a} \pm \Delta_a$),
unde Δ_a este mărimea maximă a erorii absolute.

Eroarea absolută spune cu cât diferă efectiv valoarea aproximativă de cea exactă.

Eroare relativă (pentru a nenul): $\frac{a-\tilde{a}}{a}$ sau $\frac{|a-\tilde{a}|}{|a|}$ sau $\frac{\|a-\tilde{a}\|}{\|a\|}$, și se

notează $\frac{|a-\tilde{a}|}{|a|} \leq \delta_a$

unde δ_a este eroarea relativă maximă (uzual exprimată în procente, %).

Eroarea relativă se folosește când vrem să vedem eroarea raportată la mărimea valorii exacte.

Eroare absolută, eroare relativă

Eroare absolută	Eroare relativă
Măsoară diferența directă	Măsoară diferența raportată la mărimea valorii
Depinde de unitate	Este adimensională
Bună pentru valori de aceeași scară	Bună pentru comparații între mărimi diferite

Eroare absolută, eroare relativă: exemplu

În aproximările $1\text{kg} \pm 5\text{g}$, $50\text{g} \pm 5\text{g}$ erorile absolute sunt egale (5g), dar pentru prima cantitate eroarea relativă este 0,5% ($(5*100)/1000$) iar pentru a doua eroarea relativă este 10% ($(5*100)/50$).

Formalizând, vom vedea că erorile se propagă:

$$a_1 = \tilde{a}_1 \pm \Delta_{a_1}, a_2 = \tilde{a}_2 \pm \Delta_{a_2},$$

$$a_1 \pm a_2 = (\tilde{a}_1 \pm \tilde{a}_2) \pm (\Delta_{a_1} \pm \Delta_{a_2})$$

$$\Delta_{a_1+a_2} \leq \Delta_{a_1} + \Delta_{a_2}.$$

a_1 cu eroare relativă δ_{a_1} și a_2 cu eroare relativă δ_{a_2} .

Dacă $a = a_1 * a_2$ sau $\frac{a_1}{a_2}$ rezultă $\delta_a = \delta_{a_1} + \delta_{a_2}$

Condiționare numerică

Condiționarea unei probleme caracterizează sensibilitatea soluției în raport cu perturbarea datelor \mathbf{x} de intrare, în ipoteza unor calcule exacte (independent de algoritmul folosit pentru rezolvarea problemei). Condiționarea ține de *problema matematică* în sine. Condiționarea măsoară cât de sensibilă este soluția exactă la mici modificări ale datelor de intrare.

Fie \mathbf{x} datele exacte de intrare,

$\tilde{\mathbf{x}}$ o aproximație cunoscută a acestora,

$P(\mathbf{x})$ soluția exactă a problemei și

$P(\tilde{\mathbf{x}})$ soluția problemei cu $\tilde{\mathbf{x}}$ ca date de intrare.

Se presupune că s-au făcut calcule exacte pentru obținerea soluțiilor $P(\mathbf{x})$ și $P(\tilde{\mathbf{x}})$.

O problemă se consideră a fi prost condiționată dacă $P(\mathbf{x})$ și $P(\tilde{\mathbf{x}})$ diferă mult chiar dacă eroarea relativă $\frac{\|\mathbf{x}-\tilde{\mathbf{x}}\|}{\|\mathbf{x}\|}$ este mică.

Condiționare numerică

Condiționarea numerică $k(\mathbf{x})$ a unei probleme este exprimată prin amplificarea erorii relative, anume prin formula:

$$k(\mathbf{x}) = \frac{\frac{\|P(\mathbf{x}) - P(\tilde{\mathbf{x}})\|}{\|P(\mathbf{x})\|}}{\frac{\|\mathbf{x} - \tilde{\mathbf{x}}\|}{\|\mathbf{x}\|}} \quad \text{pentru } \mathbf{x} \neq \mathbf{0} \text{ și } P(\mathbf{x}) \neq \mathbf{0}$$

O valoare mică pentru $k(\mathbf{x})$ caracterizează o problemă bine condiționată.

Condiționarea este o proprietate locală (se evaluează pentru diverse date de intrare \mathbf{x}). O problemă este bine condiționată dacă este bine condiționată în orice punct.

Eroarea relativă în datele de ieșire \approx

Număr de condiționare \times Eroarea relativă în datele de intrare

Condiționare: exemplu

Se consideră polinomul Wilkinson, cu rădăcinile întregi 1, 2, ..., 20:

$$w(x) = (x - 1)(x - 2) \cdots (x - 20) = x^{20} - 210x^{19} + P_{18}(x)$$

Dacă se schimbă coeficientul -210 al lui x^{19} cu următoarea aproximare:

$$-210 - 2^{-23} = -210.0000001192$$

atunci soluțiile (cu 5 zecimale exacte) noului polinom sunt:

**1.00000 2.00000, 3.00000, 4.00000, 5.00000, 6.00001, 6.99970, 8.00727,
8.91725, 20.84691, 10.09527 ± i0.64350, 11.79363 ± i1.65233, 13.99236 ±
i2.51883, 16.73074 ± i2.81262, 19.50244 ± i1.94033**

Stabilitate numerică

Pentru rezolvarea unei probleme P , calculatorul execută un algoritm \tilde{P} . Deoarece se folosesc numere în virgulă mobilă, calculele sunt afectate de erori :

$$P(x) \neq \tilde{P}(x)$$

Stabilitatea numerică exprimă mărimea erorilor numerice introduse de algoritm, în ipoteza unor date de intrare exacte (cum se comportă algoritmul în prezența erorilor de rotunjire de pe calculator), anume:

$$\|P(x) - \tilde{P}(x)\| \text{ sau } \frac{\|P(x) - \tilde{P}(x)\|}{\|P(x)\|}.$$

Algoritmul amplifică sau controlează erorile de rotunjire?

- Dacă erorile rămân mici \rightarrow ***algoritm numeric stabil***
- Dacă erorile cresc exploziv \rightarrow ***algoritm numeric instabil***

Stabilitate numerică

O eroare relativă de ordinul erorii de rotunjire caracterizează un *algoritm numeric stabil*.

Un *algoritm numeric stabil* aplicat unei *probleme bine condiționate* conduce la *rezultate cu precizie foarte bună*.

Un *algoritm* \tilde{P} destinat rezolvării problemei P este *numeric stabil* dacă este îndeplinită una din condițiile:

- $\tilde{P}(\mathbf{x}) \approx P(\mathbf{x})$ pentru orice date de intrare \mathbf{x}
- există $\tilde{\mathbf{x}}$ apropiat de \mathbf{x} , astfel ca $\tilde{P}(\mathbf{x}) \approx P(\tilde{\mathbf{x}})$,

unde: \mathbf{x} = datele exacte,

$P(\mathbf{x})$ = soluția exactă folosind date exacte,

$\tilde{P}(\mathbf{x})$ = soluția „calculată” folosind algoritmul \tilde{P} cu date exacte de intrare

$P(\tilde{\mathbf{x}})$ = soluția „calculată” folosind algoritmul P cu date approximate $\tilde{\mathbf{x}}$

Condiționare vs. Stabilitate numerică

Condiționare	Stabilitate
Proprietate a problemei	Proprietate a algoritmului
Nu depinde de implementare	Depinde de metoda aleasă
Ține de sensibilitatea soluției	Ține de propagarea erorilor
Nu poate fi „reparată” prin algoritm	Poate fi îmbunătățită printr-un algoritm mai bun
Cât de <i>fragil</i> este un <i>obiect</i>	Cât de <i>delicat</i> manipulezi <i>obiectul</i>

Dacă obiectul este fragil (rău condiționat), orice mică atingere îl strică.

Dacă obiectul este solid (bine condiționat), dar îl arunci pe jos (algoritm instabil), tot îl strici.

Recapitulare și/sau întrebări curs 3

1. Reprezentări ale numerelor
 - Numere în format binar
 - Numere în format zecimal
2. Reprezentarea zecimală (trunchiere și rotunjire)
3. Erori în calculele numerice (surse, propagare)
4. Condiționare și stabilitate numerică
 - Exemplu: polinomul Wilkinson