

Obiective curs.

- Crearea, analiza și implementarea de algoritmi pentru rezolvarea problemelor din *matematica continuă*
- Analiza complexității, analiza și propagarea erorilor, condiționarea problemelor și stabilitatea numerică a algoritmilor problemelor numerice
- Prezentarea metodelor numerice clasice și a celor moderne de rezolvare a problemelor științifice și ingineresti
- Alegerea celor mai potrivite metode numerice pentru o problemă dată

O problemă P_F având o natură fizică oarecare poate fi studiată *experimental* sau prin *simulare*. Aceasta poate fi transformată, utilizând legile fundamentale ale fizicii într-o problemă de natură matematică P_M . Vom spune că problema este *bine pusă* dacă admite o soluție unică.

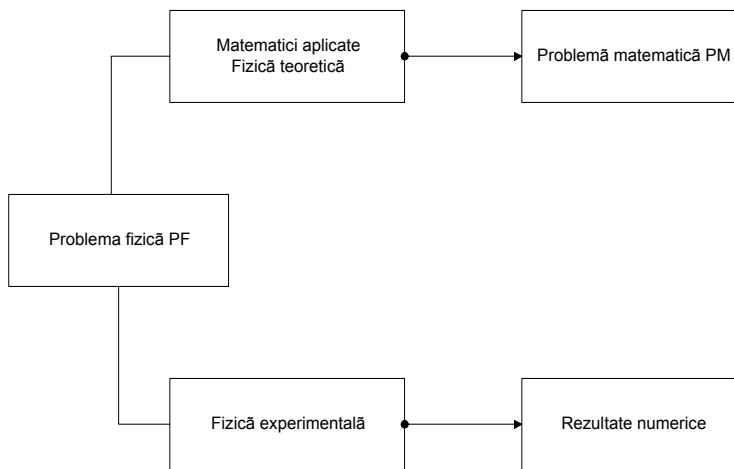


Fig.1.1. Modalități de abordare a problemelor fizice

Ca exemplu, vom considera următoarea problemă fizică:

P_F : Să se studieze propagarea temperaturii într-o bară AB de lungime l cunoscând
 -temperaturile la momentul inițial în orice punct M al barei $\theta_0(x)$, $x \in [0, l]$
 -temperaturile la cele două capete $\tau_A(t)$ și $\tau_B(t)$ în orice moment $t \in [0, t_1]$

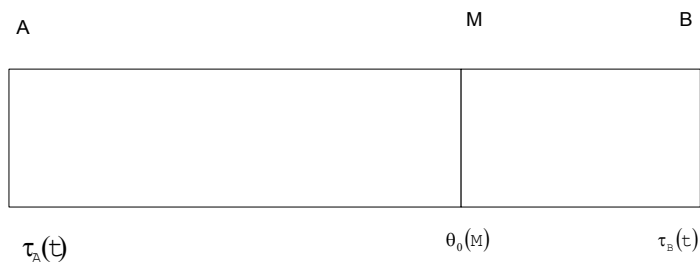


Fig.1.2. Propagarea căldurii într-o bară

Problema matematică corespunzătoare este:

P_M : Să se determine funcția:

$$\begin{aligned} \theta(x, t) &\rightarrow \theta(x, t) \\ [0, 1] \times [0, t_1] &\rightarrow \mathbb{R} \end{aligned}$$

care satisface următoarele condiții:

$$1^0 \quad \frac{\partial^2 \theta}{\partial x^2} = K \cdot \frac{\partial \theta}{\partial t} \quad \text{ecuația lui Fourier}$$

$$2^0 \quad \theta(x, 0) = \theta_0(x) \quad \text{condițiile inițiale}$$

$$3^0 \quad \theta(0, t) = \tau_A(t) \quad \text{condițiile pe frontieră}$$

$$4^0 \quad \theta(1, t) = \tau_B(t)$$

$$5^0 \quad \theta \in S, S = \text{spațiul funcțiilor de 2 ori derivabile pe } [0, 1] \times [0, t_1]$$

În acest moment intervine analiza numerică și furnizează metodele de calcul, care în urma unui număr finit $N(x, t, \varepsilon)$ de operații elementare furnizează pentru soluția $\theta(x, t)$ o aproximație $\theta'(x, t)$ efectiv calculabilă, astfel că: $|\theta(x, t) - \theta'(x, t)| < \varepsilon$.

Prezintă interes metodele de calcul în timp finit, cu: $0 < t < t_1$ care furnizează aproximații uniforme: $N(x, t, \varepsilon) = N(\varepsilon)$.

Problema continuă **P_M** este transformată într-o problemă asemănătoare **P_h** prin discretizare.

În acest scop se selectează un număr finit de puncte $(i\Delta x, n\Delta t)$ din domeniul compact $[0, 1] \times [0, t_1]$ folosind o rețea de discretizare cu pașii:

$$\begin{aligned} \Delta x &= \frac{1}{M}, \\ \Delta t &= \frac{t_1}{N}. \end{aligned}$$

și se notează: $\theta_i^n = \theta(i\Delta x, n\Delta t)$

Dacă se aproximează derivatele parțiale cu diferențele finite:

$$\begin{aligned} \frac{\partial \theta}{\partial t}(i\Delta x, n\Delta t) &= \frac{\theta_i^{n+1} - \theta_i^n}{\Delta t} \\ \frac{\partial^2 \theta}{\partial x^2}(i\Delta x, n\Delta t) &= \frac{\theta_{i+1}^n - 2 \cdot \theta_i^n + \theta_{i-1}^n}{(\Delta x)^2} \end{aligned}$$

se obține următoarea **problemă discretizată: P_h**:

Să se determine θ_i^{n+1} cu $1 < i < M-1, 0 < n < N-1$, care satisface condițiile:

$$1^0 \quad \frac{\theta_i^{n+1} - \theta_i^n}{\Delta t} = K \cdot \frac{\theta_{i+1}^n - 2 \cdot \theta_i^n + \theta_{i-1}^n}{(\Delta x)^2}$$

$$2^0 \quad \theta_i^0 = \theta_0(i\Delta x)$$

$$3^0 \quad \theta_0^n = \tau_A(n\Delta t)$$

$$4^0 \quad \theta_M^n = \tau_B(n\Delta t)$$

$$5^0 \quad \frac{\Delta t}{(\Delta x)^2} \leq \frac{1}{2 \cdot K}$$

Problema discretizată P_h constă în rezolvarea succesivă a N sisteme de ecuații liniare tridiagonale.

Diferența: $|\theta(i\Delta x, n\Delta t) - \theta_i^n|$ evaluează apropierea între soluția problemei discretizate P_h și a modelului matematic P_M în fiecare punct al discretizării.

Soluția problemei discretizate P_h trebuie să tindă spre soluția problemei continue P_M dacă $h \rightarrow 0$ (h reprezintă pasul de discretizare – în cazul problemei considerate avem pașii $\Delta x \rightarrow 0, \Delta t \rightarrow 0$ sau $N \rightarrow \infty, M \rightarrow \infty$); vom spune că trebuie satisfăcută o *condiție de consistență*:

$$\lim_{h \rightarrow 0} P_h = P_M.$$

O altă condiție importantă o reprezintă *stabilitatea*; aceasta impune ca soluția θ' a problemei perturbate P_M (manifestată prin perturbarea parametrilor $\theta', \tau'_A, \tau'_B, K'$) să fie apropiată de soluția θ a modelului matematic P_M .

Pe baza modelului matematic discretizat se va proiecta un algoritm, care va fi analizat prin prisma:

- *eficienței* (resurse folosite: timp de calcul și memorie),
- *convergenței* către soluția modelului matematic continuu,
- efectului *propagării erorilor*.

Etapele enumerate evidențiază următoarele *tipuri de erori*:

- *erori de problemă (inerente)* care apar la trecere de la modelul fizic P_F la cel matematic P_M ,
- *erori de metodă* introduse prin discretizarea modelului matematic,
- *erori de trunchere* provin din natura infinită a unor procese care descriu soluția problemei
- *erori de rotunjire* specifice rezolvării problemei pe calculatorul numeric, care utilizează aritmetica în virgulă mobilă

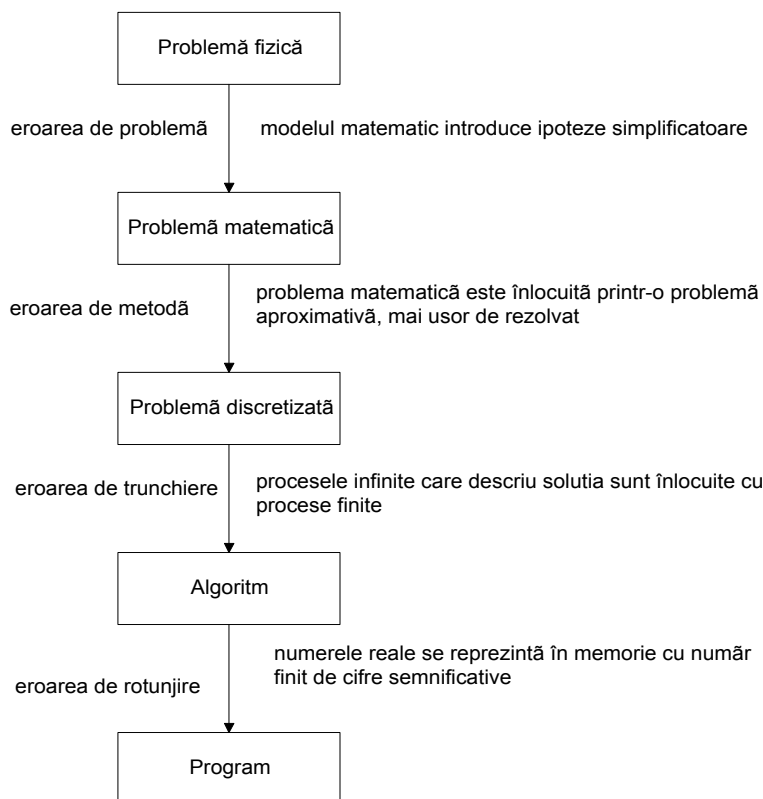


Fig. 1.3. Surse de erori

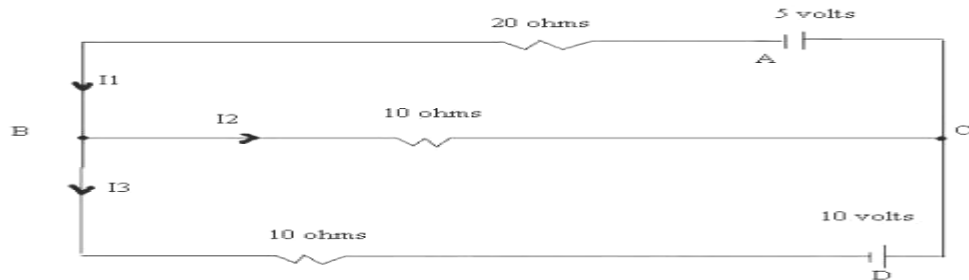
Conținut curs.

- Reprezentare în virgulă mobilă.
Standardul IEEE 754 pentru numere reale.
Condiționarea problemelor și stabilitatea numerică a algoritmilor.
- Rezolvarea sistemelor de ecuații liniare prin metode gaussiene.
Pivotare parțială și totală. Factorizare LU.
- Propagarea erorilor în rezolvarea sistemelor de ecuații liniare.
- Metode iterative de rezolvare a sistemelor de ecuații liniare
- Interpolare polinomială. Polinom de interpolare Lagrange.
- Diferențe divizate. Polinom Newton. Eroarea interpolării.
- Interpolare cu funcții spline. Interpolare trigonometrică.
- Aproximare uniformă. Polinoame Cebășev. Algoritmii lui Remes.
- Aproximare continuă și discretă în sensul celor mai mici pătrate.
- Rezolvarea sistemelor în sensul celor mai mici pătrate. Factorizare QR.
- Metodele Householder, Givens, Gram-Schmidt
- Integrare numerică. Metode Newton-Cotes. Metoda Romberg.
- Integrare gaussiană. Polinoame ortogonale. Integrale improprii.
- Integrarea ecuațiilor diferențiale ordinare. Metode Runge-Kutta.

- Metode multipas explicite și implicate. Predictor-corector.
- Convergența metodelor multipas
- Valori proprii și vectori proprii. Metodele puterii
- Algoritmul QR cu deplasare explicită. Descompunerea valorilor singulare

Aplicații ale calculului numeric.

1. Determinați curenții I_1 , I_2 , și I_3 din circuitul electric:



Se aplică legea 1 a lui Kirchhoff în nodurile **B** sau **C**:

$$I_1 - I_2 - I_3 = 0$$

Și legea a doua Kirchhoff în buclele BDCB și BCAB:

$$-10I_1 + 10I_2 = 10$$

$$20I_1 + 10I_2 = 5$$

cu soluția: $I_1 = 0.4$, $I_2 = -0.3$, $I_3 = 0.7$

2. **Modelul Leontieff** consideră economia formată din n sectoare independente: S_1 , S_2 , ..., S_n . Fiecare sector consumă bunuri produse de celelalte sectoare (inclusive cele produse de el însuși). Introducem notațiile:

m_{ij} = numărul de unități produse de sectorul S_i necesare sectorului S_j să producă o unitate

p_i = nivelul producției sectorului S_i

$m_{ij}p_j$ = numărul unităților produse de S_i și consumate de S_j

Numărul total de unități produs de S_i este: $p_1m_{i1} + p_2m_{i2} + \dots + p_nm_{in}$

Într-o economie echilibrată, tot ce se produce trebuie consumat, adică:

$$\begin{cases} m_{11}p_1 + m_{12}p_2 + \dots + m_{1n}p_n = p_1 \\ \dots \\ m_{n1}p_1 + m_{n2}p_2 + \dots + m_{nn}p_n = p_n \end{cases}$$

Adică sistemul:

$$M \cdot p = p$$

Într-un model deschis de economie, unele sectoare își satisfac unele cerințe din exterior, adică:

$$p_i = m_{i1}p_1 + m_{i2}p_2 + \dots + m_{in}p_n + d_i$$

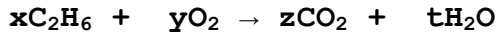
care conduce la sistemul liniar de ecuații:

$$p = M \cdot p + d$$

cu soluția:

$$p = (I - M)^{-1} \cdot d$$

3. Aplicând legea conservării masei unei ecuații de echilibru chimic:



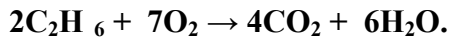
se obține sistemul de ecuații liniare:

$$\begin{cases} 2x = z \\ 6x = 2t \\ 2y = 2x + t \end{cases}$$

care are o soluție întreagă:

$$x=2, y=7, z=4, t=6.$$

deci ecuația chimică este:



Reprezentarea în virgulă mobilă.

$$f1(x) = \pm 0.a_1a_2\dots a_t \times \beta^e$$

reprezentare normalizată

$$1 \leq a_1 < \beta \quad \text{și} \quad 0 \leq a_i < \beta, \quad i=2:t$$

$$L \leq e \leq U$$

- Sistemul de reprezentare în virgulă mobilă $F(\beta, t, L, U)$ cuprinde:
- baza β
- precizia reprezentării t
- limitele (superioară și inferioară ale) exponentului L și U
- reprezentarea lui zero

Exemplu: $F(10, 1, 0, 1) = \{ \pm 0.a_1 \times 10^e \} \cup \{0\}$ cu $a_1 \in \{1:9\}$ și $e \in \{0, 1\}$, în total 37 de numere.

- $2(\beta-1)\beta^{t-1}(U-L+1)$ valori distincte
- a_1 poate lua $\beta-1$ valori distincte,
- restul de $t-1$ cifre poate lua fiecare β valori diferite, deci β^{t-1} ,
- exponentul ia $U-L+1$, și
- semnul două).

Cel mai mare număr reprezentabil Ω , (**realmax**) are forma:

$$\begin{aligned} \Omega &= 0.(\beta-1)(\beta-1)\dots(\beta-1) \times \beta^U = \\ &= [(\beta-1)/\beta^1 + (\beta-1)/\beta^2 + \dots + (\beta-1)/\beta^t] \times \beta^U \end{aligned}$$

$$= (\beta-1)/\beta(1-\beta^{-t})/(1-\beta^{-1}) \times \beta^U = \beta^U(1-\beta^{-t})$$

$$\Omega = \beta^U(1-\beta^{-t})$$

- Cel mai mic număr pozitiv reprezentabil ω numit și **realmin** este:

$$\omega = 0.10\dots 0 \times \beta^L = \beta^L/\beta = \beta^{L-1}$$

$$\omega = \beta^{L-1}$$

Surse de erori.

Un număr real $x \in F$ se reprezintă exact, dacă suma se termină înainte de t termeni și exponentul este cuprins între limite. Altfel, numărul real x se *aproximează* printr-o valoare $f1(x) \in F$

Aproximarea numărului real

$$x = (0.a_1a_2\dots) \beta^e = (a_1\beta^{-1} + a_2\beta^{-2} + \dots + a_t\beta^{-t} + a_{t+1}\beta^{-t-1} + \dots) \beta^e$$

se poate face prin *trunchiere* sau prin *rotunjire*.

- Aproximarea prin trunchiere ignoră cifrele numărului real din dreapta poziției t .

$$f1(x) = (0.a_1a_2\dots a_t) \beta^e = (a_1\beta^{-1} + a_2\beta^{-2} + \dots + a_t\beta^{-t}) \beta^e$$

- Aproximarea prin rotunjire consideră:

$$f1(x) = (0.a_1a_2\dots a_{t+1}) \beta^e \text{ dacă } a_{t+1} \geq \beta/2$$

$$f1(x) = (0.a_1a_2\dots a_t) \beta^e \text{ dacă } a_{t+1} < \beta/2$$

O *depășire superioară* apare dacă $e > U$.

Ea declanșează o *eroare la execuție*, care conduce la întreruperea calculelor.

O *depășire inferioară* apare dacă $e < L$;

ea duce la înlocuirea numărului prin zero.

- *Epsilon mașină* (notat fie **eps** în Matlab sau μ) reprezintă cel mai mic număr pozitiv cu proprietatea că:

$$f1(1+\mu) > 1$$

De exemplu în $F(10, 4, -3, 3)$ cu *rotunjire prin tăiere (trunchiere)*:

$$f1(1+0.0009) = f1(1.0009) = 1$$

$$f1(1+0.0010) = f1(1.0010) = 1.001 > 1$$

$$\text{așadar } \mu_{tr} = 0.001 = 10^{-3} > \omega = 10^{-4}.$$

- Dacă se folosește rotunjire, atunci:

$$f1(1+0.0004) = f1(1.0004) = 1$$

$$f1(1+0.0005) = f1(1.0005) = 1.001 > 1$$

$$\text{cu } \mu_{rot} = 0.0005 = 1/2 \cdot 10^{-3} = 1/2 \cdot \mu_{tr}$$

- *Eroarea absolută* la rotunjirea prin trunchiere:

$$e_x = x - f1(x) = (a_1/\beta^1 + a_2/\beta^2 + \dots + a_t/\beta^t + a_{t+1}/\beta^{t+1} + \dots) \beta^e - (a_1/\beta^1 + a_2/\beta^2 + \dots + a_t/\beta^t) \beta^e$$

$$e_x = (a_{t+1}/\beta^1 + a_{t+2}/\beta^2 + \dots) \beta^{e-t}$$

$$|e_x| \leq |(\beta-1)/\beta^1 + (\beta-1)/\beta^2 + \dots| \beta^{e-t} =$$

$$(\beta-1)\beta^{e-t} (1/\beta^1 + 1/\beta^2 + \dots) \leq (\beta-1)\beta^{e-t}/(\beta-1) = \beta^{e-t}$$

$$|e_x| \leq \beta^{e-t}$$

Dacă se folosește **rotunjirea** atunci eroarea absolută este și mai mică:

$$|e_x| \leq 1/2 \cdot \beta^{e-t}$$

• **Eroarea relativă** este:

$$\varepsilon_x = |e_x|/|x| = |x - fl(x)|/|x| \leq \beta^{e-t}/(0.a_1 \dots a_t \dots \beta^e)$$

$$\varepsilon_x \leq \beta^{e-t}/(0.10 \dots 0\beta^e) = \beta^{1-t}$$

$$\varepsilon_x \leq \beta^{1-t} \quad \text{la trunchiere}$$

$$\varepsilon_x \leq 1/2 \cdot \beta^{1-t} \quad \text{la rotunjire}$$

În general:

$$|x - fl(x)|/|x| \leq \mu$$

de unde deducem:

$$fl(x) = x(1+\varepsilon), \quad |\varepsilon| \leq \mu = K \beta^{-t}$$

• De exemplu $F(10, 4, -20, 20), \Omega = 10^{20}(1-10^{-4}) = 9.999 \times 10^{19}$,

• $\omega = 10^{-20-1} = 1.0 \times 10^{-21}$, $\mu_r = 1/2 \cdot 10^{-4+1} = 5 \times 10^{-4}$

Propagarea erorilor.

• numere aproximative - operații exacte

• operații aproximative - date exacte

1. Rezultatul exact al adunării a două numere x și y , dacă *operațiile se execută exact* este $x+y$.

În realitate, se lucrează cu *valorile inexacte* $\underline{x}$ și $\underline{y}$, în care:

$$e_x = |\underline{x} - x| \quad \text{și} \quad e_y = |\underline{y} - y|$$

$$\underline{x} + \underline{y} = x + y \pm e_{x+y} = x \pm e_x + y \pm e_y = x + y \pm (e_x + e_y)$$

$$e_{x+y} = e_x + e_y$$

$$\varepsilon_{x+y} = e_{x+y}/|x+y| = (e_x + e_y)/|x+y| = (|x|\varepsilon_x + |y|\varepsilon_y)/|x+y|$$

$$\varepsilon_{x+y} = |x|/|x+y|\varepsilon_x + |y|/|x+y|\varepsilon_y = k_x\varepsilon_x + k_y\varepsilon_y$$

Pentru scădere:

$$\underline{x} - \underline{y} = x - y \pm e_{x-y} = x \pm e_x - (y \pm e_y) = x - y \pm (e_x + e_y)$$

de unde:

$$e_{x-y} = e_x + e_y$$

$$\varepsilon_{x-y} = |x|/|x-y|\varepsilon_x + |y|/|x-y|\varepsilon_y = k_x\varepsilon_x + k_y\varepsilon_y$$

În acest caz coeficienții de ponderare:

$$k_x = |x|/|x-y| \quad \text{și} \quad k_y = |y|/|x-y|$$

pot lua valori foarte mari dacă $x \approx y$, deci *în cazul scăderii numerelor apropiate* ca ordin de mărime *se pot comite erori foarte mari*

În cazul înmulțirii:

$$\begin{aligned} \underline{xy} &= xy \pm e_{xy} = (x \pm e_x) (y \pm e_y) = xy \pm xe_y \pm ye_x + e_x e_y \approx xy \pm (xe_y + ye_x) \\ e_{xy} &= xe_y + ye_x \\ e_{xy} &= e_x + e_y \end{aligned}$$

2. Dacă operațiile se reprezintă *aproximativ*, iar numerele sunt reprezentate *exact*, adunarea a două numere $x = f_x \cdot b^{e_x}$ și $y = f_y \cdot b^{e_y}$ presupune aducerea celui mai mic (fie acesta y) la exponentul celui mai mare, producându-se o *denormalizare*

$$f1(x+y) = f1(f_x b^{e_x} + f_y b^{-(e_x - e_y)} b^{e_x}) = f1[(f_x + f_y b^{-(e_x - e_y)}) b^{e_x}]$$

$$f1(x+y) = f1[(f_x + f_y (1+\mu)) b^{e_x}] = f1[x + (1+\mu)y]$$

Rezultatul operației este *normalizat*:

$$f1(x+y) = [x + (1+\mu)y] (1+\mu)$$

Denormalizarea unuia dintre termeni poate fi evitată dacă se păstrează rezultatul intermediar într-un acumulator cu lungimea $2t$ (*acumulator dublu*). În acest caz numai rezultatul final va fi afectat de trunchiere la t cifre semnificative și *normalizare*, deci:

$$f1_2(x+y) = (x+y) (1+\mu)$$

Anularea catastrofală.

- La scăderea a două numere apropiate ca ordin de mărime, cifrele semnificative se anulează reciproc, rezultând o eroare relativă mare.

$$f1(x) = 0.a_1 a_2 \dots a_{p-1} a_p \dots \underline{a_t} \times \beta^e$$

$$f1(y) = 0.a_1 a_2 \dots a_{p-1} b_p \dots \underline{b_t} \times \beta^e$$

$$f1(y) - f1(y) = 0.0 \ 0 \ \dots 0 \ c_p \dots c_t \times \beta^e = 0.c_p \dots c_t \times \beta^{e-p}$$

- Inițial avem o singură cifră inexactă, în poziția t , cu eroarea relativă β^{1-t}
- După scădere, bitul inexact trece în poziția $t-p$ cu eroarea relativă $\beta^{1-(t-p)}$ adică amplificată de β^p ori.

Să considerăm scăderea numerelor $x = 0.120$ și $y = -0.119$ în sistemul $F(10, 2, -10, 10)$:

$$f1(x) = -f1(y) = 0.120$$

$$\varepsilon = |(x+y) - f1(x+y)| / (x+y) = (0.001 - 0) / 0.001 = 1 \quad !$$

eroarea este de **100%** !

Se evită scăderea numerelor apropiate ca ordin de mărime prin:

- înmulțire cu conjugatul,
- dezvoltare în serie Taylor,
- rearanjarea termenilor etc .

Prin rearanjarea termenilor evităm adunarea numerelor foarte diferite ca ordin de mărime.

Astfel în sistemul $F(10, 3, -10, 10)$ cu rotunjire suma: $1 + 0.002 + 0.002 + 0.002$ calculată

$$\bullet \quad f1(f1(f1(1 + 0.002) + 0.002) + 0.002) = 1$$

în timp ce asocierea:

$$\bullet \quad f1(1 + f1(0.002 + f1(0.002 + 0.002))) = 1.01.$$

În aritmetica în virgulă mobilă, *asociativitatea* nu se mai păstrează. Astfel:

$$f1(f1(x+y)+z) \neq f1(x+f1(y+z)) .$$

De exemplu:

$$f1(f1(1+\mu/2) + \mu/2) = f1(1+\mu/2) = 1,$$

în timp ce:

$$f1(1+f1(\mu/2+\mu/2)) = f1(1+\mu) > 1$$

Reprezentarea numerelor reale (standardul IEEE 754).

Permite *reprezentarea realilor* în:

1) *precizie simplă* **F(2, 24, -126, 127)**, folosind 32 biți

2) *precizie dublă* **F(2, 53, -1022, 1023)**; se folosesc 64 biți:

3) *precizie extinsă* **F(2, 65, -16382, 16382)**; se folosesc 80 biți:

Întrucât $a_1=1$, acesta nu se mai reprezintă (este ascuns), câștigându-se astfel precizie suplimentară. Bitul ascuns este evidențiat în reprezentarea:

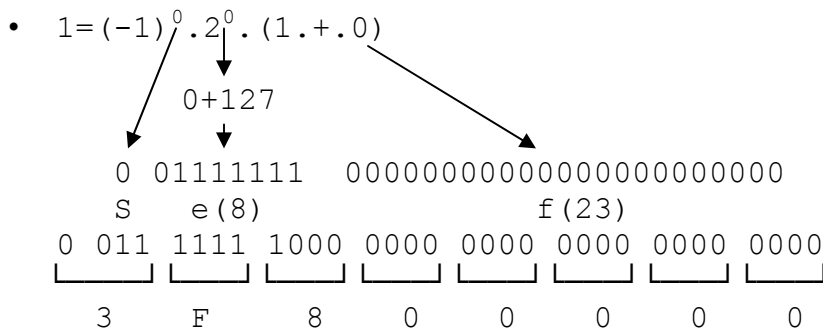
$$f1(x) = (-1)^s 2^e \cdot (1.+.f)$$

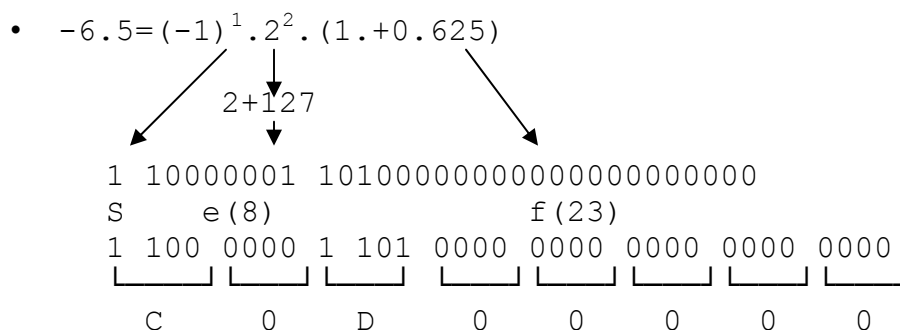
Precizie simplă

- reprezentare pe 32 biți
- baza = 2
- precizie = 24 biți (numerele normalizate păstrează numai 23 biți)
- Numărul real este păstrat prin 3 componente:
 - semnul: 1 bit
 - exponentul: 8 biți
 - mantisa: 23 biți(logic24 l)
- Cei 8 biți permit: $2^8 = 256$ valori diferite. Domeniul $[0, 255]$ este transformat în $[-127, 128]$
- La exponentul (pozitiv sau negativ) se adaugă o valoare constantă care duce la un *exponent deplasat* sau *caracteristică* pozitivă. Factorul de deplasare pentru precizie simplă este 127.
- Domeniul deplasat $[0-255]$ reprezintă xponenți în domeniul $[-127, 128]$

$$\text{exponent_deplasat} = \text{exponent} + 127$$

Valoarea numărului este: $V = (-1)^s \cdot 2^e \cdot (1.+.f)$





Un număr mai mare decât cel mai mare număr reprezentabil M (cunoscut sub numele de *modulul reprezentării*) se obține în urma unei *depășiri superioare* (de regulă o împărțire prin 0: $1/0 = \infty$, $-1/0 = -\infty$) va fi desemnat prin infinit – **Inf**, iar nedeterminările $0/0$, ∞/∞ etc, vor fi desemnate ca **NaN** (Not a Number).

Pentru toate acestea se rezervă în reprezentare cel mai mare exponent posibil 128 (adică exponentul deplasat 255).

Precizie simplă

	S	E (8biti)	e= E-127	H	f (23biti)	Valoare	
NaN	0	11111111	128	1	$\neq 0$		
+Inf	0	11111111	128	1	000 ...000	$(-1)^0 2^{128}$	0x7F800000
Ω	0	11111110	127	1	111 ...111	$(-1)^0 2^{127} (2-2^{-23}) \approx 3.4E38$	0x7F7FFFFF
$1+\epsilon$	0	01111111	0	1	000 ...001	$(-1)^0 2^0 (1+2^{-23})$ $\epsilon = 2^{-23} \approx 1.92E-7$	0x3F800001
1	0	01111111	0	1	000 ...000	$(-1)^0 2^0 = 1$	0x3F800000
ω	0	00000001	-126	1	000 ...000	$(-1)^0 2^{-126} = 2^{-126} \approx 1.18E-38$	0x00FFFFFF
MaxD	0	00000001	-126	0	111 ...111	$(-1)^0 2^{-126} (1-2^{-23}) = 2^{-126} - 2^{-149}$	
MinD	0	00000001	-126	0	000 ...001	$(-1)^0 2^{-126} 2^{-23} = 2^{-149} \approx 1.4E-45$	0x00000001
+0	0	00000000	-127	1	000 ...000	$(-1)^0 2^{-127} = 2^{-127}$	0x00000000

Precizie dublă

	S	E 11biti	E-1023	H	f (52biti)	Valoare	
NaN	0	11...11	1024	1	$\neq 0$		
+Inf	0	11...11	1024	1	000 ...000	$(-1)^0 2^{1024}$	
Ω	0	11...10	1023	1	111 ...111	$(-1)^0 2^{1023} (2-2^{-52}) \approx 1.8E308$	
$1+\epsilon$	0	01...01	0	1	000 ...001	$(-1)^0 2^0 (1+2^{-52})$ $\epsilon = 2^{-52} \approx 1.1E-16$	
1	0	01...01	0	1	000 ...000	$(-1)^0 2^0 = 1$	
ω	0	00...01	-1022	1	000 ...000	$(-1)^0 2^{-1022} = 2^{-1022} \approx 2.2E-308$	
MaxD	0	00...01	-1022	0	111 ...111	$(-1)^0 2^{-1022} (1-2^{-52}) = 2^{-1022} - 2^{-1074}$	
MinD	0	0...001	-1022	0	000 ...001	$(-1)^0 2^{-1022} 2^{-52} = 2^{-1074} \approx 5E-324$	
+0	0	0...000	-1023	1	000 ...000	$(-1)^0 2^{-1023} = 2^{-1023}$	

- Cel mai mic număr normalizat este $2^{-126} \approx 1.175 \text{E}-38$
- Cel mai mic număr denormalizat este $.00...1 * 2^{-126} = 2^{-149} \approx 1.4 \text{E}-45$
 - Infinit rezultă din calcule precum: $1/0 = \infty$, $-1/0 = -\infty$
 - Se reprezintă cu exponentul deplasat 255, (nedeplasat 128), și fracția 0. all 0's
 - Cel mai mare număr = $1.111...1 * 2^{127} \approx 3.4 \text{E}38$
- NaN (“not a number”)
 - Apare când se încearcă o operație nelegală (ca **sqrt** dintr-un număr negativ)
- Orice expresie care conține un termen NaN este evaluată ca NaN
 - Există cazuri în care apariția unui NaN nu declanșează nici o întrerupere (excepție) NaNs este “liniștit”
- Un NaN semnalizat declanșează o excepție (de exemplu o valoare neinițializată)
 - Alte NaN semnalizate:
- `sqrt(număr negativ)`
- $0 * \infty$ $0 / 0$ ∞ / ∞
- $x \% 0$ $\infty \% x$ $\infty - \infty$

Condiționarea problemelor.

- Condiționarea unei probleme caracterizează *sensibilitatea* soluției în raport cu erorile din datele de intrare.
- O problemă este *bine condiționată* dacă erori mici în date produc de asemeni erori mici în rezultate.
- Condiționarea este o *proprietate a problemei, independentă de soluția* aleasă.
- O problemă rău condiționată este „aproape nerezolvabilă” în practică (chiar dacă problema este rezolvată exact, soluția poate fi lipsită de semnificație).
- De exemplu, la evaluarea funcției $\mathbf{y} = \mathbf{f}(\mathbf{x})$, o perturbare a datelor $\mathbf{x} + \Delta \mathbf{x}$ produce o perturbare a soluției $\mathbf{y} + \Delta \mathbf{y} = \mathbf{f}(\mathbf{x} + \Delta \mathbf{x})$, în care:
- eroarea absolută $|\Delta \mathbf{y}| \approx |\mathbf{f}'(\mathbf{x})| \cdot |\Delta \mathbf{x}|$
- eroarea relativă $|\Delta \mathbf{y}| / |\mathbf{y}| \approx |\mathbf{f}'(\mathbf{x})| / |\mathbf{f}(\mathbf{x})| \cdot |\mathbf{x}| |\Delta \mathbf{x}| / |\mathbf{x}|$
- Problema este *rău condiționată* dacă factorul $|\mathbf{f}'(\mathbf{x})| / |\mathbf{f}(\mathbf{x})| \cdot |\mathbf{x}|$ este mare.

Stabilitatea numerică a algoritmilor

- Se referă la precizia algoritmului în prezența erorilor de rotunjire.
- Un algoritm este *instabil* dacă erorile de rotunjire produc erori mari în rezultate.

Exemple de algoritmi instabili:

- inversarea de matrice folosind determinanți
- rezolvarea sistemelor liniare prin factorizare LU fără pivotare

- utilizarea factorizării Cholesky în metoda celor mai mici pătrate (rezultate mult mai bune furnizează factorizarea QR).
- calculul valorilor proprii ca rădăcini ale polinomului caracteristic
- Un algoritm numeric stabil nu introduce o sensibilitate suplimentară la perturbații.
- Un algoritm stabil dă rezultate apropiate de soluția exactă pentru o problemă bine condiționată.
- Un algoritm stabil nu poate rezolva o problemă rău condiționată, dar un algoritm instabil poate da soluții slabe chiar pentru o problemă bine condiționată.

Dacă $\mathbf{f}: \mathbf{X} \rightarrow \mathbf{Y}$ este o problemă și $\tilde{\mathbf{f}}: \mathbf{X} \rightarrow \mathbf{Y}$ este un algoritm, atunci este *numeric stabil* dacă pentru $\forall \mathbf{x} \in \mathbf{X}, \exists \tilde{\mathbf{x}} \in \mathbf{X}$, astfel încât:

$$\|\tilde{\mathbf{f}}(\mathbf{x}) - \mathbf{f}(\tilde{\mathbf{x}})\| = O(\varepsilon_m) \quad \text{și} \quad \|\mathbf{x} - \tilde{\mathbf{x}}\| = O(\varepsilon_m)$$

Bibliografie.

- V.Iorga, B.Jora “Metode Numerice”, Ed. Albastră, 2005
- C.Popeea, B.Jora, B.Dumitrescu “Calcul Numeric – Algoritmi fundamentali”, Ed. ALL
- C.Moler “Numerical Computing with Matlab”
- V.Iorga, F.Pop “Metode Numerice – Îndrumar de laborator”