METODE NUMERICE

Obiective curs

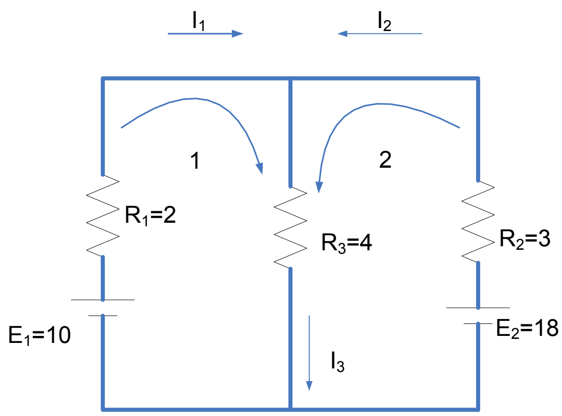
- Crearea, analiza şi implementarea de algoritmi pentru rezolvarea problemelor din *matematica continuă*
- Analiza complexităţii, analiza şi propagarea erorilor, condiţionarea problemelor şi stabilitatea numerică a algoritmilor problemelor numerice
- Prezentarea metodelor numerice clasice şi a celor moderne de rezolvare a problemelor ştiinţifice şi inginereşti
- Alegerea celor mai potrivite metode numerice pentru o problemă dată

Conţinut curs

- Reprezentare în virgulă mobilă. Standardul IEEE 754 pentru numere reale. Condiţionarea problemelor şi stabilitatea numerică a algoritmilor.
- Rezolvarea sistemelor de ecuaţii liniare prin metode gaussiene. Pivotare parţială şi totală. Factorizare LU.
- Propagarea erorilor în rezolvarea sistemelor de ecuaţii liniare.
- Metode iterative de rezolvare a sistemelor de ecuaţii liniare
- Interpolare polinomială. Polinom de interpolare Lagrange. Diferenţe divizate. Polinom Newton. Eroarea interpolării.
- Interpolare cu funcţii spline. Interpolare trigonometrică.
- Aproximare uniformă. Polinoame Cebâşev. Algoritmii lui Remes.
- Aproximare continuă şi 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 ecuaţiilor diferenţiale ordinare. Metode Runge-Kutta.
- Metode multipas explicite şi implicite. Predictor-corector.
- Convergenţa metodelor multipas
- Valori proprii şi vectori proprii. Metodele puterii
- Algoritmul QR cu deplasare explicită. Descompunerea valorilor singulare

Aplicaţii ale calculului numeric

1. Determinarea curenţilor într-un circuitul electric în regim staţionar:



conduce prin aplicarea legilor lui Kirchhoff, la un system de ecuaţii liniare:

$$\begin{cases} \mathbf{I}_1 + \mathbf{I}_2 - \mathbf{I}_3 = 0 \\ 2\mathbf{I}_1 + 4\mathbf{I}_3 = 10 \\ 3\mathbf{I}_2 + 4\mathbf{I}_3 = 18 \end{cases}$$

cu soluţia $\mathbf{I_1=1,\ I_2=2,\ I_3=3}$

2. **Modelul Leontieff** consideră economia formată din $\mathbf{n}$ sectoare independente: $S_1, S_2, \ldots, S_n$. Fiecare sector consumă bunuri produse de celelalte sectoare (inclusive cele produse de el însuşi). Introducem notaţiile:

$\mathbf{m_{ij}}$ = numărul de unităţi produse de sectorul $\mathbf{S_i}$ necesare sectorului $\mathbf{S_j}$ să producă o unitate

$\mathbf{p_i}$ = nivelul producţiei sectorului $\mathbf{S_i}$

$\mathbf{m_{ij}p_j}$ = numărul unităţilor produse de $\mathbf{S_i}$ şi consumate de $\mathbf{S_j}$

Numărul total de unităţi produs de $\mathbf{S_i}$ este: $\mathbf{p_1m_{i1}+p_2m_{i2}+\ldots+p_nm_{in}}$

Într-un system închis (autarhic) dacă economia este echilibrată, tot ce se produce trebuie consumat, adică:

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

Adică sistemul: $\mathbf{M.p = p}$ sau $\mathbf{(I-M).p=0}$, care pentru soluţii nenule, conduce la o problemă de valori şi vectori proprii.

Într-un model deschis de economie, unele sectoare îşi satisfac unele cerinţe din exterior, adică:

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

care conduce la sistemul liniar de ecuaţii:

$\mathbf{p = M.p + d}$

cu soluţia:

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

3. Coeficienţii care apar în reacţiile chimice se obţin aplicând legea conservării masei ecuaţiei de echilibru chimic. Astfel arderea etanului:

$$\mathbf{x}C_2H_6 + \mathbf{y}O_2 \rightarrow \mathbf{z}CO_2 + \mathbf{t}H_2O$$

dă sistemul de ecuaţii liniare:

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

care are o soluţie întreagă:

$$\mathbf{x=2,\ y=7,\ z=4,\ t=6.}$$

deci ecuaţia chimică este:

$$2C_2H_6 + 7O_2 \rightarrow 4CO_2 + 6H_2O.$$

O problemă 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 soluţie unică.

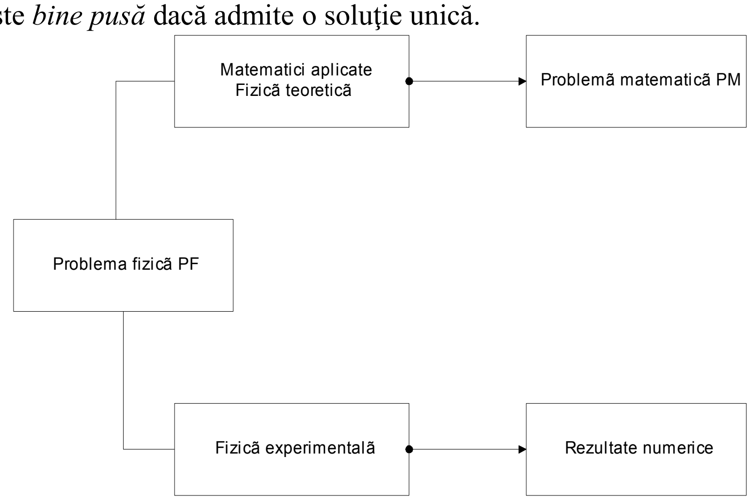


Fig.1.1. Modalităţi de abordare a problemelor fizice

Ca exemplu, vom considera următoarea problemă fizică:

P_F: ***Să se studieze propagarea temperaturii într-o bară*** $\mathbf{AB}$ ***de lungime*** $\mathbf{l}$ ***cunoscând***
-temperaturile la momentul iniţial în orice punct $\mathbf{M}$ ***al barei*** $\theta_0(\mathbf{x}),\ \mathbf{x} \in [0,\ \mathbf{l}]$
-temperaturile la cele două capete $\tau_{\mathbf{A}}(\mathbf{t})$ ***şi*** $\tau_{\mathbf{B}}(\mathbf{t})$ ***în orice moment*** $\mathbf{t} \in [0,\ \mathbf{t}1]$

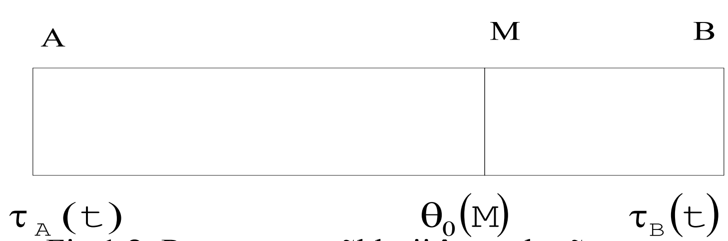


Fig.1.2. Propagarea căldurii într-o bară

Problema matematică corespunzătoare este:

$\mathbf{P_M}$ *: Să se determine funcţia:*

$$\theta:(x,t)\rightarrow\theta(x,t) \quad (1.43)$$
$$[0,l]\times[0,t1]\rightarrow\Re$$

care satisface următoarele condiţii:

1^0 $\quad \dfrac{\partial^2\theta}{\partial x^2} = K\cdot\dfrac{\partial\theta}{\partial t}$ *ecuaţia lui Fourier*

2^0 $\quad \theta(x, 0)=\theta_0(x)$ *condiţiile iniţiale*

3^0 $\quad \theta(0, t)=\tau_A(t)$ *condiţiile pe frontieră*

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

5^0 $\quad \theta\in S$, S = *spaţiul funcţiilor de 2 ori derivabile pe* $[0, l]\times[0, t1]$

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

Prezintă interes metodele de calcul în timp finit, cu: $0 < t < t1$ care furnizează aproximaţii uniforme: $N(x, t, \varepsilon)=N(\varepsilon)$.

Problema continuă $\mathbf{P_M}$ este transformată într-o problemă asemănătoare $\mathbf{P_h}$ prin *discretizare*.

În acest scop se selectează un număr finit de puncte $(i\Delta x, n\Delta t)$ din domeniul compact $[0, l]\times[0, t1]$ folosind o reţea de discretizare cu paşii:

$$\Delta x = \frac{l}{M},$$
$$\Delta t = \frac{t1}{N}.$$

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

Dacă se aproximează derivatele parţiale cu diferenţele finite:

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

se obţine următoarea **problemă discretizată**: $\mathbf{P_h}$*:*

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

$$1^0 \ \frac{\theta_i^{n+1}-\theta_i^n}{\Delta t} = K\cdot\frac{\theta_{i+1}^n - 2\cdot\theta_i^n + \theta_{i-1}^n}{(\Delta x)^2}$$
$$2^0 \ \theta_i^0 = \theta_0(i\Delta x)$$
$$3^0 \ \theta_0^n = \tau_A(n\Delta t)$$
$$4^0 \ \theta_M^n = \tau_B(n\Delta t)$$
$$5^0 \ \frac{\Delta t}{(\Delta x)^2} \le \frac{K}{2}$$

Problema discretizată $\mathbf{P_h}$ constă în rezolvarea succesivă a N sisteme de ecuaţii liniare tridiagonale.
Diferenţa: $\left|\theta(i\Delta x,\ n\Delta t)-\theta_i^n\right|$ evaluează apropierea între soluţia problemei discretizate P_h şi a modelului matematic P_M în fiecare punct al discretizării.
Soluţia problemei discretizate P_h trebuie să tindă spre soluţia problemei continue P_M dacă $h \to 0$ (h reprezintă pasul de discretizare – în cazul problemei considerate avem paşii $\Delta x \to 0, \Delta t \to 0$ sau $N \to \infty, M \to \infty$); vom spune că trebuie satisfacută o *condiţie de consistenţă*:

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

O altă condiţie importantă o reprezintă *stabilitatea*; aceasta impune ca soluţia θ' a problemei perturbate P_M (manifestată prin perturbarea parametrilor θ', τ'_A, τ'_B, K') să fie apropiată de soluţia θ a modelului matematic P_M.
Pe baza modelului matematic discretizat se va proiecta un algoritm, care va fi analizat prin prisma:

- *eficienţei* (resurse folosite: timp de calcul şi memorie),
- *convergenţei* către soluţia modelului matematic continuu,
- efectului *propagării erorilor*.

Etapele enumerate evidenţiază urmatoarele *tipuri de erori*:
- *erori deproblemă (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 soluţia problemei
- *erori de rotunjire* specifice rezolvării problemei pe calculatorul numeric, care utilizează aritmetica în virgulă mobilă mobilă

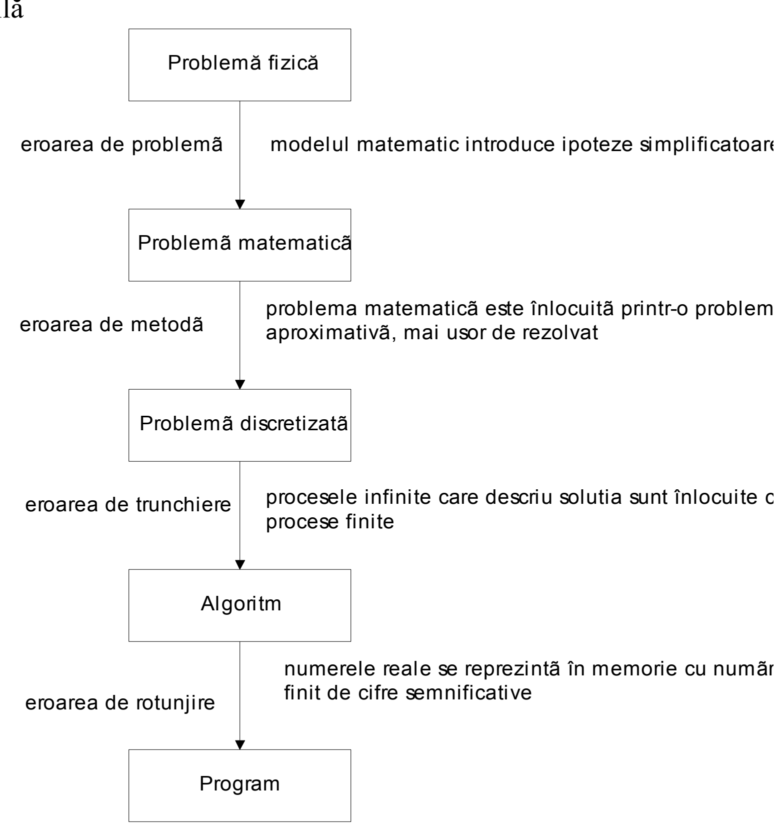


Fig. 1.3. Surse de erori

Reprezentarea în virgulă mobilă

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

reprezentare normalizată

$1 \le a_1 < \beta$ şi $0 \le a_i < \beta$, $i=2:t$

$L \le e \le U$

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

Exemplu: $F(10, 1, 0, 1)=\{\pm 0.a_1\times 10^e\}\cup\{0\}$ cu $a_1\in\{1:9\}$ şi $e\in\{0,1\}$, 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 $U-L+1$, şi
- semnul două).

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

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

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

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

Surse de erori.

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

Aproximarea numărului real

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

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

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

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

- Aproximarea prin rotunjire consideră:

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

O ***depăşire superioară*** apare dacă **e>U.**

Ea declanşează o *eroare la execuţie*, care conduce la întreruperea calculelor.

O ***depăşire inferioară*** apare dacă **e<L**;

ea duce la înlocuirea numărului prin zero.

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

$fl(1+\mu) > 1$

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

- $fl(1+0.0009)=fl(1.0009)=1$
- $fl(1+0.0010)=fl(1.0010)=1.001 > 1$

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

- Dacă se foloseşte rotunjire, atunci:

$fl(1+0.0004)=fl(1.0004)=1$
$fl(1+0.0005)=fl(1.0005)=1.001 > 1$
cu $\mu_{rot}=0.0005=1/2.10^{-3} =1/2.\mu_{tr}$

- ***Eroarea absolută*** la rotunjirea prin trunchiere**:**

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

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

Dacă se foloseşte **rotunjirea** atunci eroarea absolută este şi mai mică**:**

$$|e_x|\le 1/2.\ \beta^{e-t}$$

- ***Eroarea relativă*** este**:**

$$\varepsilon_x = |e_x|/|x| = |x-fl(x)|/|x| \le \beta^{e-t}/(0.a_1...a_t...\beta^e)$$
$$\varepsilon_x \le \beta^{e-t}/(0.10...0\beta^e)= \beta^{1-t}$$

$\varepsilon_x \le \beta^{1-t}$ la trunchiere
$\varepsilon_x \le 1/2.\ \beta^{1-t}$ la rotunjire

În general:

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

de unde deducem:

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

- De exemplu $F(10,4,-20,20)$, $\Omega=10^{20}(1-10^{-4}) =9.999\times10^{19}$,
- $\omega=10^{-20-1}=1.0\times10^{-21}$, $\mu_r=1/2.10^{-4+1}=5\times10^{-4}$

Propagarea erorilor

- numere aproximative -operaţii exacte
- operaţii aproximative - date exacte

1.Rezultatul exact al adunării a două numere **x** şi **y,** dacă *operaţiile se execută exact* este **x+y**.

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

$e_x=|\underline{x}-x|$ şi $e_y=|\underline{y}-y|$
$\underline{x}+\underline{y}=x+y\pm e_{x+y} =x\pm e_x+y\pm e_y=x+y\pm(e_x+e_y)$
$e_{x+y}=e_x+e_y$
$\varepsilon_{x+y}=e_{x+y}/|x+y|=(e_x+e_y)/|x+y|=(|x|\varepsilon_x+|y|\varepsilon_y)/|x+y|$
$\varepsilon_{x+y}=|x|/|x+y|\varepsilon_x+|y|/|x+y|\varepsilon_y=k_x\varepsilon_x+k_y\varepsilon_y$

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 coeficienţii de ponderare:

$k_x=|x|/|x-y|$ şi $k_y=|y|/|x-y|$

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 înmulţirii:

$\underline{xy}=xy\pm e_{xy}=(x\pm e_x)(y\pm e_y)=xy\pm xe_y\pm ye_x+e_xe_y\approx xy\pm(xe_y+ye_x)$

$e_{xy}=xe_y+ye_x$

$\varepsilon_{xy}=\varepsilon_x+\varepsilon_y$

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

$fl(x+y)=fl(f_xb^{ex}+f_yb^{-(ex-ey)}b^{ex})=fl[(f_x+ f_yb^{-(ex-ey)})\ b^{ex}]$

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

Rezultatul operaţiei este *normalizat*:

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

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 şi normalizare, deci:

$fl_2(x+y)=(x+y)(1+\mu)$

Anularea catastrofală

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

$fl(x)=0.a_1a_2...a_{p-1}a_p...\underline{a}_t\times\ \beta^e$

$fl(y)=0.a_1a_2...a_{p-1}b_p...\underline{b}_t\times\ \beta^e$

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

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

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

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

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

eroarea este de **100% !**

Se evită scăderea numerelor apropiate ca ordin de mărime prin:

- înmulţire 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

- **fl(fl(fl(1+0.002)+0.002)+0.002)=1**

în timp ce asocierea:

- **fl(1+fl(0.002+fl(0.002+0.002)))=1.01.**

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

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

De exemplu:

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

în timp ce:

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

Reprezentarea numerelor reale (standardul IEEE 754)

Permite ***reprezentarea realilor*** în:
1) *precizie simplă* **F(2, 24, -126, 127)**, folosind 32 biţi
2) *precizie dublă* **F(2, 53, -1022, 1023)**; se folosesc 64 biţi:
3) *precizie extinsă* **F(2, 65, -16382, 16382)**; se folosesc 80 biţi:

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

$fl(x)=(-1)^s 2^e.(1.+.f)$

Precizie simplă

- reprezentare pe 32 biţi
- baza $\beta= 2$
- precizie t= 24 biţi (numerele normalizate păstrează numai 23 biţi)
- Numărul real este păstrat prin 3 componente:
  - semnul: 1 bit
  - exponentul: 8 biţi
  - mantissa: 23 biţi (logic24)
- Cei 8 biţi 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 este127.
- Domeniul deplasat [0-255] reprezintă exponenţi în domeniul [-127, 128]

  ```
  exponent_deplasat = exponent + 127
  ```

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

- $1=(-1)^0.2^0.(1.+.0)$

```
    0+127

  0 01111111  00000000000000000000000
  S   e(8)                f(23)
0 011 1111 1000 0000 0000 0000 0000 0000
```

```
    3     F      8      0      0      0      0      0
```

- $-6.5=(-1)^{1}.2^{2}.(1.+0.625)$

```
          2+127

1 10000001 10100000000000000000000
S    e(8)            f(23)
1 100 0000 1 101 0000 0000 0000 0000 0000

   C     0     D     0     0     0     0     0
```

Un număr mai mare decât cel mai mare număr reprezentabil `M` (cunoscut sub numele de *modulul reprezentării*) se obţine în urma unei *depăşiri superioare* (de regulă o împărţire prin 0: `1/0 = ∞, -1/0 = -∞`) 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	≠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+ε	0	01111111	0	1	000 …001	$(-1)^0 2^0(1+2^{-23})$ $\varepsilon=2^{-23}\approx 1.92E-7$	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
Max D	0	00000001	-126	0	111 …111	$(-1)^0 2^{-126}(1-2^{-23})=2^{-126}-2^{-149}$	
		...					
Min D	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(11b)	E-1023	H	f(52biti)	Valoare
NaN	0	11…11	1024	1	≠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+ε	0	01…01	0	1	000 …001	$(-1)^0 2^0(1+2^{-52})$ $\varepsilon=2^{-52}\approx 1.1E-16$
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 $2^{-126}\approx 1.175E-38$
- Cel mai mic număr denormalizat este $.00…1 * 2^{-126} = 2^{-149}\approx 1.4E-45$

- Infinit rezultă din calcule precum: $1/0 = \infty, -1/0 = -\infty$
  Se reprezintă cu exponentul deplasat 255, (nedeplasat 128), şi fracţia 0. … 0
  Cel mai mare număr = $1.111...1*2^{127} \approx 3.4E38$
- NaN ("not a number")
  - Apare când se încearcă o operaţie nelegală (ca **sqrt** dintr-un număr negativ)
- Orice expresie care conţine un termen NaN este evaluată ca NaN
  - Există cazuri în care apariţia unui NaN nu declanşează nici o întrerupere (excepţie) NaNs este "liniştit"
- Un NaN semnalizat declanşează o excepţie (de exemplu o valoare neiniţializată)
  - Alte NaN semnalizate:
- sqrt(număr negativ)
- $0 * \infty$, $0 / 0$, ∞ / ∞
- x % 0, ∞ % x, $\infty - \infty$

Condiţionarea problemelor

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

Stabilitatea numerică a algoritmilor

- Stabilitatea numerică caracterizează erorile numerice introduse de algoritm, în ipoteza unor date de intrare exacte. Se referă la precizia algoritmului.
- Un algoritm este *instabil* dacă erorile de rotunjire produc erori mari în rezultate.
- Un algoritm numeric stabil nu introduce o sensibilitate suplimentară la perturbaţii.
- Un algoritm stabil dă rezultate apropiate de soluţia exactă pentru o problemă bine condiţionată.
- Un algoritm stabil nu poate rezolva o problemă rău condiţionată, dar un algoritm instabil poate da soluţii slabe chiar pentru o problemă bine condiţionată.

Dacă $\mathbf{f: X \to Y}$ este o *problemă* şi $\mathbf{f^{\sim}: X \to Y}$ este un *algoritm*, atunci acesta este ***numeric stabil*** dacă pentru $\forall \mathbf{x} \in \mathbf{X}, \exists \mathbf{x^{\sim}} \in \mathbf{X}$, astfel încât:

$$\|\mathbf{f^{\sim}(x)} - \mathbf{f(x^{\sim})}\| = O(\varepsilon_m) \text{ şi } \|\mathbf{x} - \mathbf{x^{\sim}}\| = O(\varepsilon_m)$$

Algoritmul $\tilde{f}$ destinat rezolvării problemei **f** este numeric stabil, dacă este îndeplinită una din condiţiile:

1. $\tilde{f}(x) \cong f(x)$, adică soluţia calculată aproximează bine soluţia exactă
2. există $\tilde{x}$ apropiat de **x** astfel încât $\tilde{f}(x) = f(\tilde{x})$ – soluţia calculată de algoritm cu date de intrare exacte este soluţia exactă cu date uşor perturbate.

Exemple de algoritmi instabili:

- inversarea de matrice folosind determinanţi
- 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

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"