CURS 1
INTRODUCERE ÎN CALCULUL ŞTIINŢIFIC

---

1. Analiza numerică şi calculul ştiinţific.
2. Reprezentarea numerelor în calculator (rapel).
3. Erori, măsura erorii.
4. Condiţionarea problemei. Stabilitatea metodei.
5. Exemple – Probleme rău-condiţionate şi probleme intratabile.
   - Polinomul Wilkinson; Matricea Wilkinson cu şi fără balansare.
   - SSH. Polinomul Rump.

1 Analiza numerică şi calculul ştiinţific

Obiectul Analizei numerice este studiul metodelor numerice desvoltate pentru rezolvarea efectivă a unei probleme ştiinţifice sau matematice – înţelegând prin rezolvare efectivă, găsirea unei soluţii aproximative a problemei. Studiul (analiza) metodei constă în:

- Construcţia metodei
- Analiza convergenţei şi erorii metodei
- Analiza condiţionării problemei şi stabilităţii metodei
- Compararea eficienţei metodelor care rezolvă o aceeaşi clasă de probleme

Astăzi, utilizarea unei metode numerice se face exclusiv prin programarea ei şi rulare pe calculator.

Utilizarea calculatorului a generat noi probleme de studiu (care nu făceau înainte obiectul Analizei numerice), şi anume:

- Studiul erorilor metodei, provenite din reprezentarea numerelor în calculator printr-un număr finit de cifre.
- Studiul algoritmilor pentru programarea metodei, şi a performanţei acestora.

Rezolvarea unei probleme matematice (adică, formulată prin ecuaţii), prin utilizarea unei metode numerice pe calculator constituie *calculul ştiinţific*. Analiza numerică împreună cu studiul

algoritmilor metodelor numerice, constituie *matematica calculului ştiinţific* (Kincaid & Cheney, 1996).

Utilizarea calculatorului şi a metodelor numerice, în rezolvarea unei probleme matematice sau pusă de o ştiinţă aplicată, a dat naştere unei noi discipline numită *Computational Science*.
Aceasta este un domeniu interdisciplinar, cuprinzând cunoştinţe din:

(1) o ştiinţă aplicată (fizică, inginerie, biologie, economie, ştiinţe sociale, etc.);
(2) matematică (inclusiv metode numerice);
(3) ştiinţa calculatoarelor (soft şi hard).
Specialistul în "Computational Science" îndeplineşte următoarele sarcini:

- Identifică fenomenul şi defineşte problema pusă de ştiinţa aplicată;
- Construieşte un model matematic;
- Alege o metodă numerică şi proiectează algoritmul metodei;
- Programează algoritmul şi obţine programul executabil.
- Face simulări numerice, în scopul de a valida modelul prin rezultatele obţinute:
  - Uneori, se procedează la rafinarea modelului, pentru a-l adapta fenomenului real.
  - Ades, din analiza făcută, se generează noi teorii şi proceduri numerice privind modelul şi algoritmul.

■

Principiul calculului cu numere reprezentate în virgulă flotantă (*cu un număr finit de cifre binare*):

- Rezultatul oricărei operaţii în virgulă flotantă este *imprecis*, cu o anumită toleranţă.
- Eroarea propagată într-un şir de astfel de operaţii se acumulează inevitabil.

Se poate spune:
"Un rezultat precis, este unul *tolerabil imprecis*."

2 Reprezentarea numerelor în calculator (rapel)

2.1 Întregi

INTEGER(n), n = 1, 2, 4, 8, se reprezintă pe 'n' octeţi contigui. Intregii sunt stocaţi în reprezentarea *complementului lui* 2, nume:

- O valoare pozitivă se reprezintă ca atare, în binar.
- O valoare negativă se reprezintă astfel: se inversează toţi biţii valorii pozitive şi apoi se adună 1. Pentru a regăsi o valoare negativă: se inversează toţi biţii şi se adună 1.

Reprezentarea numerelor întregi

Tipul	Număr de octeţi	Plaja de reprezentare	
		Baza 2	Zecimal
INTEGER(1)	1	-2^7 2^7-1	-128 127
INTEGER(2)	2	-2^{15} $2^{15}-1$	-32,768 32,767
INTEGER(4)	4	-2^{31} $2^{31}-1$	$\approx \pm 2.147 \times 10^9$ -2,147,483,648 2,147,483,647
INTEGER(8)	8	-2^{63} $2^{63}-1$	$\approx \pm 9.223 \times 10^{18}$ -9,223,372,036,854,775,808 9,223,372,036,854,775,807

2.2 Reali

Reprezentarea numerelor reale în calculator se zice reprezentare *în virgulă flotantă* (*Floating-Point*; abreviat "FP"). Numerele în virgulă flotantă aproximează numerele reale printr-un număr finit de cifre binare (*biţi*).

Exemplu pentru baza 10:

Fie numărul de reprezentat:

$x = 240.1308$

Acesta se poate scrie în una din formele:

$x = 0.2401308 \times 10^3$, (a)

în care: 0.2401308 este *fracţia*, $b = 10$ este baza, iar $e = 3$ este *exponentul*;

$x = 2.401308 \times 10^2$ (b)

în care: 2.401308 este *semnificandul*, iar $E = 2$ este *exponentul*.

Forma (a) este reprezentarea în modelul de reprezentare (sau, notaţia ştiinţifică); forma (b) este analoagă cu reprezentarea în calculator (unde baza este $b = 2$ – v. 2.2.2). Exponenţii reprezentărilor sunt legaţi prin $e = E + 1$.

2.2.1 Modelul de reprezentare (notaţia ştiinţifică)

Reprezentarea unui număr real *x* este una din următoarele expresii:

$x = 0$

$$x = \sigma \times \sum_{k=1}^{P} a_k \beta^{-k} \times \beta^e$$

Pentru $x \neq 0$, expresia de mai sus se scrie în reprezentare *poziţională*:

$$x = \sigma \times .a_1 a_2 \ldots a_P \times \beta^e$$

Parametrii reprezentării au semnificaţiile:

$$\sigma = sign(x) = \begin{cases} +1 & x > 0 \\ -1 & x < 0 \end{cases}$$

β = baza (întreg > 1); $a_1, a_2, \ldots, a_P$ = cifre în baza β.

$.a_1 a_2 \ldots a_P$ = mantisa (sau: fracţia) – conţine *P* cifre. Este un număr pozitiv, astfel că

$$\beta^{-1} \leq .a_1 a_2 \ldots a_P < 1$$

P = precizia reprezentării = numărul de cifre în mantisă.

e = exponentul reprezentării: $e_{\min} \leq e \leq e_{\max}$.

2.2.2 Reprezentarea în calculator binar:

$$x = (-1)^s \times b_0 . b_1 b_2 \ldots b_{P-1} \times 2^E$$

unde:

s = bitul de semn = $\begin{cases} 0 & - \text{pentru } x > 0 \\ 1 & - \text{pentru } x < 0 \end{cases}$; Pentru $x = 0$, v. – 3.5.

b_N = cifre binare: 0 sau 1

$b_0 \cdot b_1 \ldots b_{P-1}$ = *semnificandul*, care reprezintă partea non-exponent a numărului. Este un număr nenegativ care conţine *P* cifre binare.

$b_1 \ldots b_{P-1}$ se zice *fracţia*; conţine $P-1$ cifre binare. Notând semnificandul cu *sig* şi fracţia cu *f*, avem $sig = b_0.f$, unde, $b_0 = 1$ sau 0.

P = precizia reprezentării = numărul de cifre binare în semnificand.

b_0 = primul bit al semnificandului ('leading bit'). Reprezentarea standard este *normalizată*, adică întotdeauna $b_0 = 1$, şi acesta nu se mai stochează. Punctul binar se presupune la dreapta lui b_0. In consecinţă, precizia *P* este asigurată cu un număr de $P-1$ biţi, şi avem:

$1 \le semnificand < 2$.

E = exponentul reprezentării: $E_{\min} \le E \le E_{\max}$.

Observaţie

Concret, reprezentarea normalizată este

$$x = (-1)^s \times 1.\ f \times 2^E .$$

Pentru numere *denormalizate* şi valori speciale – v. 2.3.2 (aici $sig \ge 0$).

Exemplu

Simplă precizie (Format simplu): *P* = 24 biţi – asigurată cu 23 biţi.

Dublă precizie (Format dublu): *P* = 53 biţi – asigurată cu 52 biţi ■

Observaţie

În modelul ştiinţific 2.2.1, numărul este reprezentat astfel:

$$x = (-1)^s \times .b_0 b_1 b_2 \ldots b_{P-1} \times 2^e ; \quad e = E + 1$$ ■

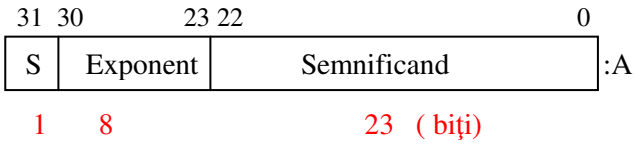
2.3 Structura formatelor

Tipul REAL(4) ocupă 4 octeţi contigui şi este stocat în formatul simplu (IEEE S_floating). Tipul REAL(8) ocupă 8 octeţi contigui şi este stocat în formatul dublu (IEEE T_floating). Figurile următoare prezintă structura formatelor: Cifrele indică numerotarea biţilor; 'S' desemnează bitul de semn. Simbolul ":A" reprezintă adresa octetului care conţine bitul 0 (adresa de start a datei reprezentate). Formatul este numit "nativ" şi convenţia de stocare "Little endian".

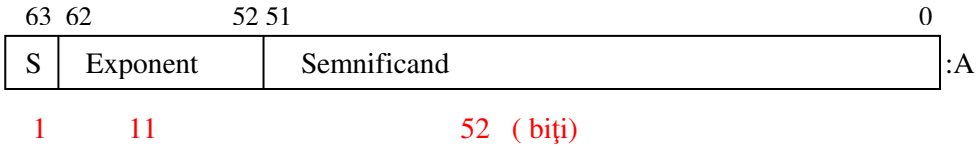
CVF utilizează convenţia "Little endian". Convenţia "Big endian" este utilizată, de exemplu, de calculatoarele IBM/370. V. pentru detalii, CVF – Programmer's Guide (2001).

■

Format simplu:



Format dublu:



2.3.1 Formate IEEE

Următorul tabel prezintă parametrii reprezentării pentru formatele IEEE în virgulă flotantă, suportate de CVF.

Formate IEEE

PARAMETRU	FORMAT		
	Simplu	Dublu	Dublu extins
Format, lungime în biţi	32	64	80
Bitul de semn, lungime în biţi	1	1	1
Semnificand, lungime câmp în biţi	23	52	64
Precizie *P*, în biţi	24	53	64
Exponent, lungime câmp în biţi	8	11	15
E_{max}	+ 127	+ 1023	+ 16383
E_{min}	- 126	- 1022	- 16382
Deplasare exponent (‘bias’)	+ 127	+ 1023	+ 16383
Precizie, în cifre zecimale	6 - 9	15 - 17	18 - 2 1
Plaja de reprezentare (aproximativ)	$10^{\pm 38}$	$10^{\pm 308}$	$10^{\pm 4932}$

Notă

- REAL(4) se reprezintă în formatul simplu. REAL(8) – în formatul dublu. Reprezentarea este *normalizată*: primul bit al semnificandului se presupune 1 şi acesta nu se mai stochează. În fapt, în câmpul semnificandului se stochează *fracţia*.
- Formatul dublu-extins nu trebuie normalizat, el utilizează 64 biţi pentru precizie. Acest format este utilizat intern, de către co-procesorul matematic. Bitul b_0 este stocat explicit (în locaţia 63 a câmpului semnificand).
- La fiecare exponent se adaugă o "deplasare" astfel încât să apară numai exponenţi stocaţi *pozitivi.* Exponentul stocat (în câmpul exponent) este

  $E_{stocat} = E + deplasare > 0$,

  iar exponentul reprezentării este

  $E = E_{stocat} - deplasare$.

  Primul bit al semnificandului se presupune egal cu 1 atât timp cât exponentul stocat nu este zero. Dacă $E_{stocat} = 0$, atunci valoarea reprezentată este o valoare specială (un număr denormalizat sau ±0) – v. 2.3.2.
- În modelul de reprezentare analitic (2.2.1) avem, conform $e = E + 1$:

  Format simplu: $e_{\max} = 128$, $e_{\min} = -125$

  Format dublu: $e_{\max} = 1024$, $e_{\min} = -1021$ ■

Exemplu

Numărul 1.0, în format simplu, se reprezintă prin următorul şir de biţi (câmpul exponent este umbrit):

|0|01111111|00000000000000000000000|

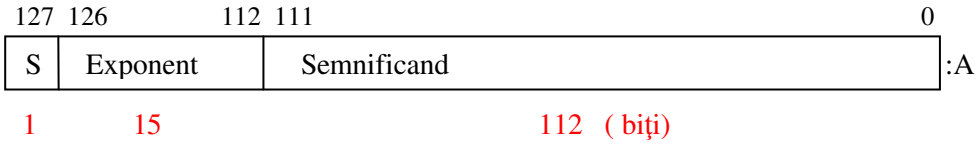
Exponentul stocat este: $E_{stocat} = 2^7\text{-}1 = 127$; exponentul real este: $E = 127\text{-}127 = 0$; astfel, reprezentarea 2.2.2 este: $+(1.0)_2 \times 2^0$.

■

Notă: REAL(16)

Standardul IEEE 754 nu specifică un format pentru pentru tipul REAL(16). Acest tip este implementat soft de unele compilatoare şi suportat hard de unele procesoare recente. Se reprezintă într-un format pe 16 octeţi (format cvadruplu – INTEL; dublu extins – SPARC), numit IEEE X_floating.

Formatul cvadruplu (Intel Fortran 11):



Parametrii formatului sunt:

- Lungime format: 128 biţi
- Câmp semnificand = 112 biţi.
- Câmp exponent: 15 biţi
- Precizie P = 113 biţi
- Precizie în cifre zecimale: 33 – 36
- Plaja de reprezentare (numere normalizate): $3.362\ldots\times10^{-4932}$ ⋯ $1.1897\ldots\times10^{4932}$.

■

2.3.2 Formatul IEEE – Valori speciale

Există patru cazuri de combinare exponent – semnificand, care conduc la valorile speciale 1- 4 prezentate mai jos. Ele se adaugă numerelor normalizate – poziţia 0 din tabel. "*f* " reprezintă fracţia.

Formatul IEEE – Valori în virgulă flotantă

Nr. crt.	Denumire	Cantitate reprezentată	Exponent *E*	Semnificand *sig*
0	Număr normalizat	$\pm 1.f\times 2^{E}$	$E_{\min} \le E \le E_{\max}$	*sig*
1	Zero cu semn	± 0	$E = E_{\min} - 1^{*}$	$sig = 0$
2	Număr denormalizat	$\pm 0.f\times 2^{E\min}$	$E = E_{\min} - 1$	$sig \neq 0$ $(f \neq 0)$
3	Infinit cu semn	$\pm \infty$	$E = E_{\max} + 1$	$sig = 0$
4	Not a Number	NaN	$E = E_{\max} + 1$	$sig \neq 0$

* Exponentul stocat este 0. Prin convenție, reprezentarea se face cu $E = 0$

2.3.3 Valori reprezentabile

a) Numere denormalizate:

Tip dată	Baza 2	Zecimal
REAL(4)		
Negative:	$-2^{-127}\times 1.11...$ $-2^{-126}\times 1.00...$	-3.403×10^{38} -1.175×10^{-38}
Zero:	0.	0.
Pozitive:	$2^{-126}\times 1.00...$ $2^{127}\times 1.11...$	1.175×10^{-38} 3.403×10^{38}
REAL(8)		
Negative:	$-2^{1023}\times 1.11...$ $-2^{-1022}\times 1.00...$	-1.798×10^{308} -2.250×10^{-308}
Zero:	0.	0.
Pozitive:	$2^{-1022}\times 1.00...$ $2^{1023}\times 1.11...$	2.250×10^{-308} 1.798×10^{308}

Notă: Tipul REAL(16) (Intel Fortran 11)

Plaja de reprezentare (zecimal):

$-1.190\times 10^{+4962}$ ... -3.362×10^{-4962}; 0.0; 3.362×10^{-4962} ... $1.190\times 10^{+4962}$

a) Numere denormalizate:

Sunt cuprinse între valorile de mai jos.

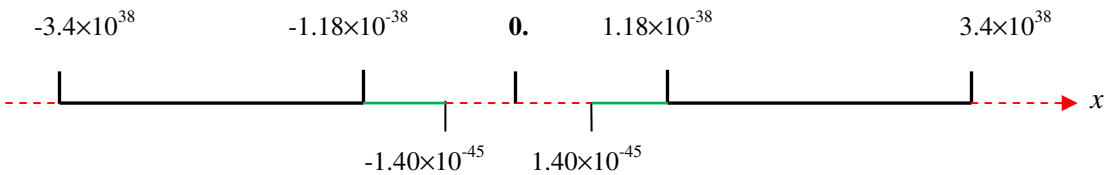
REAL(4): $-2^{-149}\times 1.00...$ $-2^{-126}\times 1.00...$ (-1.18×10^{-38} -1.40×10^{-45})

$2^{-149}\times 1.00...$ $2^{-126}\times 1.00...$ (1.40×10^{-45} 1.18×10^{-38})

REAL(8): $-2^{-1022}\times 1.00...$ $-2^{-1074}\times 1.00...$ (-2.23×10^{-308} 4.94×10^{-324})

$2^{-1074}\times 1.00...$ $2^{-1022}\times 1.00...$ (4.94×10^{-324} 2.23×10^{-308})

Valorile reprezentabile pentru tipul REAL(4) sunt indicate în figura de mai jos. Pentru tipul REAL(8) figura este analogă, cu valorile de mai sus.



Valori reprezentabile REAL(4)

Legenda:

———	Numere normalizate
———	Numere denormalizate
- - - - - -	Numere ne-reprezentabile REAL(4)

3 Erori, măsura erorii

Eroarea de rotunjire este diferenţa dintre un număr real (care nu se reprezintă exact FP) şi numărul FP cel mai apropiat care îl reprezintă.

Definiţia 1

Eroarea = Valoarea exactă – Valoarea aproximativă ■

Notăm: x = valoarea exactă; $f\ell(x)$ = reprezentarea în virgulă flotantă. Eroarea de rotunjire este:

$$Err(f\ell(x)) = x - f\ell(x)$$

Definiţia 2

$$Eroarea\ relativă = \frac{Eroarea}{Valoarea\ exacta}$$ ■

Eroarea relativă la rotunjire este:

$$Rel(fl(x)) = \frac{x - fl(x)}{x}$$

Măsurile erorii de rotunjire şi erorii relative de rotunjire, se descriu prin mărimile prezentate în continuare.

3.1 ULP (Unit in the Last Place)

Definiţia 3

ULP(*x*) este numărul cu 1 în bitul ultim (b_{P-1}), 0 (zero) în restul biţilor, şi cu acelaşi exponent ca şi *x* ■

Fie $x > 0$ reprezentat în calculator (modelul 3.2), $x = b_0.b_1 \ldots b_{P-1} \times 2^E$. Avem:

$$ULP(x) = 0.\underbrace{00\ldots0}_{1\ 2}\ \underbrace{1}_{P-1} \times 2^E = 2^{-P+1} \times 2^E,$$

unde *E* este exponentul reprezentării în calculator.

În modelul 3.1 (notaţia ştiinţifică), $x = 0.b_0 b_1 \ldots b_{P-1} \times 2^e$, şi avem

$$ULP(x) = 0.\underbrace{00\ldots0}_{1\ 2}\underbrace{1}_{P} \times 2^e = 2^{-P} \times 2^e,$$

unde *e* este exponentul lui *x* în modelul 3.1. Rezultatele sunt identice, întrucât $e = E + 1$.

Propoziţia 1

Cel mai mic număr FP care adunat la *x* produce, prin rotunjire, un număr FP mai mare decât *x* este

$$ULP(x) = 2^{-P+1} \times 2^E,$$

unde *E* este exponentul reprezentării lui *x* (în calculator, modelul 2.2.2) ■

Într-adevăr, în calculator:

$$x + ULP(x) = b_0.b_1 \ldots (b_{P-1} + 1) \times 2^E > x$$

Sau, în modelul 3.1, avem

$$x + ULP(x) = .b_0\, b_1 \ldots (b_{P-1} + 1) \times 2^e > x\,.$$

Observaţie

ULP este util in algoritmii numerici care includ o iteraţie şi un test de oprire a iteraţiei de tipul

$$| x_{n+1} - x_n | \le TOL\,.$$

Valoarea toleranţei *TOL* trebuie aleasă *mai mare sau egală cu ULP*(*x*), pentru domeniul valorilor lui *x*.

Exemplu - 1

Fie $x = 1.0$.

În calculator – modelul 2.2.2, în formatul simplu, avem reprezentările:

1.0: $1.000\ 0000\ 0000\ 0000\ 0000\ 0000 \times 2^0 = (1.0)_2 \times 2^0$

$ULP(1.0)$: $0.000\ 0000\ 0000\ 0000\ 0000\ 0001 \times 2^0 = (1.0)_2 \times 2^{-23}$.

Analitic, exponentul reprezentării este $E = 0$, şi din Propoziţia 1 rezultă

$ULP(1.0) = 2^{-P+1} \times 2^0 = 2^{-P+1}$.

Expresia este valabilă şi pentru formatul dublu, unde reprezentarea e analoagă cu cea de mai sus, tot cu $E = 0$.

De exemplu, în formatul simplu, cu $P = 24$, rezultă $ULP(1.0) = 2^{-23}$.

Analog, în modelul 3.1: $1.0 = (0.1)_2 \times 2^1$, şi avem $ULP(1.0) = 2^{-P} \times 2^1 = 2^{-P+1}$ ■

Propoziţia 2

Eroarea de rotunjire satisface:

$\left|Err\left(f\ell(x)\right)\right| \le \frac{1}{2} ULP(x)$ ■

Propoziţia 3

$\frac{1}{2} \cdot 2^{-P} \le \left|Rel\left(f\ell(x)\right)\right| \le 2^{-P}$ ■

Pentru demonstraţii – v. Chisăliţă A., "Numerical Analysis".

3.2 'ε- maşină' şi *EPS*

Definiţia 4

Limita superioară a erorii relative se numeşte *ε-maşină* = 2^{-P} ■

Astfel, avem:

- Format simplu: $P = 24$; *ε-maşină* = 2^{-24}; Zecimal: $5.96\ldots \times 10^{-8}$
- Format dublu, $P = 53$; *ε-maşină* = 2^{-53}; Zecimal: $1.11\ldots \times 10^{-16}$

Definiţia 5

EPS este cel mai mic număr în virgulă flotantă, care adunat la 1.0 dă un rezultat care nu se rotunjeşte la 1.0:

$x = 1.0 + EPS \;\Rightarrow\; x > 1.0$ ■

Propoziţia 3 *EPS* = 2×*ε-maşină* ■

Aceasta rezultă din Propoziţia 1 cu $x = 1.0$, pentru care $E = 0$ (v. Exemplu –1):

$EPS = ULP(1.0) = 2^{-P+1} = 2\times$*ε-maşină*.

Exemplu – 2:

Format simplu: $EPS = 2^{-23}$. Zecimal: $1.192\ldots \times 10^{-7}$.

Format dublu : $EPS = 2^{-52}$. Zecimal: $2.220\ldots \times 10^{-16}$.

Observaţii

- Funcţia intrinsecă `EPSILON(x)`, returnează mărimea *EPS* pentru un real de tipul argumentului *x*. (REAL(4) sau REAL(8).)
- *EPS* este cerut ca dată de intrare în unii algoritmi numerici. (Exemplu: integrarea sistemelor de ecuaţii diferenţiale).

Notă

Uneori, eroarea se măsoară în număr de *ulp*-uri, măsura erorii fiind definită prin:

$$\frac{|x_T - x_A|}{ulp(x_A)}$$

V. Demmel & Hida (2002) ■

4 Condiţionarea problemei. Stabilitatea metodei.

Condiţionarea problemei este sensibilitea soluţiei (exacte) la mici schimbări în datele problemei. Dacă soluţia este sensibilă la mici schimbări în date, problema se zice *rău-condiţionată*. In caz contrar, problema se zice *bine- condiţionată*.

Stabilitatea metodei: In anumite condiţii, soluţia calculată iterativ creşte nemărginit odată cu creşterea numărului paşilor procesului. In acest caz, metoda numerică se zice *instabilă*. V. exemplele următoare.

Concluzii:

Trebuie făcută distincţie între *condiţionarea problemei* şi *stabilitatea metodei* de calcul.

- O problemă bine-condiţionată poate fi rezolvată precis prin orice metodă stabilă pentru problema respectivă.
- O problemă *rău-condiţionată* , care are soluţie, poate fi rezolvată precis numai utilizând precizie multiplă.
- O metodă *instabilă* utilizată pentru o problemă bine condiţionată, poate da în primii paşi rezultate bune, dar va da inevitabil rezultate rele după un anumit număr de paşi – pe măsură ce erorile propagate vor ”acoperi” soluţia reală.

5 Exemple – Probleme rău-condiţionate şi probleme intratabile.

5.1 Probleme rău-condiţionate

5.1.1 Polinomul lui Wilkinson

Fie

$f(z) = (z-1)(z-2)\ldots(z-20)$

cu rădăcinile $z = 1, 2, \ldots, 20$.

Considerăm polinomul perturbat:

$\tilde{f}(z) = f(z) - 2^{-23} \cdot z^{19}$,

unde avem: $2^{-23} = 1.19209\ldots * 10^{-7}$. Acesta are 5 perechi de rădăcini complexe; exemplu: $z = 19.502 \pm 1.940i$. (Polinomul mai are 10 rădăcini reale in intervalul (1,10)).

Deci, modificarea coeficientului lui z^{19} cu aproximativ 10^{-7} (concret: din -210 în -210.000000119…), conduce nu numai la modificarea valorilor, dar chiar la modificarea tabloului rădăcinilor.

Note:

- Polinomul de mai sus se numeşte polinomul lui Wilkinson, după numele celui care l-a

introdus în 1984 (James H. Wilkinson, "The perfidious polynomial". In: G. H. Golub, editor, Studies in Numerical Analysis, volume 24 of Studies in Mathematics, pages 1-28. Mathematical Association of America, Washington, D.C., 1984).
Polinomul a generat, şi generează şi în prezent, studii privind algoritmi pentru calculul precis al rădăcinilor unui polinom.

- Pe de altă parte, chiar polinomul neperturbat este *intratabil* cu un soft în care precizia maximă este dubla precizie: termenul liber este 20! = 2432902008176640000, şi acesta are 19 cifre, în timp ce precizia maximă este de 16 cifre semnificative. De exemplu, cu coeficienţii polinomului calculaţi în dublă precizie, programul Pol_Direct în dublă precizie, cu toleranţa 1E-14 şi numărul maxim de iteraţii 100, dă pseudo-rădăcinile şi valorile polinomului:

19.9999903868239	-786592049152.000
19.0000865019199	2172793747456.00

■

Acest exemplu arată că, problema găsirii rădăcinilor unui polinom poate fi ades, o problemă rău-condiţionată.

5.1.2 Matricea Wilkinson

Considerăm matricea Wilkinson de ordinul 20 (Wilkinson (1963), Westlake (1968)):

$$\mathbf{A} = \begin{bmatrix} 1 & 20 & & & & & \\ & 2 & 20 & & & & \\ & & 3 & 20 & & & \\ & & & \ddots & & & \\ & & & & 19 & 20 & \\ \varepsilon & & & & & 20 \end{bmatrix}$$

Analitic, matricea $n \times n$, este definită prin $a(i, j) = 0$, cu excepţia elementelor:

$a(i,i) = i; \qquad i = \overline{1,n}$

$a(i,i+1) = n\,; \;\; i = \overline{1,n-1}\,;$

$a(n,1) = \varepsilon\,.$

Exemplul Wilkinson ia $n = 20$.

Dacă perturbaţia $\varepsilon = 0$, matricea este superior triunghiulară, şi valorile proprii sunt $\lambda(i) = i$. Pentru $\varepsilon = 1E-10$, cu metoda QR *fără rafinare* (cu *EPS* = 1E-6), se obţin următoarele rezultate:

Fără balansare: Valorile proprii rezultă toate reale. Ele reprezintă perturbaţii ale valorilor *i*, $i = \overline{20,1}$, anume (ordonate descrescător):

```
19.99999998462896,    19.00001298980951,    17.99999341364806,    ...,
3.000002187998112, 1.999999854534825, 1.000000000000000
```

Cu balansare: Valorile proprii calculate conţin 14 valori complexe, şi anume (ordonate după *i* descrescător):

20	20.00424815126970	
19	18.89075561522547	
18	18.42511929989719	
17	17.03466897322695	1.087736309816453
16	17.03466897322695	-1.087736309816453
15	15.10602239289448	1.948529250030004
14	15.10602239289448	-1.948529250030004
13	12.88192664561770	2.529181675892914
12	12.88192664561770	-2.529181675892914
11	10.49999998763635	2.733397362516592
10	10.49999998763635	-2.733397362516592
9	8.118073427255256	2.529181726294424
8	8.118073427255256	-2.529181726294424
7	5.893977535006919	1.948529282139898
6	5.893977535006919	-1.948529282139898
5	3.965330675018651	1.087735665091806
4	3.965330675018651	-1.087735665091806
3	2.574881415008950	
2	2.109241835064037	
1	0.9957544102238627	

Se observă necesitatea balansării în acest caz.

Explicaţia rezultatelor foarte diferite se poate da analizând condiţionarea rădăcinilor polinomului caracteristic al matricii **A**. Acesta este

$$p(\lambda) = (\lambda - 1)(\lambda - 2)\ldots(\lambda - 20) - 20^{19}\varepsilon$$

Notăm rădăcinile cu $\lambda_i(\varepsilon)$, iar $\lambda_i(0) = \lambda_i = i$. Numărul de condiţie al rădăcinii λ_i este $K_i = \left| q(\lambda_i) / p'(\lambda_i) \right|$, unde $q(z) = -20^{19}$. Avem:

$$K_i = \frac{20^{19}}{(i-1)!(20-i)!}$$

Numerele de condiţie maxim şi minim se ating pentru $i = 10, 11$ şi $i = 1, 20$, respectiv. Anume:

$K_{10} = K_{11} = 20^{19} / (10 \times 9!) \approx 10^{12} \times 3.98$; $\quad K_1 = K_{20} = 20^{19} / 19! \approx 10^7 \times 4.31$.

Numerele de condiţie foarte mari arată senzitivitatea mare a rădăcinilor la mici perturbaţii în coeficienţi.

În acest caz, valorile proprii ale matricii sunt rău-condiţionate. V. § 6.

5.2 Sumarea

Sumarea este una din cele mai importante operaţii în calculul ştiiţific.

Sumarea intervine în aproape orice calculaţie. Exemple:

- Suma unor numere: $S = \sum_{i=1}^{n} x_i$
- Produsul scalar: $P = \sum_{i=1}^{n} x_i y_i$
- Produsul unei matrici cu un vector (şi produsul a două matrici): $\mathbf{b} = \mathbf{Ax}$,

  $b_i = \sum_{j=1}^{n} a_{ij} x_j$
- Calculul valorii unui polinom: $p(x) = \sum_{i=0}^{n} a_i x^i$
- Calculul funcţiilor intrinseci: acestea se calculează prin serii trunchiate, adică prin polinoame.
- Etc.

Datorită reprezentării numerelor cu un număr finit de cifre în mantisă, suma în calculator poate să nu mai aibă proprietăţile din algebră, anume suma nu mai este comutativă sau asociativă.

Pentru a releva problemele pe care le pune sumarea prezentăm următoarele exemple.

5.2.1 Înălţimea suprafeţei mării (SSH - Sea Surface Height)

În 1999, în cadrul unui program de cercetare, s-au făcut măsurători (din satelit) ale înălţimii suprafeţei oceanului, în scopul calculării înălţimii medii – şi utilizării acesteia într-un model al circulaţiei oceanului (He & Ding(2001)).

Reţeaua cuprindea 120×64 = 7680 de puncte (120 longitudini şi 64 de latitudini). Datele sunt numere de ordinul de mărime 10^{12} …10^{15} şi de semne diferite. Datele corespunzând observaţiilor dintr-o zi s-au stocat într-un tablou `ssh(120,64)`, de date cu precizia de 64 biţi.

Ordonarea sumei după diferite criterii produce rezultatele de mai jos (calculate cu precizia de 64 biţi):

Ordonarea sumei după	Valoarea
Longitude crescător	34.4147682189941410
Longitude descrescător	32.302734375
Latitude crescător	0.67326545715332031
Latitude descrescător	0.734375
Valoarea corectă	0.35798583924770355

Valoarea corectă s-a obţinut cu un pachet soft de precizie multiplă (MPFUN, Bailey)

5.2.2 Polinomul lui Rump

Se consideră ”polinomul”

$$f(x,y) = 333.75y^6 + x^2(11x^2y^2 - y^6 - 121y^4 - 2) + 5.5y^8 + x/(2y)$$

şi evaluarea lui pentru $x = 77617$ and $y = 33096$.

Rezultatul calculului, cu 3 nivele de precizie (oferite de compilatorul CVF 6.6), sunt date mai jos.

Simplă precizie:	-1.180592E+21
Dublă precizie:	-1.180591620717411E+21
Precizie extinsă (64 biţi):	5.764607523034234880E+17
Valoarea corectă:	-0.82739 60599 46821 36814 11650 95479 81629 ...

(obţinută cu precizie de 128 biţi ≈ 38-39 cifre zecimele de precizie).

Pentru rezulate similare, obţinute cu compilatorul FORTE Developer 6 (Fortran 95, Sun) v. Loh & Walster (2002).

Explicaţie:

Cei patru termeni ai polinomului au valorile:

```
t1:   4.386057508463932E+29
t2:  -7.917111779274714E+36
t3:   7.917111340668962E+36
t4:    1.17260394005318
```

Astfel, nu se produce depăşire de format nici pentru simpla precizie. Dar termenii doi şi trei sunt egali cu aproximativ 2^{123}, şi astfel, pentru reprezentarea cu toate cifrele pentru evitarea pierderii de semnificaţie, este necesară o precizie de cel puţin 123 biţi. Ori, chiar precizia cvadruplă (real(16)) nu oferă decât 113 biţi. Astfel, pentru evaluarea polinomului în forma dată, trebuie apelat la un pachet multi-precizie – cum este, de exemplu, MPFUN.

Rezultate obţinute cu MPFUN – cu diferite nivele de precizie:

```
* Multiple Precision (    21 decimal digits;     5 words):
10 ^ 0 x  1.17260394005317863185883490452331093O,

* Multiple Precision (    28 decimal digits;     6 words):
10 ^ -1 x -8.27396059946821368141165095479816291699033115784384819917,

 * Multiple Precision (    36 decimal digits;     7 words):
10 ^ -1 x -8.27396059946821368141165095479816291999033115784384819917,

 * Multiple Precision (    43 decimal digits;     8 words):
10 ^ -1 x -8.27396059946821368141165095479816291999033115784384819917,

 * Multiple Precision (    50 decimal digits;     9 words):
10 ^ -1 x -8.27396059946821368141165095479816291999033115784384819917,

 * Multiple Precision (    57 decimal digits;    10 words):
10 ^ -1 x -8.27396059946821368141165095479816291999033115784384819917,
```

■

Explicaţia *analitică* este următoarea (Loh & Walster (2002)): *x* şi *y* daţi, satisfac relaţia $x^2 = 5.5y^2 + 1$. Cu aceasta, termenii de ordin superior lui 1 în *x* şi *y* se reduc, şi expresia polinomului devine

$$f_1(x, y) = \frac{x}{2y} - 2.$$

Rezultat:

```
* x/(2*y)-2 =       -0.82739605994682131       (Double Precision)
```
