

CURS 1

INTRODUCERE ÎN CALCULUL ȘTIINȚIFIC

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

1 Analiza numerică și calculul științific

Obiectul Analizei numerice este studiul metodelor numerice dezvoltate pentru rezolvarea efectivă a unei probleme științifice sau matematice – înțelegând prin rezolvare efectivă, găsirea unei soluții aproximative a problemei. Studiul (analiza) metodei constă în:

- Construcția metodei
- Analiza convergenței și erorii metodei
- Analiza condiționării problemei și stabilității metodei
- Compararea eficienței metodelor care rezolvă o aceeași clasă de probleme

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

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

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

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

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

Utilizarea calculatorului și a metodelor numerice, în rezolvarea unei probleme matematice sau pusă de o știință aplicată, a dat naștere unei noi discipline numită *Computational Science*.

Aceasta este un domeniu interdisciplinar, cuprinzând cunoștințe din:

- (1) o știință aplicată (fizică, inginerie, biologie, economie, științe sociale, etc.);
- (2) matematică (inclusiv metode numerice);
- (3) știința calculatoarelor (soft și hard).

Specialistul în “Computational Science” îndeplinește următoarele sarcini:

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

■

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

- Rezultatul oricărei operații în virgulă flotantă este *imprecis*, cu o anumită toleranță.
- Eroarea propagată într-un șir de astfel de operații 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*' octeți contigui. Intregii sunt stocați în reprezentarea *complementului lui 2*, nume:

2.2.2 Reprezentarea în calculator binar:

$$x = (-1)^s \times b_0 \cdot b_1 b_2 \dots b_{p-1} \times 2^E$$

unde:

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

b_N = cifre binare: 0 sau 1

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

$b_1 \dots b_{p-1}$ se zice *fracția*; conține $P - 1$ cifre binare. Notând semnificandul cu *sig* și fracția cu *f*, avem $sig = b_0 \cdot f$, 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 $b_0 = 1$, și acesta nu se mai stochează. Punctul binar se presupune la dreapta lui b_0 . În consecință, precizia P este asigurată cu un număr de $P - 1$ biți, și avem:

$$1 \leq \text{semnificand} < 2.$$

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

Observație

Concret, reprezentarea normalizată este

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

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

Exemplu

Simplă precizie (Format simplu): $P = 24$ biți – asigurată cu 23 biți.

Dublă precizie (Format dublu): $P = 53$ biți – asigurată cu 52 biți ■

Observație

În modelul științific 2.2.1, numărul este reprezentat astfel:

$$x = (-1)^s \times b_0 b_1 b_2 \dots b_{p-1} \times 2^e ; \quad e = E + 1 \quad \blacksquare$$

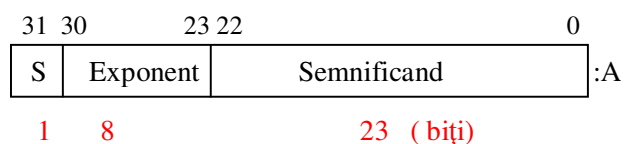
2.3 Structura formatelor

Tipul REAL(4) ocupă 4 octeți contigui și este stocat în formatul simplu (IEEE S_floating). Tipul REAL(8) ocupă 8 octeți contigui și este stocat în formatul dublu (IEEE T_floating). Figurile următoare prezintă structura formatelor: Cifrele indică numerotarea biților; ‘S’ desemnează bitul de semn. Simbolul “:A” reprezintă adresa octetului care conține bitul 0 (adresa de start a datei reprezentate). Formatul este numit “nativ” și convenția de stocare “Little endian”.

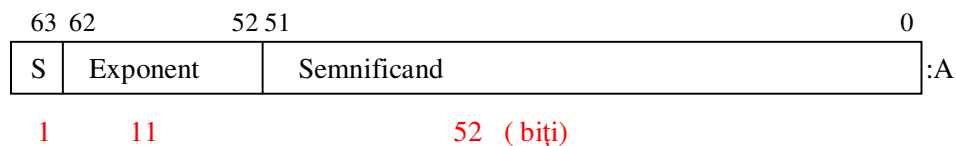
CVF utilizează convenția "Little endian". Convenția "Big endian" este utilizată, de exemplu, de calculatoarele IBM/370. V. pentru detalii, CVF – Programmer's Guide (2001).

■

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 biți	32	64	80
Bitul de semn, lungime în biți	1	1	1
Semnificand, lungime câmp în biți	23	52	64
Precizie P , în biți	24	53	64
Exponent, lungime câmp în biți	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 - 21
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 și acesta nu se mai stochează. În fapt, în câmpul semnificandului se stochează *fracția*.
- Formatul dublu-extins nu trebuie normalizat, el utilizează 64 biți pentru precizie. Acest format este utilizat intern, de către co-procesorul matematic. Bitul b_0 este stocat explicit (în locația 63 a câmpului semnificand).
- La fiecare exponent se adaugă o "deplasare" astfel încât să apară numai exponenți stocați *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 $e = E + 1$:

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

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 – poziția 0 din tabel. “*f*” reprezintă fracția.

Formatul IEEE – Valori în virgulă flotantă

Nr. crt.	Denumire	Cantitate reprezentată	Exponent E	Semnificand sig
0	Număr normalizat	$\pm 1.f \times 2^E$	$E_{\min} \leq E \leq E_{\max}$	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	$E = E_{\max} + 1$	$sig \neq 0$

2.3.3 Valori reprezentabile

a) Numere denormalizate:

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

Notă: Tipul REAL(16) (Intel Fortran 9)

Plaja de reprezentare (zecimal):

$$-1.190 \times 10^{+4962} \dots -3.362 \times 10^{-4962}; 0.0; \quad 3.362 \times 10^{-4962} \dots 1.190 \times 10^{+4962}$$

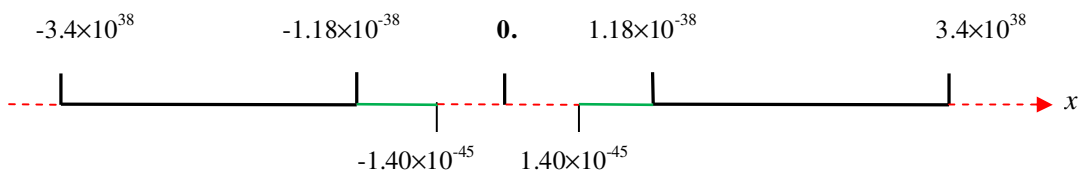
a) Numere denormalizate:

Sunt cuprinse între valorile de mai jos.

$$\begin{array}{l} \text{REAL(4):} \quad -2^{-149} \times 1.00\dots \quad -2^{-126} \times 1.00\dots \quad (-1.18 \times 10^{-38} \quad -1.40 \times 10^{-45}) \\ \quad \quad \quad 2^{-149} \times 1.00\dots \quad 2^{-126} \times 1.00\dots \quad (1.40 \times 10^{-45} \quad 1.18 \times 10^{-38}) \\ \text{REAL(8):} \quad -2^{-1022} \times 1.00\dots \quad -2^{-1074} \times 1.00\dots \quad (-2.23 \times 10^{-308} \quad 4.94 \times 10^{-324}) \\ \quad \quad \quad 2^{-1074} \times 1.00\dots \quad 2^{-1022} \times 1.00\dots \quad (4.94 \times 10^{-324} \quad 2.23 \times 10^{-308}) \end{array}$$

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 diferența dintre un număr real (care nu se reprezintă exact FP) și numărul FP cel mai apropiat care îl reprezintă.

Definiția 1

$Eroarea = Valoarea\ exactă - Valoarea\ aproximativă$ ■

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

$$Err(fl(x)) = x - fl(x)$$

Definiția 2

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

Eroarea relativă la rotunjire este:

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

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

3.1 ULP (Unit in the Last Place)

Definiția 3

$ULP(x)$ este numărul cu 1 în bitul ultim (b_{p-1}), 0 (zero) în restul biților, și cu același exponent ca și x ■

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

$$ULP(x) = 0.\underset{1}{0}0 \dots \underset{p-1}{0}1 \times 2^E = 2^{-p+1} \times 2^E,$$

unde E este exponentul reprezentării în calculator.

În modelul 3.1 (notația științifică), $x = 0.b_0b_1 \dots b_{p-1} \times 2^e$, și avem

$$ULP(x) = 0.\underset{1}{0}0 \dots \underset{p}{0}1 \times 2^e = 2^{-p} \times 2^e,$$

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

Propoziția 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:

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

Sau, în modelul 3.1, avem

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

Observație

ULP este util în algoritmi numerici care includ o iterație și un test de oprire a iterației de tipul

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

Valoarea toleranței 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:

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

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

Analitic, exponentul reprezentării este $E = 0$, și din Propoziția 1 rezultă

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

Expresia este valabilă și pentru formatul dublu, unde reprezentarea e analogă 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: $1.0 = (0.1)_2 \times 2^1$, și avem $ULP(1.0) = 2^{-P} \times 2^1 = 2^{-P+1}$ ■

Propoziția 2

Eroarea de rotunjire satisface:

$$|Err(fl(x))| \leq \frac{1}{2} ULP(x) \quad \blacksquare$$

Pentru demonstrație, vezi Cap. 2, 3.1.

Propoziția 3

$$\frac{1}{2} \cdot 2^{-P} \leq |Rel(fl(x))| \leq 2^{-P} \quad \blacksquare$$

Cu $\frac{1}{2} \cdot 2^e \leq |x| \leq 2^e$ și $|Err(fl(x))| \leq \frac{1}{2} \cdot 2^{-P} \cdot 2^e$, avem:

$$\frac{\frac{1}{2} \cdot 2^{-P} \cdot 2^e}{2^e} \leq |Rel(fl(x))| \leq \frac{\frac{1}{2} \cdot 2^{-P} \cdot 2^e}{\frac{1}{2} \cdot 2^e},$$

de unde rezultă afirmația din Propoziția 3. V. și vezi Cap. 2, 3.2.

3.2 ‘ε- mașină’ și EPS

Definiția 4

Limita superioară a erorii relative se numește ε -mașină = 2^{-P} ■

Astfel, avem:

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

Definiția 5

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

$$x = 1.0 + EPS \Rightarrow x > 1.0 \blacksquare$$

Propoziția 3 $EPS = 2 \times \varepsilon$ -mașină ■

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

$$EPS = ULP(1.0) = 2^{-P+1} = 2 \times \varepsilon\text{-mașină}.$$

Exemplantu – 2:

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

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

Observații

- Funcția 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. (Exemplantu: integrarea sistemelor de ecuații diferențiale).

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 Condiționarea problemei. Stabilitatea metodei.

Condiționarea problemei este sensibilitatea soluției (exacte) la mici schimbări în datele problemei. Dacă soluția este sensibilă la mici schimbări în date, problema se zice *rău-condiționată*. In caz contrar, problema se zice *bine-condiționată*.

Stabilitatea metodei: In anumite condiții, soluția calculată iterativ crește nemărginit odată cu creșterea numărului pașilor procesului. In acest caz, metoda numerică se zice *instabilă*. V. exemplele următoare.

Concluzii:

Trebuie făcută distincție între *condiționarea problemei* și *stabilitatea metodei* de calcul.

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

5 Exemple – Probleme rău-condiționate și probleme intratabile.

5.1 Probleme rău-condiționate

5.1.1 Polinomul lui Wilkinson

Fie

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

cu rădăcinile $z = 1, 2, \dots, 20$.

Considerăm polinomul perturbat:

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

unde avem: $2^{-23} = 1.19209 \dots \cdot 10^{-7}$. 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 numește polinomul lui Wilkinson, după numele celui care l-a

Dacă perturbația $\varepsilon = 0$, matricea este superior triunghiulară, și valorile proprii sunt $\lambda(i) = i$.

Pentru $\varepsilon = 1E-10$, cu metoda QR *fără rafinare* (cu $EPS = 1E-6$), se obțin următoarele rezultate:

Fără balansare: Valorile proprii rezultă toate reale. Ele reprezintă perturbații 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 conțin 14 valori complexe, și 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.

Explicația rezultatelor foarte diferite se poate da analizând condiționarea rădăcinilor polinomului caracteristic al matricii \mathbf{A} . Acesta este

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

Notăm rădăcinile cu $\lambda_i(\varepsilon)$, iar $\lambda_i(0) = \lambda_i = i$. Numărul de condiție 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 condiție maxim și minim se ating pentru $i = 10, 11$ și $i = 1, 20$, 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 condiție foarte mari arată sensibilitatea mare a rădăcinilor la mici perturbații în coeficienți.

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

5.2 Sumarea

Sumarea este una din cele mai importante operații în calculul științific.

Sumarea intervine în aproape orice calculație. 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 (și 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 funcțiilor 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 proprietățile 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 Înălțimea suprafeței mării (SSH - Sea Surface Height)

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

Rețeaua cuprindea $120 \times 64 = 7680$ de puncte (120 longitudini și 64 de latitudini). Datele sunt numere de ordinul de mărime $10^{12} \dots 10^{15}$ și de semne diferite. Datele corespunzând observațiilor dintr-o zi s-au stocat într-un tablou `ssh(120, 64)`, de date cu precizia de 64 biți.

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

<u>Ordonarea sumei după</u>	<u>Valoarea</u>
Longitudine crescător	34.4147682189941410
Longitudine descrescător	32.302734375
Latitudine crescător	0.67326545715332031
Latitudine descrescător	0.734375
Valoarea corectă	0.35798583924770355

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

5.2.2 Polinomul lui Rump

Se consideră ”polinomul”

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

și 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 biți):	5.764607523034234880E+17

Valoarea corectă: -0.82739 60599 46821 36814 11650 95479 81629 ...

(obținută cu precizie de 128 biți \approx 38-39 cifre zecimale de precizie).

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

Explicație:

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

Rezultate obținute cu MPFUN – cu diferite nivele de precizie:

* Multiple Precision (21 decimal digits; 5 words):

$10^0 \times 1.172603940053178631858834904523310930,$

* Multiple Precision (28 decimal digits; 6 words):

$10^{-1} \times -8.273960599468213681411650954798162916293306,$

* Multiple Precision (36 decimal digits; 7 words):

$10^{-1} \times -8.273960599468213681411650954798162919990331124134,$

* Multiple Precision (43 decimal digits; 8 words):

$10^{-1} \times -8.273960599468213681411650954798162919990331157843838320,$

* Multiple Precision (50 decimal digits; 9 words):

$10^{-1} \times -8.27396059946821368141165095479816291999033115784384819917,$

* Multiple Precision (57 decimal digits; 10 words):

$10^{-1} \times -8.27396059946821368141165095479816291999033115784384819917,$

■

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

20

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

Rezultat:

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