Evaluarea algoritmilor: complexitate şi erori.

Prof.dr.ing. Gabriela Ciuprina

Universitatea "Politehnica" Bucureşti, Facultatea de Inginerie Electrică,
Departamentul de Electrotehnică

Suport didactic pentru disciplina *Algoritmi Numerici*, 2017-2018

Cuprins

T = O(?)

Complexitatea unui algoritm din punct de vedere al timpului de calcul

relaţia dintre timpul de calcul exprimat în număr de operaţii elementare şi dimensiunea problemei

Operaţie elementară

operaţia care durează cel mai mult.

Algoritmi polinomiali - de ordinul 1 (liniar)

p = 0;
pentru $i = 1, n$
$p = p + a_i b_i$ $T = O(2n) \approx O(n)$

•

Algoritmi polinomiali - de ordinul 2 (pătratic)

- $\underline{\texttt{pentru}}\ i = 1, n$
  - $b_i = 0$
  - $\underline{\texttt{pentru}}\ j = 1, n$
    - $b_i = b_i + a_{ij} x_j$ $\qquad T = O(2n^2) \approx O(n^2)$
  - •
- •

Algoritmi polinomiali - de ordinul 3 (cubic)

$\underline{\texttt{pentru}}\ i = 1, n$

$\quad \underline{\texttt{pentru}}\ j = 1, n$

$\quad\quad c_{ij} = 0$

$\quad\quad \underline{\texttt{pentru}}\ k = 1, n$

$\quad\quad\quad c_{ij} = c_{ij} + a_{ik} b_{kj}$

$\quad\quad \bullet$

$\quad \bullet$

$\bullet$

$$T = O(2n^3) \approx O(n^3)$$

Algoritmi polinomiali - de ordinul k

$$T = O(n^k) \iff (\exists)\ C > 0 \text{ şi } n_0 \text{ a.i. } T \leq Cn^k\ (\forall)\ n \geq n_0$$

Algoritm de ordin 0: $T = O(1)$ - nu depinde de dimensiunea problemei.

$$O(1) < O(\log n) < O(n) < O(n \log n) < O(n^2) < \ldots O(e^n) < O(n!)$$

Ex1: Evaluarea unui polinom - varianta I

$$P(x) = a_0 + a_1 x + \ldots + a_n x^n. \quad (1)$$

$$P(x) = a_0 + \sum_{i=1}^{n} a_i x^i \quad (2)$$

; Varianta 1
$P = a_0$;
pentru $i = 1, n$
    $t = a_i$
    pentru $j = 1, i$
        $t = t * x$
    •
    P = P + t;
•

Nr. de operaţii elementare:
$\sum_{i=1}^{n}(i+1) \approx n^2/2$

$T_1 = \mathrm{O}(n^2/2) \approx \mathrm{O}(n^2)$

Ex1: Evaluarea unui polinom - varianta II

$$P(x) = a_0 + a_1 t_1 + a_2 t_2 + \ldots + a_n t_n = a_0 + \sum_{i=1}^{n} a_i t_i, \qquad (3)$$

unde

$$t_1 = x, \quad t_i = t_{i-1} x, \quad i = 2, \ldots, n. \qquad (4)$$

; Varianta 2
$P = a_0$;
$t = 1$;
pentru $i = 1, n$ $\qquad T_2 = \mathrm{O}(3n) \approx \mathrm{O}(n)$
$\quad t = t * x$
$\quad P = P + a_i * t$
•

Ex1: Evaluarea unui polinom - varianta III

$$P(x) = a_0 + x(a_1 + x(a_2 + \ldots x(a_{n-1} + a_n x)\ldots)). \qquad (5)$$

; Varianta 3
$P = a_n$;
pentru $i = n-1, 0, -1$ $\qquad T_3 = \mathrm{O}(2n) \approx \mathrm{O}(n)$
$\qquad P = a_i + P * x$

•

Ex2: Reducerea timpului de calcul

; Varianta A
întreg i, j, n
real a, b
...
tablou real $c[n][n]$
pentru $i = 1, n$
    pentru $j = 1, n$
        $c_{ij} = f(i * a) + f(j * b)$; f definită în altă parte
    •
•

Operaţie elementară: evaluarea funcţiei $f \Rightarrow T_A = \mathrm{O}(2n^2)$.

Ex2: Reducerea timpului de calcul

; Varianta B
întreg i, j, n
real a, b, p
...
tablou real $c[n][n]$
pentru $i = 1, n$
    $p = f(i * a)$
    pentru $j = 1, n$
        $c_{ij} = p + f(j * b)$
    •
•

$$T_B = \mathrm{O}(n(n+1)) = \mathrm{O}(n^2 + n) \approx \mathrm{O}(n^2).$$

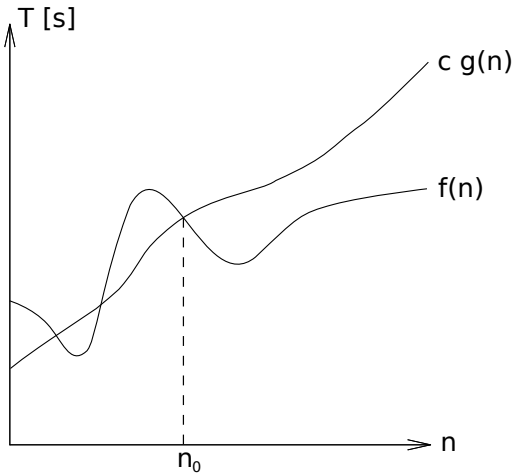
Ex2: Reducerea timpului de calcul

; Varianta C
întreg i, j, n
real a, b
...
tablou real $c[n][n]$
tablou real $p[n], q[n]$
pentru $i = 1, n$
    $p_i = f(i * a)$
    $q_i = f(i * b)$

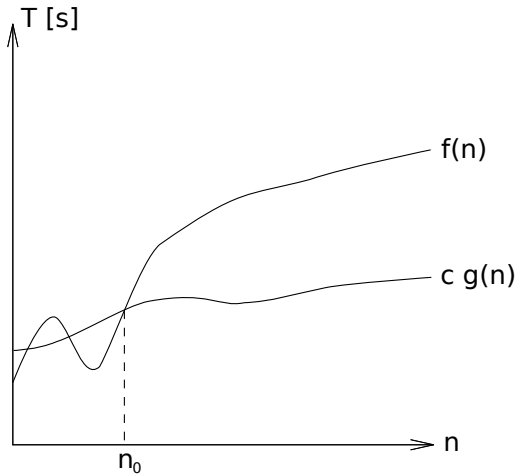
$$T_C = O(2n)$$

•
pentru $i = 1, n$
    pentru $j = 1, n$
        $c_{ij} = p_i + q_j$
    •
•

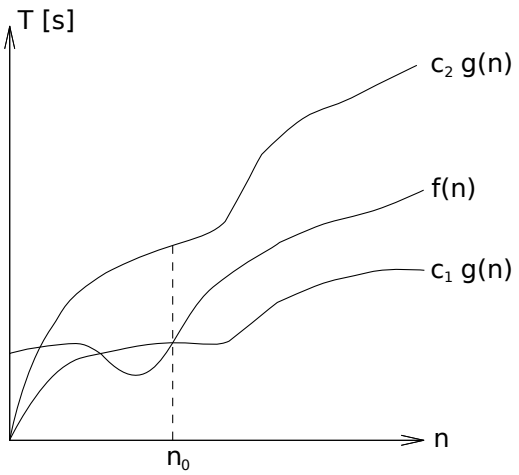
Notaţii asimptotice



$$T = f(n) = \mathrm{O}(g(n))$$



$$T = f(n) = \Omega(g(n))$$



$$T = f(n) = \Theta(g(n))$$

T = $\mathrm{O}(g(n)) \iff \exists\ C > 0$ şi n_0 a.i. $T \leq Cg(n)$ $(\forall)$ $n \geq n_0$
T = $\Omega(g(n)) \iff \exists\ C > 0$ şi n_0 a.i. $Cg(n) \leq T$ $(\forall)$ $n \geq n_0$

T = $\Theta(g(n)) \iff \exists\ C_1$ şi C_2 şi n_0 a.i. $C_1 g(n) \leq T \leq C_2 g(n)$ $(\forall)$ $n \geq n_0$.

Exemplu - căutarea unei valori

Fie $x_1 < x_2 < \ldots < x_n$ şi *xcrt*.
Se cere subintervalul $[x_k, x_{k+1}]$ se află valoarea *xcrt*.

Parcurgerea subintervalelor:
$T = \mathrm{O}(n)$
$T = \Omega(1)$

```
funcţie caută_v0(n, x, xcrt)
; caută valoarea xcrt în şirul ordonat x cu n valori
întreg n
tablou real x[n] ; şirul este presupus ordonat
real xcrt
...
dacă (xcrt < x_1)
        rez = 1
altfel dacă (xcrt > x_n)
        rez = n - 1
altfel
        flag = 0
        k = 1
        cât timp ((k <= n - 1) şi (flag = 0))
                dacă xcrt ≤ x_{k+1}
                        rez = k
                        flag = 1
                altfel
                        k = k + 1
                •
        •
•
întoarce rez
```

Exemplu - căutarea unei valori (căutarea binară)

Fie $x_1 < x_2 < \ldots < x_n$ şi $xcrt$.
Se cere subintervalul $[x_k, x_{k+1}]$ se află valoarea $xcrt$.

Căutarea binară:
$T = \mathrm{O}(log(n))$

$n/2 + n/4 + n/8 + \cdots n/2^k = n - 1$
$\Rightarrow 2^k = n$
$\Rightarrow k = \log_2(n)$

```
funcţie caută(n, x, xcrt)
...
dacă (xcrt < x1)
        rez = 1
altfel dacă (xcrt > xn)
        rez = n − 1
altfel
        k1 = 1
        k2 = n
        cât timp k2 − k1 ≠ 1
                km = [(k1 + k2)/2]      ; [· · ·] este partea întreagă
                dacă xcrt < xkm
                        k2 = km
                altfel
                        k1 = km
                •
        •
•
rez = k1
întoarce rez
```

Exemplu - căutarea binară, implementare recursivă

```
funcţie caută_v2(n, x, xcrt)
...
dacă (xcrt < x_1)
        rez = 1
altfel dacă (xcrt > x_n)
        rez = n − 1
altfel
rez = binary_search(x, xcrt, 1, n);
●
întoarce rez
;
funcţie binary_search(x, xcrt, k1, k2)
; presupuneri:
; x - ordonate, k1 < k2
; xcrt se afla in interiorul vectorului x
...
km = [(k1 + k2)/2]
dacă k2 = k1 + 1
        rez = km
altfel dacă x_km > xcrt
        rez = binary_search(x, xcrt, k1, km)
altfel
        rez = binary_search(x, xcrt, km, k2)
●
întoarce rez
```

M = O(?)

Complexitatea unui algoritm din punct de vedere al necesarului de memorie

relaţia dintre necesarul de memorie exprimat în număr de locaţii elementare şi dimensiunea problemei

Locaţia elementară de memorie

locaţia corespunzătoare unui număr real.

$M = O(n^k) \iff (\exists)\ C$ a.i. $M \leq Cn^k$.

Ex2: Reducerea timpului de calcul

; Varianta A
întreg i, j, n
real a, b
...
tablou real $c[n][n]$
pentru $i = 1, n$
    pentru $j = 1, n$
        $c_{ij} = f(i * a) + f(j * b)$; f definită în altă parte
    •
•

$T_A = O(2n^2)$
$M_A = O(n^2 + 2) \approx O(n^2)$

Ex2: Reducerea timpului de calcul

; Varianta B
întreg i, j, n
real a, b, p
...
tablou real $c[n][n]$
pentru $i = 1, n$
    $p = f(i * a)$
    pentru $j = 1, n$
        $c_{ij} = p + f(j * b)$
    •
•

$T_B = \mathrm{O}(n(n+1)) = \mathrm{O}(n^2 + n) \approx \mathrm{O}(n^2)$
$M_B = \mathrm{O}(n^2 + 3) \approx \mathrm{O}(n^2)$

Ex2: Reducerea timpului de calcul

; Varianta C
întreg i, j, n
real a, b
...
tablou real $c[n][n]$
tablou real $p[n], q[n]$
pentru $i = 1, n$
$p_i = f(i * a)$ $\quad T_C = \mathrm{O}(2n)$
$q_i = f(i * b)$ $\quad M_C = \mathrm{O}(n^2 + 2n + 2) \approx \mathrm{O}(n^2)$
•
pentru $i = 1, n$
pentru $j = 1, n$
$c_{ij} = p_i + q_j$
•
•

Concluzii (complexitate)

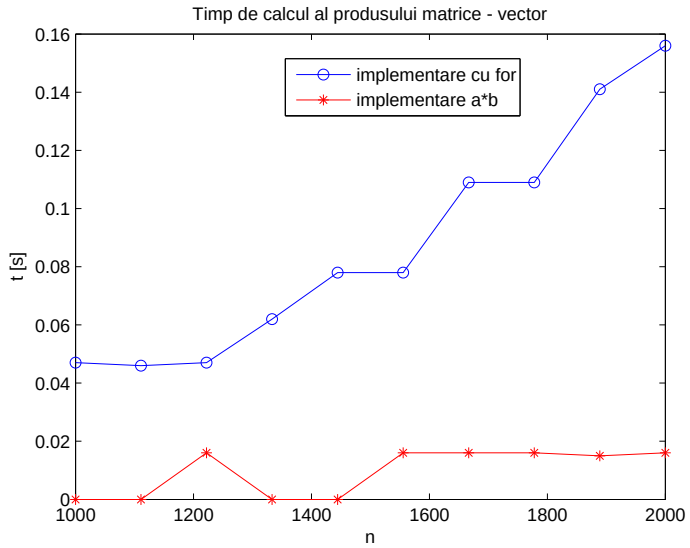
- Resursele de calcul necesare se exprimă cu ajutorul noţiunii de *complexitate*, care se poate referi la două aspecte: timpul de calcul (T) şi necesarul de memorie (M).
- Pentru analiza complexităţii trebuie stabilite una sau mai multe variabile de care depinde dimensiunea problemei.
- Complexitatea T - se exprimă în număr de operaţii elementare;
- Complexitatea M - se exprimă în număr de locaţii elementare de memorie.

Concluzii (complexitate)

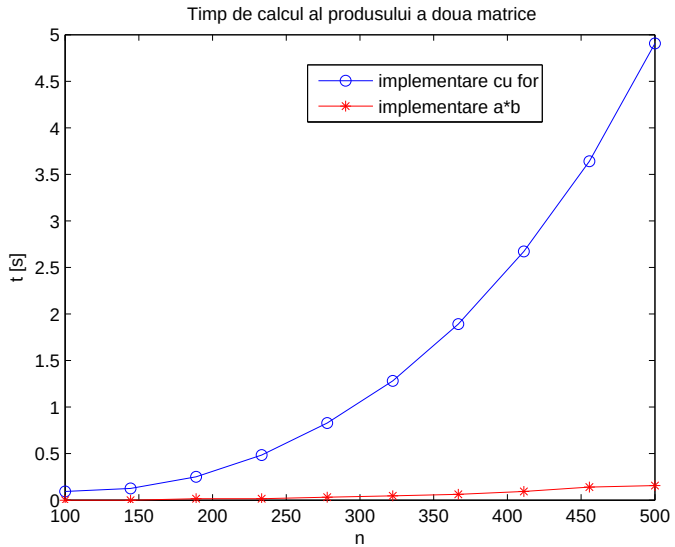
- Cea mai simpla analiză asimptotică a complexităţii este dată de notaţia "big O".
- După elaborarea unui algoritm, este necesară analiza complexităţii lui şi investigarea unor modalităţi de reducere a resurselor de calcul necesare.
- Elaborarea unui algoritm înseamnă întotdeauna stabilirea unui compromis între timp de calcul şi necesar de memorie.
- Nu neglijaţi facilităţile specifice mediului de programare în care lucraţi pentru a scrie coduri eficiente (vedeţi exemplul următor).

Implementări eficiente în Matlab

Comparaţie între două implementări posibile în Matlab.



Timpul de calcul al produsului dintre o matrice pătrată şi un vector în funcţie de dimensiunea problemei.



Timpul de calcul al produsului dintre două matrice pătrate în funcţie de dimensiunea problemei.

Tipuri de erori

În funcţie de tipul cauzelor care le generează:

1. Erori de rotunjire - datorate reprezentării finite a numerelor reale;
2. Erori de trunchiere - datorate reprezentării finite a algoritmului;
3. Erori inerente - datorate reprezentării imprecise a datelor de intrare.

Mărimi utile - eroarea absolută şi marginea ei

Fie:
$\mathbf{x} \in \mathbb{R}^n$ - valoarea exactă a unei mărimi;
$\bar{\mathbf{x}}$ - valoarea aproximativă.
Eroarea absolută $\mathbf{e_x} \in \mathbb{R}^n$:

$$\mathbf{e_x} = \bar{\mathbf{x}} - \mathbf{x}. \tag{6}$$

Marginea erorii absolute $a_x \in \mathbb{R}$:

$$\|\mathbf{e_x}\| \leq a_x. \tag{7}$$

Dacă $n = 1$ rezultă

$$\bar{x} - a_x \leq x \leq \bar{x} + a_x. \tag{8}$$

Echivalentă cu: $x \in [\bar{x} - a_x, \bar{x} + a_x]$.
Scrisă pe scurt ca:

$$"x = \bar{x} \pm a_x". \tag{9}$$

Mărimi utile - eroarea relativă şi marginea ei

Eroarea relativă $\varepsilon_x \in \mathbb{R}^n$:

$$\varepsilon_{\mathbf{x}} = \frac{\mathbf{e_x}}{\|\mathbf{x}\|}. \tag{10}$$

Marginea erorii relative $r_x \in \mathbb{R}$

$$\|\varepsilon_{\mathbf{x}}\| \leq r_x. \tag{11}$$

Cel mai adesea, r_x se exprimă în procente.
Scriere pe scurt:

$$''\mathbf{x} = \bar{\mathbf{x}} \pm r_x\%''. \tag{12}$$

Exemplu: π

$x = 3.1415\ldots$
$\bar{x} = 3.14$
$e_x = -0.0015\ldots$
$a_x = 0.0016$
$\varepsilon_x = -0.0015\ldots/3.1415\ldots$
$r_x = 0.0016/3 \leq 0.0006 = 0.06\%.$

$$\pi = 3.14 \pm 0.0016 \quad \text{sau} \quad \pi = 3.14 \pm 0.06\%.$$

Concluzii

Relaţia "$\mathbf{x} = \bar{\mathbf{x}} \pm a_x$"

unde $\mathbf{x}, \bar{\mathbf{x}} \in \mathbb{R}^n$ şi $a_x \in \mathbb{R}$ se interpretează astfel:

$$(\exists)\mathbf{e_x} \in \mathbb{R}^n, \|\mathbf{e_x}\| \leq a_x, \quad \text{astfel încât} \quad \bar{\mathbf{x}} = \mathbf{x} + \mathbf{e_x}, \tag{13}$$

Relaţia "$\mathbf{x} = \bar{\mathbf{x}} \pm r_x\%$"

unde $\mathbf{x}, \bar{\mathbf{x}} \in \mathbb{R}^n$, $r_x\% = 100 r_x$ şi $r_x \in \mathbb{R}$ se interpretează astfel:

$$(\exists)\varepsilon_{\mathbf{x}} \in \mathbb{R}^n, \|\varepsilon_{\mathbf{x}}\| \leq r_x, \quad \text{astfel încât} \quad \bar{\mathbf{x}} = \mathbf{x} + \|\mathbf{x}\|\varepsilon_{\mathbf{x}}. \tag{14}$$

În cazul unei mărimi scalare ($n = 1$), relaţia (14) se scrie

$$\bar{x} = x(1 \pm \varepsilon_x), \tag{15}$$

semnul plus corespunzând unei valori x pozitive, iar semnul minus uneia negative.

Cifre semnificative

Reprezentarea unui număr real în baza 10:

$$\bar{x} = f \cdot 10^n. \tag{16}$$

unde $0.1 \leq |f| < 1$.
Cifrele părţii fracţionare se numesc cifre semnificative.
Exemple:
$3.14 = 0.314 \cdot 10^1$
$-0.007856 = -0.7856 \cdot 10^{-2}$.

Rotunjirea afectează reprezentarea numerelor reale

$$\bar{x} = \overbrace{0.\underbrace{***\cdots*}_{k \text{ cifre}}}^{f} \cdot 10^n, \tag{17}$$

$$x = 0.\underbrace{***\cdots*}_{k \text{ cifre}}\#\#\#\cdots \cdot 10^n, \tag{18}$$

$$e_x = \bar{x} - x = -0.\underbrace{000\cdots0}_{k \text{ cifre}}\#\#\#\cdots \cdot 10^n = -0.\#\#\#\cdots \cdot 10^{n-k}, \tag{19}$$

Rotunjirea afectează reprezentarea numerelor reale

$$\varepsilon_x = \frac{e_x}{x} = \frac{-0.\#\#\#\cdots 10^{n-k}}{0.\underbrace{***\cdots *}_{k \text{ cifre}}\#\#\#\cdots 10^{n}} = -\frac{0.\#\#\#\cdots}{0.***\cdots}10^{-k} \tag{20}$$

$$|\varepsilon_x| \leq \frac{1}{0.1}10^{-k} = 10^{-k+1}. \tag{21}$$

Marginea erorii relative de rotunjire a unui sistem de calcul depinde doar de numărul de cifre semnificative ce pot fi memorate. Pentru un sistem de calcul ce lucrează cu k cifre semnificative, marginea erorii relative de rotunjire este 10^{-k+1}.

Rotunjirea afectează calculele

Adunarea a două numere reale
Intuitiv: pp. $k = 3$, $x_1 + x_2 = ?$
$x_1 = 3.73 = 0.373 \cdot 10^1$
$x_2 = 0.006 = 6 \cdot 10^{-3}$
$x_2 = 6 \cdot 10^{-4} \cdot 10^1 = 0.0006 \cdot 10^1 = 0.000 \cdot 10^1$
Rezultat: $x_1 + x_2 = x_1$.

Zeroul maşinii

Zeroul (acurateţea, precizia,"epsilon-ul") maşinii = cel mai mic `eps` pentru care `1 + eps > 1`.

- $(\forall) a <$ `eps`, $1 + a = 1$ (în calculator)
- în mod uzual `eps` $= 2.22 \cdot 10^{-16}$.
- Matlab: `eps`
- Scilab `%eps`.
- Zeroul maşinii nu trebuie confundat cu cel mai mic număr reprezentabil în calculator şi care, în mod uzual are valoarea $2.23 \cdot 10^{-308}$.

Consecinţă: adunarea numerelor reale în calculator nu este asociativă.

Determinarea eps într-un mediu de programare

```
funcţie zeroul_maşinii ()
real eps
eps = 1
cât timp (1 + eps > 1)
        eps = eps/2
●
eps = eps*2
întoarce eps
```

Notaţii pentru un număr real

- *Notaţie ştiinţifică*: $x = a \cdot 10^b$ unde $0.1 \leq |a| < 1$
- *Notaţia ştiinţifică normalizată*: $x = a \cdot 10^b$ unde $1 \leq |a| < 10$ (pune în evidenţă ordinul de mărime)
- *Notaţia inginerească* în care b se alege numai dintre multiplii lui 3, şi, deci a poate avea valori între $1 \leq a < 1000$. (se citeşte cu prefixe: mega, kilo, mili, micro etc.)

Reprezentarea numerelor reale în calculator este un fel de echivalent hardware al notaţiei ştiinţifice.

Necesitatea unui standard

Calculatoarele folosesc un număr finit de biţi pentru a reprezenta un număr real. Din acest motiv, în calculator poate fi reprezentată numai o mulţime finită de numere reale.

- numerele reale reprezentate nu pot fi oricât de mari sau oricât de mici (probleme numite *overflow*,*underflow*) ;
- există spaţii între numerele reale care se pot reprezenta - standardul din 1985 (IEEE şi ANSI) care reglementează reprezentarea în virgulă mobilă.

Institute of Electrical and Electronics Engineers (IEEE) www.ieee.org

American National Standards Institute (ANSI) www.ansi.org

Toate calculatoarele construite după 1985 folosesc acest standard IEEE pentru reprezentarea în virgulă mobilă: există un model independent de calculator pentru modul în care se fac calculele aritmetice cu numere reale.

Reprezentarea unui număr (standardul IEEE)

$$\bar{x} = \pm(1+f)\cdot 2^e, \tag{22}$$

unde f se numeşte *mantisă* şi e *exponent*.
$0 \leq f < 1$ şi $-1022 \leq e \leq 1023$ se reprezintă în calculator ca numere binare (în baza 2).
Exemplu: numărul 0.1 nu poate fi reprezentat exact în acest standard deoarece reprezentarea sa în binar necesită un număr infinit de biţi:

$$\frac{1}{10} = \frac{1}{2^4} + \frac{1}{2^5} + \frac{0}{2^6} + \frac{0}{2^7} + \frac{1}{2^8} + \frac{1}{2^9} + \frac{0}{2^{10}} + \frac{0}{2^{11}} + \frac{1}{2^{12}} + \cdots$$

după primul termen secvenţa $1, 0, 0, 1$ repetându-se la infinit. În consecinţă, reprezentarea numărului 0.1 este un pic mai mare decât valoarea exactă.
Nu numai numerele iraţionale sunt afectate de erori de rotunjire.

Reprezentarea unui număr (standardul IEEE)

- Reprezentarea intervalului [1,2]:
  $\{1, 1+2^{-52}, 1+2\cdot 2^{-52}, 1+3\cdot 2^{-52}, \ldots, 2\}$.
  Spaţiile dintre numerele vecine = zeroul maşinii.
- Reprezentarea intervalului $[2^j, 2^{j+1}]$: mulţimea de mai sus înmulţită cu 2^j. Spaţiile dintre numerele vecine sunt scalate în funcţie de dimensiunea numerelor.

Zeroul maşinii reflectă rezoluţia mulţimii reale discrete $\mathcal{R}$ ce poate fi reprezentată în calculator. Proprietate:

$$(\forall) x \in \mathbb{R}, (\exists)\bar{x} \in \mathcal{R} \quad \text{astfel încât} \quad |x-\bar{x}| \leq \mathtt{eps}|x|. \qquad (23)$$

Eroarea relativă dintre numărul real x şi reprezentarea lui discretă $\bar{x}$ este mărginită superior de zeroul maşinii.
$(\exists)\varepsilon$, unde $|\varepsilon| \leq \mathtt{eps}$ a.i:

$$\bar{x} = x(1+\varepsilon). \qquad (24)$$

Exemplu

$$f(x) = f(x_0) + \frac{x - x_0}{1!} f'(x_0) + \frac{(x - x_0)^2}{2!} f''(x_0) + \cdots \tag{25}$$

sinus, $x_0 = 0$:

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots = \sum_{k=0}^{\infty} (-1)^k \frac{x^{2k+1}}{(2k+1)!}. \tag{26}$$

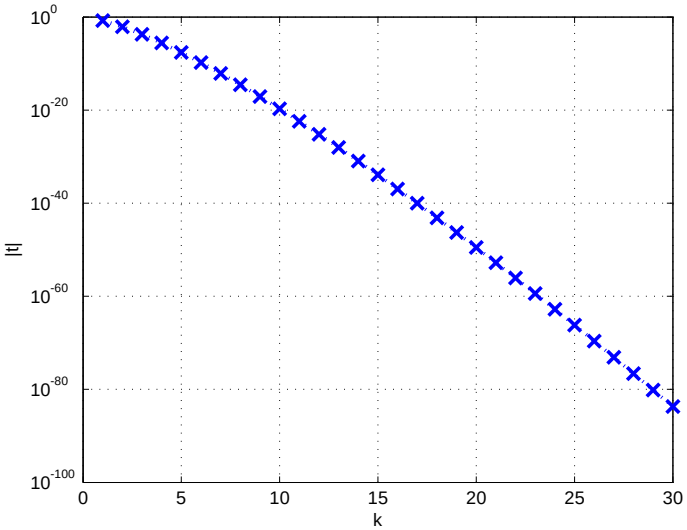
$$\bar{s} = \sum_{k=0}^{n} (-1)^k \frac{x^{2k+1}}{(2k+1)!}. \tag{27}$$

$$|e_s| = |\bar{s} - s| \leq \frac{x^{2n+1}}{(2n+1)!}. \tag{28}$$

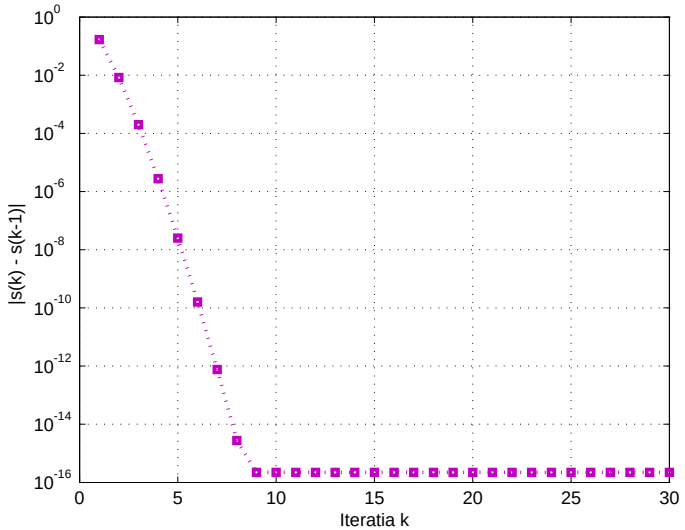
Algoritm cu controlul erorii de trunchiere

funcţie sinus(x, e)
; întoarce valoarea funcţiei sinus in punctul x
; prin trunchierea seriei Taylor dezvoltată in 0
real x ; punctul în care se va evalua funcţia sin
real e ; eroarea de trunchiere impusă
real t, s
întreg k
$t = x$
$s = t$
$k = 0$
cât timp ($|t| > e$)
$k = k + 1$
$t = (-1) * t * \frac{x^2}{(2k)(2k+1)}$
$s = s + t$
•
întoarce s

Rezultate numerice



Modulul termenului curent al dezvoltării în serie Taylor a funcţiei sinus.



Modulul diferenţei dintre sume parţiale consecutive la dezvoltarea în serie Taylor a funcţiei sinus.

Efectul perturbaţiilor datelor de intrare

$$y = f(x_1, x_2, \ldots, x_n). \tag{29}$$

$$\mathrm{d}y = \frac{\partial f}{\partial x_1}\mathrm{d}x_1 + \frac{\partial f}{\partial x_2}\mathrm{d}x_2 + \ldots \frac{\partial f}{\partial x_n}\mathrm{d}x_n. \tag{30}$$

$$\Delta y \approx \frac{\partial f}{\partial x_1}\Delta x_1 + \frac{\partial f}{\partial x_2}\Delta x_2 + \ldots \frac{\partial f}{\partial x_n}\Delta x_n. \tag{31}$$

$$\Delta x_k = \bar{x_k} - x_k = e_{x_k}, \tag{32}$$

Eroarea absolută a rezultatului şi marginea ei

$e_y = \bar{y} - y = \Delta y$:

$$e_y = \sum_{k=1}^{n} \frac{\partial f}{\partial x_k} e_{x_k}. \tag{33}$$

$$\left| \sum_{k=1}^{n} \frac{\partial f}{\partial x_k} e_{x_k} \right| \leq \sum_{k=1}^{n} \left| \frac{\partial f}{\partial x_k} e_{x_k} \right| = \sum_{k=1}^{n} \left| \frac{\partial f}{\partial x_k} \right| |e_{x_k}| \leq \sum_{k=1}^{n} \left| \frac{\partial f}{\partial x_k} \right| a_{x_k}, \tag{34}$$

unde $|e_{x_k}| \leq a_{x_k}$.

Marginea erorii absolute a rezultatului

$$a_y = \sum_{k=1}^{n} \left| \frac{\partial f}{\partial x_k} \right| a_{x_k}. \tag{35}$$

Eroarea relativă a rezultatului şi marginea ei

$$\varepsilon_y = e_y/|y|$$

$$\varepsilon_y = \frac{\sum_{k=1}^{n} \frac{\partial f}{\partial x_k} e_{x_k}}{|y|} = \sum_{k=1}^{n} \frac{\partial f}{\partial x_k} \frac{e_{x_k}}{|y|} = \sum_{k=1}^{n} \frac{\partial f}{\partial x_k} \frac{|x_k|}{|y|} \varepsilon_{x_k}. \qquad (36)$$

Marginea erorii relative a rezultatului

$$r_y = \sum_{k=1}^{n} \left| \frac{\partial(\ln f)}{\partial x_k} \right| |x_k| r_{x_k}. \qquad (37)$$

Cazuri particulare: +, -

Erori	**Adunare** $y = x_1 + x_2$	**Scădere** $y = x_1 - x_2$
Eroare absolută: $e_y =$	$e_{x_1} + e_{x_2}$	$e_{x_1} - e_{x_2}$
majorată de: $a_y =$	$a_{x_1} + a_{x_2}$	$a_{x_1} + a_{x_2}$
Eroare relativă: $\varepsilon_y =$	$\left\lvert \frac{x_1}{x_1+x_2} \right\rvert \varepsilon_{x_1} + \left\lvert \frac{x_2}{x_1+x_2} \right\rvert \varepsilon_{x_2}$	$\left\lvert \frac{x_1}{x_1-x_2} \right\rvert \varepsilon_{x_1} - \left\lvert \frac{x_2}{x_1-x_2} \right\rvert \varepsilon_{x_2}$
majorată de $r_y =$	$\left\lvert \frac{x_1}{x_1+x_2} \right\rvert r_{x_1} + \left\lvert \frac{x_2}{x_1+x_2} \right\rvert r_{x_2}$	$\left\lvert \frac{x_1}{x_1-x_2} \right\rvert r_{x_1} + \left\lvert \frac{x_2}{x_1-x_2} \right\rvert r_{x_2}$

Erorile rezultatului adunării şi scăderii a două numere reale în funcţie de erorile datelor de intrare.

NB! La adunare şi scădere marginile erorilor absolute se adună.

- Adunarea este o operaţie bine condiţionată.
- Scăderea este o operaţie prost condiţionată.

Exemplu

$x_1 = 1.23 \pm 1\%$, $x_2 = 1.22 \pm 1\%$

- Scădere:
  $r = |1.23/0.01 \cdot 1/100 + 1.22/0.01 \cdot 1/100 =$
  $1.23 + 1.22 = 2.45 = 245\%$
  $x_1 - x_2 = 0.01 \pm 245\%.$
- Adunare:
  $r = |1.23/2.45 \cdot 1/100 + 1.22/2.45 \cdot 1/100 \approx$
  $0.5 \cdot 1/100 + 0.5 \cdot 1/100 = 1/100 = 1\%.$
  $x_1 + x_2 = 2.45 \pm 1\%.$

Cazuri particulare: *, /

Erori	**Înmulţire** $y = x_1 x_2$	**Împărţire** $y = \frac{x_1}{x_2}$
Eroare absolută: $e_y =$	$x_2 e_{x_1} + x_1 e_{x_2}$	$\frac{1}{x_2} e_{x_1} - \frac{x_1}{x_2^2} e_{x_2}$
majorată de: $a_y =$	$\lvert x_2 \rvert a_{x_1} + \lvert x_1 \rvert a_{x_2}$	$\frac{1}{\lvert x_2 \rvert} a_{x_1} + \frac{\lvert x_1 \rvert}{x_2^2} a_{x_2}$
Eroare relativă: $\varepsilon_y =$	$\varepsilon_{x_1} + \varepsilon_{x_2}$	$\varepsilon_{x_1} - \varepsilon_{x_2}$
majorată de $r_y =$	$r_{x_1} + r_{x_2}$	$r_{x_1} + r_{x_2}$

Erorile rezultatului înmulţirii şi împărţirii a două numere reale în funcţie de erorile datelor de intrare.

NB! La înmulţire şi împărţire marginile erorilor relative se adună.

- Înmulţirea şi împărţirea sunt operaţii bine condiţionate.

Scăderea trebuie evitată

$ax^2 + bx + c = 0$
$x_{1,2} = (-b \pm \sqrt{b^2 - 4ac})/(2a)$
pp. $b > 0$ şi că $b^2 \gg 4ac$

dacă $b > 0$
    $x1 = (-b - \sqrt{b^2 - 4ac})/(2a)$
altfel
    $x1 = (-b + \sqrt{b^2 - 4ac})/(2a)$
•
$x2 = c/(a * x1)$

Extragerea radicalului

$y = \sqrt{x}$

$$e_y = \frac{df}{dx} e_x = \frac{1}{2\sqrt{x}} e_x, \tag{38}$$

$$\varepsilon_y = \frac{e_y}{y} = \frac{1}{2\sqrt{x}\sqrt{x}} e_x = \frac{e_x}{2x} = \frac{\varepsilon_x}{2}. \tag{39}$$

Dar rotunjirea nu poate fi ignorata!

Superpoziţia erorilor

eroarea relativă într-un calcul aproximativ
=
eroarea relativă produsă de calculul aproximativ cu numere exacte (eroarea de rotunjire)
+
eroarea relativă produsă de calculul exact cu numere aproximative (afectate deci de erori inerente).

$$\bar{y} = y_i(1+\mathtt{eps}) = y(1+\varepsilon_y)(1+\mathtt{eps}) \approx y(1+\varepsilon_y+\mathtt{eps}),$$

de unde $(\bar{y}-y)/y = \varepsilon_y + \mathtt{eps}$.

$$\varepsilon_{\sqrt{x}} = \frac{\varepsilon_x}{2} + \mathtt{eps}. \tag{40}$$

Eroarea relativă a oricărui rezultat numeric este cel puţin egală cu zeroul maşinii.

Condiţionare vs. stabilitate

Condiţionarea

se referă la comportarea problemei matematice la perturbaţii ale datelor.

Stabilitatea

se referă la comportarea algoritmului la perturbaţii ale datelor.

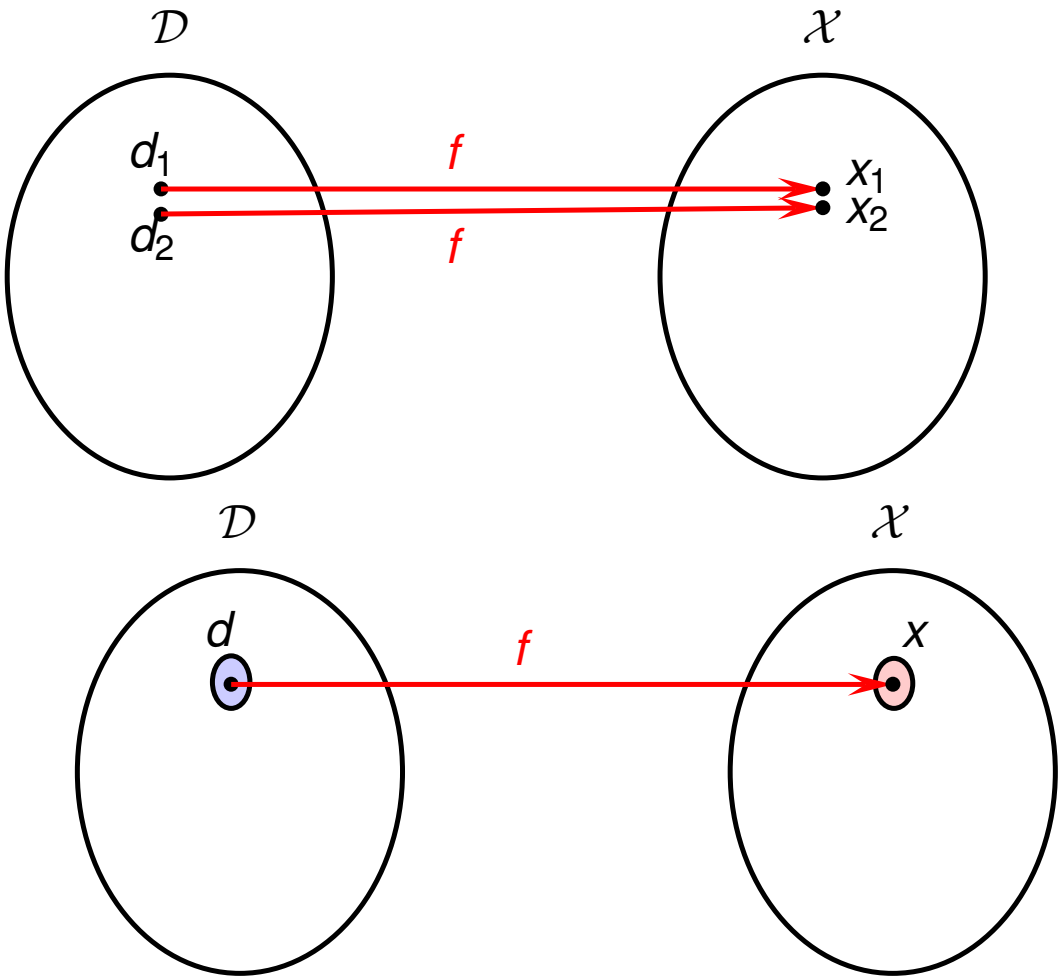
Condiţionare

Problemă matematică f formulată explicit:

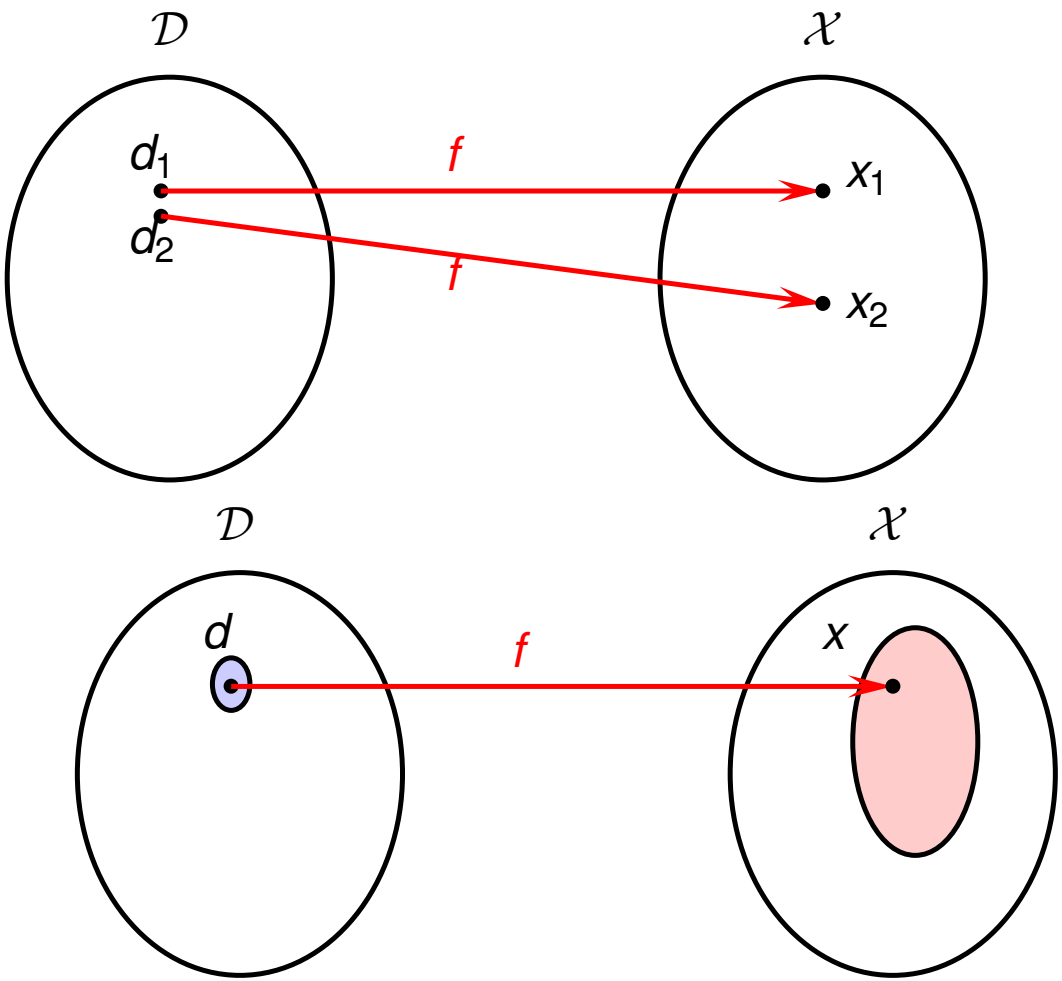
$$\text{Fie} \quad f : \mathcal{D} \to \mathcal{X} \quad \text{şi} \quad \mathbf{d} \in \mathcal{D}.$$
$$\text{Să se găsească} \quad \mathbf{x} \in \mathcal{X} \quad \text{astfel încât} \quad f(\mathbf{d}) = \mathbf{x}. \quad (41)$$

O problemă este bine condiţionată dacă perturbaţii mici ale datelor conduc la perturbaţii mici ale rezultatului.

Reprezentări intuitive - problemă bine condiţionată



Reprezentări intuitive - problemă prost condiţionată

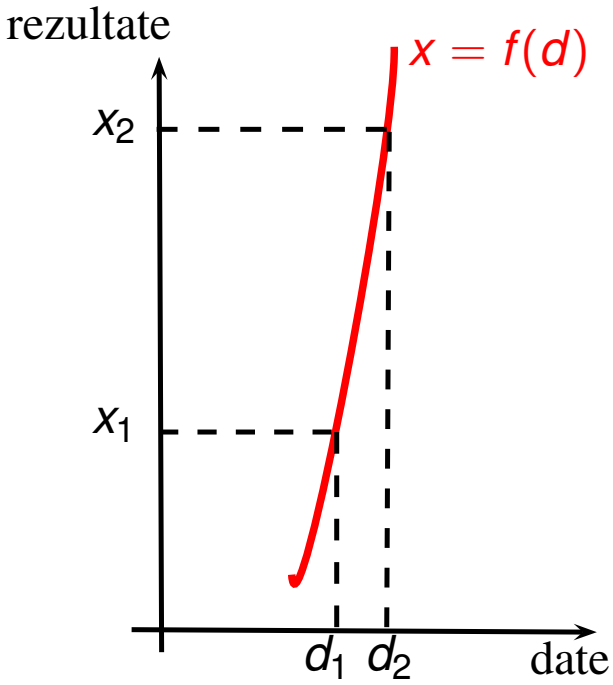


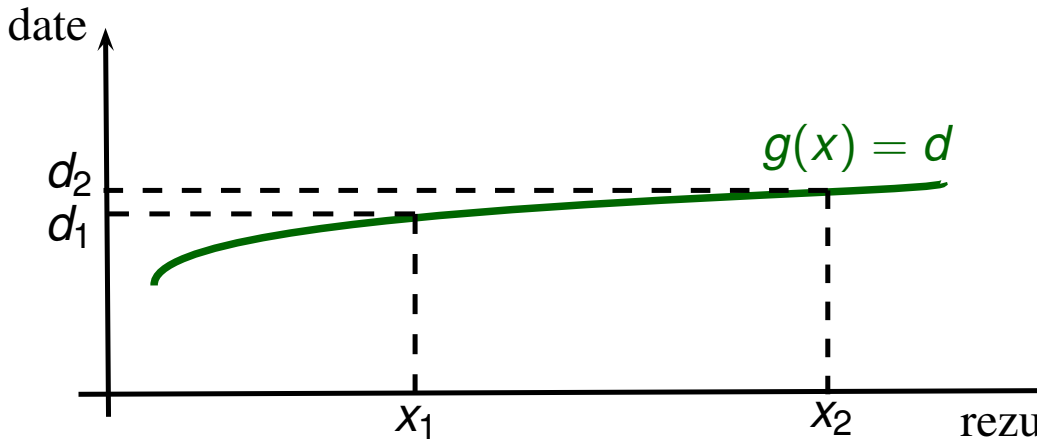
Condiţionare

Problemă matematică poate fi formulată şi implicit:

Fie $g : \mathcal{X} \to \mathcal{D}$ şi $\mathbf{d} \in \mathcal{D}$.
Să se găsească $\mathbf{x} \in \mathcal{X}$ astfel încât $g(\mathbf{x}) = \mathbf{d}$. (42)

Reprezentări intuitive - problemă prost condiţionată





Numărul de condiţionare absolut

$$\hat{\kappa} = \lim_{\delta \to 0} \sup_{\|\delta \mathbf{d}\| < \delta} \frac{\|\delta f\|}{\|\delta \mathbf{d}\|}, \tag{43}$$

sau, într-o scriere simplificată

$$\hat{\kappa} = \sup_{\|\delta \mathbf{d}\|} \frac{\|\delta f\|}{\|\delta \mathbf{d}\|}, \tag{44}$$

unde $\delta f = f(\mathbf{d} + \delta \mathbf{d}) - f(\mathbf{d})$ şi $\delta \mathbf{d}$ sunt mărimi infinitezimale. Dacă f este derivabilă, atunci perturbaţia δf se poate aproxima în condiţia $\|\delta \mathbf{d}\| \to 0$ ca

$$\delta f = \mathbf{J}(\mathbf{d}) \delta \mathbf{d}, \tag{45}$$

$$\hat{\kappa} = \|\mathbf{J}(\mathbf{d})\|. \tag{46}$$

Numărul de condiţionare relativ

$$\kappa = \lim_{\delta \to 0} \sup_{\|\delta \mathbf{d}\| < \delta} \frac{\|\delta f\| / \|f(\mathbf{d})\|}{\|\delta \mathbf{d}\| / \|\mathbf{d}\|}, \tag{47}$$

sau, scris mai simplu în ipoteza unor variaţii infinitezimale

$$\kappa = \sup_{\|\delta \mathbf{d}\|} \frac{\|\delta f\| / \|f(\mathbf{d})\|}{\|\delta \mathbf{d}\| / \|\ \mathbf{d}\|}. \tag{48}$$

Dacă f este derivabilă, atunci

$$k = \frac{\|\mathbf{J}(\mathbf{d})\|}{\|f(\mathbf{d})\| / \|\mathbf{d}\|}. \tag{49}$$

O problemă este bine condiţionată dacă valoarea lui κ este mică şi prost condiţionată dacă valorea lui κ este mare. Ce înseamnă mic sau mare, depinde de problemă.

Exemplu

Numărul de condiţionare al scăderii a două numere reale $f(\mathbf{d}) = d_1 - d_2$, unde $\mathbf{d} = [d_1, d_2]^T$.
$\mathbf{J} = [\partial f/\partial d_1, \partial f/\partial d_2]^T = [1, -1]^T$

$$\kappa = \frac{\|\mathbf{J}(\mathbf{d})\|}{\|f(\mathbf{d})\|/\|\mathbf{d}\|} = \frac{1}{|d_1 - d_2|/\max\{|d_1|, |d_2|\}}. \qquad (50)$$

Scăderea este prost condiţionată dacă $d_1 \approx d_2$.

Relaţia între numărul de condiţionare, eroare şi reziduu

Deoarece

$$\kappa = \sup_{\|\delta \mathbf{d}\|} \frac{\|\delta f\|/\|f(\mathbf{d})\|}{\|\delta \mathbf{d}\|/\|\ \mathbf{d}\|}. \tag{51}$$

rezultă

$$\frac{\|\delta f\|/\|f\|}{\|\delta \mathbf{d}\|/\|\mathbf{d}\|} \leq \kappa. \tag{52}$$

Perturbaţia rezultatului este o distanţa în spaţiul soluţiilor $\mathcal{X}$, deci reprezintă o eroare absolută $\mathbf{e_x} = \delta f$ sau relativă $\varepsilon_{\mathbf{x}} = \delta f/\|f\|$.
Perturbaţia datelor, numită şi reziduu este o distanţă în spaţiul $\mathcal{D}$. Reziduul poate fi absolut $\mathbf{e_d} = \delta \mathbf{d}$, unde $\delta \mathbf{d} = \bar{\mathbf{d}} - \mathbf{d}$, sau relativ $\varepsilon_{\mathbf{d}} = \delta \mathbf{d}/\|\mathbf{d}\|$.
Cu aceste notaţii:

Condiţionare - concluzie

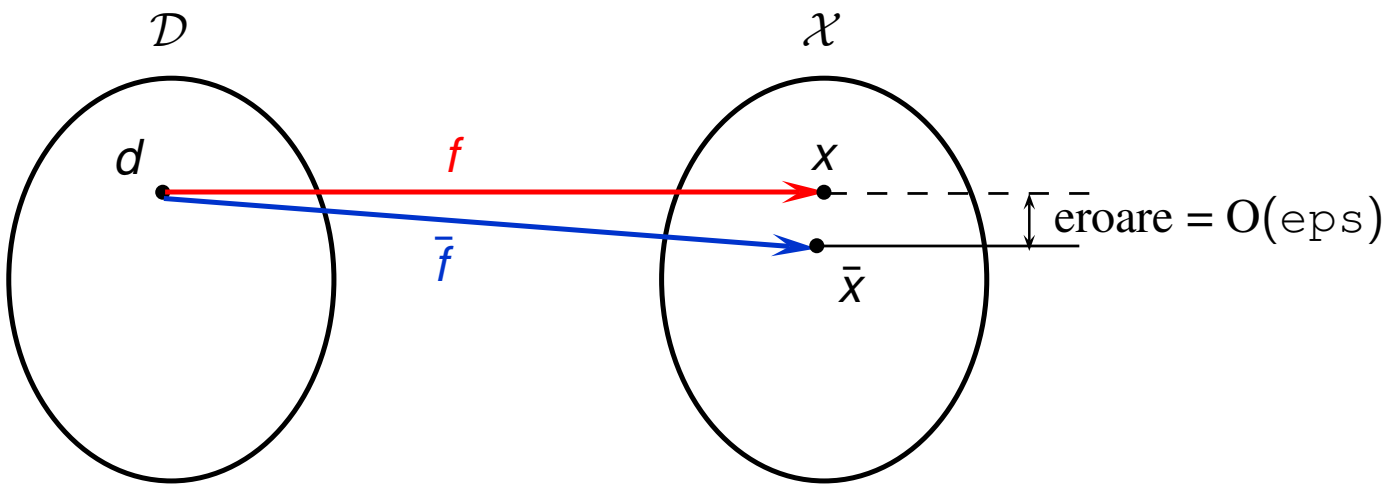
$$\|\mathbf{e_x}\| \leq \kappa \|\varepsilon_\mathbf{d}\|. \tag{54}$$

- Eroarea şi reziduul sunt legate prin numărul de condiţionare.
- Pentru o problemă cu număr de condiţionare mic, o perturbaţie mică în date va duce la o perturbaţie mică a rezultatului.
- Problemele matematice care au κ mare sunt prost condiţionate şi ele nu pot fi rezolvate cu ajutorul calculatorului. Pentru astfel de probleme, trebuie găsită o formulare matematică echivalentă din punct de vedere al rezultatului, dar bine condiţionată.

În cele ce urmează vom presupune că problema f este bine condiţionată şi pentru rezolvarea ei a fost conceput un algoritm $\overline{f}$.

Acurateţea unui algoritm

Acurateţea unui algoritm se referă la eroarea soluţiei numerice.



Reprezentarea intuitivă a unui algoritm a cărui precizie este ideală.

În mod ideal, un algoritm este precis dacă:

$$\frac{\|\bar{f}(\mathbf{d}) - f(\mathbf{d})\|}{\|f(\mathbf{d})\|} = \mathrm{O}(\mathtt{eps}). \tag{55}$$

$\bar{f}(\mathbf{d})$ = "rezultatul algoritmului $\bar{f}$ aplicat datelor $\mathbf{d}$".

Stabilitatea unui algoritm

Dar, rotunjirea datelor este inevitabilă, erorile se acumulează şi perturbă rezultatul. Este mai util să se ţintească stabilitatea algoritmului.
Stabilitatea unui algoritm se referă la comportarea algoritmului atunci când datele de intrare sunt perturbate.
Un algoritm $\bar{f}$ folosit pentru rezolvarea unei probleme f este stabil dacă

$$\frac{\|\bar{f}(\bar{\mathbf{d}}) - f(\mathbf{d})\|}{\|f(\mathbf{d})\|} = \mathrm{O}(\mathtt{eps}), \tag{56}$$

pentru $(\forall)\bar{\mathbf{d}}, \mathbf{d}$ care satisfac $\|\bar{\mathbf{d}} - \mathbf{d}\|/\|\mathbf{d}\| = \mathrm{O}(\mathtt{eps})$.
Pe scurt, un algoritm stabil dă raspunsul aproape corect pentru date reprezentate aproape precis.

Ilustrarea stabilităţii unui algorim - problema

$$\mathbf{Ax} = \mathbf{b}, \quad \text{unde} \quad \mathbf{A} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

$$\begin{aligned} x_2 &= 1 \\ x_1 + x_2 &= 0 \end{aligned} \tag{57}$$

$x_1 = -1$, $x_2 = 1$. $\mathbf{x} = f(\mathbf{d}) = [-1, 1]^T$.
Să considerăm acum că datele au fost perturbate:

$$\bar{\mathbf{A}} = \begin{bmatrix} 10^{-20} & 1 \\ 1 & 1 \end{bmatrix},$$

$$\begin{aligned} 10^{-20} x_1 + x_2 &= 1 \\ x_1 + x_2 &= 0 \end{aligned} \tag{58}$$

$x_1' = -x_2' = 1/(10^{-20} - 1) \approx -1$. Se poate demonstra că această problemă este bine condiţionată.

Ilustrarea stabilităţii unui algorim - algoritmul $\bar{f}_1$

- **Pasul 1:** se înmulţeşte prima ecuaţie a sistemului cu (-10^{20}) şi se adună cu a doua, rezultând x_2;
- **Pasul 2:** se calculează x_1 din prima ecuaţie.

La pasul 1 se ajunge la ecuaţia $(1 - 10^{20})x_2 = -10^{20}$ care, în calculator devine datorită rotunjirilor $-10^{20}x_2 = -10^{20}$, de unde va rezulta $x_2 = 1$, ceea ce este corect.
La pasul 2 ecuaţia de rezolvat devine $10^{-20}x_1 + 1 = 1$, de unde va rezulta $x_1 = 0$, ceea ce este greşit, foarte departe de valoarea adevărată.
Acest algoritm este instabil.

Ilustrarea stabilităţii unui algorim - algoritmul $\bar{f}_2$

- **Pasul 1:** se înmulţeşte a doua ecuaţie a sistemului cu (-10^{-20}) şi se adună cu prima, rezultând x_2;
- **Pasul 2:** se calculează x_1 din a doua ecuaţie.

La pasul 1 se ajunge la ecuaţia $(1 - 10^{-20})x_2 = 1$, care în calculator devine $x_2 = 1$.
La pasul 2 ecuaţia de rezolvat este $x_1 + 1 = 0$, de unde $x_1 = -1$, ceea ce este corect.
Algoritmul $\bar{f}_2$ este stabil. Stabilitatea lui este foarte puternică, el a dat raspunsul exact pentru date de intrare aproape precise.

Concluzii - estimarea acurateţii unei solutii numerice

1. Se estimează numărul de condiţionare al problemei. Se continuă numai dacă problema matematică este bine condiţionată.
2. Se investighează stabilitatea algoritmului. Cel mai simplu este ca acest lucru să se realizeze experimental, rulându-se algoritmul pentru date perturbate. Dacă dispersia rezultatelor este mare atunci algoritmul este instabil şi trebuie schimbat.
3. Dacă algoritmul este stabil, atunci acurateţea finală (modulul erorii relative) este majorată de produsul dintre numărul de condiţionare şi modulul reziduului relativ.

Despre un algoritm stabil care generează erori mici pentru probleme bine condiţionate se spune că este robust.

Referinţe (pseudocod, complexitate)

- [Ciuprina13a] Gabriela Ciuprina - Algoritmi numerici pentru calcule ştiinţifice în ingineria electrică , Editura MatrixROM, 2013, paragrafele 1.1. şi 1.2.

  disponibilă la http://www.lmn.pub.ro/∼gabriela/books/AlgNr_MatrixRom2013.pdf

- [Cormen09] T. Cormen, C. Leiserson, R. Rivest, C. Stein, *Introduction to Algorithms*, MIT Press, cap 1 (The role of algorithms in computing), cap 2 (Getting started), cap 3 (Growth of functions).

  ediţia mai veche este diponibilă online aici

Referinţe (Matlab)

- [MatlabDoc] Documentaţia Matlab
  disponibilă online la

  http://www.mathworks.com/access/helpdesk/help/helpdesk.html
- [Moler04] Clever Moler - Numerical Computing with Matlab, SIAM, 2004

  disponibilă online la http://www.mathworks.com/moler/
- [Getreuer09] Pascal Getreuer - Writing fast Matlab code, 2009

  disponibilă online la

  https://www.mathworks.com/matlabcentral/fileexchange/5685-writing-fast-matlab-code

Referinţe (erori, condiţionare, stabilitate)

- [Ciuprina13a] Gabriela Ciuprina - Algoritmi numerici pentru calcule ştiinţifice în ingineria electrică , Editura MatrixROM, 2013, paragraful 1.3.

  disponibilă la http://www.lmn.pub.ro/∼gabriela/books/AlgNr_MatrixRom2013.pdf

- [Trefethen97] L. N. Trefethen and D. Bau. *Numerical Linear Algebra*. SIAM. 1997. Philadelphia, PA. (Lecture 12: Conditioning and Condition Numbers; Lecture 13: Floating Point Arithmetic; Lecture 14: Stability; Lecture 15: More on Stability;)

  Cartea v-o pot împrumuta la cerere.
  Vă recomand cu căldură să vizitaţi pagina Prof. Trefethen disponibilă la

  https://people.maths.ox.ac.uk/trefethen/