

# Formula lui Taylor

Radu Trîmbițaș

23 februarie 2021

## 1 Formula lui Taylor

- $I$  interval,  $f : I \rightarrow \mathbb{R}$  o funcție derivabilă de  $n$  ori în punctul  $a \in I$ .  
Polinomul lui Taylor de gradul  $n$ , atașat funcției  $f$  în punctul  $a$ :

$$(T_n f)(x) = f(a) + \frac{x-a}{1!} f'(a) + \dots + \frac{(x-a)^n}{n!} f^{(n)}(a)$$

- *Restul* de ordinul  $n$  al formulei lui Taylor în punctul  $x$

$$(R_n f)(x) = f(x) - (T_n f)(x)$$

- *Formula lui Taylor* de ordinul  $n$  pentru funcția  $f$  în vecinătatea punctului  $a$ :

$$f(x) = (T_n f)(x) + (R_n f)(x)$$

sau

$$f(x) = f(a) + \frac{x-a}{1!} f'(a) + \frac{(x-a)^2}{2!} f''(a) + \dots + \frac{(x-a)^n}{n!} f^{(n)}(a) + (R_n f)(x)$$

## 2 Restul

- Are loc

$$(R_n f)(x) = \frac{(x-a)^n}{n!} \omega(x), \text{ cu } \lim_{x \rightarrow a} \omega(x) = 0.$$

- Dacă  $f \in C^{n+1}(I)$ , atunci  $\exists \theta \in (0, 1)$  astfel încât

$$(R_n f)(x) = \frac{(x-a)^{n+1} f^{(n+1)}(a + \theta(x-a))}{(n+1)!}$$

(restul în forma Lagrange)

$$(R_n f)(x) = \frac{(x-a)^{n+1} (1-\theta)^n f^{(n+1)}(a + \theta(x-a))}{n!}$$

(restul în forma Cauchy)

$$(R_n f)(x) = \int_a^x \frac{(x-t)^n}{n!} f^{(n+1)}(t) dt$$

(restul în formă integrală)

### 3 Formula lui Maclaurin

- Dacă în formula lui Taylor se ia  $a = 0$ , se obține formula lui MacLaurin

$$f(x) = f(0) + x f'(0) + \dots + \frac{x^n}{n!} f^{(n)}(0) + (R_n f)(x),$$

unde

$$(R_n f)(x) = \frac{x^{n+1}}{(n+1)!} f^{(n+1)}(\theta x), \quad \theta \in (0, 1).$$

- Exemple de dezvoltări uzuale

$$e^x = 1 + x + \frac{x^2}{2!} + \dots + \frac{x^n}{n!} + R_n(x); \quad (1)$$

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \dots + (-1)^n \frac{x^{2n+1}}{(2n+1)!} + R_{2n+1}(x); \quad (2)$$

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \dots + (-1)^n \frac{x^{2n}}{(2n)!} + R_{2n}(x); \quad (3)$$

- Alte dezvoltări uzuale

$$\ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \dots + (-1)^n \frac{x^n}{n+1} + R_{n+1}(x); \quad (4)$$

$$(1+x)^k = 1 + \binom{k}{1}x + \binom{k}{2}x^2 + \dots + \binom{k}{n}x^n + R_n(x), \quad (5)$$

unde

$$\binom{k}{n} = \frac{k(k-1)\dots(k-n+1)}{n!}.$$

### 4 Aplicații

**Problema 1.** Să se scrie formula lui MacLaurin pentru funcția  $f : [-a, \infty) \rightarrow \mathbb{R}$ ,  $f(x) = \sqrt{a+x}$ ,  $a > 0$ .

**Soluție.** Scriem  $f(x) = \sqrt{a+x} = \sqrt{a} \left(1 + \frac{x}{a}\right)^{\frac{1}{2}}$ ; se obține

$$f(x) = \sqrt{a} \left[ 1 + \frac{1}{2} \frac{x}{a} + (-1)^1 \frac{1}{2^2} \frac{1}{2!} \left(\frac{x}{a}\right)^2 + (-1)^2 \frac{1}{2^3} \frac{1}{3!} \left(\frac{x}{a}\right)^3 + \dots \right. \\ \left. + (-1)^{n-1} \frac{1 \cdot 3 \cdot 5 \dots (2n-3)}{n! 2^n} \left(\frac{x}{a}\right)^n + (R_n f)(x) \right].$$

■

**Problema 2.** Să se determine numărul natural  $n$ , astfel ca pentru  $a = 0$  și  $f : \mathbb{R} \rightarrow \mathbb{R}$ ,  $f(x) = e^x$   $T_n f$  să aproximeze  $f$  în  $[-1, 1]$  cu trei zecimale exacte.

**Soluție.** Impunem condiția  $|(R_n f)(x)| = \left| \frac{x^{n+1} e^{\theta x}}{(n+1)!} \right| < 10^{-3}$ . Deoarece  $\theta x < 1$ ,  $e^{\theta x} < e < 3$ , avem

$$\left| \frac{x^{n+1}}{(n+1)!} e^{\theta x} \right| < \frac{3}{(n+1)!} < 10^{-3} \Rightarrow n = 6.$$

În particular, luând  $x = 1$ , obținem

$$e - \left( 1 + \frac{1}{1!} + \cdots + \frac{1}{6!} \right) < \frac{1}{1000}.$$

■

**Problema 3.** Să se aproximeze  $\sqrt[3]{999}$  cu 12 zecimale exacte.

**Soluție.** Avem

$$\sqrt[3]{999} = 10 \left( 1 - \frac{1}{1000} \right)^{\frac{1}{3}}.$$

Folosim formula (5) pentru  $k = 1/3$ ,  $x = -\frac{1}{1000}$ . Într-o serie alternată modulul erorii este mai mic decât modulul primului termen neglijat.

$$|(R_n f)(x)| < \left| \binom{\frac{1}{3}}{n} 10^{-3n} \right|.$$

Pentru  $n = 4$ , avem  $|(R_n f)(x)| < \frac{10}{243} 10^{-12} = \frac{1}{24300000000000} = 4.1152 \times 10^{-14}$ .

■

## 5 Probleme propuse

**Problema 4.** Dezvoltați funcția eroare

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$$

în serie utilizând seria pentru exponențială și integrând. Calculați seria Taylor a lui  $\operatorname{erf}(x)$  în jurul lui zero direct. Sunt cele două serii identice? Evaluați  $\operatorname{erf}(1)$  adunând patru termeni ai seriei și comparați cu valoarea  $\operatorname{erf}(1) \approx 0.8427$ , care este dată cu patru zecimale corecte. *Indicație:* Din teorema fundamentală a calculului integral rezultă că

$$\frac{d}{dx} \int_0^x f(t) dt = f(x).$$

**Problema 5.** Deduceți seria Taylor pentru  $\ln(1+x)$  și aproximați  $\ln 2$  folosind primii 8 termeni. Câți termeni sunt necesari pentru a obține  $\ln 2$  cu 5 zecimale corecte? La fel pentru  $\ln \frac{1+x}{1-x}$ .

**Problema 6.** Deduceți seria Taylor pentru arctangentă. Câți termeni sunt necesari pentru a obține  $\pi/4$  cu 5 zecimale corecte.

**Problema 7** (Aproximare cu serii MacLaurin). O funcția  $f \in C^n[a, b]$  se poate aproxima, utilizând seria Maclaurin trunchiată, printr-un polinom de grad  $n$

$$f(x) \approx T_n(x) = \sum_{i=0}^n c_i x^i,$$

unde  $c_i = f^{(i)}(0)/i!$ .

- (a) Reprezentați grafic și comparați graficele lui  $f(x) = e^x$  și ale polinoamelor  $(T_2 f)(x)$ ,  $(T_3 f)(x)$ ,  $(T_4 f)(x)$ ,  $(T_5 f)(x)$ . Aproximează mulțumitor polinoamele  $T_n f$  de grad mare funcția  $e^x$  pe un interval din ce în ce mai mare centrat în jurul originii?

- (b) Repetați pentru  $g(x) = \ln(1+x)$ .

**Problema 8** (Aproximare Padé rațională). *Aproximarea Padé rațională* este cea mai bună aproximare a unei funcții printr-o funcție rațională de ordin  $(m, k)$  dat. Se definește ca fiind o aproximare rațională de grad  $(m, k)$  dat care reproduce valorile funcției și derivatelor ei până la ordinul  $m+k$ . Ea dă adesea aproximări mai bune decât seriile Taylor trunchiate și uneori lucrează chiar și atunci când seria Taylor nu converge! În loc să utilizăm polinoame de grad mare, putem utiliza câțuri de polinoame de grad mic. Aceste aproximări se numesc *aproximări raționale*. Fie

$$f(x) \approx \frac{p_m(x)}{q_k(x)} = \frac{\sum_{i=0}^m a_i x^i}{\sum_{j=0}^k b_j x^j} = R_{m,k}(x),$$

unde  $b_0 = 1$ . Aici am normalizat prin  $b_0 \neq 0$  iar valorile lui  $m$  și  $k$  se presupun a fi modeste. Alegem cei  $k$  coeficienți  $b_j$  și cei  $m+1$  coeficienți  $a_i$  din  $R_{m,k}$  astfel încât  $R_{m,k}$  să reproducă valorile lui  $f$  și ale unui număr specificat de derivate ale ei în punctul fixat  $x = 0$ . Construim întâi seria Maclaurin trunchiată  $\sum_{i=0}^n c_i x^i$ , unde  $c_i = f^{(i)}(0)/i!$  și  $c_i = 0$  pentru  $i < 0$ . Apoi, egalăm primele  $m+k+1$  derivate ale lui  $R_{m,k}$  în raport cu  $x$  în  $x = 0$  cu primii  $m+k+1$  coeficienți  $c_i$ . Se obține sistemul:

$$\begin{bmatrix} c_m & c_{m-1} & \cdots & c_{m-(k-2)} & c_{m-(k-1)} \\ c_{m+1} & c_m & \cdots & c_{m-(k-3)} & c_{m-(k-2)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ c_{m+(k-2)} & c_{m+(k-3)} & \cdots & c_m & c_{m-1} \\ c_{m+(k-1)} & c_{m+(k-2)} & \cdots & c_{m+1} & c_m \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_{k-1} \\ b_k \end{bmatrix} = \begin{bmatrix} -c_{m+1} \\ -c_{m+2} \\ \vdots \\ -c_{m+(k-1)} \\ -c_{m+k} \end{bmatrix}.$$

Deoarece  $b_0 = 1$ , rezolvând acest sistem de dimensiune  $k \times k$  vom obține coeficienții  $b_1, b_2, \dots, b_k$ . Valorile lui  $a_0, a_1, \dots, a_m$  se obțin din

$$a_j = \sum_{\ell=0}^j c_{j-\ell} b_\ell \quad (j = 0, 1, \dots, m).$$

De notat că  $a_j = 0$  pentru  $j > m$  și  $b_j = 0$  pentru  $j > k$ . De asemenea, dacă  $k = 0$ , atunci  $R_{m,0}$  este seria Maclaurin trunchiată a lui  $f$ . Mai mult, aproximarea Padé poate avea singularități.

- (a) Implementați aproximarea Padé pentru  $f$ ,  $k$ ,  $m$  date.
- (b) Determinați funcțiile raționale  $R_{1,1}(x)$  și  $R_{2,2}(x)$  pentru  $f(x) = e^x$ . Reprezentați grafic și comparați graficele lui  $f(x) = e^x$ ,  $R_{1,1}$  și  $R_{2,2}$ . Sunt satisfăcătoare aceste aproximații raționale ale lui  $e^x$  pe  $[-1, 1]$ ? Cum se comportă comparativ cu seriile Maclaurin trunchiate din problemele precedente?
- (c) Repetați pentru aproximările  $R_{2,2}(x)$  și  $R_{3,1}(x)$  ale funcției  $g(x) = \ln(1 + x)$ .

**Problema 9.** Calculați dezvoltarea MacLaurin a funcției Bessel  $J_0(2x)$ . Determinați  $R_{2,2}(x)$ ,  $R_{4,3}(x)$  și  $R_{2,4}(x)$  și comparați graficele. Funcțiile Bessel  $J_n$  se definesc prin

$$J_n(x) = \frac{1}{\pi} \int_0^\pi \cos(x \sin \theta - n\theta) d\theta.$$

## Teoria erorilor

precizia rez. unui calcul

erori la datele de intrare

erori de rotunjire (calcul cu nr. repr. într-un nr. finit de cifre)

erori de aproximare. (multe metode care se apropie de sol. exactă)

Exemplu: (P): calculați  $e$

$$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \dots + \frac{1}{n!}$$

(P)' - simplă

calculăm  $e$  astfel

$$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \dots + \frac{1}{n!}$$

TRUNCHIERE

Exemplu:  $\int_a^b f(x) dx \approx \sum_{k=0}^n \dots$

$$f'(a) = \lim_{x \rightarrow a} \frac{f(x) - f(a)}{x - a} \quad \left| \quad \lim_{x \rightarrow h} \frac{f(x+h) - f(x)}{h} = f'(h) \right.$$

$f' \approx$  diferență

## DISCRETIZARE

Prob. numerică = Prob. matematică + niste specificații de precizie asupra rezultatului.

Exemplu:  $f: \mathbb{R} \rightarrow \mathbb{R}, y = ?$  unde  $f(x) = y$

În PC avem:

fie  $x^* \approx x$   
poate fi repr. în calc.

$f^* \approx f$   
poate fi scrisă în calc.

dat

PN:

PM:  $x, f$ , se cere  $f(x)$

SP:  $|f(x) - f^*(x^*)| < \epsilon$

$\epsilon$  dat

de pildă  $x = 5$   
 $x^* = 5.00000002$

$$|x - x^*| < \epsilon \text{ OK}$$

$$\epsilon = \frac{1}{100} \quad \epsilon = \frac{1}{10^{16}}$$

$$\text{nu e OK } |x - x^*| < \epsilon$$

$$1.2 + 1.3 = 2.4999 \dots$$

PN  $\rightarrow$  evaluarea unei funcționale: derivată numerică, integrala numerică  
rez. ec. algebrice: sisteme de ec. liniare  
rez. ec. analitice: rez. numerică de EB, EBP  
pb. de optimizare

PM:  $Fx = y$

Alg. (input) = output

Alg. = ? - pb. de identificare

input = ? - pb. inversă - ex. sist. de ec.

output = ? - pb. directă - află cât e  $\int f$

Măsurile erorilor  $x, x^*$  aprox.

$$\Delta x = x - x^* \text{ eroare}$$

$$\|\Delta x\| = \|x - x^*\| \text{ eroare absolută}$$

$$\delta x = \frac{\|\Delta x\|}{\|x\|} \text{ eroare relativă}$$

$$f(x) = y \quad \leftarrow \text{propagată}$$

$$f(x + \Delta x) = y^* \quad \leftarrow \text{ceva mic}$$

## Aritmetică în virgulă flotantă (Reprezentarea)

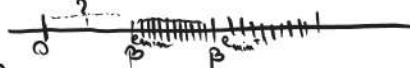
• reprez. are urm. param.: baza  $\beta$  (mereu pară)  
precizia  $p$   
exponent minim  $e_{\min}$   
exponent maxim  $e_{\max}$

$$x - \text{repr. lui în virgulă flotantă este: } x = \pm d_0.d_1d_2\dots d_{p-1} \cdot \beta^e, \quad 0 \leq d_i < \beta$$

semnificativ/mantisă

$$\text{valoarea efectivă a lui } x \text{ este } \pm (d_0 + d_1\beta^{-1} + d_2\beta^{-2} + \dots + d_{p-1}\beta^{-(p-1)})\beta^e$$

$$d_0 \neq 0 \text{ și } 0 = 1.0 \times \beta^{e_{\min}-1}$$



$x \in \mathbb{R}$

$$|x| > \beta^{e_{\max}} \text{ depășire sup.}$$

se introduce și nr. de forma  $0.d_1\dots d_{p-1}$  cu exp.  $\beta^{e_{\min}+1}$

$$|x| < 1.0 \beta^{e_{\min}} \text{ depășire inferioară}$$

IEEE 754 unde  $\beta = 2$ .

# Teoria erorilor și aritmetica în virgula flotantă

Radu T. Trîmbițaș

6 martie 2022

## 1 Probleme

- P1.** Scrieți funcții MATLAB pentru a calcula epsilon-ul mașinii, cel mai mare număr reprezentabil în VF și cel mai mic număr normalizat și nenormalizat reprezentabil în VF. Comparați rezultatele cu cele returnate de funcțiile MATLAB `eps`, `realmin`, `realmax`.
- P2.** Scrieți funcții MATLAB pentru calculul lui  $\sin x$  și  $\cos x$  folosind formula lui Taylor:

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \dots + (-1)^n \frac{x^{2n+1}}{(2n+1)!} + \dots$$

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \dots + (-1)^n \frac{x^{2n}}{(2n)!} + \dots$$

se reduce la primul cadran (-2 pi de cate ori de poate), apoi fiecare "sfert" se reduce la [0, 90]  
[90, 180] -> [0, 90], [180, 270] -> [0, 90], [270, 360] -> [0, 90]  
cu formulele de sin cos si nebunii)

Știm de la cursul de Analiză matematică următoarele:

- modulul erorii este mai mic decat modulul primului termen neglijat;
- raza de convergență este  $R = \infty$ .

Ce se întâmplă pentru  $x = 10\pi$  (și în general pentru  $x = 2k\pi$ ,  $k$  mare)?

Explicați fenomenul și propuneți un remediu.

- P3.** Scrieți funcții MATLAB pentru calculul lui  $\sin x$  și  $\cos x$  folosind aproximarea Padé în locul formulei lui Taylor. Atenție la reducerea rangului.

- P4.** Scrieți o funcție MATLAB care primește la intrare un număr flotant (simplă sau dublă precizie) și returnează reprezentarea sa binară pe componente: semn, exponent deplasat și semnificantul (așa cum este acesta reprezentat intern).

## 2 Probleme suplimentare

- S1.** Fie două numere reale  $x_1, x_2 \in \mathbb{R}$ ,  $x_1 \neq x_2$ . Considerăm reprezentările lor în virgulă flotantă  $x_1^*$  și  $x_2^*$  astfel încât  $x_1^* = \text{fl}(x_1) = x_1(1 + \delta_1)$ ,  $x_2^* = \text{fl}(x_2) = x_2(1 + \delta_2)$  și  $|\delta_1| < \delta$ ,  $|\delta_2| < \delta$ . Cât de mic trebuie să fie  $\delta$ , astfel încât să putem testa corect (în virgulă flotantă cu precizia mașinii eps), dacă  $x_1 \neq x_2$ .
- S2.** Același enunț ca la problema P1, dar în Maple.



## Conditionarea unei probleme

$$f: \mathbb{R}^m \rightarrow \mathbb{R}^n, y = f(x)$$

$x \rightarrow x + \Delta x$  ? perturbatia lui  $y$  fata de perturbatia lui  $x$   
perturbatie a lui  $x$

gradul de sensibilitate  $\leftarrow$  nr. de conditionare al aplicatiei  $f$  in punctul  $x$

$\text{cond } f$  - o prop. a functiei / aceasta nu depinde de implem.  
 dar  $\text{cond } f$  este f. relevantă pt. + sol. algoritmică

$$y^* = f(x^*)$$

$x^* = x + \delta$ , iar  $\delta = \|x - x^*\|$  poate fi estim.  
 in termeni de precizie  
 ai masurii

Obs.  $f: \mathbb{R} \rightarrow \mathbb{R}$   $x \neq 0, y \neq 0, (\text{cond } f)(x) = \left| \frac{x f'(x)}{f(x)} \right|$   
 $f(x) = y$   
 $x = 0, y \neq 0, (\text{cond } f)(x) = \left| \frac{f'(x)}{f(x)} \right|$

$$x = 0, y = 0, (\text{cond } f)(x) = f'(x)$$

Ex. sistem de ec. alg. (lin.)  $Ax = b \Rightarrow x = A^{-1}b = f(b)$   
 $A$  fixată  
 $b$  perturbat

$$\text{cond } A = \|A\| \cdot \|A^{-1}\|$$

$x \in \mathbb{R}, \|x\| = \text{norma lui } x \text{ pe } \mathbb{R} \text{ (modul)}$

$$x = (x_1, x_2) \in \mathbb{R}, \|x\|_2 = \sqrt{x_1^2 + x_2^2}$$

$$A = (a_{ij})_{\substack{1 \leq i \leq m \\ 1 \leq j \leq n}}, \|A\|_2 = \left( \sum_{i=1}^m \sum_{j=1}^n a_{ij}^2 \right)^{\frac{1}{2}} \leftarrow$$

$$\|A\| = \max_{1 \leq j \leq n} \|a_{ij}\|$$

## Conditionarea unui algoritim

$$f: \mathbb{R}^m \rightarrow \mathbb{R}^n, y = f(x)$$

$f$  deja in alg. repr.

$$f + \text{algoritim } A \Rightarrow y_A = f_A(x)$$

$$(\text{cond } A)(x) \leq \frac{\|x_A - x\|}{\|x\|} \cdot \frac{\|x\|}{\text{eps}}$$

Eroare globală  $\xrightarrow{\text{erorile din date} + \text{eps}}$  eroarea totală

$$\frac{\|y_A^* - y^*\|}{\|y^*\|} \leq (\text{cond } f)(x) \left[ \frac{\|x^* - x\|}{\|x\|} + \text{cond } A(x^*) \cdot \text{eps} \right]$$

## Pb. prost conditionate / incorect puse

$(\text{cond } f)(x) \gg 1$  : erori rel. mici putem să ne așteptăm  
mult mai mare la erori foarte mari în output

Specificațiile de precizie  $\begin{cases} < \text{b.c.} \\ < \text{p.c.} \end{cases}$

$$\text{P.C.: } \frac{\|y^* - y_A\|}{\|y\|} < \tau \text{ și } (\text{cond } f)(x) \frac{\|x^* - x\|}{\|x\|} \geq \tau$$

$\Rightarrow$  problema este P.C.

Ex. nr. răd. reale pt. o fct. / rangul unei matrici

# Conditionarea rădăcinilor ecuațiilor polinomiale

Radu Trîmbițaș

30 aprilie 2020

## 1 Senzitivitatea rădăcinilor polinomiale

Vom considera condiționarea unei rădăcini  $\xi$  a unei ecuații polinomiale  $p(x)$ . Prin aceasta se înțelege influența asupra lui  $\xi$  a unei perturbații a coeficienților polinomului  $p(x)$ :

$$p_\varepsilon(x) = p(x) + \varepsilon g(x),$$

unde  $g(x) \neq 0$  este un polinom arbitrar nenul.

Se poate arăta că dacă  $\xi$  este o rădăcină simplă a lui  $p$ , atunci pentru valori absolute suficient de mici ale lui  $\varepsilon$  există o funcție analitică  $\xi(\varepsilon)$ , cu  $\xi(0) = \xi$ , astfel încât  $\xi(\varepsilon)$  să fie o rădăcină simplă a polinomului perturbat  $p_\varepsilon(x)$ :

$$p(\xi(\varepsilon)) + \varepsilon g(\xi(\varepsilon)) \equiv 0.$$

De aici, prin derivare în raport cu  $\varepsilon$ , se obține pentru  $k := \xi'(0)$  ecuația

$$kp'(\xi(0)) + g(\xi(0)) = 0,$$

deci

$$k := \frac{-g(\xi)}{p'(\xi)}.$$

Astfel, până la o aproximare de ordinul întâi (adică, eliminând termenii în  $\varepsilon$  cu puteri mai mari decât 1), se obține prin dezvoltarea Taylor a lui  $\xi(\varepsilon)$  relația

$$\xi(\varepsilon) := \xi - \varepsilon \frac{g(\xi)}{p'(\xi)}. \quad (1)$$

În cazul unei rădăcini multiple  $\xi$ , de ordin  $m$ , se poate arăta că  $p(x) + \varepsilon g(x)$  are o rădăcină de forma

$$\xi(\varepsilon) = \xi + h\left(\varepsilon^{1/m}\right),$$

unde, pentru  $|t|$  mic,  $h(t)$  este o funcție analitică cu  $h(0) = 0$ . Derivând de  $m$  ori în raport cu  $t$  și observând că  $p(\xi) = p'(\xi) = \dots = p^{(m-1)}(\xi) = 0$ ,  $p^{(m)}(\xi) \neq 0$ , din

$$0 \equiv p_\varepsilon(\xi(\varepsilon)) = p(\xi + h(t)) + t^m g(\xi + h(t)), \quad t^m = \varepsilon,$$

pentru  $k = h'(0)$  ecuația

$$p^{(m)}(\xi) k^m + m!g(\xi) = 0,$$

de unde

$$k := \left[ -\frac{m!g(\xi)}{p^{(m)}(\xi)} \right]^{1/m}.$$

Din nou, până la o aproximare de ordinul I

$$\xi(\varepsilon) = \xi + \varepsilon^{1/m} \left[ -\frac{m!g(\xi)}{p^{(m)}(\xi)} \right]^{1/m}. \quad (2)$$

Pentru  $m = 1$  această formulă se reduce la (1) pentru rădăcini simple.

Să presupunem că polinomul  $p(x)$  este dat sub formă de listă de coeficienți

$$p(x) = a_0x^n + \dots + a_n.$$

Pentru

$$g_i(x) = a_i x^{n-i}$$

polinomul  $p_\varepsilon(x)$  se obține prin înlocuirea lui  $a_i$  în  $p(x)$  cu  $a_i(1 + \varepsilon)$ . Formula (2) ne dă următoarea estimare a efectului asupra rădăcinii  $\xi$  a unei erori relative  $\varepsilon$  asupra lui  $a_i$ :

$$\xi(\varepsilon) - \xi = \varepsilon^{1/m} \left[ -\frac{m!a_i\xi^{n-i}}{p^{(m)}(\xi)} \right]. \quad (3)$$

Astfel, este clar că în cazul rădăcinilor multiple, schimbările în valoarea rădăcinii  $\xi(\varepsilon) - \xi$  sunt proporționale cu  $\varepsilon^{1/m}$ ,  $m > 1$ , pe când în cazul unei rădăcini simple ele sunt proporționale doar cu  $\varepsilon$ : rădăcinile multiple sunt întotdeauna prost condiționate. Rădăcinile simple pot fi și ele prost condiționate. Aceasta se întâmplă dacă factorul lui  $\varepsilon$  în (3),

$$k(i, \xi) := \left| \frac{a_i \xi^{n-i}}{p'(\xi)} \right|,$$

este mare comparativ cu  $\xi$ , ceea ce se poate întâmpla și în cazul unor polinoame în aparență „nevătămătoare”.

**Exemplul 1 (Wilkinson 1959)** Rădăcinile  $\xi_k = k$ ,  $k = 1, 2, \dots, 20$ , ale polinomului

$$p(x) = (x-1)(x-2)\dots(x-20) = \sum_{i=0}^{20} a_i x^{20-i}$$

sunt bine separate. Pentru  $\xi_{20} = 20$ , găsim  $p'(20) = 19!$ , și înlocuind coeficientul  $a_1 = -(1+2+\dots+20) = -210$  cu  $a_1(1+\varepsilon)$  cauzează o schimbare estimată de

$$\xi_{20}(\varepsilon) - \xi_{20} = \varepsilon \frac{210 \times 20^{19}}{19!} \approx \varepsilon \cdot 0.9 \times 10^{10}.$$

Cele mai drastice schimbări schimbări sunt cauzate în  $\xi_{16}$  prin perturbarea lui  $a_5$ . Deoarece  $\xi_{16} = 16$  și  $a_5 \approx 10^{-10}$ ,

$$\xi_{16}(\varepsilon) - \xi_{16} = -\varepsilon a_5 \frac{16^5}{4!15!} \approx \varepsilon \cdot 3.7 \times 10^{14}.$$

Aceasta înseamnă că rădăcinile lui  $p$  sunt atât de prost condiționate încât chiar și calculul cu o aritmetică de 14 cifre nu va garanta nici o cifră corectă în  $\xi_{16}$ .

**Exemplul 2** Prin contrast, rădăcinile polinomului

$$p(x) = \sum_{i=0}^{20} a_i x^{20-i} := \prod_{j=1}^{20} (x - 2^{-j}), \quad \xi_j = 2^{-j},$$

deși nu sunt bine separate și se acumulează spre zero, sunt toate bine condiționate. De exemplu, schimbând  $a_{20}$  în  $a_{20}(1 + \varepsilon)$  cauzează o variație a lui  $\xi_{20}$  care până la o aproximare de ordinul întâi poate fi delimitată sub forma

$$\left| \frac{\xi_{20}(\varepsilon) - \xi_{20}}{\xi_{20}} \right| = \left| \varepsilon \frac{1}{(2^{-1} - 1)(2^{-2} - 1) \dots (2^{-19} - 1)} \right| \leq 4|\varepsilon|.$$

Mai general, se poate arăta că pentru toate rădăcinile  $\xi_j$  și toate schimbările  $a_i \rightarrow a_i(1 + \varepsilon)$

$$\left| \frac{\xi_j(\varepsilon) - \xi_j}{\xi_j} \right| \leq 64|\varepsilon|.$$

Totuși, rădăcinile sunt bine condiționate numai în raport cu schimbări relative ale coeficienților  $a_i$  și nu pentru schimbări absolute mici. Dacă înlocuim  $a_{20} = 2^{-210}$  cu  $\bar{a}_{20} = a_{20} + \Delta a_{20}$ ,  $\Delta a_{20} = 2^{-48}$  ( $\approx 10^{-14}$ ) – acesta poate fi considerată o schimbare absolută mică – atunci polinomul modificat are rădăcinile  $\bar{\xi}_i$  cu

$$\bar{\xi}_1 \dots \bar{\xi}_{20} = \bar{a}_{20} = 2^{-120} + 2^{-48} = (2^{162} + 1) \xi_1 \dots \xi_{20}.$$

Cu alte cuvinte, există cel puțin un indice  $r$  cu  $|\bar{\xi}_r/\xi_r| \geq (2^{162} + 1)^{1/20} > 2^8 = 256$ .

Trebuie accentuat că formula (3) se referă doar la sensibilitatea rădăcinilor unui polinom

$$p(x) := \sum_{i=0}^n a_i x^{n-i}$$

în reprezentarea uzuală a coeficienților. Există și alte moduri de reprezentare a polinoamelor – de exemplu, ca polinoame caracteristice ale matricelor tridiagonale prin elementele matricei. Efectul unei modificări a parametrilor asupra rădăcinilor într-o astfel de reprezentare poate diferi printr-un ordin de mărime de cel din formula (3). Condiționarea rădăcinilor se definește întotdeauna cu un tip particular de reprezentare în minte.

**Exemplul 3** *Se știe că pentru orice matrice tridiagonală*

$$J = \begin{bmatrix} \alpha_1 & \beta_2 & & & \\ \beta_2 & \alpha_2 & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \ddots \\ & & & \ddots & \alpha_{19} & \beta_{20} \\ & & & & \beta_{20} & \alpha_{20} \end{bmatrix}$$

*al cărei polinom caracteristic este  $p(x) \equiv (x-1)(x-2)\cdots(x-20)$ , schimbări relative mici în  $\alpha_i$  și  $\beta_i$  cauzează schimbări relative mici ale rădăcinilor  $\xi_j = j$ . În raport cu această reprezentare, toate rădăcinile sunt bine condiționate, deși în raport cu reprezentarea uzuală prin coeficienți sunt foarte prost condiționate.*

## Bibliografie

- [1] J. Stoer, R. Bulirsch, Introduction to Numerical Analysis, 2nd edition, Springer, 1992

# Condiționarea unei probleme

Radu T. Trîmbițaș

6 martie 2022

## 1 Probleme

1. Fie sistemul (exemplul este datorat lui Wilson)

$$\begin{bmatrix} 10 & 7 & 8 & 7 \\ 7 & 5 & 6 & 5 \\ 8 & 6 & 10 & 9 \\ 7 & 5 & 9 & 10 \end{bmatrix} x = \begin{bmatrix} 32 \\ 23 \\ 33 \\ 31 \end{bmatrix}$$

cu soluția  $\begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}^T$ .

- (a) Ce se obține dacă perturbăm membrul drept astfel încât el să devină  $\begin{bmatrix} 32.1 & 22.9 & 33.1 & 30.9 \end{bmatrix}$ ? Care este eroarea relativă la intrare, la ieșire și raportul lor? formula de eroare relativă de la lab2
- (b) Aceeași întrebare dacă perturbăm matricea sistemului:

$$\begin{bmatrix} 10 & 7 & 8.1 & 7.2 \\ 7.08 & 5.04 & 6 & 5 \\ 8 & 5.98 & 9.89 & 9 \\ 6.99 & 4.99 & 9 & 9.98 \end{bmatrix} \quad \text{eroare relativă pt matrice}$$

Explicați și analizați fenomenul.

2. Să se studieze condiționarea matricei Hilbert  $H_n$  în raport cu norma euclidiană,  $n = \overline{10, 15}$ . pt matricea hilbert, exista comanda (hilb?) for n=10:15 etc prin 3 metode: cu cond, cum e in lab3 + slide-uri de la curs
- ✕ 3. (a) Să se studieze condiționarea matricei Vandermonde  $V_n(t)$ , pentru  $t_k = -1 + k \frac{2}{n}$  (puncte echidistante în  $[-1, 1]$ ) și  $n = \overline{10, 15}$  în raport cu norma Cebîșev.

(b) Aceeași întrebare pentru  $t_k = \frac{1}{k}$ ,  $k = \overline{1, n}$  și  $n = \overline{10, 15}$ .

4. Să se studieze teoretic și experimental condiționarea problemei determinării rădăcinilor ecuației polinomiale

$$x^n + a_1x^{n-1} + a_2x^{n-2} + \dots + a_n = 0 \quad (1)$$

cunoscându-se coeficienții. Se va scrie o rutină pentru calculul numărului de condiționare al fiecărei rădăcini și se va studia grafic efectul perturbării fiecărui coeficient cu o variabilă aleatoare normală cu media 0 și dispersia  $10^{-10}$ . Aplicație pentru ecuațiile

$$(x-1)(x-2)\dots(x-n) = 0$$

și (1) pentru  $a_k = 2^{-k}$ . Se va lua ca exemplu practic pentru testare  $n = 20$ . Ce se întâmplă dacă perturbația urmează legea uniformă?

5. Fie

$$E_n = \int_0^1 x^n e^{x-1} dx.$$

Se observă că  $E_1 = 1/e$  și  $E_n = 1 - nE_{n-1}$ ,  $n = 2, 3, \dots$

Se poate arăta că

$$0 < E_n < \frac{1}{n+1}$$

și dacă  $E_1 = c$  avem

$$\lim_{n \rightarrow \infty} E_n = \begin{cases} 0, & \text{pentru } c = 1/e \\ \infty & \text{altfel.} \end{cases}$$

Explicați fenomenul, găsiți un remediu și calculați  $e$  cu precizia eps.

vezi PS pt  
nebuniile astea

+ fisier ajutator

## 2 Probleme facultative

1. Să se studieze condiționarea unei rădăcini multiple a unei ecuații algebrice. Scrieți o rutină MATLAB pentru calculul numerelor de condiționare dacă se dau ecuația (coeficienții), rădăcinile și multiplicitățile lor. Repetați experimentul aleator de la problema 4 pentru ecuația

$$(x - 1)^2(x - 2)^2 \dots (x - n)^2 = 0$$



# Sisteme de ecuații liniare

## Metode directe

### metoda lui Gauss

$$\begin{cases} x - y + 6z = 1 \\ 2x + y - 13z = 2 \\ x + y - z = 15 \end{cases}$$

$$\left( \begin{array}{ccc|c} 1 & -1 & 6 & 1 \\ 2 & 1 & -13 & 2 \\ 1 & 1 & -1 & 15 \end{array} \right) \xrightarrow[L_3 - L_1]{L_2 - 2L_1} \left( \begin{array}{ccc|c} 1 & -1 & 6 & 1 \\ 0 & 3 & -25 & 0 \\ 0 & 2 & -7 & 14 \end{array} \right)$$

$$\xrightarrow{L_3 - \frac{2}{3}L_2} \left( \begin{array}{ccc|c} 1 & -1 & 6 & 1 \\ 0 & 3 & -25 & 0 \\ 0 & 0 & \frac{29}{3} & 14 \end{array} \right)$$

$$\begin{aligned} -7 + \frac{2}{3} \cdot 25 &= -7 + \frac{50}{3} = -\frac{21}{3} + \frac{50}{3} = \frac{29}{3} \\ 14 - \frac{2}{3} \cdot 0 &= 14 \end{aligned}$$

Am obținut sistemul echivalent:

$$\begin{cases} x - y + 6z = 1 \\ 3y - 25z = 0 \\ \frac{29}{3}z = 14 \end{cases} \Rightarrow z = \frac{42}{29}$$

$$\begin{aligned} x + y + z &= 14 \\ -x + y + z &= 0 \\ y + x + z &= 14 \\ y - x + z &= 0 \end{aligned}$$

Altă versiune: Gauss-Jordan (în matricea extinsă se fac 0 sub diagonala și deasupra diagonalei)

### descompunere LU

L - lower triangular  
U - upper triangular

$$\text{ex. } L = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

$$U = \begin{pmatrix} 1 & 7 & 2 \\ 0 & 4 & 1 \\ 0 & 0 & 9 \end{pmatrix}$$

rez. un sistem de ec. lin. cu aj. desc. LU.

$$AX = b$$

$$LUX = b \Rightarrow \begin{cases} LY = b \text{ unde } Y = UX \\ UX = Y \end{cases}$$

$$A = \begin{pmatrix} 1 & 0 & 2 \\ 2 & 1 & 4 \\ 1 & -1 & 3 \end{pmatrix} \Rightarrow \text{matrice triu. superioară (faceți 0 sub diag. prime)} \quad L = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & -1 & 1 \end{pmatrix}$$

$$\left( \begin{array}{ccc|c} 1 & 0 & 2 & 1 \\ 2 & 1 & 4 & 2 \\ 1 & -1 & 3 & 1 \end{array} \right) \xrightarrow{L_2 - 2L_1} \left( \begin{array}{ccc|c} 1 & 0 & 2 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & -1 & 3 & 1 \end{array} \right) \xrightarrow{L_3 - L_1} \left( \begin{array}{ccc|c} 1 & 0 & 2 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \end{array} \right) \xrightarrow{L_3 + L_2} \left( \begin{array}{ccc|c} 1 & 0 & 2 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{array} \right) \Rightarrow U$$

$$LU = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & -1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 2 \\ 2 & 1 & 4 \\ 1 & -1 & 3 \end{pmatrix} = A \quad \checkmark$$

matrice hermitiană: A hermitiană ( $\Rightarrow A = A^t$ )

matrice simetrică  $A = A^t$

$$\overline{a+bi} = a-bi$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad \begin{pmatrix} 1 & 4 \\ 0 & 2 \\ 4 & 1 & 3 \end{pmatrix}$$

matrice pozitiv definită: A poz. def. ( $\Leftrightarrow \forall x \in \mathbb{R}^n \setminus \{0\}: x^T A x > 0$ )

Criteriul lui Sylvester

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

$$\Delta_1 = |1|$$

$$\Delta_2 = \begin{vmatrix} 1 & 2 \\ 4 & 5 \end{vmatrix} < 0 \text{ nu e poz. def.}$$

$$\Delta_3 = \begin{vmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{vmatrix}$$

A poz. def. dacă  $\Delta_1 > 0, \Delta_2 > 0, \Delta_3 > 0$

De rezolvat: 1-4.

# Sisteme liniare - metode directe

Radu T. Trîmbițaș

14 martie 2021

## 1 Eliminare gaussiană

Să considerăm sistemul liniar cu  $n$  ecuații și  $n$  necunoscute

$$Ax = b, \quad (1)$$

unde  $A \in \mathbb{K}^{n \times n}$ ,  $b \in \mathbb{K}^{n \times 1}$  sunt date, iar  $x \in \mathbb{K}^{n \times 1}$  trebuie determinat, sau scris pe componente

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 & (E_1) \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 & (E_2) \\ \vdots & \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n & (E_n) \end{cases} \quad (2)$$

### 1.1 Eliminare gaussiană cu pivotare parțială

Metoda este dată de algoritmul 1.

### 1.2 Eliminare gaussiană cu pivot scalat pe coloană

O tehnică care micșorează eroarea și preîntâmpină anularea flotantă este *pivotarea parțială cu pivot scalat pe coloană*. Definim la început un factor de scară pentru fiecare linie

$$s_i = \max_{j=1, n} |a_{ij}| \text{ or } s_i = \sum_{j=1}^n |a_{ij}|.$$

---

**Algoritmul 1** Rezolvă sistemul  $Ax = b$  prin metoda eliminării a lui Gauss

---

**Intrare:** Matricea extinsă  $A = (a_{ij})$ ,  $i = \overline{1, n}$ ,  $j = \overline{1, n+1}$

**Ieșire:** Soluțiile  $x_1, \dots, x_n$  sau un mesaj de eroare

```
{Eliminare}
1: for  $i := 1$  to  $n - 1$  do
2:   Fie  $p$  cel mai mic întreg  $i \leq p \leq n$ ,  $a_{pi} \neq 0$  și  $|a_{pi}| = \max_{i \leq j \leq n} |a_{ji}|$ 
3:   if  $\nexists p$  then
4:     mesaj (' $\nexists$  soluție unică'); STOP
5:   end if
6:   if  $p \neq i$  then
7:      $(E_p) \leftrightarrow (E_i)$ 
8:   end if
9:   for  $j := i + 1$  to  $n$  do
10:     $m_{ji} := a_{ji}/a_{ii}$ ;
11:     $(E_j - m_{ji}E_i) \rightarrow (E_j)$ ;
12:  end for
13: end for
14: if  $a_{nn} = 0$  then
15:   mesaj (' $\nexists$  soluție unică'); STOP
16: end if
   {Substituție inversă}
17:  $x_n := a_{n,n+1}/a_{nn}$ ;
18: for  $i := n - 1$  downto  $1$  do
19:    $x_i = \left[ a_{i,n+1} - \sum_{j=i+1}^n a_{ij}x_j \right] / a_{ii}$ ;
20: end for
21: Returnează  $(x_1, \dots, x_n)$  {succes} STOP.
```

---

Dacă există un  $i$  a.î.  $s_i = 0$ , matricea este singulară. Pașii următori vor stabili interschimbările care se vor face. La al  $i$ -lea pas vom găsi cel mai mic întreg  $p$ ,  $i \leq p \leq n$ , a.î.

$$\frac{|a_{pi}|}{s_p} = \max_{i \leq j \leq n} \frac{|a_{ji}|}{s_j}$$

și apoi,  $(E_i) \leftrightarrow (E_p)$ . Scalarea ne garantează că cel mai mare element din fiecare coloană are înainte de comparațiile necesare pentru schimbare mărimea relativă 1. Scalarea se realizează doar în comparații, nu efectiv în matrice, astfel că împărțirea cu factorul de scalare nu produce nici o eroare de rotunjire.

## 2 Descompunere (factorizare) LUP

Ideea din spatele descompunerii LUP este de a găsi 3 matrice pătratice de ordinul  $n$   $L, U$  și  $P$  astfel încât

$$PA = LU \tag{3}$$

unde

- $L$  este o matrice triunghiulară inferior;
- $U$  este o matrice triunghiulară superior;
- $P$  este o matrice de permutare.

Tripletul  $(L, U, P)$  se va numi **descompunere LUP** a matricei  $A$ . Orice matrice nesară posedă o astfel de descompunere.

Sistemul

$$Ax = b \tag{4}$$

se poate rezolva astfel

$$Ax = b \iff LUx = Pb \iff Ly = Pb \wedge Ux = y, \tag{5}$$

deoarece

$$Ax = P^{-1}LUx = P^{-1}Ly = P^{-1}Pb = b. \tag{6}$$

Având descompunerea LUP sistemul se poate rezolva cu algoritmul 2.

---

**Algoritmul 2** Rezolvă sistemul  $Ax = b$  având descompunerea LUP

---

**Intrare:** Matricele  $L$ ,  $U$ , vectorul  $b$ , vectorul de permutare  $\pi$ , toate de dimensiune  $n$

**Ieșire:** Soluțiile  $x_1, \dots, x_n$

```
1: for  $i := 1$  to  $n$  do
2:    $y_i := b_{\pi[i]} - \sum_{j=1}^{i-1} l_{ij}y_j$ ;
3: end for
4: for  $i := n$  downto  $1$  do
5:    $x_i = \left[ y_i - \sum_{j=i+1}^n u_{ij}x_j \right] / u_{ii}$ ;
6: end for
```

---

**Observație.** Am presupus că matricea  $P$  este reprezentată prin vectorul  $\pi$ .

Procedura care urmează (algoritmul 3) calculează descompunerea LUP. Ea reprezintă  $P$  ca un vector  $\pi$ , iar  $L$  și  $U$  sunt calculate în locul lui  $A$ , adică la terminare

$$a_{ij} = \begin{cases} l_{ij}, & \text{pentru } i > j, \\ u_{ij}, & \text{pentru } i \leq j. \end{cases}$$

---

**Algoritmul 3** Descompunere LUP

---

**Intrare:** Matricea  $A$ , de dimensiune  $m$

**Ieșire:** Matricele  $L$ ,  $U$  și  $P$ , toate de dimensiune  $m$

```
 $p = 1 : m$ ;
for  $k := 1$  to  $m - 1$  do
  {Pivotare}
  Alege  $i \geq k$  care maximizează  $|u_{ik}|$ ;
   $A_{k,:} \leftrightarrow A_{i,:}$ ; {interschimbare}
   $p_k \leftrightarrow p_i$ ;
   $lin := i + 1 : m$ ;
  {Calculez complementul Schur}
   $A_{lin,k} := A_{lin,k} / A_{k,k}$ ;
   $A_{lin,lin} := A_{lin,lin} - A_{lin,k}A_{k,lin}$ ;
end for
Extrage  $L$ ,  $U$ , generează  $P$ ;
```

---

**Exemplu.** Să se calculeze descompunerea LUP a matricei

```
A =  
 2.0000  0      2.0000  0.6000  
 3.0000  3.0000  4.0000 -2.0000  
 5.0000  5.0000  4.0000  2.0000  
-1.0000 -2.0000  3.4000 -1.0000
```

Calcululele decurg astfel

```
>> [l,u,p]=lup(A)
```

interschimb liniile 1 și 3

```
A =  
  
 5.0000  5.0000  4.0000  2.0000  
 3.0000  3.0000  4.0000 -2.0000  
 2.0000  0      2.0000  0.6000  
-1.0000 -2.0000  3.4000 -1.0000
```

calculez complementul Schur

```
A =  
 5.0000  5.0000  4.0000  2.0000  
 0.6000  3.0000  4.0000 -2.0000  
 0.4000  0      2.0000  0.6000  
-0.2000 -2.0000  3.4000 -1.0000
```

```
A =  
 5.0000  5.0000  4.0000  2.0000  
 0.6000  0      1.6000 -3.2000  
 0.4000 -2.0000  0.4000 -0.2000  
-0.2000 -1.0000  4.2000 -0.6000
```

interschimb liniile 2 și 3

A =

5.0000	5.0000	4.0000	2.0000
0.4000	-2.0000	0.4000	-0.2000
0.6000	0	1.6000	-3.2000
-0.2000	-1.0000	4.2000	-0.6000

calculez complementul Schur

A =

5.0000	5.0000	4.0000	2.0000
0.4000	-2.0000	0.4000	-0.2000
0.6000	0	1.6000	-3.2000
-0.2000	0.5000	4.2000	-0.6000

A =

5.0000	5.0000	4.0000	2.0000
0.4000	-2.0000	0.4000	-0.2000
0.6000	0	1.6000	-3.2000
-0.2000	0.5000	4.0000	-0.5000

interschimb liniile 3 și 4

A =

5.0000	5.0000	4.0000	2.0000
0.4000	-2.0000	0.4000	-0.2000
-0.2000	0.5000	4.0000	-0.5000
0.6000	0	1.6000	-3.2000

calculez complementul Schur

A =

5.0000	5.0000	4.0000	2.0000
0.4000	-2.0000	0.4000	-0.2000
-0.2000	0.5000	4.0000	-0.5000
0.6000	0	0.4000	-3.2000

A =

5.0000	5.0000	4.0000	2.0000
--------	--------	--------	--------

```

    0.4000 -2.0000 0.4000 -0.2000
-0.2000  0.5000 4.0000 -0.5000
    0.6000  0      0.4000 -3.0000

```

Rezultatele finale sunt

l =

```

    1.0000  0      0      0
    0.4000  1.0000  0      0
-0.2000  0.5000  1.0000  0
    0.6000  0      0.4000  1.0000

```

u =

```

    5.0000  5.0000  4.0000  2.0000
    0      -2.0000  0.4000 -0.2000
    0      0      4.0000 -0.5000
    0      0      0      -3.0000

```

p =

```

    0  0  1  0
    1  0  0  0
    0  0  0  1
    0  1  0  0

```

verificare:

>> disp(l\*u)

```

    5.0000  5.0000  4.0000  2.0000
    2.0000  0.0000  2.0000  0.6000
-1.0000 -2.0000  3.4000 -1.0000
    3.0000  3.0000  4.0000 -2.0000

```

>> disp(p\*A)

```

    5.0000  5.0000  4.0000  2.0000
    2.0000  0.0000  2.0000  0.6000
-1.0000 -2.0000  3.4000 -1.0000
    3.0000  3.0000  4.0000 -2.0000

```



### 3 Descompunere (factorizare) Cholesky

O matrice hermitiană și pozitiv definită se poate factoriza sub forma  $A = LL^*$  sau  $A = R^*R$ , unde  $L$  este o matrice triunghiulară inferior, iar  $R$  este triunghiulară superior.

Pentru algoritm a se vedea notele de curs sau algoritmul 4.

---

**Algoritmul 4** Descompunere Cholesky

---

**Intrare:** Matricea  $A$ , hermitiană și pozitiv definită

**Ieșire:** Matricea  $R$ , triunghiulară superior

```
R := A;
for k := 1 to m do
  for j := k + 1 to m do
     $R_{j,j:m} := R_{j,j:m} - R_{k,j:m} \overline{R_{k,j}} / R_{k,k}$ 
  end for
   $R_{k,k:m} := R_{k,k:m} / \sqrt{R_{k,k}}$ 
end for
```

---

### 4 Probleme

- ✓ **Problema 1** Implementați eliminarea gaussiană cu pivotare parțială sau scalată pe coloană (la alegere) în MATLAB.
- ✓ **Problema 2** Să se implementeze descompunerea LUP. Să se scrie rutine pentru rezolvarea unui sistem folosind descompunerea LUP.
- ✓ **Problema 3** Generați sisteme cu matrice aleatoare nesingulare ce au soluția  $[1, \dots, 1]^T$ . Rezolvați-le cu eliminare gaussiană și descompunere LUP. nesingulare :  $\det \neq 0$
- ✓ **Problema 4** Să se scrie rutine pentru descompunerea Cholesky a unei matrice hermitiene și pozitiv definite și rezolvarea unui sistem cu o astfel de matrice prin descompunere Cholesky. Testați rutinele pentru matrice generate aleator și sisteme cu matrice aleatoare, dar cu soluție cunoscută.

✕ **Problema 5** *Rezolvați sistemul*

$$\begin{bmatrix} 1 & 0 & 0 & \dots & 1 \\ -1 & 1 & 0 & \dots & 1 \\ -1 & -1 & 1 & \dots & 1 \\ -1 & -1 & -1 & 1 & \dots & 1 \\ \vdots & & \ddots & \ddots & \vdots \\ -1 & \dots & -1 & -1 & 1 \end{bmatrix} x = \begin{bmatrix} 2 \\ 1 \\ 0 \\ -1 \\ \vdots \\ -n+2 \end{bmatrix}$$

*prin descompunere LUP și QR. Ce se observă? Explicați.*

## 5 Probleme suplimentare

**Problema 6** *Scrieți rutine pentru descompunerea LUP în care permutarea să se facă fizic și logic (cu vectori de permutări) și comparați timpii de execuție al ambelor variante pentru sisteme cu dimensiunea între 100 și 300.*

## Sist. de ec. lin. - Met. iterative

Exemplu:

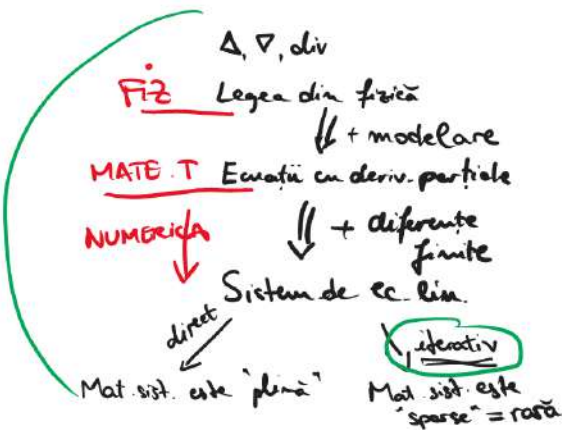
$$\begin{cases} 5x + 4y + 3z = 7 \\ 8x + 3y + z = 15 \\ 9x + y + 4z = 11 \end{cases}$$

$$\begin{cases} 5x_1 + 4x_2 + 3x_3 = 7 \\ 8x_1 + 3x_2 + x_3 = 15 \\ 9x_1 + x_2 + 4x_3 = 11 \end{cases}$$

$$\begin{cases} 9x_1 + x_2 + 4x_3 = 11 \\ 5x_1 + 4x_2 + 3x_3 = 7 \\ 8x_1 + 3x_2 + x_3 = 15 \end{cases}$$

$$\begin{cases} 9x_1 = 11 - x_2 - 4x_3 \\ 4x_2 = 7 - 5x_1 - 3x_3 \\ x_3 = 15 - 8x_1 - 3x_2 \end{cases}$$

$$\Rightarrow \begin{cases} x_1 = 11/9 - (1/9)x_2 - (4/9)x_3 \\ x_2 = 7/4 - (5/4)x_1 - (3/4)x_3 \\ x_3 = 15 - 8x_1 - 3x_2 \end{cases}$$



## Metoda lui Jacobi

- se alege o soluție inițială  $x^{(0)} = (x_1^{(0)}, x_2^{(0)}, x_3^{(0)})$ .
- se calculează  $x_1^{(1)}, x_2^{(1)}, x_3^{(1)}, \dots$  până când  $\|x^{(m+1)} - x^{(m)}\| < \epsilon$  unde  $\epsilon$  este dat.

$$x^{(0)} = (1, 1, 1)$$

Pe sist. de mai sus, iteratia Jacobi arată în felul următor:

$$\begin{cases} x_1^{(m+1)} = 11/9 - (1/9)x_2^{(m)} - (4/9)x_3^{(m)} \\ x_2^{(m+1)} = 7/4 - (5/4)x_1^{(m)} - (3/4)x_3^{(m)} \\ x_3^{(m+1)} = 15 - 8x_1^{(m)} - 3x_2^{(m)} \end{cases}$$

$$x_1^{(1)} = 11/9 - (1/9)x_2^{(0)} - (4/9)x_3^{(0)} = \frac{11}{9} - \frac{1}{9} - \frac{4}{9} = \frac{6}{9} = \frac{2}{3}$$

$$x_2^{(1)} = 7/4 - (5/4)x_1^{(0)} - (3/4)x_3^{(0)} = \frac{7}{4} - \frac{5}{4} - \frac{3}{4} = -\frac{1}{4}$$

$$x_3^{(1)} = 15 - 8x_1^{(0)} - 3x_2^{(0)} = 15 - 8 - 3 = 4$$

$$\|x^{(1)} - x^{(0)}\| = \left\| \left( \frac{2}{3}, -\frac{1}{4}, 4 \right) - (1, 1, 1) \right\| = \left\| \left( -\frac{1}{3}, -\frac{5}{4}, 3 \right) \right\| = \dots$$

$$\Rightarrow x^{(1)} = \left( \underline{\underline{\frac{2}{3}}}, \underline{\underline{-\frac{1}{4}}}, \underline{\underline{4}} \right)$$

$< \epsilon$   
STOP ALG.  
 $> \epsilon$   
calculați  $x^{(2)}$   
mai departe

## Metoda Gauss-Seidel

- se alege o sol. inițială  $x^{(0)} = (x_1^{(0)}, x_2^{(0)}, x_3^{(0)})$
- calculați  $x_1^{(1)}, x_2^{(1)}, x_3^{(1)}, \dots$  până când  $\|x^{(m+1)} - x^{(m)}\| < \epsilon$ , unde  $\epsilon$  este dat

$$x^{(0)} = (1, 2, 3)$$

Pe sist. de mai sus, iteratia Gauss-Seidel arată în felul următor:

$$\begin{cases} x_1^{(m+1)} = 11/9 - (1/9)x_2^{(m)} - (4/9)x_3^{(m)} \\ x_2^{(m+1)} = 7/4 - (5/4)x_1^{(m+1)} - (3/4)x_3^{(m)} \\ x_3^{(m+1)} = 15 - 8x_1^{(m+1)} - 3x_2^{(m+1)} \end{cases}$$

$$x_1^{(1)} = 11/9 - (1/9)x_2^{(0)} - (4/9)x_3^{(0)} = \frac{11}{9} - \frac{2}{9} - \frac{12}{9} = -\frac{3}{9} = -\frac{1}{3}$$

$$x_2^{(1)} = 7/4 - (5/4)x_1^{(1)} - (3/4)x_3^{(0)} = \frac{7}{4} - \frac{5}{4} \cdot \left(-\frac{1}{3}\right) - \frac{3}{4} \cdot 3 = \frac{7}{4} + \frac{5}{12} - \frac{9}{4} = \frac{5}{12} - \frac{2}{4} = -\frac{1}{6}$$

$$x_3^{(1)} = 15 - 8x_1^{(1)} - 3x_2^{(1)} = 15 - 8 \cdot \left(-\frac{1}{3}\right) - 3 \cdot \left(-\frac{1}{6}\right) = \dots$$

$$\Rightarrow x^{(1)} = \left( -\frac{1}{3}, -\frac{1}{6}, \dots \right)$$

$$\|x^{(1)} - x^{(0)}\| < \epsilon \text{ STOP PRINT } x^{(1)} \\ \not< \epsilon \text{ continuă cu } x^{(2)} \dots$$

# Sisteme liniare - metode iterative

Radu T. Trîmbițaș

21 martie 2022

## 1 Metode iterative staționare

Dorim să calculăm soluția sistemului

$$Ax = b, \quad (1)$$

când  $A$  este inversabilă. Presupunem că am găsit o matrice  $T$  și un vector  $c$  astfel încât  $I - T$  este inversabilă și punctul fix unic al ecuației

$$x = Tx + c \quad (2)$$

coincide cu soluția sistemului  $Ax = b$ . Fie  $x^*$  soluția lui (1) sau, echivalent, a lui (2).

Iterația:  $x^{(0)}$  dat; se definește  $(x^{(k)})$  prin

$$x^{(k+1)} = Tx^{(k)} + c, \quad k \in \mathbb{N}. \quad (3)$$

Metoda (3) este convergentă dacă și numai dacă  $\rho(T) < 1$ , unde  $\rho(T)$  este raza spectrală a lui  $T$ .

Criteriul de oprire este

$$\|x^{(k)} - x^{(k-1)}\| \leq \frac{1 - \|T\|}{\|T\|} \varepsilon. \quad (4)$$

Presupunem că putem descompune  $A$  sub forma  $A = M - N$ . Dacă  $M$  este ușor de inversat (diagonală, triunghiulară, ș.a.m.d.) este mai ușor să realizăm calculele în modul următor

$$Ax = b \Leftrightarrow Mx = Nx + b \Leftrightarrow x = M^{-1}Nx + M^{-1}b$$

Ultima ecuație este de forma  $x = Tx + c$ , unde  $T = M^{-1}N = I - M^{-1}A$ . Se obține șirul

$$x^{(k+1)} = M^{-1}Nx^{(k)} + M^{-1}b, \quad k \in \mathbb{N}, \quad (5)$$

unde  $x^{(0)}$  este un vector arbitrar. Considerăm descompunerea  $A = D - L - U$ , unde

$$(D)_{ij} = a_{ij}\delta_{ij}, \quad (-L)_{ij} = \begin{cases} a_{ij}, & i > j \\ 0, & \text{altfel} \end{cases}$$

$$(-U)_{ij} = \begin{cases} a_{ij}, & i < j \\ 0, & \text{altfel} \end{cases}$$

Pentru diverse alegeri ale lui  $M$  și  $N$  se obține:

- metoda lui Jacobi:  $M = D$ ,  $N = L + U$ . În acest caz,  $T = D^{-1}(L + U)$ ,  $c = D^{-1}b$ . Scrisă pe componente, metoda are forma

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left( b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k)} - \sum_{j=i+1}^n a_{ij}x_j^{(k)} \right)$$

- metoda Gauss-Seidel:  $M = D - L$ ,  $N = U$ . În acest caz,  $T = (D - L)^{-1}U$ ,  $c = (D - L)^{-1}b$ . Scrisă pe componente, metoda are forma

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left( b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^n a_{ij}x_j^{(k)} \right)$$

- metoda SOR (Successive OverRelaxation). În acest caz,  $M = \frac{D}{\omega} - L$ . Se obține

$$T = (D - \omega L)^{-1}((1 - \omega)D + \omega U),$$

$$c = \omega(D - \omega L)^{-1}b.$$

Pe componente,

$$x_i^{(k+1)} = (1 - \omega)x_i^{(k)} + \frac{\omega}{a_{ii}} \left( b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^n a_{ij}x_j^{(k)} \right).$$

Valoarea optimă a lui  $\omega$ , valabilă doar pentru anumite tipuri de matrice (tridiagonale, tridiagonal pe blocuri, ordonate consistent, etc.) este

$$\omega_O = \frac{2}{1 + \sqrt{1 - \rho^2}}, \quad (6)$$

unde  $\rho$  este raza spectrală a matricei metodei lui Jacobi.

## 2 Considerații de implementare

Dacă putem descompune  $A$  sub forma  $A = M - N$ , unde  $M$  este inversabilă, iterația (5) se scrie sub forma

$$x^{(k+1)} = M^{-1} (Nx^{(k)} + b), \quad k \in \mathbb{N}.$$

Folosind operatorul `\` din MATLAB putem scrie iterația sub forma

$$\mathbf{x} = \mathbf{M} \setminus (\mathbf{N} * \mathbf{x} + \mathbf{b})$$

unde  $\mathbf{x}$  este aproximația precedentă, iar  $\mathbf{x}$  aproximația curentă.

Dacă  $\varepsilon_a$  este eroarea absolută, iar  $\varepsilon_r$  eroarea relativă, criteriul de oprire se poate scrie sub forma (criteriu mixt)

$$\|x^{(k)} - x^{(k-1)}\| \leq \frac{1 - \|T\|}{\|T\|} \varepsilon_a + \varepsilon_r.$$

Pentru efectivitatea criteriului, cei doi termeni  $\frac{1 - \|T\|}{\|T\|} \varepsilon_a$  și  $\varepsilon_r$  trebuie să aibă același ordin de mărime. La implementarea criteriului de oprire pute lua în considerare și reziduul

$$r^{(k)} = b - Ax^{(k)}$$

și norma sa  $\|r^{(k)}\|$ . Alegerea unui criteriu de oprire eneral este o problemă dificilă.

Metodele iterative dau rezultate bune dacă sunt aplicate sistemelor mari cu matrice rară sau structurată.

## 3 Probleme

**Problema 1** *Implementați metoda lui Jacobi în MATLAB.*

**Problema 2** *Implementați metoda SOR în MATLAB. Găsiți  $\omega$  optim utilizând (6). Atenție: aceasta nu este o metodă practică pentru calculul lui  $\omega_o$ ; ea are numai scop didactic.*

**Problema 3** *Rezolvați sistemele:*

$$\begin{bmatrix} 5 & -1 & 0 & 0 & \dots & & 0 \\ -1 & 5 & -1 & 0 & \dots & & 0 \\ 0 & -1 & 5 & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & -\ddots & 0 & \vdots \\ \vdots & & \ddots & -1 & 5 & -1 & 0 \\ 0 & \dots & \dots & 0 & -1 & 5 & -1 \\ 0 & \dots & \dots & \dots & 0 & -1 & 5 \end{bmatrix} x = \begin{bmatrix} 4 \\ 3 \\ 3 \\ \vdots \\ 3 \\ 3 \\ 4 \end{bmatrix}$$

$$\begin{bmatrix} 5 & -1 & 0 & -1 & \dots & & \dots & 0 \\ -1 & 5 & -1 & 0 & -1 & & & \vdots \\ 0 & -1 & 5 & -1 & \ddots & \ddots & & \vdots \\ -1 & & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & -1 & \ddots & -1 & 5 & -1 & 0 & -1 \\ 0 & \dots & \ddots & 0 & -1 & 5 & -1 & 0 \\ 0 & \dots & & -1 & 0 & -1 & 5 & -1 \\ 0 & \dots & & & -1 & 0 & -1 & 5 \end{bmatrix} x = \begin{bmatrix} 3 \\ 2 \\ 2 \\ 1 \\ \vdots \\ 2 \\ 2 \\ 3 \end{bmatrix}.$$

cu toate metodele implementate.

**Problema 4** *Generați sisteme cu matrice diagonal dominante aleatoare ce au soluția  $[1, \dots, n]^T$  și rezolvați-le cu metodele Jacobi, Gauss-Seidel și SOR.*

## 4 Probleme suplimentare

**Problema 5** *Testați rutinele implementate pentru matrice rare de diverse dimensiuni și comparați timpii de execuție cu cei necesari pentru matrice dense.*

**Problema 6** *Pentru rutinele implementate generați curbe de convergență, adică curbe semilogaritmice care au pe abscisă numărul pasului curent, iar pe ordonată logaritmul normei reziduiului. (folosiți funcția MATLAB `semilogy`).*

## Interpolare Lagrange

anul	1960	1970	1980	1990	2000
pop. t�ntori �n Cluj	725.000	615.000	1.000.000	400.000	510.000

$$\text{pop}(2005) = ?$$

$$\text{pop}(1975) = ?$$

$$\text{pop}(2020) = ?$$

Poate fi prea complicat s  g sim func ia analitic  care modeleaz  evolu ia datelor noastre!

Alegem iar si polinoame.

Teorem  (Weierstrass) Fie  $f: [a, b] \rightarrow \mathbb{R}$  continu . Atunci  $\forall \epsilon > 0, \exists P(x)$  un polinom a.i.  $|f(x) - P(x)| < \epsilon, \forall x \in [a, b]$ .

Taylor: acurate ea era concentrat   n jurul punctului

Teorem  Fie  $x_0, \dots, x_m$ :  $m+1$  moduri distincte  i  $f$  o func ie pentru care  $f(x_0), \dots, f(x_m)$  sunt cunoscute. Atunci  $\exists!$  un polinom  $P(x)$  de grad cel mult  $m$  a.i.

$$P(x_k) = f(x_k), \forall k = 0, \dots, m$$

 i  $P$  este dat de:

$$P(x) = f(x_0) \cdot L_{m,0}(x) + \dots + f(x_m) \cdot L_{m,m}(x)$$

$$\text{unde } L_{m,k}(x) = \prod_{\substack{i=0 \\ i \neq k}}^m \frac{(x - x_i)}{(x_i - x_k)}$$

$P$   
polinomul  
de  
interpolare  
Lagrange.

Exemplu:  $x_0 = 2, x_1 = 2.75, x_2 = 4, f(x) = \frac{1}{x}$

Pol. Lagrange

$f(3)$  de aproximat.

$$\text{Rez. } f(x_0) = \frac{1}{2} = 0.5, f(x_1) = \frac{1}{2.75} = \frac{100}{275} = \frac{4}{11}, f(x_2) = \frac{1}{4}$$

noduri	2	2.75	4
valori	$\frac{1}{2}$	$\frac{4}{11}$	$\frac{1}{4}$

$$P(x) = f(x_0) \cdot L_{2,0}(x) + f(x_1) \cdot L_{2,1}(x) + f(x_2) \cdot L_{2,2}(x)$$

$$P(x) = \frac{1}{2} L_{2,0}(x) + \frac{4}{11} L_{2,1}(x) + \frac{1}{4} L_{2,2}(x)$$

$$L_{2,0}(x) = \prod_{\substack{i=0 \\ i \neq 0}}^2 \frac{(x - x_i)}{(x_i - x_0)} = \frac{(x - x_1)(x - x_2)}{(x_1 - x_0)(x_2 - x_0)} = \frac{(x - 2.75)(x - 4)}{(2.75 - 2)(4 - 2)} = \frac{x^2 - 6.75x + 11}{0.75 \cdot 2}$$

$$= \frac{x^2 - 6.75x + 11}{1.5} = \frac{10}{15} (x^2 - 6.75x + 11)$$

$$L_{2,1}(x) = \frac{x^2 - 6x + 8}{0.9375}$$

$$L_{2,2}(x) = \frac{x^2 - 4.75x + 5.5}{2.5}$$

$$P(x) = \frac{1}{2} \cdot \frac{10}{15} (x^2 - 6.75x + 11) + \frac{4}{11} \cdot \frac{1}{0.9375} (x^2 - 6x + 8) + \frac{1}{4} \cdot \frac{1}{2.5} (x^2 - 4.75x + 5.5)$$

$$P(x) = 0.33(x^2 - 6.75x + 11) + 0.38(x^2 - 6x + 8) + 0.1(x^2 - 4.75x + 5.5)$$

$$P(x) = 0.81x^2 - 4.98x + 7.22$$

$$P(2) = \frac{1}{2} = 0.5$$

$$P(3) \approx \frac{1}{3} = 0.333 \dots$$

$$P(3) = 0.81 \cdot 9 - 4.98 \cdot 3 + 7.22 = -0.43 \approx 0.33$$

Restul:  $f(x) = P(x) + \frac{f^{(m+1)}(\xi(x))}{(m+1)!} (x - x_0)(x - x_1) \dots (x - x_m)$

$$f \in C^{m+1} [a, b], \xi(x) \in [\min\{x_0, \dots, x_m\}, \max\{x_0, \dots, x_m\}]$$



# Interpolare Lagrange

Radu Trîmbițaș

27 martie 2021

## 1 Forma clasică

Fie  $f : [a, b] \rightarrow \mathbb{R}$ ,  $x_i \in [a, b]$ ,  $i = 0, \dots, m$ . Dacă  $x_i \neq x_j$ , pentru  $i \neq j$ , atunci există un polinom unic de gradul  $m$  (numit polinomul de interpolare Lagrange), astfel încât:

$$(L_m f)(x_i) = f(x_i), i = 0, \dots, m.$$

Formula de interpolare Lagrange este

$$f = L_m f + R_m f,$$

unde  $L_m$  este polinomul de interpolare Lagrange:

$$(L_m f)(x) = \sum_{k=0}^m \ell_k(x) f(x_k), \quad (1)$$

$\ell_k$  sunt polinoamele fundamentale de interpolare Lagrange

$$\ell_k(x) = \frac{\prod_{\substack{j=0 \\ j \neq k}}^m (x - x_j)}{\prod_{\substack{j=0 \\ j \neq k}}^m (x_k - x_j)}, \quad (2)$$

iar  $R_m$  este termenul rest:

$$(R_m f)(x) = \frac{(x - x_0) \dots (x - x_m)}{(m+1)!} f^{(m+1)}(x). \quad (3)$$

Dacă valorile funcției sunt tabelate, evaluarea lui  $\ell_k$  necesită  $2(n-1)$  înmulțiri, o împărțire și  $2n$  scăderi. Întreaga evaluare necesită  $2n(n+1)$  \* / și  $n(2n+3)$  +| -.

## 2 Algoritmul lui Aitken

Uneori gradul este necunoscut sau precizia dorită poate fi atinsă utilizând un număr mai mic de noduri. Să introducem notațiile:

$$\begin{aligned}(L_{m-1}f)_{1,m}(x) &= \sum_{k=1}^m \ell_k(x)f(x_k), \\ (L_{m-1}f)_{0,m-1}(x) &= \sum_{k=0}^{m-1} \ell_k(x)f(x_k), \\ (L_m f)_{0,m}(x) &= \sum_{k=0}^m \ell_k(x)f(x_k).\end{aligned}\tag{4}$$

Algoritmul lui Aitken se bazează pe relația

$$(L_m f)_{0,m}(x) = \frac{\begin{vmatrix} (L_{m-1}f)_{1,m}(x) & x_0 - x \\ (L_{m-1}f)_{0,m-1}(x) & x_m - x \end{vmatrix}}{x_m - x_0}.$$

Metoda generează tabela următoare:

$$\begin{array}{cccccc} x_0 & f_{0,0} & & & & \\ x_1 & f_{1,0} & f_{1,1} & & & \\ x_2 & f_{2,0} & f_{2,1} & f_{2,2} & & \\ \vdots & \vdots & \vdots & \vdots & \ddots & \\ x_i & f_{i,0} & f_{i,1} & f_{i,2} & \cdots & f_{i,i} \\ \vdots & \vdots & \vdots & \vdots & & \vdots & \ddots \\ x_n & f_{n,0} & f_{n,1} & f_{n,2} & \cdots & f_{n,i} & \cdots & f_{n,n} \end{array}$$

unde  $f_{i,0} = f(x_i)$ ,  $i = 0, \dots, m$ , și

$$f_{i,j+1} = \frac{1}{x_i - x_j} \begin{vmatrix} f_{j,j} & x_j - x \\ f_{i,j} & x_i - x \end{vmatrix}.\tag{5}$$

Se verifică ușor că  $(L_i f)(x) = f_{i+1,i+1}$ ,  $i = 0, \dots, n-1$ , datorită ecuației (??). Dacă interpolarea Lagrange converge, atunci  $(f_{i,i})_{i \in \mathbb{N}}$  converge către  $f(x)$  și  $|f_{i,i} - f_{i-1,i-1}| \rightarrow 0$  când  $i \rightarrow \infty$ , deci relația  $|f_{i,i} - f_{i-1,i-1}| \leq \varepsilon$  ar putea fi utilizată drept criteriu de oprire.

Algoritmul poate fi accelerat dacă sortăm nodurile crescător după distanța lor la  $x$ , i.e.  $|x_i - x| \leq |x_j - x|$ , dacă  $i < j$ .

Intrare:  $m \in \mathbb{N}$ ,  $x$ ,  $x_i$ ,  $f_i \in \mathbb{R}$ ,  $i = 0, \dots, m$ ,  $\varepsilon > 0$ .

Ieșire:  $f_{i,i}$ .

P1. Sortează  $x_i$  crescător după  $a_i = |x - x_i|$ .

P2. For  $i = 0, \dots, m$  set  $f_{i,1} := f(x_i)$ .

P3. For  $i = 1, \dots, m$  do

P3.1. For  $j = 0, \dots, i - 1$  do  
 $y_{i,j} := x_i - x_j$ ;  
 $f_{i,j+1} := ((x - x_i) * f_{jj} - (x - x_j) * f_{ij}) / y_{ij}$ ;  
P3.2. If  $|f_{i,i} - f_{i-1,i-1}| \leq \varepsilon$  go to P4.  
P4. Extrage  $f_{i,i}$ .

### 3 Interpolare Lagrange baricentrică

Forma clasică a interpolării Lagrange are următoarele dezavantaje:

1. fiecare evaluare a lui  $p(x)$  necesită  $\Theta(m^2)$  adunări și înmulțiri;
2. adăugarea unei noi perechi de date  $(x_{m+1}, f_{m+1})$  necesită reluarea tuturor calculelor.
3. procesul de calcul este numeric instabil.

Metoda lui Newton, odată ce s-a generat tabele de diferențe divizate, necesită un timp  $\Theta(m)$ , dar este instabilă.

#### 3.1 O formulă Lagrange îmbunătățită

Notăm cu  $f_i = f(x_i)$ . Vom rescrie formulele (1)+(2) astfel ca  $(L_m f)(x)$  să poată fi evaluat și actualizat cu  $O(m)$  operații. Introducând

$$\ell(x) = (x - x_0)(x - x_1) \cdots (x - x_m) \quad (6)$$

$\ell_j$  se poate scrie ca  $\ell_j(x) = \ell(x)/(x - x_j)$ . Definind ponderile baricentrice prin

$$w_j = \frac{1}{\prod_{k \neq j} (x_j - x_k)}, \quad j = 0, \dots, m, \quad (7)$$

adică,  $w_j = 1/\ell'(x_j)$ , putem scrie  $\ell_j$  sub forma

$$\ell_j(x) = \ell(x) \frac{w_j}{x - x_j}.$$

Acum PIL se scrie

$$(L_m f)(x) = \ell(x) \sum_{j=0}^m \frac{w_j}{x - x_j} f_j. \quad (8)$$

Avantajul este că putem calcula interpolantul Lagrange cu o formulă ce necesită  $O(m^2)$  flops pentru calculul unor cantități independente de  $x$ , numerele  $w_j$ , urmate de  $O(m)$  flops pentru evaluarea lui  $p$ , odată ce aceste numere sunt cunoscute.

Din (8) rezultă că actualizarea polinomului de interpolare la inserția unui nod nou necesită următoarele calcule:

- se împarte fiecare  $w_j$ ,  $j = 0..m$ , prin  $x_j - x_{m+1}$  (un flop pentru fiecare punct), cu un cost de  $m + 1$  flops;
- se calculează  $w_{m+1}$  cu formula (7) cu alte  $m + 1$  flops.

### 3.2 Metoda baricentrică

Interpolând funcția constantă 1 obținem

$$1 = \sum_{j=0}^m \ell_j(x) = \ell(x) \sum_{j=0}^m \frac{w_j}{x - x_j}. \quad (9)$$

Împărțind (8) cu expresia de mai sus și simplificând cu  $\ell(x)$ , obținem

$$p(x) = \frac{\sum_{j=0}^m \frac{w_j}{x - x_j} f_j}{\sum_{j=0}^m \frac{w_j}{x - x_j}}, \quad (10)$$

numită *formula baricentrică*.

La fel ca în (8), în (10) se poate adăuga o nouă pereche de date  $(x_{m+1}, f_{m+1})$  și actualiza  $w_j$  în  $O(m)$  flops.

### 3.3 Distribuții remarcabile

În cazul unor noduri particulare se pot da formule explicite pentru ponderile baricentrice  $w_j$ . Pentru noduri echidistante în intervalul  $[-1, 1]$ , la distanța  $h = 2/m$ , se obține  $w_j = (-1)^m \binom{m}{j} / (h^m m!)$ , care după anularea (simplificarea) factorilor independenți de  $j$  ne dă

$$w_j = (-1)^j \binom{m}{j}. \quad (11)$$

Același rezultat se obține și pentru un interval arbitrar  $[a, b]$ , deoarece formula originală pentru  $w_j$  se înmulțește cu  $2^m(b-a)^{-m}$ , dar acest factor poate fi înlăturat prin simplificare.

Familia de *puncte Cebîșev* se poate obține proiectând puncte egal spațiate pe cercul unitate pe intervalul  $[-1, 1]$ . Pornind de la formula

$$w_j = \frac{1}{\ell'(x_j)}, \quad (12)$$

se pot obține formule explicite pentru ponderile  $w_j$ .

*Punctele Cebîșev de speța I* sunt date de

$$x_j = \cos \frac{(2j+1)\pi}{2m+2}, \quad j = 0, \dots, m.$$

Simplificând factorii independenți de  $j$  se obține

$$w_j = (-1)^j \sin \frac{(2j+1)\pi}{2m+2}. \quad (13)$$

*Punctele Cebîșev de speța II* sunt date de

$$x_j = \cos \frac{j\pi}{m}, \quad j = 0, \dots, m,$$

iar ponderile corespunzătoare sunt

$$w_j = (-1)^j \delta_j, \quad \delta_j = \begin{cases} 1/2, & j = 0 \text{ sau } j = m, \\ 1, & \text{altfel.} \end{cases}$$

Dăm codul MATLAB pentru interpolarea Lagrange baricentrică.

Funcția `barycentricInterpolation` calculează valorile polinomului de interpolare Lagrange folosind metoda baricentrică.

```
function ff=barycentricInterpolation(x,y,xx,c)
%BARYCENTRICINTERPOLATION - barycentric Lagrange interpolation
%call ff=barycentricInterpolation(x,y,xx,c)
%x - nodes
%y - function values
%xx - interpolation points
%c - barycentric weights
%ff - values of interpolation polynomial

n=length(x)-1;
numer = zeros(size(xx));
denom = zeros(size(xx));
exact = zeros(size(xx)); %test if xx=nodes
for j=1:n+1
    xdiff = xx-x(j);
    temp = c(j)./xdiff;
    numer = numer+temp*y(j);
    denom = denom+temp;
    exact(xdiff==0) = j;
end
ff = numer ./ denom;
jj = find(exact);
ff(jj) = y(exact(jj));
```

Funcția `barycentricweights` calculează ponderile baricentrice.

```
function c = barycentricweights( x )
%BARYCENTRICWEIGHTS - compute barycentric weights(coefficients)
%call c = barycentricweights( x )
%x - nodes
%c - weights

n=length(x)-1;
c=ones(1,n+1);
```

```

for j=1:n+1
    c(j)=prod(x(j)-x([1:j-1,j+1:n+1]));
end
c=1./c;
end

```

În cazul nodurilor Cebîșev de speța a doua sursa MATLAB este

```

function ff=ChebLagrange(y,xx,a,b)
%CHEBLAGRANGE - Lagrange interpolation for Chebyshev #2 points- barycentric
%call ff=ChebLagrange(y,xx,a,b)
%y - function values;
%xx - evaluation points
%a,b - interval
%ff - values of Lagrange interpolation polynomial

n = length(y)-1;
if nargin==2
    a=-1; b=1;
end
c = [1/2; ones(n-1,1); 1/2].*(-1).^((0:n)');
x = sort(cos((0:n)'*pi/n))*(b-a)/2+(a+b)/2;
ff=barycentricInterpolation(x,y,xx,c);
end

```

### Probleme astea aplicate pe problemele practice de mai jos

- ✓ 1. Implementați o rutină pentru calculul valorilor polinomului de interpolare Lagrange când se dau punctele, nodurile și valorile funcției în noduri.
- ✓ 2. Reprezentați grafic polinoamele fundamentale când se dau gradul și nodurile.
- ✓ 3. Reprezentați pe același grafic  $f$  și  $L_m f$ .
- ✓ 4. Dându-se  $x$ ,  $f$ ,  $m$  și nodurile, aproximați  $f(x)$  utilizând interpolarea Lagrange. m = grad polinom
- ✓ 5. Implementați metoda baricentrică.

### Probleme practice

- ✓ 1. Datele de mai jos dau populația SUA în perioada 1900 – 2000 (în milioane de locuitori)

t	y
1900	75.995
1910	91.972
1920	105.711
1930	123.203
1940	131.669
1950	150.697
1960	179.323
1970	203.212
1980	226.505
1990	249.633
2000	281.422
2010	308.786

Approximați populația din 1975 și 2018.



2. Fie

$$f(x) = e^{x^2-1}.$$

Aproximați  $f(1.25)$  utilizând valorile lui  $f$  în 1, 1.1, 1.2, 1.3 și 1.4 și dați o delimitare a erorii.

3. Aproximați  $\sqrt{115}$  cu 3 zecimale exacte prin interpolare Lagrange.
4. Dați contraexemple pentru convergența interpolării Lagrange și studiați-le grafic:

- (a) contraexemplul lui Runge  $f : [-5, 5] \rightarrow \mathbb{R}$ ,  $f(x) = \frac{1}{1+x^2}$ ;
- (b) contraexemplul lui Bernstein  $g : [-1, 1] \rightarrow \mathbb{R}$ ,  $g(x) = |x|$ ;

ambele cu noduri echidistante și noduri Cebîșev de speța a doua.

5. Aproximați  $\sin 5^\circ$  și  $\cos 5^\circ$  folosind interpolarea Lagrange și tabela de valori

$x$	0	$\frac{\pi}{6}$	$\frac{\pi}{4}$	$\frac{\pi}{3}$	$\frac{\pi}{2}$
$\sin x$	0	$\frac{1}{2}$	$\frac{\sqrt{2}}{2}$	$\frac{\sqrt{3}}{2}$	1
$\cos x$	1	$\frac{\sqrt{3}}{2}$	$\frac{\sqrt{2}}{2}$	$\frac{1}{2}$	0

## Probleme suplimentare

1. Se consideră funcția cu valori complexe  $f : [-1, 1] \rightarrow \mathbb{C}$ , data de  $f(x) = (3 + \sin 10\pi x + \sin 61e^{0.8 \sin \pi x + 0.7}) e^{i\pi x}$ . Să se reprezinte grafic valorile polinomului său de interpolare Lagrange pentru 671 de noduri Cebîșev de speța a doua în cel puțin 10000 de puncte din  $[-1, 1]$ . (Indicație: `plot(z)` pentru  $z$  complex este echivalent cu `plot(real(z), imag(z))`.)

## Diferențe divizate și interpolare Hermite

### Diferențe divizate

$P_n(x)$  - polinomul de interpolare de grad  $\leq n$  pt care  $P(x_i) = f(x_i)$ ,  $i = \overline{0, n}$ .

! Cu toate că  $P_n$  este unic, reprezentarea algebrică nu este unică.

Ideea:  $P_n(x) = a_0 + a_1(x-x_0) + a_2(x-x_0)(x-x_1) + \dots + a_n(x-x_0)\dots(x-x_{n-1})$

Pentru găsirea constantelor o să fimuți cont de urm. rel.:

$$P_n(x_0) = a_0 = f(x_0)$$

Mai departe  $a_1 = ?$

$$P_n(x_1) = f(x_1) = a_0 + a_1(x_1 - x_0) \Rightarrow a_1 = \frac{f(x_1) - f(x_0)}{x_1 - x_0}$$

Atunci diferențele divizate sunt:

$$f[x_0, x_1] = f[x_1] - f[x_0] \text{ - dif. div. de ordin 0}$$

Definiții recurente:

$$f[x_0, x_1, x_2] = \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0} \text{ - dif. div. de ordin 1}$$

$$f[x_0, x_1, x_2, x_3] = \frac{f[x_1, x_2, x_3] - f[x_0, x_1, x_2]}{x_3 - x_0} \text{ - dif. div. de ord. 2}$$

$$f[x_0, \dots, x_{i+k}] = \frac{f[x_1, \dots, x_{i+k}] - f[x_0, \dots, x_{i+k-1}]}{x_{i+k} - x_0} \text{ - dif. div. de ord. } k$$

$$f[x_0, \dots, x_n] = \frac{f[x_1, \dots, x_n] - f[x_0, \dots, x_{n-1}]}{x_n - x_0} \text{ - dif. div. de ord. } n$$

Newton (ND):  $P_n(x) = f[x_0] + \sum_{k=1}^n f[x_0, \dots, x_k] (x - x_0) \dots (x - x_{k-1})$

Pentru calcul dif. div. se folosește urm. schemă / tabel:

$x_0$	$f[x_0]$	$f[x_0, x_1] = \frac{f[x_1] - f[x_0]}{x_1 - x_0}$	$f[x_0, x_1, x_2] = \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0}$
$x_1$	$f[x_1]$	$f[x_1, x_2]$	$f[x_1, x_2, x_3]$
$x_2$	$f[x_2]$	$f[x_2, x_3]$	$f[x_2, x_3, x_4]$
$x_3$	$f[x_3]$	$f[x_3, x_4]$	$f[x_3, x_4, x_5]$
$x_4$	$f[x_4]$	$f[x_4, x_5]$	
$x_5$	$f[x_5]$		

$x$	$x_0$	$x_1$	$x_2$
$f(x)$	1	0.5	0.5

$N_n(x) = ?$

$$N_2(x) = f[x_0] + \sum_{k=1}^2 f[x_0, \dots, x_k] (x - x_0) \dots (x - x_{k-1})$$

$$= f[x_0] + f[x_0, x_1] (x - x_0) + f[x_0, x_1, x_2] (x - x_0) (x - x_1)$$

$$= 1 + f[x_0, x_1] (x - 1) + f[x_0, x_1, x_2] (x - 1) (x - 2)$$

$x_k$	$f(x_k)$	$f[x_0, x_1] = \frac{f(x_1) - f(x_0)}{x_1 - x_0} = \frac{0.5 - 1}{2 - 1} = -0.5$	$f[x_0, x_1, x_2] = \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0} = \frac{0.33 - (-0.5)}{3 - 1} = \frac{0.83}{2} = 0.415$
1	0.5		
2	0.33		
3	0.33		

$$= 1 + 0.5(x - 1) + 0.415(x - 1)(x - 2)$$

### Interpolarea Hermite

**Teoremă** De:  $f \in C^1[a, b]$ ,  $x_0, \dots, x_m \in [a, b]$  distincte, at. unicul polinom  $P$  de grad minim pentru care:

$$P(x_i) = f(x_i) \text{ și } P'(x_i) = f'(x_i), \text{ } i = \overline{0, m}$$

Este polinomul Hermite de grad  $2m+1$ , dat de:

$$P(x) = \sum_{j=0}^m f(x_j) H_{m,j}(x) + \sum_{j=0}^m f'(x_j) \hat{H}_{m,j}(x)$$

$$\text{unde } H_{m,j}(x) = [1 - 2(x - x_j)] L_{m,j}'(x_j) L_{m,j}^1(x_j)$$

$$\hat{H}_{m,j}(x) = (x - x_j) L_{m,j}^2(x_j)$$

Hermite cu diferențe divizate:

$$H_{2m+1}(x) = f[x_0] + \sum_{k=1}^{2m+1} f[x_0, \dots, x_k] (x - x_0) \dots (x - x_{k-1})$$

unde, ipse exemplul, schema de calcul este:

$(x_0, \dots, x_k)$

$x$	$0$			
$x_0 = x_0$	$f(x_0)$	$f[x_0, x_1]$	$f[x_0, x_1, x_2]$	$f[x_0, x_1, x_2, x_3]$
$x_1 = x_0$	$f'(x_0)$	$f[x_1, x_2]$	$f[x_1, x_2, x_3]$	
$x_2 = x_1$	$f(x_1)$	$f[x_2, x_3]$	$f[x_2, x_3, x_4]$	
$x_3 = x_1$	$f'(x_1)$	$f[x_3, x_4]$		
$x_4 = x_2$	$f(x_2)$	$f[x_4, x_5]$		
$x_5 = x_2$	$f'(x_2)$			



# Metode de interpolare bazate pe diferențe divizate

Radu Trîmbițaș

April 4, 2021

## 1 Forma Newton a polinomului de interpolare Lagrange

Algoritmul nostru se bazează pe forma Newton a polinomului de interpolare Lagrange:

$$(N_m f)(x) = f[x_0] + \sum_{k=1}^m f[x_0, \dots, x_k] \prod_{i=0}^{k-1} (x - x_i), \quad (1)$$

și formula iterativă:

$$\begin{aligned} (N_k f)(x) &= (N_{k-1} f)(x) + (x - x_0) \dots (x - x_{k-1}) f[x_0, \dots, x_k], & k = 1, \dots, m, \\ (N_0 f)(x) &= f(x_0). \end{aligned}$$

Notăția  $f[x_0, \dots, x_k]$  înseamnă diferența divizată de ordinul  $k$  a funcției  $f$  cu nodurile  $x_0, \dots, x_k$ .

Calculul diferențelor divizate se poate face în formă tabelară cu algoritmul:

**Intrare:**  $x_0, x_1, \dots, x_m, f(x_0), f(x_1), \dots, f(x_m)$ , ca primă coloană  $Q_{0,0}, Q_{1,0}, \dots, Q_{n,0}$  a tabelului  $Q$ .

**Ieșire:** numerele  $Q_{0,0}, Q_{1,1}, \dots, Q_{n,n}$ , unde  $Q_{i,i} = f[x_0, \dots, x_i]$ .

**P1.** for  $i = 1, 2, \dots, n$  do  
    for  $j = 1, 2, \dots, i$  do

$$Q_{i,j} := \frac{Q_{i,j-1} - Q_{i-1,j-1}}{x_i - x_{i-j}}, \quad (Q_{i,j} = f[x_{i-j}, \dots, x_i].)$$

**P2.** returnează  $Q_{0,0}, Q_{1,1}, \dots, Q_{n,n}$ .

**Exemplul 1.** Fie funcția dată mai jos (tabela 1); dorim să aproximăm  $f(1.5)$  utilizând polinomul de interpolare Newton. Diferențele divizate apar pe prima linie a tabelului. Polinomul Newton corespunzător este

$$\begin{aligned} (N_4 f)(x) &= 0.7651977 - 0.4837057(x - 1.0) - 0.1087339(x - 1.0)(x - 1.3) + \\ &\quad 0.0658784(x - 1.0)(x - 1.3)(x - 1.6) + \\ &\quad 0.0018251(x - 1.0)(x - 1.3)(x - 1.6)(x - 1.9). \end{aligned}$$

$x_i$	$f(x_i)$	$\mathcal{D}^1$	$\mathcal{D}^2$	$\mathcal{D}^3$	$\mathcal{D}^4$
1.0	0.7651977	-0.4837057	-0.1087339	0.0658784	0.0018251
1.3	0.6200860	-0.5489460	-0.0494433	0.0680685	0.0000000
1.6	0.4554022	-0.5786120	0.0118183	0.0000000	0.0000000
1.9	0.2818186	-0.5715210	0.0000000	0.0000000	0.0000000
2.2	0.1103623	0.0000000	0.0000000	0.0000000	0.0000000

Table 1: Date pentru exemplul 1

Se verifică ușor că  $(N_4 f)(1.5) = 0.5118200$ .

Într-o implementare practică a algoritmului este convenabil să se sorteze în prealabil nodurile.

**Intrare:**  $x_0, x_1, \dots, x_m, f(x_0), f(x_1), \dots, f(x_m), x$ .

**Ieșire:** Valoarea  $P_i$  a polinomului de interpolare Newton.

**P1.** Sortează  $x_i$  crescător după  $a_i = |x_i - x|$ .

**P2.**  $D_{0,0} := f(x_i); P_0 := D_{0,0}; S_1 := 1;$

**P3.** for  $i = 1, \dots, m$  do

**P3.1.**  $D_{i,1} := f(x_i);$

**P3.2.** for  $j = 0, \dots, i - 1$  do  $y_{i,j} := x_i - x_j$

**P3.3.** for  $j = 1, \dots, i$  do

$$D_{i,j} := \frac{D_{i,j-1} - D_{i-1,j-1}}{y_{i,i-j+1}}.$$

**P3.4.**  $S_i := S_{i-1} * a_{i-1}; P_i := P_{i-1} + S_i * D_{i,1};$

**P3.5.** if  $|P_i - P_{i-1}| < \varepsilon$  go to P4.

**P4.** returnează  $P_i$ .

Se poate obține o implementare satisfăcătoare, dacă se renunță la sortare și la if-ul final, iterând până la sfârșit după  $i$  și  $j$ .

## 2 Interpolare Hermite

Fie  $x_k \in [a, b]$ ,  $k = 0, \dots, m$ ,  $x_i \neq x_j$  pentru  $i \neq j$ ,  $r_k \in \mathbb{N}$ ,  $k = 0, \dots, m$ , și  $f : [a, b] \rightarrow R$  astfel încât  $\exists f^{(j)}(x_k)$ ,  $k = 0, \dots, m$ ,  $j = 0, \dots, r_k$ . Problema de interpolare Hermite cere determinarea unui polinom  $P$  de grad minim, care verifică

$$P^{(j)}(x_k) = f^{(j)}(x_k), \quad k = 0, \dots, m, j = 0, \dots, r_k.$$

Fie  $n + 1 = m + r_0 + \dots + r_m = (r_0 + 1) + \dots + (r_m + 1)$ . Polinomul de interpolare Hermite are expresia:

$$(H_n f)(x) = \sum_{k=0}^m \sum_{j=0}^{r_k} h_{kj}(x) f^{(j)}(x_k), \quad (2)$$

unde  $h_{kj}$  sunt polinoamele fundamentale Hermite date de

$$h_{kj}(x) = \frac{(x - x_k)^j}{j!} u_k(x) \sum_{\nu=0}^{r_k-j} \frac{(x - x_k)^\nu}{\nu!} \left[ \frac{1}{u_k(x)} \right]_{x=x_k}^{(\nu)}, \quad (3)$$

și

$$u(x) = \prod_{k=0}^m (x - x_k)^{r_k},$$

$$u_k(x) = \frac{u(x)}{(x - x_k)^{r_k}}.$$

Expresia restului este:

$$(R_n f)(x) = \frac{u(x)}{(n+1)!} f^{(n+1)}(\xi), \quad \xi \in (\alpha, \beta)$$

unde  $\alpha = \min(x, x_0, \dots, x_m)$ ,  $\beta = \max(x, x_0, \dots, x_m)$ .

Vom da o metodă de calcul a polinomului de interpolare Hermite cu noduri duble dedusă din polinomul de interpolare Newton. Dându-se  $x_i$  și  $f(x_i)$ ,  $f'(x_i)$ ,  $i = 0, \dots, m$ , definim secvența  $z_0, z_1, \dots, z_{2m+1}$

$$z_{2i} = z_{2i+1} = x_i, \quad i = 0, \dots, m.$$

Construim acum tabela de diferențe divizate a lui  $f$  pentru nodurile  $z_i$ ,  $i = 0, \dots, 2m+1$ . Deoarece  $z_{2i} = z_{2i+1} = x_i$  pentru  $i = 0, \dots, m$ ,  $f[z_{2i}, z_{2i+1}]$  este o diferență divizată cu noduri duble și este egală cu  $f'(x_i)$ , deci în locul diferențelor divizate de ordinul I  $f[z_0, z_1]$ ,  $f[z_2, z_3]$ , ...,  $f[z_{2m}, z_{2m+1}]$  vom utiliza derivatele  $f'(x_0)$ ,  $f'(x_1)$ , ...,  $f'(x_m)$ . Celelalte diferențe divizate se obțin în mod obișnuit, așa cum arată tabela de mai jos. Acest algoritm se poate extinde și la alte tipuri de interpolare Hermite. Se pare că metoda este datorată lui Powell.

$z$	$\mathcal{D}^0$	$\mathcal{D}^1$	$\mathcal{D}^2$
$z_0 = x_0$	$f[z_0] = f(x_0)$	$f[z_0, z_1] = f'(x_0)$	$f[z_0, z_1, z_2]$
$z_1 = x_0$	$f[z_1] = f(x_0)$	$f[z_1, z_2]$	$f[z_1, z_2, z_3]$
$z_2 = x_1$	$f[z_2] = f(x_1)$	$f[z_2, z_3] = f'(x_1)$	$f[z_2, z_3, z_4]$
$z_3 = x_1$	$f[z_3] = f(x_1)$	$f[z_3, z_4]$	$f[z_3, z_4, z_5]$
$z_4 = x_2$	$f[z_4] = f(x_2)$	$f[z_4, z_5] = f'(x_2)$	
$z_5 = x_2$	$f[z_5] = f(x_2)$		

**Algoritm de interpolare Hermite.** Calculează coeficienții polinomului de interpolare Hermite (diferențele divizate) a lui  $f$  cu nodurile duble  $x_k \in [a, b]$ ,  $k = 0, \dots, m$ .

**Intrare.**  $x_0, \dots, x_m, f(x_0), \dots, f(x_m), f'(x_0), \dots, f'(x_m)$ .

**Ieșire.** Numerele  $Q_{0,0}$ ,  $Q_{1,1}$ , ...,  $Q_{2m+1,2m+1}$  (diferențele divizate de pe prima linie a tablei), unde

$$(H_{2m+1}f)(x) = Q_{0,0} + Q_{1,1}(x - x_0) + Q_{2,2}(x - x_0)^2 + Q_{3,3}(x - x_0)^2(x - x_1) + Q_{4,4}(x - x_0)^2(x - x_1)^2 + \dots + Q_{2m+1,2m+1}(x - x_0)^2(x - x_1)^2 \dots (x - x_{m-1})^2(x - x_m).$$

**P1.** for  $i = 0, \dots, m$  do pașii P2 și P3.  
**P2.**  $z_{2i} := x_i; z_{2i+1} := x_i; Q_{2i,0} := f(x_i); Q_{2i+1,0} := f(x_i); Q_{2i+1,1} := f'(x_i);$   
**P3.** if  $i \neq 0$  then  $Q_{2i,1} = \frac{Q_{2i,0} - Q_{2i-1,0}}{z_{2i} - z_{2i-1}}.$   
**P4.** for  $i = 2, 3, \dots, 2m+1$  do  
for  $j=2,3,\dots, i$  do  $Q_{i,j} = \frac{Q_{i,j-1} - Q_{i-1,j-1}}{z_i - z_{i-j}}.$   
**P5.** Extrage  $Q_{0,0}, Q_{1,1}, \dots, Q_{2m+1,2m+1}$ . Stop.

**Exemplu.** Fie datele de intrare

$k$	$x$	$f(x)$	$f'(x)$
0	1.3	0.6200860	-0.5220232
1	1.6	0.4554022	-0.5698959
2	1.9	0.2818186	-0.5811571

Utilizând diferențele divizate se obține(datele de intrare sunt subliniate):

$$\begin{aligned}
H_5(1.5) &= 0.6200860 + (1.5 - 1.3)(-0.5220232) + (1.5 - 1.3)^2(-0.897427) + \\
&\quad (1.5 - 1.3)^2(1.5 - 1.6)(0.0663657) + \\
&\quad (1.5 - 1.3)^2(1.5 - 1.6)^2(1.5 - 1.9)(-0.0027738) \\
&= 0.5118277
\end{aligned}$$

1.3	<u>0.6200860</u>	<u>-0.5220232</u>	-0.0897427	0.0663657	0.0026663	-0.0027738
1.3	<u>0.6200860</u>	-0.5489460	-0.0698330	0.0679655	0.0010020	
1.6	<u>0.4554022</u>	<u>-0.5698959</u>	-0.0290937	0.0685667		
1.6	<u>0.4554022</u>	-0.5786120	-0.0084837			
1.9	<u>0.2818186</u>	<u>-0.5811571</u>				
1.9	<u>0.2818186</u>					

### 3 Probleme

1. Implementați o rutină pentru calculul valorilor polinomului de interpolare Hermite cu noduri duble, dându-se punctele în care se face evaluarea, nodurile, valorile funcției și ale derivatei în noduri.
2. Reprezentați pe același grafic  $f$  și polinomul său de interpolare Hermite.
3. Scrieți o rutină care reprezintă grafic o cubică parametrică Hermite (o curbă care trece prin două puncte date și are în acele puncte tangente date).

## 4 Probleme practice

1. Pentru  $f(x) = e^x$  și nodurile de interpolare 0, 1, 2, aproximați  $f(0.25)$  prin interpolare Hermite și comparați rezultatul cu cel obținut prin interpolare Lagrange. Dați o delimitare a erorii. Comparați cu rezultatul furnizat de software-ul utilizat.
2. Utilizați valorile date mai jos pentru a aproxima  $\sin 0.34$  utilizând interpolarea Hermite.

$x$	$\sin x$	$(\sin x)' = \cos x$
0.30	0.29552	0.95534
0.32	0.31457	0.94924
0.35	0.34290	0.93937

Dați o delimitare a erorii și comparați-o cu eroarea exactă. Adăugați datele pentru nodul  $x = 0.33$  și refaceți calculele.

3. Un automobil care se deplasează pe un drum drept este cronometrat în mai multe puncte. Datele de observație se dau în tabela de mai jos. Utilizați interpolarea Hermite pentru a prevedea poziția și viteza automobilului la momentul  $t = 10$ .

Timpul	0	3	5	8	13
Distanța	0	225	383	623	993
Viteza	75	77	80	74	72

## 5 Probleme suplimentare

1. Implementați interpolarea Hermite pentru noduri de multiplicitate arbitrară. Intrare: nodurile  $x_k$ , multiplicitățile  $r_k$ , valorile  $f^{(j)}(x_k)$ ,  $k = 0, \dots, m$ ,  $j = 0, \dots, r_k$ , punctul (sau punctele)  $t$  în care se face evaluarea. Ieșire: valorile polinomului de interpolare în punctul (sau punctele)  $t$ . (Indicație: folosiți pentru valorile funcție și derivatelor un tablou de celule.)

## Interpolare Spline

De ce? Polinoame de grad mare - fluctuatii & pe un subinterval  $\rightarrow$   
 $\rightarrow$  fluctuatii mari pe tot intervalul.

Pt. a interpolare: împartim intervalul în subintervale - aprox. pe  $f$  pe fiecare subinterval cu un anumit polinom.

- alegem  $p(x) = ax+b$ , atunci functia de interpolare  $P$  nu e metoda ( $P \in C^1$  de obicei)



- alegem Hermite: mod.  $f, f'$ , calc.  $H_3$ , calc.  $L_0, \dots$   
 însă de obicei nu se cunoaște  $f'$

Soluție: **Spline cubic** - un polinom de gradul 3 (pe porțiuni)

$\Downarrow$   
 4 constante  
 + (Av:  $f$  care interpoalează  
 este chiar de clasă  $C^3$ )

Obs. Construcția nu cere ca deriv.  $f$  de interpolare să fie aceeași  
 cu deriv. funcției pe care o aprox. (nici măcar pe noduri).

Def.  $f: [a, b] \rightarrow \mathbb{R}$  unde  $a = x_0 < x_1 < \dots < x_n = b$  noduri. Spunem  
 că funcția  $S$  este interpolant spline cubic, dacă  $S$  satisface  
 simultan urm. cond:

- (a)  $S(x)$  este un polinom cubic notat  $S_j(x)$  pe  $[x_j, x_{j+1}]$   
 $\forall j = \overline{0, n-1}$ .

$$S(x) = \begin{cases} S_0 & [x_0, x_1] \\ S_1 & [x_1, x_2] \\ \vdots & \vdots \\ S_{n-1} & [x_{n-1}, x_n] \end{cases}$$

- (b)  $S_j(x_j) = f(x_j)$  și  $S_j(x_{j+1}) = f(x_{j+1})$ ,  $\forall j = \overline{0, n-1}$

- (c)  $S_{j+1}(x_{j+1}) = S_j(x_{j+1})$ ,  $\forall j = \overline{0, n-2}$

- (d)  $S'_{j+1}(x_{j+1}) = S'_j(x_{j+1})$ ,  $\forall j = \overline{0, n-2}$

- (e)  $S''_{j+1}(x_{j+1}) = S''_j(x_{j+1})$ ,  $\forall j = \overline{0, n-2}$

(f) unul dintre următoarele seturi de condiții pe  $f$ , să fie  
 satisfăcut:

(i)  $S''(x_0) = S''(x_n) = 0$  (naturală)

(ii)  $S'(x_0) = f'(x_0)$  și  $S'(x_n) = f'(x_n)$  (clamped)

Naturală - natural spline - a lung flexible rod

Clamped - mai multă informație despre funcția, dar ea să  
 aprox. cu clamped spline, la capete ai nevoie de  
 $f'(x_0), f'(x_n)$  sau de aprox.  $f$  bune ale  
 acestora.

Exemplu: Să se construiască spline-ul natural care trece prin  
 punctele  $(1, 2), (2, 3)$  și  $(3, 5)$ .

$$\begin{array}{lcl} \text{Rez.} & x_0=1 & f(x_0)=2 \\ & x_1=2 & f(x_1)=3 \\ & x_2=3 & f(x_2)=5 \end{array}$$

Funcția căutată  $S$  are 2 componente cubice (doar 2 intervale  
 $[1, 2]$  și  $[2, 3]$ ).

General:  $S_j(x) = a_j + b_j(x-x_j) + c_j(x-x_j)^2 + d_j(x-x_j)^3$ ,  $j = \overline{0, n-1}$

$$\text{Pt. noi: } S(x) = \begin{cases} S_0(x), & x \in [1, 2] \\ S_1(x), & x \in [2, 3] \end{cases}$$

$$S_0(x) = a_0 + b_0(x-1) + c_0(x-1)^2 + d_0(x-1)^3$$

$$S_1(x) = a_1 + b_1(x-2) + c_1(x-2)^2 + d_1(x-2)^3$$

$a_0, \dots, d_1 = 8$  constante de găsit  $\Rightarrow$  8 condiții sau relații  
 necesare.

$$S_0(x_0) = f(x_0) \Rightarrow a_0 = 2$$

$$S_0(x_1) = f(x_1) \Rightarrow a_0 + b_0 + c_0 + d_0 = 3$$

$$S_1(x_1) = f(x_1) \Rightarrow a_1 = 3$$

$$S_1(x_2) = f(x_2) \Rightarrow a_1 + b_1 + c_1 + d_1 = 5$$

$$\bullet \text{ deriv de ord 1: } S'_0(2) = S'_1(2) \Rightarrow b_0 + 2c_0 + b_1 = b_1$$

$$\bullet \text{ deriv de ord 2: } S''_0(2) = S''_1(2) \Rightarrow 2c_0 + 6d_0 = 2c_1$$

$$\bullet \text{ naturală: } S''_0(1) = 0 \Rightarrow 2c_0 = 0$$

$$S''_1(3) = 0 \Rightarrow 2c_1 + 6d_1 = 0$$

$$S(x) = \begin{cases} 2 + \frac{3}{4}(x-1) + \frac{1}{4}(x-1)^3, & x \in [1, 2] \\ 3 + \frac{3}{2}(x-2) + \frac{3}{4}(x-2)^2 - \frac{1}{4}(x-2)^3, & x \in [2, 3] \end{cases}$$

ex. clamped spline,  $S'(1)=2$ ,  $S'(3)=1$ .

Tema: implementați pt. toate cele 4 metode de  
 calcul plus exemple.

toate ex. de la Probleme 1-3.

# Interpolare spline

Radu T. Trîmbițaș

Cele 4 metode + pb 1, 2, 3

12 aprilie 2021

Implementați următoarele tipuri de spline cubice: spline complete, spline care reproduc derivatele de ordinul al doilea, spline naturale și spline deBoor.

Pentru algoritmi a se vedea notele de curs sau slide-urile.

## Probleme

1. Pentru fiecare tip de spline scrieți o funcție care calculează coeficienții spline-ului, dacă se dau nodurile și valorile funcției.
2. Evaluați spline-ul pe o mulțime de puncte, dacă se dau nodurile, punctele și coeficienții.
3. Desenați o curbă spline cubică parametrică ce trece printr-o mulțime de puncte date.

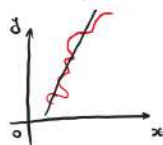
## Probleme suplimentare

1. Scrieți o funcție MATLAB pentru calculul coeficienților unui spline periodic de clasă  $C^2[a, b]$ . Aceasta înseamnă că datele trebuie să verifice  $f_n = f_1$  și că interpolantul rezultat trebuie să fie periodic, de perioadă  $x_n - x_1$ . Condițiile de periodicitate de la capete se pot impune mai ușor considerând două puncte suplimentare  $x_0 = x_1 - \Delta x_{n-1}$  și  $x_{n+1} = x_n + \Delta x_1$ , în care funcția să ia valorile  $f_0 = f_{n-1}$  și respectiv  $f_{n+1} = f_2$ .
2. Scrieți o funcție MATLAB pentru calculul coeficienților unui spline Hermite de clasă  $C^1[a, b]$ . Interpolantul va trebui să verifice condițiile  $s_3(f; x_i) = f(x_i)$  și  $s'_3(f; x_i) = f'(x_i)$ ,  $i = 1, \dots, n$ .

Metoda celor mai mici pătrate

$x_i$	$y_i$

reprezentare grafică



→ pare că dreapta aprox bine însă un polinom care să interpoleze nodurile și replică val. fct. pe noduri introduce osculații care nu sunt prezente în date.

Abordarea: Găsește "cea mai bună" dreaptă de aproximare pentru datele noastre. (chiar dacă nu are exact aceeași valoare pe date)

$$y_i \approx \underbrace{a_1}_{\text{m.c.}} x_i + \underbrace{a_0}_{\text{m.c.}}$$

În loc să folosesc  $|x - y|$  vom folosi  $(x - y)^2$

$$E(a_0, a_1) = \sum_{i=1}^m (y_i - (a_1 x_i + a_0))^2 \longrightarrow \min$$

Rezolvând ac. pb. de minimizare, se obține:

$$a_0 = \frac{\sum_{i=1}^m x_i^2 \sum_{i=1}^m y_i - \sum_{i=1}^m x_i y_i \sum_{i=1}^m x_i}{m(\sum_{i=1}^m x_i^2) - (\sum_{i=1}^m x_i)^2}$$

$$a_1 = \frac{m \sum_{i=1}^m x_i y_i - \sum_{i=1}^m x_i \sum_{i=1}^m y_i}{m(\sum_{i=1}^m x_i^2) - (\sum_{i=1}^m x_i)^2}$$

$$y = a_1 x + a_0$$

dr. căutată.

Obs.

$x_i$	$y_i$	$x_i y_i$	$x_i^2$

$\sum$

MCMMP pt. polinoame

$x_i$	$y_i$

$P_m$  - polinom care aproximează datele a.î.  $m < m-1$ .

$$P_m(x) = a_m x^m + a_{m-1} x^{m-1} + \dots + a_1 x + a_0$$

Căutăm constantele  $a_0, \dots, a_m$  a.î.

$$E(a_0, \dots, a_m) = E = \sum_{i=1}^m (y_i - P_m(x_i))^2 \longrightarrow \min$$

Rezolvând pb. de minimizare, avem că  $a_0, \dots, a_m$  sunt sol. sistemului:

$$\begin{cases} a_0 \sum_{i=1}^m x_i^0 + a_1 \sum_{i=1}^m x_i^1 + a_2 \sum_{i=1}^m x_i^2 + \dots + a_m \sum_{i=1}^m x_i^m = \sum_{i=1}^m y_i x_i^0 \\ a_0 \sum_{i=1}^m x_i^1 + a_1 \sum_{i=1}^m x_i^2 + a_2 \sum_{i=1}^m x_i^3 + \dots + a_m \sum_{i=1}^m x_i^{m+1} = \sum_{i=1}^m y_i x_i^1 \\ \vdots \\ a_0 \sum_{i=1}^m x_i^m + a_1 \sum_{i=1}^m x_i^{m+1} + a_2 \sum_{i=1}^m x_i^{m+2} + \dots + a_m \sum_{i=1}^m x_i^{2m} = \sum_{i=1}^m y_i x_i^m \end{cases}$$

Sistemul are sol. unică dacă  $x_i$  sunt distincte.

- MCMMP pentru aprox. funcțiilor

$f \in C[a, b]$  ce trebuie aproximat

Obiectivul: găsirea unui polinom de un grad  $\leq m$  a.î.

$$\int_a^b (f(x) - P_m(x))^2 dx \longrightarrow \min$$

$$E(a_0, \dots, a_m) = \int_a^b (f(x) - \sum_{k=0}^m a_k x^k)^2 dx \longrightarrow \min$$

În urma rezolvării acestei probleme de minimizare, se obțin ec.:

$$\sum_{k=0}^m a_k \int_a^b x^{j+k} dx = \int_a^b x^j f(x) dx, \quad \forall j=0, \dots, m$$

( $m+1$ ) - ec. :  $a_0, \dots, a_m$

Sist. de ec. are sol. unică dacă  $f \in C[a, b]$

□ sistemul are sol. numerică mai greu de calculat

□ calculul lui  $P_m$  nu minimizează dificultatea calculului lui  $P_{m+1}$

Vrem să aproximăm cât mai general o funcție  $f$  pe  $[a, b]$ .

Def. O funcție  $w: I \rightarrow \mathbb{R}$  s.m. fct. pondere dacă

$$w(x) \geq 0 \text{ pe } I$$

$$w(x) \neq 0 \text{ pe orice subinterval al lui } I.$$

$$\begin{aligned} & \varphi_1, \varphi_2, \dots, \varphi_n \\ & \lambda_1, \lambda_2, \dots, \lambda_n \\ & \forall \lambda_1, \lambda_2, \dots, \lambda_n \in \mathbb{R} \quad \lambda_1 \varphi_1 + \lambda_2 \varphi_2 + \dots + \lambda_n \varphi_n = 0 \\ & \Rightarrow \alpha_1 = \dots = \alpha_n = 0 \end{aligned}$$

$$\text{Ex. } w \equiv 1, \quad w(x) = \frac{1}{\sqrt{1-x^2}}, \quad x \in (-1, 1)$$

Fie  $\{\phi_1, \dots, \phi_n\}$  o mult. de fct. lin. indep. pe  $[a, b]$ .

$$\text{Ex. } \{1, x, x^2, \dots, x^n\} \text{ - l.i. pe } [a, b].$$

Fie  $w$  o fct. pondere pe  $[a, b]$

Obiectivul: pentru  $f \in C[a, b]$ , căutăm  $P(x) = \sum_{k=0}^m a_k \phi_k(x)$  a.î.

$$E = E(a_0, \dots, a_m) = \int_a^b w(x) (f(x) - \sum_{k=0}^m a_k \phi_k(x))^2 dx \longrightarrow \min$$

Sist. care se obține este:

$$\int_a^b w(x) f(x) \phi_j(x) dx = \sum_{k=0}^m a_k \int_a^b w(x) \phi_k(x) \phi_j(x) dx, \quad \forall j=0, \dots, m$$

$$\text{Dacă } \phi_1, \dots, \phi_m \text{ pot fi alese a.î.: } \int_a^b w(x) \phi_k(x) \phi_j(x) dx = \begin{cases} 0, & j \neq k \\ \alpha_j > 0, & j = k \end{cases}$$

atunci sistemul devine:  $\int_a^b w(x) f(x) \phi_j(x) dx = \alpha_j a_j$

$$\Rightarrow a_j = \frac{1}{\alpha_j} \int_a^b w(x) f(x) \phi_j(x) dx \quad (\text{pb. este simplificată})$$



de facut 1 3 4

# Aproximare prin metoda celor mai mici pătrate

Radu T. Trîmbițaș

25 mai 2020

Fie  $f \in L_w^2[a, b]$  și  $\Phi \leq L_w^2[a, b]$  de dimensiune  $n + 1$ . Dorim să găsim o aproximantă  $\varphi^* \in \Phi$  astfel încât  $\|\varphi^* - f\|^2 \leq \|\varphi - f\|^2, \forall \varphi \in \Phi$ .

Scriem

$$\varphi^*(x) = a_0\varphi_0(x) + \dots + a_n\varphi_n(x), \quad (1)$$

unde  $\{\varphi_k | k = 0, \dots, n\}$  este o bază a lui  $\Phi$ .

Coeficienții sunt soluțiile ecuațiilor normale

$$a_0(\varphi_0, \varphi_k) + a_1(\varphi_1, \varphi_k) + \dots + a_n(\varphi_n, \varphi_k) = (f, \varphi_k), \quad k = 0, \dots, n. \quad (2)$$

Dacă sistemul  $\{\varphi_k | k = 0, \dots, n\}$  este ortogonal, coeficienții se pot obține cu ajutorul formulelor

$$a_k = \frac{(f, \varphi_k)}{(\varphi_k, \varphi_k)}, \quad k = 0, \dots, n. \quad (3)$$

Aproximanta poate fi continuă sau discretă, în funcție de măsura aleasă în definiția produsului scalar. În cazul continuu produsul scalar are forma

$$(g, h) = \int_a^b w(x)g(x)h(x)dx,$$

iar în cazul discret

$$(g, h) = \sum_{k=0}^N w_k g(x_k)h(x_k).$$

Să considerăm relația de recurență pentru polinoamele ortogonale monice

$$\begin{aligned}\pi_{k+1}(x) &= (x - \alpha_k)\pi_k(x) - \beta_k\pi_{k-1}(x), k = 0, 1, 2, \dots \\ \pi_0(x) &= 1, \pi_{-1}(x) = 0,\end{aligned}$$

unde

$$\beta_0 = \int_a^b w(x)f(x)dx = \mu_0.$$

Coefficienții din relația de recurență (2) au expresia

$$\alpha_k = \frac{(x\pi_k, \pi_k)}{(\pi_k, \pi_k)}, \beta_k = \frac{(\pi_k, \pi_k)}{(\pi_{k-1}, \pi_{k-1})}.$$

Reamintim câteva dintre polinoamele ortogonale clasice și coeficienții din relațiile lor de recurență:

Polinoamele	Notăția	Pondere	interval	$\alpha_k$	$\beta_k$
Legendre	$P_n(l_n)$	1	$[-1,1]$	0	$2 \ (k=0)$ $(4-k^2)^{-1} \ (k>0)$
Cebîșev #1	$T_n$	$(1-t^2)^{-\frac{1}{2}}$	$[-1,1]$	0	$\pi \ (k=0)$ $\frac{1}{2} \ (k=1)$ $\frac{1}{4} \ (k>0)$
Cebîșev #2	$U_n(Q_n)$	$(1-t^2)^{\frac{1}{2}}$	$[-1,1]$	0	$\frac{1}{2}\pi \ (k=0)$ $\frac{1}{4} \ (k>0)$
Jacobi	$P_n^{(\alpha,\beta)}$	$(1-t)^\alpha(1+t)^\beta$ $\alpha>-1, \beta>-1$	$[-1,1]$		
Laguerre	$L_n^{(\alpha)}$	$t^\alpha e^{-t} \ \alpha>-1$	$[0,\infty)$	$2k+\alpha+1$	$\Gamma(1+\alpha) \ (k=0)$ $k(k+\alpha) \ (k>0)$
Hermite	$H_n$	$e^{-t^2}$	$\mathbb{R}$	0	$\sqrt{\pi} \ (k=0)$ $\frac{1}{2}k \ (k>0)$

Tabela 1: Polinoame ortogonale

**Observația 1** Pentru polinoamele Jacobi avem

$$\alpha_k = \frac{\beta^2 - \alpha^2}{(2k + \alpha + \beta)(2k + \alpha + \beta + 2)}$$

și

$$\beta_0 = 2^{\alpha+\beta+1} B(\alpha+1, \beta+1),$$

$$\beta_k = \frac{4k(k+\alpha)(k+\alpha+\beta)(k+\beta)}{(2k+\alpha+\beta-1)(2k+\alpha+\beta)^2(2k+\alpha+\beta+1)}, \quad k > 0.$$

**Probleme propuse.**

1. Să se găsească aproximanta discretă prin metoda celor mai mici pătrate pentru ponderea  $w(x)=1$  și baza  $1, x, x^2, \dots, x^n$ .
2. Un asteroid ce orbitează în jurul Soarelui a putut fi observat timp de câteva zile înainte să dispară. Iată 10 observații

$x_{1:5}$	-1.024940	-0.949898	-0.866114	-0.773392	-0.671372
$x_{6:10}$	-0.559524	-0.437067	-0.302909	-0.159493	-0.007464
$y_{1:5}$	-0.389269	-0.322894	-0.265256	-0.216557	-0.177152
$y_{6:10}$	-0.147582	-0.128618	-0.121353	-0.127348	-0.148895

Se dorește calcularea traiectoriei pe baza acestor observații pentru a putea prevedea situația când orbita va fi din nou vizibilă. Se presupune un model elipsoidal pentru orbită

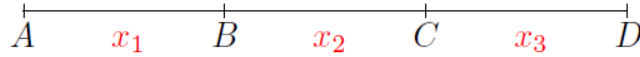
$$x^2 = ay^2 + bxy + cx + dy + e.$$

El ne conduce la un sistem supradeterminat, care trebuie rezolvat în sensul celor mai mici pătrate pentru a determina parametrii  $a, b, c, d, e$ . Realizați o estimare a erorii și un test de încredere în model. Faceți același lucru pentru modelul parabolic

$$x^2 = ay + e.$$

Care este mai probabil?

3. La măsurarea unui segment de drum, presupunem că am efectuat 5 măsurători



$$AD = 89m, \quad AC = 67m, \quad BD = 53m, \quad AB = 35m \text{ și } CD = 20m,$$

Să se determine lungimile segmentelor  $x_1 = AB$ ,  $x_2 = BC$  și  $x_3 = CD$ .

4. Datele următoare dau populația SUA (în milioane) determinată la recensăminte de US Census, între anii 1900 și 2010. Dorim să modelăm populația și să o estimăm pentru anii 1975 și 2010.

An	Populația	An	Populația
1900	75.995	1960	179.320
1910	91.972	1970	203.210
1920	105.710	1980	226.510
1930	123.200	1990	249.630
1940	131.670	2000	281.420
1950	150.700	2010	308.790

Modelați populația printr-un model polinomial de gradul 3

$$y = c_0 + c_1t + c_2t^2 + c_3t^3,$$

și printr-un model exponențial

$$y = Ke^{\lambda t}.$$

### Probleme suplimentare

1. Să se gasească aproximanta discretă prin metoda celor mai mici pătrate pentru ponderea  $w(x) = 1$  și baza formată din polinoamele Cebîșev de speța I. Produsul scalar are forma

$$(g, h) = \sum_{k=1}^{n+1} g(\xi_k)h(\xi_k),$$

unde  $\xi_k$ ,  $k = 1, \dots, n+1$  sunt rădăcinile polinomului Cebîșev de speța I  $T_{n+1}$ .

2. Se dă un polinom prin coeficienții săi relativ la o bază ortogonală  $\{\pi_j\}$ :

$$p(t) = \sum_{i=0}^n c_i \pi_i(t).$$

Să se dea o metodă de evaluare analoagă schemei lui Horner. (*Indicație:* se va folosi relația de recurență și coeficienții ei.)

## Integrare numerică

$$\int_a^b f(x) dx$$



- nu se poate afla explicit primitiva
- se poate afla foarte greu

$$\int_a^b f(x) dx$$

$$\int_a^b f(x) dx \approx \sum_{i=0}^n a_i f(x_i)$$

Se consideră un set de noduri  $\{x_0, \dots, x_n\} \subset [a, b]$ . Apoi integrăm polinomul de interpolare Lagrange.

$$P_m(x) = \sum_{i=0}^m f(x_i) L_i(x)$$

$$f = P_m + R_m \quad \Big| \quad \int_a^b \Rightarrow \dots$$

În acest caz:  $a_i = \int_a^b L_i(x) dx$  iar  $E(f) = \frac{1}{(n+1)!} \int_a^b \prod_{i=0}^n (x-x_i) f^{(n+1)}(\xi) dx$

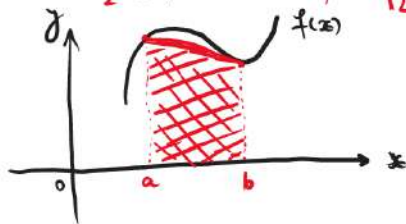
termenul eroare (al f. de c.)

Dim această regulă putem deduce:

- regula trapezului -  $x_0 = a, x_1 = b, h := b - a$

$$P_1(x) = \frac{x-x_1}{x_0-x_1} f(x_0) + \frac{x-x_0}{x_1-x_0} f(x_1)$$

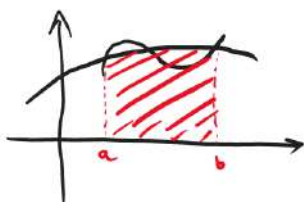
$$\int_a^b f(x) dx = \frac{h}{2} (f(x_0) + f(x_1)) - \frac{h^3}{12} f''(\xi), \quad \xi \in (x_0, x_1)$$



- regula lui Simpson:  $x_0 = a, x_2 = b, x_1 = a + h$  unde  $h = \frac{b-a}{2}$ .

Apelăm la polinomul lui Lagrange de gradul 2:

$$\int_a^b f(x) dx = \frac{h}{3} [f(x_0) + 4f(x_1) + f(x_2)] - \frac{h^5}{90} f^{(4)}(\xi), \quad \xi \in (x_0, x_2)$$



Formule Newton-Cotes închise cu  $(n+1)$  pct.:

(conțin capetele  
interv. = ac. capete sunt incluse)

$$x_i = x_0 + ih, \quad i = 0, \dots, n$$

$$x_0 = a, \quad x_n = b, \quad h = \frac{b-a}{n}$$

$$\int_a^b f(x) dx \approx \sum_{i=0}^n a_i f(x_i)$$

$$a_i = \int_a^b L_i(x) dx = \int_a^b \prod_{\substack{j=0 \\ j \neq i}}^n \frac{x-x_j}{x_i-x_j} dx$$

# Integrare numerică

Radu T. Trîmbițaș

25 aprilie 2021

## 1 Formule repetate

Fie  $f : [a, b] \rightarrow \mathbb{R}$  integrabilă pe  $[a, b]$ ,  $n \in \mathbb{N}$ ,  $h = (b - a)/n$  și nodurile echidistante  $x_k = a + kh$ ,  $k = 0, 1, \dots, n$ .

Formula repetată a trapezului:

$$\int_a^b f(x)dx = \frac{b-a}{2n} \left[ f(a) + f(b) + 2 \sum_{k=1}^{n-1} f(x_k) \right] + R_1(f),$$

unde

$$R_1(f) = -\frac{(b-a)^3}{12n^2} f''(\xi), \quad \xi \in (a, b).$$

Formula repetată a lui Simpson:

$$\int_a^b f(x)dx = \frac{b-a}{6m} \left[ f(a) + f(b) + 2 \sum_{k=1}^{m-1} f(x_{2k}) + 4 \sum_{k=1}^m f(x_{2k-1}) \right] + R_2(f),$$

unde  $n = 2m$ ;  $h = (b - a)/(2m)$ ;  $x_k = a + kh$ ;  $k = 0, 1, \dots, 2m$ , iar

$$R_2(f) = -\frac{(b-a)^5}{2880m^4}.$$

Formula dreptunghiurilor:

$$\int_a^b f(x)dx = \frac{b-a}{n} \sum_{k=1}^n f\left(\frac{x_{k-1} + x_k}{2}\right) + R_1(f)$$

unde

$$R_1(f) = \frac{(b-a)^3}{24n^2} f''(\xi), \quad \xi \in (a, b).$$

## 2 Cuadraturi adaptive

Fie  $\text{met}(a, b, f, n)$  be a composite formula. o formulă repetată oarecare. Ideea este de a împărți  $[a, b]$  în subintervale și de a folosi un număr mic de noduri pe subintervalele pe care oscilația lui  $f$  este lentă și un număr mai mare de puncte pe subintervalele pe care oscilația lui  $f$  este mai rapidă. Algoritmul este de tip divide and conquer:

```
function adaptquad(a,b:real; f:func; tol:real):real;
  if |met(a,b,f,m)-met(a,b,f,2m)|<tol
    then adaptquad:=met(a,b,f,2*m)
    else adaptquad:=adaptquad(a,(a+b)/2,f,tol)+
      adaptquad((a+b)/2,b,f,tol);
end
```

Cantitatea  $m$  este o constantă convenabilă (4 sau 5).

## 3 Metoda lui Romberg

Se bazează pe metoda trapezelor și pe extrapolarea Richardson. Fie  $h_1 = b - a$ . Vom utiliza formulele

$$R_{1,1} = \frac{h_1}{2} [f(a) + f(b)] = \frac{b-a}{2} [f(a) + f(b)]$$

$$R_{k,1} = \frac{1}{2} \left[ R_{k-1,1} + h_{k-1} \sum_{i=1}^{2^{k-2}} f \left( a + \left( i - \frac{1}{2} \right) h_{k-1} \right) \right], \quad k = \overline{2, n}$$

$$R_{k,j} = \frac{4^{j-1} R_{k,j-1} - R_{k-1,j-1}}{4^{j-1} - 1}, \quad k = \overline{2, n}$$

$$h_k = \frac{h_{k-1}}{2} = \frac{b-a}{2^{k-1}}.$$

**Exemplu.** Aproximați  $\int_0^\pi \sin x dx$  prin metoda lui Romberg,  $\varepsilon = 10^{-2}$ .

**Soluție.**

$$I = \int_0^\pi \sin x dx = 2$$

$$R_{1,1} = \frac{\pi}{2} (0 + 0) = 0$$

$$R_{2,1} = \frac{1}{2} \left( R_{1,1} + \pi \sin \frac{\pi}{2} \right) = 1.571$$

$$R_{2,2} = 1.571 + (1,571 - 0)/3 = 2.094$$

$$(R_{2,2} - R_{1,1}) > 0.01$$

$$R_{3,1} = \frac{1}{2} \left[ R_{2,1} + \frac{\pi}{2} \left( \sin \frac{\pi}{4} + \sin \frac{3\pi}{4} \right) \right] = 1.895$$

$$R_{3,2} = 1,895 + \frac{1.895 - 1.571}{3} = 2.004$$

$$R_{3,3} = 2.004 + (2.004 - 2.094)/15 = 1.999$$

$$|R_{3,3} - R_{2,2}| < 0.1$$

Valoarea exactă a integralei este  $I = 2$ . Pentru formula trapezelor cu același număr de argumente se obține  $I \approx 1,895$ ,

iar pentru formula repetată a lui Simpson cu 4 noduri,  $I \approx 2.005$ .

Calculule se pot realiza în mod tabelar.

$$\begin{array}{cccc} R_{1,1} & & & \\ R_{2,1} & R_{2,2} & & \\ R_{3,1} & R_{3,2} & R_{3,3} & \\ \vdots & \vdots & \vdots & \ddots \\ R_{n,1} & R_{n,2} & R_{n,3} & \dots & R_{n,n} \end{array}$$

Un posibil criteriu de oprire este  $|R_{n,n} - R_{n-1,n-1}| < \varepsilon$ .

## 4 Cuadraturi adaptive II

Coloana a doua din metoda lui Romberg corespunde aproximării prin metoda lui Simpson. Notăm

$$S_{k,1} = R_{k,2}.$$

Coloana a treia este deci o combinație a două aproximante de tip Simpson:

$$S_{k,2} = S_{k,1} + \frac{S_{k,1} - S_{k-1,1}}{15} = R_{k,2} + \frac{R_{k,2} - R_{k-1,2}}{15}.$$

Relația

$$S_{k,2} = S_{k,1} + \frac{S_{k,1} - S_{k-1,1}}{15}, \quad (1)$$

va fi folosită la elaborarea unui algoritm de quadratură adaptivă. Fie  $c = (a + b)/2$ . Formula elementară a lui Simpson este

$$S = \frac{h}{6} (f(a) + 4f(c) + f(b)).$$



Pentru două subintervale se obține

$$S_2 = \frac{h}{12} (f(a) + 4f(d) + 2f(c) + 4f(e) + f(b)),$$

unde  $d = (a + c)/2$  și  $e = (c + b)/2$ . Cantitatea  $Q$  se obține aplicând (1) celor două aproximante:

$$Q = S_2 + (S_2 - S)/15.$$

Putem să dam acum un algoritm recursiv pentru aproximarea integralei. Funcția *adquad* evaluează integrandul aplicând regula lui Simpson. Ea apelează recursiv *quadstep* și aplică extrapolarea. Descrierea se dă în algoritmul 1.

---

**Algorithm 1** Cuadratură adaptivă bazată pe metoda lui Simpson și extrapolare

---

**Intrare:** funcția  $f$ , intervalul  $[a, b]$ , eroarea  $\varepsilon$

**Ieșire:** Valoarea aproximativă a integralei

```

function adquad( $f, a, b, \varepsilon$ ) : real
   $c := (a + b)/2$ ;
   $fa = f(a)$ ;  $fb := f(b)$ ;  $fc := f(c)$ ;
   $Q := \text{quadstep}(f, a, b, \varepsilon, fa, fc, fb)$ ;
  return  $Q$ ;
function quadstep( $f, a, b, \varepsilon, fa, fc, fb$ ) : real
   $h := b - a$ ;  $c := (a + b)/2$ ;
   $fd := f((a + c)/2)$ ;  $fe := f((c + b)/2)$ ;
   $Q1 := h/6 * (fa + 4 * fc + fb)$ ;
   $Q2 := h/12 * (fa + 4 * fb + 2 * fc + 4 * fe + fb)$ ;
  if  $|Q1 - Q2| < \varepsilon$  then
     $Q := Q2 + (Q2 - Q1)/15$ ;
  else
     $Qa := \text{quadstep}(f, a, c, \varepsilon, fa, fd, fc)$ ;
     $Qb := \text{quadstep}(f, c, b, \varepsilon, fc, fe, fb)$ ;
     $Q := Qa + Qb$ ;
  end if
  return  $Q$ ;

```

---

## 5 Probleme

1. Implementați formula repetată a trapezului, dreptunghiului și a lui Simpson.
2. Concepeți o reprezentare grafică intuitivă pentru formula trapezelor și formula repetată a lui Simpson (facultativ).
3. Implementați o metoda de cuadratură adaptivă pentru formula repetată a lui Simpson, una pentru metoda trapezelor și una pentru metoda dreptunghiurilor.
4. Implementați metoda lui Romberg.
5. Implementați `adquad`, algoritmul 1.

## 6 Probleme practice

1. Generați formule Newton-Cotes închise pentru un număr de noduri dat.
2. Testați rutinele de integrare de la această temă pentru diverse funcții a caror primitivă nu este exprimabilă prin funcții elementare și comparați rezultatele cu cele furnizate de funcțiile MATLAB `quad` și `integral`.

## Cuadratură Gauss

Formule Newton - Cotes (formule de cuadratură) - integrezi un polinom de interpolare

↳ eroarea în această formulă cere derivata de ordinul  $n+1$  a funcției pe care o aproximăm

⇒ formula este exactă când aproximăm integrala unui polinom de grad  $\leq n$ .

□ Newton - Cotes - noduri echidistante - problemă: acuratețea



Cuadratură Gauss: alegem nodurile optime + coeficienții optime

Alegem  $x_1, \dots, x_n \in [a, b]$  și coef.  $c_1, \dots, c_n$  pt. a minimiza erorile apărute în aprox:

$$\int_a^b f(x) dx \approx \sum_{i=1}^n c_i f(x_i)$$

Ex. Să se găsească  $c_1, c_2, x_1, x_2$  a. i. cuadratură:

$$\int_{-1}^1 f(x) dx \approx c_1 f(x_1) + c_2 f(x_2)$$

să fie exactă pentru  $f$  polinom de grad  $\leq 3$ .

Rez. 4 nec. → gradul 3.

$$f(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3$$

Formula este exactă pt.  $\forall$  polinom de grad  $\leq 3$

⇒  $(x^3, x^2, x, 1)$  - formula va fi exactă pentru aceste polinoame.

$$\int_{-1}^1 1 dx = c_1 \cdot 1 + c_2 \cdot 1 \quad (\Rightarrow) \quad 2 = c_1 + c_2$$

$$\int_{-1}^1 x dx = c_1 \cdot x_1 + c_2 \cdot x_2 \quad (\Rightarrow) \quad \dots$$

$$\int_{-1}^1 x^2 dx = c_1 \cdot x_1^2 + c_2 \cdot x_2^2 \quad (\Rightarrow) \quad \dots$$

$$\int_{-1}^1 x^3 dx = c_1 \cdot x_1^3 + c_2 \cdot x_2^3 \quad (\Rightarrow) \quad \dots$$

sistem ce se rezolvă  
pentru a obține  
 $x_1, x_2, c_1, c_2$ .

$$c_1 = c_2 = 1, \quad x_1 = -\frac{\sqrt{3}}{3}, \quad x_2 = \frac{\sqrt{3}}{3}$$

$$\Rightarrow \int_{-1}^1 f(x) dx \approx f\left(-\frac{\sqrt{3}}{3}\right) + f\left(\frac{\sqrt{3}}{3}\right)$$

Obs. Acuratețe îmbunătățită → se consideră polinoame ortogonale → ex. se aleg nodurile  $x_1, \dots, x_n$  = rădăcinile polinoamelor Legendre.

# Generarea unor cuadraturi de tip Gauss

Radu T. Trîmbițaș

10 mai 2021

Formula

$$\int_a^b w(x)f(x)dx = \sum_{k=1}^n A_k f(x_k) + R_n(f)$$

se numește de tip Gauss dacă coeficienții și nodurile sunt alese astfel încât gradul de exactitate să fie maxim. Nodurile  $x_k$ ,  $k = 1, \dots, n$  sunt rădăcinile polinomului

$$\pi_n(x) = \prod_{k=1}^n (x - x_k).$$

ortogonal pe  $[a, b]$ , în raport cu ponderea  $w$ .

Să considerăm relația de recurență pentru polinoamele ortogonale monice

$$\begin{aligned} \pi_{k+1}(x) &= (x - \alpha_k)\pi_k(x) - \beta_k\pi_{k-1}(x), & k &= 0, 1, 2, \dots \\ \pi_0(x) &= 1, & \pi_{-1}(x) &= 0. \end{aligned} \quad (1)$$

unde

$$\beta_0 = \int_a^b w(x)f(x)dx = \mu_0. \quad (2)$$

Coeficienții din relația de recurență (1) au expresia

$$\alpha_k = \frac{(x\pi_k, \pi_k)}{(\pi_k, \pi_k)}, \quad \beta_k = \frac{(x\pi_k, \pi_{k-1})}{(\pi_{k-1}, \pi_{k-1})}.$$

Matricea Jacobi de ordinul  $n$  a funcției pondere  $w$  este o matrice tridia-

gonală simetrică, definită prin

$$J_n(w) = \begin{bmatrix} \alpha_0 & \sqrt{\beta_1} & & & 0 \\ \sqrt{\beta_1} & \alpha_1 & \sqrt{\beta_2} & & \\ & \sqrt{\beta_2} & & \ddots & \\ & & \ddots & \ddots & \sqrt{\beta_{n-1}} \\ 0 & & & \sqrt{\beta_{n-1}} & \alpha_{n-1} \end{bmatrix}.$$

Nodurile  $x_k$  sunt valori proprii ale lui  $J_n$

$$J_n v_k = x_k v_k, \quad v_k^T v_k = 1, \quad k = 1, 2, \dots, n, \quad (3)$$

iar coeficienții  $A_k$  se pot exprima cu ajutorul primei componente  $v_{k,1}$  a vectorilor proprii corespunzători normalizați:

$$w_k = \beta_0 v_{k,1}^2, \quad k = 1, 2, \dots, n \quad (4)$$

Dacă  $f \in C^{2n}[a, b]$ , pentru rest avem expresia

$$R_n(f) = \frac{f^{(2n)}(\xi)}{(2n)!} \int_a^b w(x) \pi_n^2(x) dx.$$

Tabela 1 dă câteva dintre polinoamele ortogonale clasice și coeficienții din relațiile lor de recurență.

Cu datele din tabel se formează matricea  $J$  și se găsesc  $x_k$  și  $A_k$ . Polinoamele ortogonale se aleg în funcție de intervalul de definiție și de pondere. De exemplu, pentru  $w = 1$  se lucrează cu polinoamele Legendre pe  $[-1, 1]$  și se face apoi schimbarea de variabilă pentru a trece la  $[a, b]$ .

**Observația 1** Pentru polinoamele Jacobi avem

$$\alpha_k = \frac{\beta^2 - \alpha^2}{(2k + \alpha + \beta)(2k + \alpha + \beta + 2)}$$

și

$$\beta_0 = 2^{\alpha+\beta+1} B(\alpha + 1, \beta + 1),$$

$$\beta_k = \frac{4k(k + \alpha)(k + \alpha + \beta)(k + \beta)}{(2k + \alpha + \beta - 1)(2k + \alpha + \beta)^2(2k + \alpha + \beta + 1)}, \quad k > 0.$$

Polinoamele	Notăție	Ponderea	interval	$\alpha_k$	$\beta_k$
Legendre	$P_n(l_n)$	1	$[-1,1]$	0	$2 \ (k=0)$ $(4-k^{-2})^{-1} \ (k>0)$
Cebîșev #1	$T_n$	$(1-t^2)^{-\frac{1}{2}}$	$[-1,1]$	0	$\pi \ (k=0)$ $\frac{1}{2} \ (k=1)$ $\frac{1}{4} \ (k>1)$
Cebîșev #2	$u_n(Q_n)$	$(1-t^2)^{\frac{1}{2}}$	$[-1,1]$	0	$\frac{1}{2}\pi \ (k=0)$ $\frac{1}{4} \ (k>0)$
Jacobi	$P_n^{(\alpha,\beta)}$	$(1-t)^\alpha (1-t)^\beta$ $\alpha > -1, \beta > -1$	$[-1,1]$		
Laguerre	$L_n^{(\alpha)}$	$t^\alpha e^{-t} \ \alpha > -1$	$[0,\infty)$	$2k+\alpha+1$	$\Gamma(1+\alpha) \ (k=0)$ $k(k+\alpha) \ (k>0)$
Hermite	$H_n$	$e^{-t^2}$	$\mathbb{R}$	0	$\sqrt{\pi} \ (k=0)$ $\frac{1}{2}k \ (k>0)$

Tabela 1: Polinoame ortogonale

### Probleme.

1. Implementați funcții ce generează formule de cuadratură gaussiene pentru ponderile clasice date în tabela 1.
2. Să se calculeze integralele  $\int_{-1}^1 \sin x^2 dx$  și  $\int_{-1}^1 \cos x^2 dx$  cu precizia  $\varepsilon = 10^{-7}$  folosind o cuadratură gaussiană. Câte noduri sunt necesare?
3. Determinați o formulă de cuadratură de forma

$$\int_{-1}^1 \frac{f(x)}{\sqrt{1-x^2}} dx = \sum_{k=1}^{10} A_k f(x_k) + R(f)$$

care să aibă grad maxim de exactitate. Să se aplice formula pentru a calcula  $\int_{-1}^1 \frac{x e^{-x^2}}{\sqrt{1-x^2}} dx$ . Verificare.

4. Determinați o formulă de cuadratură de forma

$$\int_{-1}^1 \sqrt{1-x^2} f(x) dx = \sum_{k=1}^{10} A_k f(x_k) + R(f)$$

care să aibă grad maxim de exactitate. Să se aplice formula pentru a calcula  $\int_{-1}^1 \sqrt{1-x^2} e^{-x^2} dx$ . Cât este eroarea?

5. Calculați

$$\int_0^\infty e^{-x} \sin(x) dx, \quad \int_0^\infty e^{-x} \cos(x) dx,$$

cu 8 zecimale exacte folosind o cuadratură Gauss-Laguerre.

6. Calculați

$$\int_{\mathbb{R}} e^{-x^2} \sin x dx, \quad \int_{\mathbb{R}} e^{-x^2} \cos x dx$$

utilizând o formulă de cuadratură Gauss-Hermite.

7. Aproximați  $\int_0^{\frac{\pi}{2}} \frac{1}{\sqrt{\sin x}} dx$  cu 9 zecimale exacte, folosind o cuadratură Gauss-Jacobi.

# Rezolvarea numerică a ecuațiilor neliniare

Radu T. Trîmbițaș

16 mai 2021

## 1 Ecuații neliniare în $\mathbb{R}$

Fie  $f : \mathbb{R} \rightarrow \mathbb{R}$ . Dorim să aproximăm o soluție sau toate soluțiile ecuației  $f(x) = 0$ . Vom prezenta câteva metode importante.

### 1.1 Metoda Newton-Raphson (a tangentei)

Determină o soluție a ecuației  $f(x) = 0$ , dându-se o aproximație inițială  $p_0$ .

**Intrare.** Funcția  $f$ , derivata sa  $f'$ , o aproximație inițială  $p_0$ ; eroarea  $\varepsilon$ ; numărul maxim de iterații  $N_0$ .

**Ieșire.** O soluție aproximativă  $p$  sau un mesaj de eroare.

**P1.**  $i := 1$ .

**P2.** While  $i \leq N_0$  execută pașii P3-P6.

**P3.**  $p := p_0 - f(p_0)/f'(p_0)$ ; {Calculează  $p$ }

**P4.** If  $|p - p_0| < \varepsilon$  then  
    Returnează  $p$ ; {Succes}

**P5.**  $i := i + 1$ ;

**P6.**  $p_0 := p$ ; {actualizează  $p$ }

**P7.** {eșec} eroare('precizia nu poate fi atinsă cu numărul dat de iterații').  
Stop.

**Observație.** În plus față de

$$|p_n - p_{n-1}| < \varepsilon$$

putem utiliza drept criteriu de oprire

$$|p_n - p_{n-1}| < \varepsilon |p_n|, \quad p_n \neq 0,$$



sau

$$|f(p_n)| < \varepsilon.$$

**Exemplu numeric.** Fie ecuația  $x = \cos x$ . Punem  $f(x) = \cos x - x$ . Ecuația noastră are o soluție în  $[0, \pi/2]$ , care poate fi obținută ca punct fix al lui  $g(x) = \cos x$  (vezi figura 1). Deoarece  $f'(x) = -\sin x - 1$ , iterația Newton este

$$p_n = p_{n-1} - \frac{\cos p_{n-1} - p_{n-1}}{-\sin p_{n-1} - 1}, \quad n \geq 1.$$

Ca valoare de pornire se poate alege  $p_0 = \pi/4$ . Valorile calculate se dau în tabela următoare

$n$	$p_n$	$n$	$p_n$
0	0.7853981635	3	0.7390851332
1	0.7395361337	4	0.7390851332
2	0.7390851781	5	0.7390851332

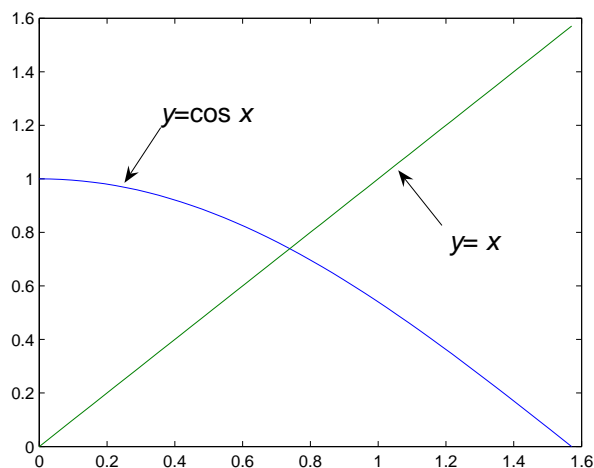


Figura 1: Ecuația  $\cos x = x$

## 1.2 Metoda secantei

Determină o soluție a ecuației  $f(x) = 0$ , dându-se aproximațiile inițiale  $p_0$  și  $p_1$ .

**Intrare.** Funcția  $f$ , aproximațiile inițiale  $p_0$  și  $p_1$ ; eroarea  $\varepsilon$ ; numărul maxim de iterații  $N_0$ .

**Ieșire.** Soluția aproximativă  $p$  sau un mesaj de eroare.

**P1.**  $i := 2$ ;  $q_0 := f(p_0)$ ;  $q_1 := f(p_1)$ ;

**P2.** while  $i \leq N_0$  execută pașii P3-P6.

**P3.**  $p := p_1 - \frac{q_1(p_1 - p_0)}{q_1 - q_0}$ ;

**P4.** if  $|p - p_1| < \varepsilon$  then  
returnează  $p$ ; {succes}

**P5.**  $i := i + 1$ ;

**P6.**  $p_0 := p_1$ ;  $q_0 := q_1$ ;  $p_1 := p$ ;  $q_1 := f(p)$ ;

**P7.** {eșec} eroare('precizia nu poate fi atinsă cu numărul dat de iterații').  
Stop.

**Exemplu numeric.** Considerăm din nou ecuația  $\cos x - x = 0$ . Ca valori de pornire alegem  $p_0 = 0.5$  and  $p_1 = \pi/4$ . Calculele se dau în tabela de mai jos:

$n$	$p_n$
0	0.5
1	0.7853981635
2	0.7363841390
3	0.7390581394
4	0.7390851492
5	0.7390851334

## 1.3 Metoda lui Steffensen

Determină o soluție a ecuației  $p = g(p)$ , dându-se o aproximație inițială  $p_0$ .

**Intrare.** Funcția  $f$ , valoarea de pornire  $p_0$ ; eroarea  $\varepsilon$ ; numărul maxim de iterații  $N_0$ .

**Ieșire.** Soluția aproximativă  $p$  sau un mesaj de eroare.

**P1.**  $i := 1$ .

**P2.** While  $i \leq N_0$  execută pașii P3-P6.

**P3.**

$$\begin{aligned} p_1 &:= g(p_0); \quad \{\text{calculează } p_1^{(i-1)}\} \\ p_2 &:= g(p_1); \\ p &:= p_0 - \frac{(p_1 - p_0)^2}{p_2 - 2p_1 + p_0}; \quad \{\text{calculează } p_0^{(i)}\} \end{aligned}$$

**P4.** if  $|p - p_0| < \varepsilon$  then  
returnează  $p$ ; {success}

**P5.**  $i := i + 1$ ;

**P6.**  $p_0 := p$ ; {actualizează  $p$ }

**P7.** {eșec} eroare('precizia nu poate fi atinsă cu numărul dat de iterații').  
Stop.

**Exemplu numeric.** Pentru a rezolva ecuația  $x^3 + 4x^2 - 10 = 0$ , o rescriem sub forma  $x^3 + 4x^2 = 10$  și obținem

$$x = g(x), \quad g(x) = \sqrt{\frac{10}{x+4}}.$$

Luând  $p_0 = 1.5$  avem succesiv

$k$	$p_0$	$p_1$	$p_2$
0	1.5	1.348399725	1.367376372
1	1.365265224	1.365225534	
2	1.365230013	1.365230583	

## 1.4 Probleme

1) Implementați metodele Newton, secantă, Steffensen și metoda aproximațiilor succesive.

newton secanta steffesen

## 2 Sisteme de ecuații neliniare

Fie

$$\begin{aligned} f : D \subseteq \mathbb{R}^n &\rightarrow \mathbb{R}^n \\ f &= \begin{pmatrix} f_1 \\ \vdots \\ f_n \end{pmatrix}. \end{aligned} \tag{1}$$

Dorim să rezolvăm ecuația (sistemul neliniar)  $f(x) = 0$ .

## 2.1 Metoda lui Newton

Formula iterativă este

$$x^{(n+1)} = x^{(n)} - [f'(x^{(n)})]^{-1} f(x^{(n)}), \quad (2)$$

unde  $f'(x^{(n)})$  este jacobianul lui  $f$  în punctul  $x^{(n)}$ .

**Algoritmul.**

**Intrare.**  $f$ ,  $f'$ ,  $\varepsilon$  (toleranța), valoarea de pornire  $x^{(0)}$  și numărul maxim de iterații  $N$ .

**Ieșire.** O aproximare a rădăcinii sau un mesaj de eroare.

$n := 0$ ;

repeat

$$x^{(n+1)} = x^{(n)} - [f'(x^{(n)})]^{-1} f(x^{(n)}); \quad n := n + 1;$$

until

$$\|x^{(n)} - x^{(n-1)}\| < \varepsilon$$

or „s-a depășit numărul maxim de iterații“.

## 2.2 Metoda aproximațiilor succesive

Transformăm ecuația noastră într-una de forma  $x = \varphi(x)$ . Căutăm  $\varphi$  de forma  $\varphi(x) = x - \Lambda f(x)$ . Avem

$$\varphi'(x^{(0)}) = 0 \implies \Lambda = -[f'(x^{(0)})]^{-1}.$$

**Algoritmul.**

**Intrare:**  $f$ ,  $\varepsilon$  (toleranța), valoarea de pornire  $x_0$  și numărul maxim de iterații  $N$ .

**Ieșire:** O aproximare a rădăcinii sau un mesaj de eroare: ”precizia dorită nu poate fi atinsă în  $N$  iterații”.

Repeat

$$x^{(n+1)} = \varphi(x^{(n)})$$

until  $\|x^{(n+1)} - x^{(n)}\| < \varepsilon$  or „s-a depășit numărul maxim de iterații“.

## 2.3 Probleme

da

1. Implementați metoda lui Newton și metoda aproximațiilor succesive.

## 2.4 Probleme practice

1. Rezolvați sistemul

$$\begin{cases} x^2 + y^2 = 1. \\ x^3 - y = 0. \end{cases}$$

2. Rezolvați numeric sistemul

de facut cu alea doua de mai sus

$$\begin{cases} 9x^2 + 36y^2 + 4z^2 - 36 = 0, \\ x^2 - 2y^2 - 20z = 0, \\ x^2 - y^2 + z^2 = 0 \end{cases}$$

utilizând metoda lui Newton și metoda aproximațiilor succesive. *Indicație.* Sunt 4 soluții. Valori bune de pornire  $[\pm 1, \pm 1, 0]^T$ .