LAB1

Formula lui Taylor

Radu Trîmbiţaş

25 februarie 2017

1 Formula lui Taylor

- I interval, $f : I \to \mathbb{R}$ o funcţie derivabilă de n ori în punctul $a \in I$. Polinomul lui Taylor de gradul n, ataşat funcţiei f în punctul a:

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

- *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 funcţia f în vecinătatea punctului a:

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

sau

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

2 Restul

- Are loc

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

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

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

(restul în forma Lagrange)

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

(restul în forma Cauchy)

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

(restul în formă integrală)

3 Formula lui Maclaurin

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

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

unde

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

- Example de dezvoltări uzuale

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

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

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

- Alte dezvoltări uzuale

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

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

unde

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

4 Aplicaţii

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

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

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

■

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

Soluţie. Impunem condiţia $|(R_n f)(x)| = \left| \frac{x^{n+1} e^{\theta x}}{(n+1)!} \right| < 10^{-3}$.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 $x = 1$, obţinem

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

■

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

Soluţie. 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. Dezvoltaţi funcţia eroare

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

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

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

Problema 5. Deduceţi seria Taylor pentru $\ln(1 + x)$ şi aproximaţi $\ln 2$ folosind primii 8 termeni. Câţi termeni sunt necesari pentru a obţine $\ln 2$ cu 5 zecimale corecte? La fel pentru $\ln \frac{1+x}{1-x}$.

Problema 6. Deduceţi seria Taylor pentru arctangentă. Câţi termeni sunt necesari pentru a obţine $\pi/4$ cu 5 zecimale corecte.

Problema 7 (Aproximare cu serii MacLaurin)**.** O funcţia $f \in C^n[a,b]$ 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) Reprezentaţi grafic şi comparaţi graficele lui $f(x) = e^x$ şi ale polinoamelor $(T_2 f)(x)$, $(T_3 f)(x)$, $(T_4 f)(x)$, $(T_5 f)(x)$. Aproximează mulţumitor polinoamele $T_n f$ de grad mare funcţia e^x pe un interval din ce in ce mai mare centrat în jurul originii?

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

Problema 8 (Aproximare Padé raţională)**.** *Aproximarea Padé raţională* este cea mai bună aproximare a unei funcţii printr-o funcţie raţională de ordin (m,n) dat. Se defineşte ca fiind o aproximare raţională de grad (m,n) dat care reproduce valorile funcţiei si derivatelor ei pâna la ordinul $m+k$. Ea dă adesea aproximări mai bune decât seriile Taylor trunchiate şi uneori lucrează chiar şi atunci când seria Taylor nu converge! În loc să utilizăm polinoame de grad mare, putem utiliza câturi de polinoame de grad mic. Aceste aproximări se numesc *aproximări raţionale*. 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 şi k se presupun a fi modeste. Alegem cei k coeficienţi b_j şi cei $m+1$ coeficienţi a_i din $R_{m,k}$ astfel încât $R_{m,k}$ să reproducă valorile lui f şi 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 $c_i = f^{(i)}(0)/i!$ şi $c_i = 0$ pentru $i < 0$. Apoi, egalăm primele $m+k+1$ derivate ale lui $R_{m,k}$ în raport cu x în $x = 0$ cu primii $m+k+1$ coeficienţi c_i. Se obţine 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 obţine coeficienţii $b_1, b_2, \ldots, b_k$. Valorile lui $a_0, a_1, \ldots, a_m$ se obţin din

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

De notat că $a_j = 0$ pentru $j > m$ şi $b_j = 0$ 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 singularităţi.

(a) Implementaţi aproximarea Padé pentru f, k, m date.

(b) Determinaţi funcţiile raţionale $R_{1,1}(x)$ şi $R_{2,2}(x)$ pentru $f(x) = e^x$. Reprezentaţi grafic şi comparaţi graficele lui $f(x) = e^x$, $R_{1,1}$ şi $R_{2,2}$. Sunt satisfăcătoare aceste aproximaţii raţionale ale lui e^x pe $[-1, 1]$? Cum se comportă comparativ cu seriile Maclaurin trunchiate din problemele precedente?

(c) Repetaţi pentru aproximările $R_{2,2}(x)$ şi $R_{3,1}(x)$ ale funcţiei $g(x) = \ln(1 + x)$.

Problema 9. Calculaţi dezvoltarea MacLaurin a funcţiei Bessel $J_0(2x)$. Determinaţi $R_{2,2}(x)$, $R_{4,3}(x)$ şi $R_{2,4}(x)$ şi comparaţi graficele. Funcţiile Bessel J_n se definesc prin

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

LAB2

Teoria erorilor şi aritmetica în virgula flotantă

Radu T. Trîmbiţaş

12 martie 2020

1 Probleme

P1. Scrieţi funcţii MATLAB pentru a calcula epsilon-ul maşinii, cel mai mare număr reprezentabil în VF şi cel mai mic număr normalizat şi nenormalizat reprezentabil în VF. Comparaţi rezultatele cu cele returnate de funcţiile MATLAB `eps`, `realmin`, `realmax`.

P2. Scrieţi funcţii MATLAB pentru calculul lui $\sin x$ şi $\cos x$ folosind formula lui Taylor:

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} + ... + (-1)^n \frac{x^{2n+1}}{(2n+1)!} + ...$$
$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} + ... + (-1)^n \frac{x^{2n}}{(2n)!} + ...$$

Ştim de la cursul de Analiză matematică următoarele:

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

Ce se întâmplă pentru $x = 10\pi$ (şi în general pentru $x = 2k\pi$, k mare)? Explicaţi fenomenul şi propuneţi un remediu.

P3. Scrieţi funcţii MATLAB pentru calculul lui $\sin x$ şi $\cos x$ folosind aproximarea Padé în locul formulei lui Taylor. Atenţie la reducerea rangului.

P4. Scrieţi o funcţie MATLAB care primeşte la intrare un număr flotant (simplă sau dublă precizie) şi returnează reprezentarea sa binară pe componente: semn, exponent deplasat şi semnificantul (aşa 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^* şi x_2^* astfel încât $x_1^* = \text{fl}(x_1) = x_1(1+\delta_1)$, $x_2 = \text{fl}(x_2) = x_2(1+\delta_2)$ şi $|\delta_1| < \delta$, $|\delta_2| < \delta$. Cât de mic trebuie să fie δ, astfel incât să putem testa corect (în virgulă flotantă cu precizia maşinii eps), dacă $x_1 \neq x_2$.

S2. Acelaşi enunţ ca la problema P1, dar în Maple.

LAB3

Condiţionarea unei probleme

Radu T. Trîmbiţaş

13 martie 2019

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

   (a) Ce se obţine dacă perturbăm membrul drept astfel încât el sa devină $\begin{bmatrix} 32.1 & 22.9 & 33.1 & 30.9 \end{bmatrix}$? Care este eroarea relativă la intrare, la ieşire şi raportul lor?

   (b) Aceeaşi intrebare dacă perturbăm matricea sistemului:

$$\begin{bmatrix} 10 & 7 & 8.1 & 7.2 \\ 7.08 & 5.04 & 6 & 5 \\ 8 & 5.98 & 9.89 & 9 \\ 6.99 & 4.99 & 9 & 9.98 \end{bmatrix}.$$

Explicaţi şi analizaţi fenomenul.

1. Să se studieze condiţionarea matricei Hilbert H_n în raport cu norma euclidiană, $n = \overline{10, 15}$.

2. (a) Să se studieze condiţionarea matricei Vandermonde $V_n(t)$, pentru $t_k = -1 + k\frac{2}{n}$ (puncte echidistante in [-1,1]) şi $n = \overline{10, 15}$ în raport cu norma Cebîşev.

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

3. Să se studieze teoretic şi experimental condiţionarea problemei determinării rădăcinilor ecuaţiei polinomiale

$$x^n + a_1 x^{n-1} + a_2 x^{n-2} + \cdots + a_n = 0 \tag{1}$$

cunoscându-se coeficienţii. Se va scrie o rutină pentru calculul numărului de condiţionare al fiecărei rădăcini şi se va studia grafic efectul perturbării fiecărui coeficient cu o variabilă aleatoare normală cu media 0 şi dispersia 10^{-10}. Aplicaţie pentru ecuaţiile

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

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

4. Fie

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

Se observă că $E_1 = 1/e$ şi $E_n = 1 - nE_{n-1}, \;\; n = 2, 3, \ldots$

Se poate arătă că

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

şi dacă $E_1 = c$ avem

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

Explicaţi fenomenul, găsiţi un remediu şi calculaţi e cu precizia eps.

2 Probleme facultative

1. Să se studieze condiţionarea unei rădăcini multiple a unei ecuaţii algebrice. Scrieţi o rutină MATLAB pentru calculul numerelor de condiţionare

dacă se dau ecuaţia (coeficienţii), rădăcinile şi multiplicităţile lor. Repetaţi experimentul aleator de la problema 4 pentru ecuaţia

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

LAB4

Sisteme liniare - metode directe

Radu T. Trîmbiţaş

16 martie 2020

1 Eliminare gaussiană

Să considerăm sistemul liniar cu n ecuaţii şi n necunoscute

$$Ax = b, \tag{1}$$

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

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

1.1 Eliminare gaussiană cu pivotare parţială

Metoda este dată de algoritmul 1.

1.2 Eliminare gaussiană cu pivot scalat pe coloană

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

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

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

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

Ieşire: Soluţiile $x_1, \ldots, x_n$ sau un mesaj de eroare

{Eliminare}

1: **for** $i := 1$ **to** $n-1$ **do**
2: Fie p cel mai mic întreg $i \leq p \leq n$, $a_{p_i} \neq 0$ şi $|a_{pi}| = \max_{i \leq j \leq n} |a_{ji}|$
3: **if** $\nexists\ p$ **then**
4: mesaj ('$\nexists$ soluţie 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** $a_{mn} = 0$ **then**
15: mesaj ('$\nexists$ soluţie unică'); STOP
16: **end if**

{Substituţie 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 $(x_1, \ldots, x_n)$ {succes} STOP.

Dacă există un i a.î. $s_i = 0$, matricea este singulară. Paşii 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.

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

şi apoi, $(E_i) \leftrightarrow (E_p)$. Scalarea ne garantează că cel mai mare element din fiecare coloană are înainte de comparaţiile necesare pentru schimbare mărimea relativă 1. Scalarea se realizează doar în comparaţii, nu efectiv în matrice, astfel că împărţirea 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 şi 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 nesingulară posedă o astfel de descompunere.

Sistemul

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

se poate rezolva astfel

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

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 π, toate de dimensiune n

Ieşire: Soluţiile $x_1, \ldots, x_n$

1: **for** $i := 1$ **to** n **do**

2: $\quad y_i := b_{\pi[i]} - \sum_{j=1}^{i-1} l_{ij} y_j$;

3: **end for**

4: **for** $i := n$ **downto** 1 **do**

5: $\quad x_i = \left[y_i - \sum_{j=i+1}^{n} u_{ij} x_j \right] / u_{ii}$;

6: **end for**

Observaţie. Am presupus că matricea P este reprezentată prin vectorul π.

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

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

Algoritmul 3 Descompunere LUP

Intrare: Matricea A, de dimensiune m

Ieşire: Matricele L, U şi P, toate de dimensiune m

$p = 1 : m$;
for $k := 1$ **to** $m - 1$ **do**
{Pivotare}
Alege $i \geq k$ care maximizeaza $|u_{ik}|$;
$A_{k,:} \leftrightarrow A_{i,:}$; {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
```

Calculele decurg astfel

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

interschimb liniile 1 şi 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 şi 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 şi 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 şi 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 şi pozitiv definită
Ieşire: 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 *Implementaţi eliminarea gaussiană cu pivotare parţială 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 *Generaţi sisteme cu matrice aleatoare nesingulare ce au soluţia* $[1, \ldots, 1]^T$. *Rezolvaţi-le cu eliminare gaussiană şi descompunere LUP.*

Problema 4 *Să se scrie rutine pentru descompunerea Cholesky a unei matrice hermitiene şi pozitiv definite şi rezolvarea unui sistem cu o astfel de matrice prin descompunere Cholesky. Testaţi rutinele pentru matrice generate aleator şi sisteme cu matrice aleatoare, dar cu soluţie cunoscută.*

Problema 5 *Rezolvaţi 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 şi QR. Ce se observă? Explicaţi.

5 Probleme suplimentare

Problema 6 *Scrieţi rutine pentru descompunerea LUP in care permutarea sa se facă fizic şi logic (cu vectori de permutări) şi comparaţi timpii de execuţie al ambelor variante pentru sisteme cu dimensiunea între 100 şi 300.*

LAB5

Sisteme liniare - metode iterative

Radu T. Trîmbiţaş

1 iunie 2020

Dorim să calculăm soluţia sistemului

$$Ax = b, \tag{1}$$

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

$$x = Tx + c \tag{2}$$

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

Iteraţia: $x^{(0)}$ dat; se defineşte $(x^{(k)})$ prin

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

Criteriul de oprire este

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

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

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

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

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

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 şi N se obţine:

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

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

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

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

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

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

Pe componente,

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

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

$$\omega_O = \frac{2}{1 + \sqrt{1 - \rho^2}}, \tag{5}$$

unde ρ este raza spectrală a matricei metodei lui Jacobi.

1 Probleme

Problema 1 *Implementaţi metoda lui Jacobi în MATLAB.*

Problema 2 *Implementaţi metoda SOR în MATLAB. Găsiţi ω optim utilizând (5). Atenţie: aceasta nu este o metodă practică pentru calculul lui ω_o; ea are numai scop didactic.*

Problema 3 *Rezolvaţi sistemele:*

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

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

cu toate metodele implementate.

Problema 4 *Generaţi sisteme cu matrice diagonal dominante aleatoare ce au soluţia $[1, \dots, n]^T$ şi rezolvaţi-le cu metodele Jacobi, Gauss-Seidel şi SOR.*

2 Probleme suplimentare

Problema 5 *Testaţi rutinele implementate pentru matrice rare de diverse dimensiuni şi comparaţi timpii de execuţie cu cei necesari pentru matrice dense.*

Problema 6 *Pentru rutinele implementate generaţi curbe de convergenţă, adică curbe semilogaritimice care au pe abscisă numărul pasului curent, iar pe ordonată logaritmul normei reziduului. (folosiţi funcţia MATLAB* `semilogy`*).*

LAB6

Interpolare Lagrange

Radu Trîmbiţaş

20 martie 2020

1 Forma clasică

Fie $f : [a, b] \to \mathbb{R}$, $x_i \in [a, b]$, $i = 0, \ldots, m$. 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:

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

Formula de interpolare Lagrange este

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

unde L_m este polinomul de interpolare Lagrange:

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

ℓ_k sunt polinoamele fundamentale de interpolare Lagrange

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

iar R_m este termenul rest:

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

Dacă valorile funcţiei sunt tabelate, evaluarea lui ℓ_k necesită $2(n-1)$ înmulţiri, o împărţire şi $2n$ scăderi. Întreaga evaluare necesită $2n(n+1)$ * |/ şi $n(2n+3)$ +|-.

2 Algoritmul lui Aitken

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

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

Algoritmul lui Aitken se bazează pe relaţia

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

Metoda generează tabela următoare:

$$\begin{array}{ccccccccc} x_0 & f_{0,0} & & & & & & \\ x_1 & f_{1,0} & f_{1,1} & & & & & \\ x_2 & f_{2,0} & f_{2,1} & f_{22} & & & & \\ \vdots & \vdots & \vdots & \vdots & \ddots & & & \\ x_i & f_{i,0} & f_{i,1} & f_{i,2} & \cdots & f_{i,i} & & \\ \vdots & \vdots & \vdots & \vdots & & \vdots & \ddots & \\ x_n & f_{n,0} & f_{n,1} & f_{n,2} & \cdots & f_{n,i} & \cdots & f_{n,n} \end{array}$$

unde $f_{i,0} = f(x_i), \quad i = 0, ..., m$, şi

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

Se verifică uşor că $(L_i f)(x) = f_{i+1,i+1}$, $i = 0, \ldots, n-1$, datorită ecuaţiei (**??**). Dacă interpolarea Lagrange converge, atunci $(f_{i,i})_{i\in\mathbb{N}}$ converge către $f(x)$ şi $|f_{i,i} - f_{i-1,i-1}| \to 0$ când $i \to \infty$, deci relaţia $|f_{i,i}$ - $f_{i-1,i-1}| \leq \varepsilon$ ar putea fi utilizată drept criteriu de oprire.

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

Intrare: $m \in N$, x, x_i, $f_i \in \mathbb{R}$, $i = 0, \ldots, m$, $\varepsilon > 0$.
*Ie*şire: $f_{i,i}$.

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

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

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

P3.1. For $j = 0, ..., i-1$ 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 şi înmulţiri;
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 diferenţe divizate, necesită un timp $\Theta(m)$, dar este instabilă.

3.1 O formulă Lagrange îmbunătăţită

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

$$\ell(x) = (x - x_0)(x - x_1) \cdots (x - x_m) \tag{6}$$

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

$$w_j = \frac{1}{\prod\limits_{k \neq j} (x_j - x_k)}, \qquad j = 0, \ldots, m, \tag{7}$$

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

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

Acum PIL se scrie

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

Avantajul este că putem calcula interpolantul Lagrange cu o formulă ce necesită $O(m^2)$ flops pentru calculul unor cantităţi 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 inserţia 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 funcţia constantă 1 obţinem

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

Împărţind (8) cu expresia de mai sus şi simplificând cu $\ell(x)$, obţinem

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

numită *formula baricentrică*.

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

3.3 Distribuţii remarcabile

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

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

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

Familia de *puncte Cebîşev* se poate obţine proiectând puncte egal spaţiate pe cercul unitate pe intervalul $[-1, 1]$. Pornind de la formula

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

se pot obţine formule explicite pentru ponderile w_j.

Punctele Cebîşev de speţa I sunt date de

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

Simplificând factorii independenţi de j se obţine

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

Punctele Cebîşev de speţa II sunt date de

$$x_j = \cos \frac{j\pi}{m}, \qquad j = 0, \ldots, m,$$

iar ponderile corespunzătoare sunt

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

Dăm codul MATLAB pentru interpolarea Lagrange baricentrică

```
function ff=baryLagrange(x,y,xx)
%BARYLAGRANGE - barycentric Lagrange interpolation
%call ff=baryLagrange(x,y,xi)
%x -  nodes
%y - function values
%xx - interpolation points
%ff - values of interpolation polynomial


%compute weights
n=length(x)-1;
w=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;
numer = zeros(size(xx));
denom = zeros(size(xx));
exact = zeros(size(xx));
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));
```

În cazul nodurilor Cebîşev de speţa a doua sursa MATLAB este

```
function ff=ChebLagrange(y,xx,a,b)
%CHEBLAGRANGE - Lagrange interpolation for Chebyshev 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;
numer = zeros(size(xx));
denom = zeros(size(xx));
exact = zeros(size(xx));
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));
```

Probleme

1. Implementaţi o rutină pentru calculul valorilor polinomului de interpolare Lagrange când se dau punctele, nodurile şi valorile funcţiei în noduri.
2. Reprezentaţi grafic polinoamele fundamentale când se dau gradul şi nodurile.
3. Reprezentaţi pe acelaşi grafic f şi $L_m f$.
4. Dându-se x, f, m şi nodurile, aproximaţi $f(x)$ utilizând interpolarea Lagrange.
5. Implementaţi metoda baricentrică.

Probleme practice

1. Datele de mai jos dau populaţia 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

Approximaţi populaţia din 1975 şi 2018.

2. Fie
$$f(x) = e^{x^2-1}.$$
Aproximaţi $f(1.25)$ utilizând valorile lui f în 1, 1.1, 1.2, 1.3 şi 1.4 şi daţi o delimitare a erorii.

3. Aproximaţi $\sqrt{115}$ cu 3 zecimale exacte prin interpolare Lagrange.

4. Daţi contraexemple pentru convergenţa interpolării Lagrange şi studiaţi-le grafic:

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

   ambele cu noduri echidistante şi noduri Cebîşev de speţa a doua.

LAB7

Metode de interpolare bazate pe diferenţe divizate

Radu Trîmbiţaş

24 martie 2020

1 Forma Newton a polinomului de interpolare Lagrange

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

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

şi formula iterativă:

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

Notaţia $f[x_0, \ldots, x_k]$ înseamnă diferenţa divizată de ordinul k a funcţiei f cu nodurile x_0, ..., x_k.

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

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

Ieşire: numerele $Q_{0,0}$, $Q_{1,1}$, ..., $Q_{n,n}$, unde $Q_{i,i} = f[x_0, ..., x_i]$.

P1. for $i = 1, 2, ..., n$ do

for j=1,2,...,n do

$$Q_{i,j} := \frac{Q_{i,j-1} - Q_{i-1,j-1}}{x_i - x_{i-j}}$$

P2. returnează $Q_{0,0}$, $Q_{1,1}$, ..., $Q_{n,n}$.

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

$$\begin{aligned}(N_4 f)(x) = 0.7651977 - 0.4837057(x-1.0) - 0.1087339(x-1.0)(x-1.3)+ \\ 0.658784(x-1.0)(x-1.3)(x-1.6)+ \\ 0.0018251(x-1.0)(x-1.3)(x-1.6)(x-1.9).\end{aligned}$$

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

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	0.6587840	0.0018251
1.3	0.6200860	-0.5489460	-0.0494433	0.0680685	
1.6	0.4554022	-0.5786120	0.0118183		
1.9	0.2818186	-0.5715210			
2.2	0.1103623				

Tabela 1: Date pentru exemplul 1

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

Intrare: x_0, x_1, ..., x_m, $f(x_0)$, $f(x_1)$, ..., $f(x_m)$, x.
Ieşire: 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 $i = 1, ..., m$ do
P3.1. $D_{i,1} := f(x_i);$
P3.2. for $j = 0, ..., i-1$ do $y_{i,j} := x_i - x_j$
P3.3. for $j = 1, ..., i$ 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.

2 Interpolare Hermite

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

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

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

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

unde h_{kj} sunt polinoamele fundamentale Hermite date de

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

şi

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

Expresia restului este:

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

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

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

$$z_{2i} = z_{2i+1} = x_i, \qquad i=0,\ldots,m.$$

Construim acum tabela de diferenţe divizate a lui f pentru nodurile z_i, $i = 0, \ldots, 2m+1$. Deoarece $z_{2i} = z_{2i+1} = x_i$ pentru $i = 0, \ldots, m$, $f[z_{2i}, z_{2i+1}]$ este o diferenţă divizată cu noduri duble şi este egală cu $f'(x_i)$, deci în locul diferenţelor divizate de ordinul I $f[z_0, z_1]$, $f[z_2, z_3]$, ..., $f[z_{2m}, z]$ vom utiliza derivatele $f'(x_0)$, $f'(x_1)$,..., $f'(x_n)$. Celelalte diferenţe divizate se obţin în mod obşnuit, aşa cum arată tabela de mai jos. Acest algoritm se poate extinde şi la alte tipuri de interpolare Hermite. Se pare că metoda este datorată lui Powell.

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

Algoritmul de interpolare Hermite. Calculează coeficienţii polinomului de interpolare Hermite (diferenţele divizate) a lui f cu nodurile duble $x_k \in [a, b]$, $k = 0, ..., m$.
Intrare. $x_0, ..., x_m, f(x_0), ..., f(x_m), f'(x_0), ..., f'(x_m)$.
Ieşire. Numerele $Q_{0,0}$, $Q_{1,1}$, ..., $Q_{2m+1,2m+1}$ (diferenţele divizate de pe prima linie a tabelei), unde

$$(H_{2m+1}f)(x) = Q_{0,0} + Q_{1,1}(x-x_0) + Q_{2,2}(x-x_0)^2 + Q_{33}(x-x_0)^2(x-x_1) + Q_{44}(x-x_0)^2(x-x_1)^2 + \cdots + Q_{2m+1,2m+1}(x-x_0)^2(x-x_1)^2 \ldots (x-x_{m-1})^2(x-x_m).$$

P1. for $i = 0, ..., m$ do paşii P2 şi 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 $i = 2, 3, ..., 2m+1$ do
for j=2,3,..., i do $Q_{i,j} = \frac{Q_{i,j-1} - Q_{i-1,j-1}}{z_i - z_{i-j}}$
P5. Extrage $Q_{0,0}$, $Q_{1,1}$, ..., $Q_{2m+1,2m+1}$. Stop.

Exemplu. Fie datele de intrare

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

Utilizând diferenţele divizate se obţine(datele de intrare sunt subliniate):

$$H_5(1.5) = 0.6200860 + (1.5 - 1.3)(-0.5220232) + (1.5 - 1.3)^2(-0.897427) + (1.5 - 1.3)^2(1.5 - 1.6)(0.0663657) + (1.5 - 1.3)^2(1.5 - 1.6)^2(1.5 - 1.9)(-0.0027738) = 0.5118277$$

1.3	<u>0.6200860</u>	<u>-0.5220232</u>	-0.0897427	0.0663657	0.0026663	-.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. Implementaţi o rutină pentru calculul valorilor polinomului de interpolare Hermite cu noduri duble, dându-se punctele în care se face evaluarea, nodurile, valorile funcţiei şi ale derivatei în noduri.

2. Reprezentaţi pe acelaşi grafic f şi polinomul său de interpolare Hermite.

3. Scrieţi o rutină care reprezintă grafic o cubică parametrică Hermite (o curbă care trece prin două puncte date şi are în acele puncte tangente date).

4 Probleme practice

1. Pentru $f(x) = e^x$ şi nodurile de interpolare 0, 1, 2, aproximaţi $f(0.25)$ prin interpolare Hermite şi comparaţi rezultatul cu cel obţinut prin interpolare Lagrange. Daţi o delimitare a erorii. Comparaţi cu rezultatul furnizat de software-ul utilizat.

2. Utilizaţi 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

Daţi o delimitare a erorii şi comparaţi-o cu eroarea exactă. Adăugaţi datele pentru nodul $x = 0.33$ şi refaceţi calculele.

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

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

5 Probleme suplimentare

1. Implementaţi interpolarea Hermite pentru noduri de multiplicitate arbitrară. Intrare: nodurile x_k, multiplicităţile r_k, valorile $f^{(j)}(x_k)$, $k = 0, \ldots, m$, $j = 0, \ldots, r_k$, punctul (sau punctele) t în care se face evaluarea. Ieşire: valorile polinomului de interpolare în punctul (sau punctele t). (Indicaţie: folosiţi pentru valorile funcţie şi derivatelor un tablou de celule.)

LAB8
Interpolare spline

Radu T. Trîmbiţaş

28 septembrie 2005

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

Pentru algoritmi a se vedea notele de curs.

Probleme

1. Pentru fiecare tip de spline scrieţi o funcţie care calculează coeficienţii spline-ului, dacă se dau nodurile şi valorile funcţiei.
2. Evaluaţi spline-ul pe o mulţime de puncte, dacă se dau nodurile, punctele şi coeficienţii.
3. Desenaţi o curbă spline cubică parametrică ce trece printr-o mulţime de puncte date.

LAB9

Aproximare prin metoda celor mai mici pătrate

Radu T. Trîmbiţaş

25 mai 2020

Fie $f \in L_w^2[a,b]$ şi $\Phi \leq L_w^2[a,b]$ 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

$$\varphi^*(x) = a_0\varphi_0(x) + \cdots + a_n\varphi_n(x), \tag{1}$$

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

Coeficienţii sunt soluţiile ecuaţiilor normale

$$a_0(\varphi_0, \varphi_k) + a_1(\varphi_1, \varphi_k) + \cdots + a_n(\varphi_n, \varphi_k) = (f, \varphi_k), \quad k = 0, \ldots, n. \tag{2}$$

Dacă sistemul $\{\varphi_k | k = 0, \ldots, n\}$ este ortogonal, coeficienţii se pot obţine cu ajutorul formulelor

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

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

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

iar in cazul discret

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

Să consideram relaţia de recurenţă pentru polinoamele ortogonale monice

$$\pi_{k+1}(x) = (x - \alpha_k)\pi_k(x) - \beta_k\pi_{k-1}(x), k = 0, 1, 2, \ldots$$
$$\pi_0(x) = 1, \pi_{-1}(x) = 0,$$

unde

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

Coeficienţii din relaţia de recurenţă (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 şi coeficienţii din relaţiile lor de recurenţă:

Polinoamele	Notaţia	Pondere	interval	α_k	β_k
Legendre	$P_n(l_n)$	1	[-1,1]	0	2 $(k{=}0)$ $(4{-}k^{-2})^{-1}$ $(k{>}0)$
Cebîşev #1	T_n	$(1{-}t^2)^{-\frac{1}{2}}$	$[-1,1]$	0	π $(k{=}0)$ $\frac{1}{2}$ $(k{=}1)$ $\frac{1}{4}$ $(k{>}0)$
Cebîşev #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

Observaţia 1 *Pentru polinoamele Jacobi avem*

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

şi

$$\beta_0 = 2^{\alpha+\beta+1} B(\alpha+1, \beta+1),$$
$$\beta_k = \frac{4k(k+\alpha)(k+\alpha+\beta)(k+\beta)}{(2k+\alpha+\beta-1)(2k+\alpha+\beta)^2(2k+\alpha+\beta+1)}, \quad k > 0.$$

Probleme propuse.

1. Să se gasească aproximanta discretă prin metoda celor mai mici pătrate pentru ponderea w(x)=1 şi baza $1, x, x^2, \ldots, x^n$.

2. Un asteroid ce orbitează în jurul Soarelui a putut fi observat timp de câteva zile înainte să dispară. Iată 10 observaţii

$x_{1:5}$	-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 doreşte calcularea traiectoriei pe baza acestor observaţii pentru a putea prevedea situaţia când orbita va fi din nou vizibilă. Se presupune un model elipsoidal pentru orbită

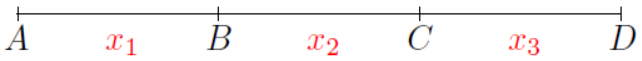
$$x^2 = ay^2 + bxy + cx + dy + e.$$

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. Realizaţi o estimare a erorii şi un test de încredere în model. Faceţi acelaşi 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



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

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

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

An	Populaţia	An	Populaţia
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

Modelaţi populaţia printr-un model polinomial de gradul 3

$$y = c_0 + c_1 t + c_2 t^2 + c_3 t^3,$$

şi printr-un model exponenţial

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

Probleme suplimentare

1. Să se gasească aproximanta discretă prin metoda celor mai mici pătrate pentru ponderea $w(x) = 1$ şi baza formata din polinoamele Cebîşev de speţa I. Produsul scalar are forma

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

unde $\xi_k,\ k = 1, \ldots, n+1$ sunt rădăcinile polinomului Cebîşev de speţa I T_{n+1}.

2. Se dă un polinom prin coeficienţii 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. (*Indicaţie*: se va folosi relaţia de recurenţă şi coeficienţii ei.)

LAB10

Integrare numerică

Radu T. Trîmbiţaş

27 aprilie 2020

1 Formule repetate

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

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 $n = 2m$; $h = (b - a)/(2m)$; $x_k = a + kh$; $k = 0, 1, ..., 2m$, 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 met(a, b, f, n) be a composite formula. o formulă repetată oarecare. Ideea este de a împărţi $[a, b]$ în subintervale şi de a folosi un număr mic de noduri pe subintervalele pe care oscilaţia lui f este lentă şi un număr mai mare de puncte pe subintervalele pe care oscilaţia lui f este mai rapidă. Algoritmul este de tip divide and conquer:

function adaptquad(a, b:real; f:funct; 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 şi pe extrapolarea Richardson. Fie $h_1 = b - a$. Vom utiliza formulele

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

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

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

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

Examplu. Aproximaţi $\int_0^\pi \sin x dx$ prin metoda lui Romberg, $\varepsilon = 10^{-2}$.
Soluţie.

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

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

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

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

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

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

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

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

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

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

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

Calculele se pot realiza în mod tabelar.

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

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 combinaţie a două aproximante de tip Simpson:

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

Relaţia

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

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

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

Pentru două subintervale se obţine

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

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

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

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

Algorithm 1 Cuadratură adaptivă bazată pe metoda lui Simpson şi extrapolare

Intrare: funcţia f, intervalul $[a, b]$, eroarea ε
Ieşire: Valoarea aproximativă a integralei

function $adquad(f, a, b, \varepsilon) : real$
$c := (a + b)/2;$
$fa = f(a);\ fb := f(b);\ fc := f(c);$
$Q := quadstep(f, a, b, \varepsilon, fa, fc, fb);$
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
$Qa := quadstep(f, a, c, \varepsilon, fa, fd, fc);$
$Qb := quadstep(f, c, b, \varepsilon, fc, fe, fb);$
$Q := Qa + Qb;$
end if
return $Q;$

5 Probleme

1. Implementaţi formula repetată a trapezului, dreptunghiului şi a lui Simpson.

2. Concepeţi o reprezentare grafică intuitivă pentru formula trapezelor şi formula repetată a lui Simpson (facultativ).

3. Implementaţi o metoda de cuadratură adaptivă pentru formula repetata a lui Simpson, una pentru metoda trapezelor şi una pentru metoda dreptunghiurilor.

4. Implementaţi metoda lui Romberg.

5. Implementaţi adquad.

6 Probleme practice

1. Generaţi formule Newton-Cotes închise pentru un număr de noduri dat.

2. Testaţi rutinele de integrare de la această temă pentru diverse funcţii a caror primitivă nu este exprimabilă prin funcţii elementare şi comparaţi rezultatele cu cele furnizate de funcţiile MATLAB `quad` şi `integral`.

LAB11

Generarea unor cuadraturi de tip Gauss

Radu T. Trîmbiţaş

25 mai 2020

Formula

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

se numeşte de tip Gauss dacă coeficienţii şi nodurile sunt alese astfel încât gradul de exactitate să fie maxim. Nodurile x_k, $k = 1, ..., n$ sunt rădăcinile polinomului

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

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

Să considerăm relaţia de recurenţă pentru polinoamele ortogonale monice

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

unde

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

Coeficienţii din relaţia de recurenţă (1) au expresia

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

Matricea Jacobi de ordinul n a funcţiei pondere w este o matrice tridiag-

onală 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

$$J_n v_k = x_k v_k, \quad v_k^T v_k = 1, \quad k = 1, 2, \ldots, n, \tag{3}$$

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

$$w_k = \beta_0 v_{k,1}^2, \quad k = 1, 2, \ldots, n \tag{4}$$

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 şi coeficienţii din relaţiile lor de recurenţă.

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

Observaţia 1 *Pentru polinoamele Jacobi avem*

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

şi

$$\begin{aligned} \beta_0 =& 2^{\alpha+\beta+1} B(\alpha + 1, \beta + 1), \\ \beta_k =& \frac{4k(k + \alpha)(k + \alpha + \beta)(k + \beta)}{(2k + \alpha + \beta - 1)(2k + \alpha + \beta)^2(2k + \alpha + \beta + 1)}, \quad k > 0. \end{aligned}$$

Polinoamele	Notaţie	Ponderea	interval	α_k	β_k
Legendre	$P_n(l_n)$	1	[-1,1]	0	2 (k=0) $(4-k^{-2})^{-1}$ (k>0)
Cebîşev #1	T_n	$(1-t^2)^{-\frac{1}{2}}$	$[-1,1]$	0	π (k=0) $\frac{1}{2}$ (k=1) $\frac{1}{4}$ (k>1)
Cebîşev #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. Implementaţi funcţii ce generează formule de cuadratură gaussiene pentru ponderile clasice date în tabela 1.

2. Aproximaţi
$$\int_{-1}^{1} \sin x^2 dx$$
printr-o formulă de cuadratură Gauss-Legendre şi comparaţi rezultatul cu cel returnat de `quad` sau `quadl`.

3. Calculaţi
$$\int_{\mathbb{R}} e^{-x^2} \sin x dx, \qquad \int_{\mathbb{R}} e^{-x^2} \cos x dx$$
utilizând o formulă de cuadratură Gauss-Hermite.

LAB12

Rezolvarea numerică a ecuaţiilor neliniare

Radu T. Trîmbiţaş

28 aprilie 2020

1 Ecuaţii neliniare în $\mathbb{R}$

Fie $f : \mathbb{R} \to \mathbb{R}$. Dorim să aproximăm o soluţie sau toate soluţiile ecuaţiei $f(x) = 0$. Vom prezenta câteva metode importante.

1.1 Metoda Newton-Raphson (a tangentei)

Determină o soluţie a ecuaţiei $f(x) = 0$, dându-se o aproximaţie iniţială p_0.

Intrare. Funcţia f, derivata sa f', o aproximaţie iniţială p_0; eroarea ε; numărul maxim de iteraţii N_0.

Ieşire. O soluţie aproximativă p sau un mesaj de eroare.

P1. $i := 1$.

P2. While $i \leq N_0$ execută paşii 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. {eşec} eroare('precizia nu poate fi atinsă cu numărul dat de iteraţii'). Stop.

Observaţie. În plus faţă 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 ecuaţia $x = \cos x$. Punem $f(x) = \cos x - x$. Ecuaţia noastră are o soluţie în $[0, \pi/2]$, care poate fi obţinută ca punct fix al lui $g(x) = \cos x$ (vezi figura 1). Deoarece $f'(x) = -\sin x - 1$, iteraţia Newton este

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

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

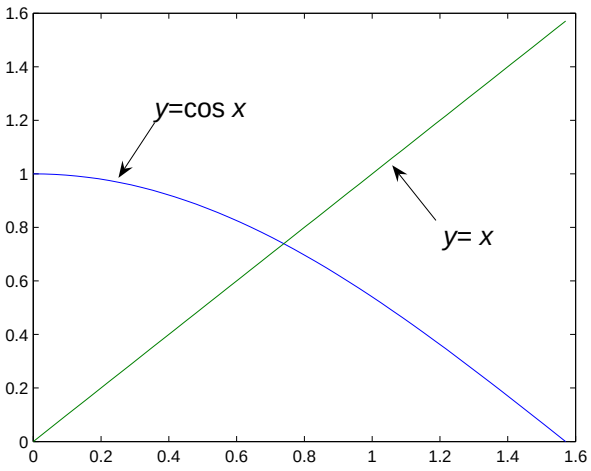


Figura 1: Ecuaţia $\cos x = x$

1.2 Metoda secantei

Determină o soluţie a ecuaţiei $f(x) = 0$, dându-se aproximaţiile iniţiale p_0 şi p_1.

Intrare. Funcţia f, aproximaţiile iniţiale p_0 şi p_1; eroarea ε; numărul maxim de iteraţii N_0.

Ieşire. Soluţia 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 paşii 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. {eşec} eroare('precizia nu poate fi atinsă cu numărul dat de iteraţii'). Stop.

Examplu numeric. Considerăm din nou ecuaţia $\cos x - x = 0$. 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 soluţie a ecuaţiei $p = g(p)$, dându-se o aproximaţie iniţială p_0.

Intrare. Funcţia f, valoarea de pornire p_0; eroarea ε; numărul maxim de iteraţii N_0.

Ieşire. Soluţia aproximativă p sau un mesaj de eroare.

P1. $i := 1$.

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

P3.

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

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

P5. $i := i + 1$;

P6. $p_0 := p$; {actualizează p}

P7. {eşec} eroare('precizia nu poate fi atinsă cu numărul dat de iteraţii'). Stop.

Exemplu numeric. Pentru a rezolva ecuaţia $x^3 + 4x^2 - 10 = 0$, o rescriem sub forma $x^3 + 4x^2 = 10$ şi obţinem

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

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) Implementaţi metodele Newton, secantă, Steffensen.

2 Sisteme de ecuaţii neliniare

Fie

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

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

2.1 Metoda lui Newton

Formula iterativă este

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

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

Algoritmul.

Intrare. f, f', ε(toleranţa), valoarea de pornire $x^{(0)}$ şi numărul maxim de iteraţii N.

Ieşire. O aproximare a rădăcinii sau un mesaj de eroare.

$n := 0$;

repeat

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

until

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

or „s-a depăşit numărul maxim de iteraţii“.

2.2 Metoda aproximaţiilor succesive

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

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

Algoritmul.

Intrare: f, ε (toleranţa), valoarea de pornire x_0 şi numărul maxim de iteraţii N.

Ieăşire: O aproximare a rădăcinii sau un mesaj de eroare: "precizia dorită nu poate fi atinsă în N iteraţii".

Repeat

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

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

2.3 Probleme

1. Implementaţi metoda lui Newton şi metoda aproximaţiilor succesive.

2.4 Probleme practice

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

2. Rezolvaţi numeric sistemul
$$\begin{cases} 9x^2 + 36y^2 + 4z^2 - 36 = 0, \\ x^2 - 2y^2 - 20z = 0, \\ x^2 - y^2 + z^2 = 0 \end{cases}$$
utilizând metoda lui Newton şi metoda aproximaţiilor succesive. *Indicaţie.* Sunt 4 soluţii. Valori bune de pornire $[\pm 1, \pm 1, 0]^T$.