Bor Plestenjak

Numerične metode 2 (finančna matematika)

delovna verzija

verzija: 11. februar 2013

Kazalo

Poglavje 1

Nesimetrični problem lastnih vrednosti

1.1 Uvod

Dana je matrika $A \in \mathbb{R}^{n\times n}$. Če neničelen vektor $x \in \mathbb{C}^n$ in skalar $\lambda \in \mathbb{C}$ zadoščata enačbi $Ax = \lambda x$, potem je λ *lastna vrednost*, x pa *(desni) lastni vektor* matrike A. Neničelen vektor $y \in \mathbb{C}^n$, za katerega velja $y^H A = \lambda y^H$, je *levi lastni vektor* matrike A. Levi in desni lastni vektorji, ki pripadajo različnim lastnim vrednostim, so med seboj ortogonalni, saj velja naslednja lema.

Lema 1.1 *Naj bosta λ in μ različni lastni vrednosti matrike A. Če je x desni lastni vektor, ki pripada λ, y pa levi lastni vektor, ki pripada μ, potem sta x in y ortogonalna.*

Dokaz. Enakost $Ax = \lambda x$ pomnožimo z leve z y^H, enakost $y^H A = \mu y^H$ pa z desne z x. Dobimo

$$y^H Ax = \lambda y^H x = \mu y^H x,$$

to pa je zaradi $\lambda \neq \mu$ lahko izpolnjeno le v primeru, ko je $y^H x = 0$. ■

V posebnem primeru, ko je matrika A simetrična, iz zgornje leme sledi, da so lastni vektorji, ki pripadajo različnim lastnim vrednostim, paroma ortogonalni. To je posledica tega, da so za simetrično matriko desni lastni vektorji enaki levim.

Pri algoritmih za računanje lastnih vektorjev zadošča, da obravnavamo le desne lastne vektorje. Namreč, če je y levi lastni vektor matrike A za lastno vrednost λ, potem je y desni lastni vektor matrike A^H za lastno vrednost $\overline{\lambda}$.

Pravimo, da se da matriko A diagonalizirati, če obstajata nesingularna matrika $X = [x_1 \ \cdots \ x_n]$ in diagonalna matrika $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$, da je $A = X\Lambda X^{-1}$. V tem primeru je $Ax_i = \lambda_i x_i$ za $i = 1, \ldots, n$, kar pomeni, da so stolpci matrike X lastni vektorji, na diagonali matrike Λ pa ležijo lastne vrednosti.

Če je S nesingularna matrika, potem pravimo, da sta matriki A in $B = S^{-1}AS$ *podobni*. V tem primeru imata A in B iste lastne vrednosti. Velja, da je x desni lastni vektor za matriko A natanko tedaj, ko je $S^{-1}x$ desni lastni vektor za matriko B.

Lastne vrednosti matrike A so ničle karakterističnega polinoma $p(\lambda) = \det(A - \lambda I)$. Vsaka $n \times n$ matrika A ima tako n lastnih vrednosti $\lambda_1, \ldots, \lambda_n$, pri čemer večkratne ničle štejemo

večkrat. Če je λ lastna vrednost matrike A, potem je njena *algebraična večkratnost* enaka večkratnosti λ kot ničle karakterističnega polinoma matrike A, *geometrijska večkratnost* pa je enaka dimenziji podprostora $\ker(A - \lambda I)$ in je vedno manjša ali enaka algebraični večkratnosti. Če je algebraična večkratnost lastne vrednosti ena, potem pravimo, da je lastna vrednost enostavna.

Ker se ničel splošnega polinoma stopnje pet ali več ne da izračunati drugače kot numerično, to velja tudi za lastne vrednosti in direktne metode za računanje lastnih vrednosti ne obstajajo. Lastne vrednosti tako vedno računamo z iterativnimi postopki.

Računanje lastnih vrednosti preko ničel eksplicitno izračunanega karakterističnega polinoma v splošnem ni priporočljivo. Algoritmi za računanje koeficientov karakterističnega polinoma namreč niso stabilni, vemo pa, da so ničle polinoma lahko zelo občutljive na motnje koeficientov, kot kaže npr. Wilkinsonov primer iz zgleda **??**.

Zgled 1.1 *Pri eksplicitni uporabi karakterističnega polinoma lahko izgubimo natančnost. Naj bo*

$$A = \begin{bmatrix} 1 & \epsilon \\ \epsilon & 1 \end{bmatrix}$$

za $\epsilon = \sqrt{u}/2$, *kjer je* u *osnovna zaokrožitvena napaka. Če izračunamo karakteristični polinom, potem zaradi zaokroževanja dobimo* $\det(A - \lambda I) = \lambda^2 - 2\lambda + 1$. *Na ta način bi izračunali dvojno lastno vrednost* 1, *pravi lastni vrednosti matrike* A *pa sta* $1 - \epsilon$ *in* $1 + \epsilon$. □

Računanje lastnih vrednosti preko eksplicitno izračunanega karakterističnega polinoma torej ni numerično stabilno. V nadaljevanju bomo spoznali več različnih stabilnih algoritmov za izračun lastnih vrednosti in vektorjev.

1.2 Schurova forma

Vemo, da za vsako $n \times n$ matriko A obstajata taka nesingularna matrika X in bločno diagonalna matrika $J = \operatorname{diag}(J_1, \ldots, J_k)$, ki ji pravimo *Jordanova*[1] *forma*, da je $X^{-1}AX = J$, kjer je

$$J_i = \begin{bmatrix} \lambda_i & 1 & & \\ & \lambda_i & \ddots & \\ & & \ddots & 1 \\ & & & \lambda_i \end{bmatrix}$$

Jordanova kletka velikosti $m_i \times m_i$ za $i = 1, \ldots, k$ in $n = m_1 + \cdots + m_k$.

Jordanova forma pove veliko o matriki A. Iz nje lahko npr. preberemo vse lastne vrednosti matrike in njihove algebraične in geometrijske večkratnosti. Zelo pomembna je tudi za računanje vrednosti funkcij matrik. Tako ima npr. rešitev sistema diferencialnih enačb s konstantnimi koeficienti $y'(t) = Ay(t)$ obliko $y(t) = y(t_0)e^{A(t-t_0)}$. Za rešitev je potrebno znati izračunati matriko e^A, ki je definirana z razvojem v vrsto

$$e^A = \sum_{j=0}^{\infty} \frac{1}{j!} A^j.$$

[1] Francoski matematik Marie Ennemond Camille Jordan (1838–1922) jo je objavil leta 1870.

Če je funkcija f definirana in dovoljkrat zvezno odvedljiva v lastnih vrednostih matrike A, potem s pomočjo Jordanove forme $A = XJX^{-1}$ velja enakost $f(A) = Xf(J)X^{-1}$, kjer je $f(J) = \mathrm{diag}(f(J_1), \ldots, f(J_k))$ in

$$f(J_i) = \begin{bmatrix} f(\lambda_i) & f'(\lambda_i) & \cdots & \frac{f^{(m_i-1)}(\lambda_i)}{(m_i-1)!} \\ & f(\lambda_i) & \ddots & \vdots \\ & & \ddots & f'(\lambda_i) \\ & & & f(\lambda_i) \end{bmatrix}$$

za $i = 1, \ldots, k$.

Na žalost je Jordanova forma zelo občutljiva in neprimerna za numerično računanje. Ker ni zvezna funkcija elementov matrike A, jo lahko v primeru večkratnih lastnih vrednosti majhne motnje popolnoma spremenijo. Zato računanje funkcij matrik preko Jordanove forme ni stabilno.

Zgled 1.2 *Matrika*

$$A = \begin{bmatrix} 0 & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ & & & 0 \end{bmatrix}$$

je že kar v Jordanovi formi z eno samo kletko $n \times n$, za poljuben $\epsilon > 0$ pa ima Jordanova forma matrike

$$A(\epsilon) = \begin{bmatrix} 0 & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ \epsilon & & & 0 \end{bmatrix}$$

n kletk velikosti 1×1 z lastnimi vrednostmi $\sqrt[n]{\epsilon}$. □

Računanje Jordanove forme tudi ni obratno stabilno. Denimo, da smo za matriko A numerično izračunali $\widetilde{X}$ in $\widetilde{J}$. Sedaj nas zanima, ali je $\widetilde{X}^{-1}(A + \delta A)\widetilde{X} = \widetilde{J}$ za neko matriko δA, ki je blizu 0. Pri tem predpostavimo, da je $\widetilde{X}$ točna, za $\widetilde{J}$ pa velja $\widetilde{J} = J + \delta J$. Iz $X^{-1}(A + \delta A)X = J + \delta J$ sledi $X^{-1}\delta A X = \delta J$, od tod pa lahko ocenimo

$$\|\delta A\| \leq \|X\| \cdot \|X^{-1}\| \cdot \|\delta J\| = \kappa(X)\|\delta J\|.$$

Ker je v splošnem občutljivost $\kappa(X)$ lahko poljubno velika, računanje Jordanove forme ni obratno stabilno.

Za stabilnost bi bilo bolje, če bi bila prehodna matrika X kar unitarna. Izkaže se, da z unitarno podobnostno transformacijo lahko matriko na stabilen način transformiramo v zgornjo trikotno matriko.

Izrek 1.2 (Schur) [2] *Za vsako matriko A obstajata unitarna matrika U in zgornja trikotna matrika T, da je $U^H A U = T$.*

[2] Izrek je leta 1909 zapisal nemški matematik Issai Schur (1875–1941), ki je znan tudi po svojih rezultatih iz teorije grup. S svojim mentorjem Frobeniusom sta bila med prvimi, ki so se ukvarjali s teorijo matrik.

Razcep $A = UTU^H$ iz zgornjega izreka imenujemo *Schurov razcep*, samo zgornjo trikotno matriko T pa *Schurova forma*. Vemo, da ima realna matrika lahko kompleksne lastne vrednosti in vektorje, ki nastopajo v konjugiranih parih. Zaradi tega sta matriki T in U iz Schurovega razcepa tudi za realno matriko lahko kompleksni.

Dokaz. Naredimo indukcijo po n. Za $n = 1$ izrek očitno velja, saj vzamemo $U = [1]$ in $T = A$.

Predpostavimo, da izrek velja za $n-1$ in ga dokažimo za n. Naj bo λ lastna vrednost matrike A in x njen normiran lastni vektor. Obstaja taka unitarna matrika U_1, da je $U_1 e_1 = x$ (skonstruiramo jo lahko npr. kar s kompleksnim Householderjevim zrcaljenjem). Matrika $B = U_1^H A U_1$ ima obliko

$$B = \begin{array}{c} \\ 1 \\ n-1 \end{array} \begin{array}{c} \begin{array}{cc} 1 & n-1 \end{array} \\ \begin{bmatrix} \lambda & \times \cdots \times \\ 0 & C \end{bmatrix} \end{array},$$

saj je $Be_1 = U_1^H A U_1 e_1 = U_1^H A x = U_1^H \lambda x = \lambda e_1$. Po indukcijski predpostavki obstaja Schurova forma za $(n-1) \times (n-1)$ matriko C. Torej obstaja taka unitarna matrika V_1, da je $V_1^H C V_1 = T_1$ zgornja trikotna matrika. Sedaj je

$$\underbrace{\begin{bmatrix} 1 & 0 \\ 0 & V_1^H \end{bmatrix}}_{V^H} B \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & V_1 \end{bmatrix}}_{V} = \begin{bmatrix} \lambda & \times \cdots \times \\ 0 & T_1 \end{bmatrix}$$

zgornja trikotna matrika in $\underbrace{V^H U_1^H}_{U^H} A \underbrace{U_1 V}_{U}$ je Schurova forma matrike A. ∎

Tako kot Jordanova tudi Schurova forma ni enolična, saj je npr. vrstni red diagonalnih elementov λ_i lahko poljuben.

V primeru realne matrike se lahko izognemo kompleksnim matrikam, če dopustimo, da v Schurovem razcepu namesto zgornje trikotne matrike dobimo *kvazi zgornjo trikotno matriko*. Ta ima na diagonali lahko bloke 2×2, v katerih so skriti konjugirani pari kompleksnih lastnih vrednosti.

Izrek 1.3 *Za vsako realno matriko A obstajata ortogonalna matrika Q in kvazi zgornja trikotna matrika T, da je $Q^T A Q = T$ (realna Schurova forma).*

Dokaz. Spet uporabimo indukcijo. Če je $\lambda \in \mathbb{R}$, lahko nadaljujemo tako kot pri dokazu izreka 1.2, zato predpostavimo, da velja $\lambda \notin \mathbb{R}$. Potem imamo poleg lastnega para (λ, x) tudi lastni par $(\overline{\lambda}, \overline{x})$. Če definiramo realna vektorja

$$x_R = \frac{1}{2}(x + \overline{x}), \quad x_I = \frac{1}{2i}(x - \overline{x}),$$

potem obstaja taka ortogonalna matrika

$$U = \begin{array}{c} \begin{array}{cc} 2 & n-2 \end{array} \\ \begin{bmatrix} U_1 & U_2 \end{bmatrix} \end{array},$$

da je $\mathrm{Lin}(U_1) = \mathrm{Lin}(\{x_R, x_I\}) = \mathrm{Lin}(\{x, \overline{x}\})$. Ker je $\mathrm{Lin}(U_1)$ invarianten podprostor za A, velja

$$U^T A U = \begin{array}{c} \\ 2 \\ n-2 \end{array} \begin{array}{c} \begin{array}{cc} 2 & n-2 \end{array} \\ \begin{bmatrix} B & \times \cdots \times \\ 0 & C \end{bmatrix} \end{array},$$

kjer sta λ in $\overline{\lambda}$ lastni vrednosti 2×2 matrike B, preostale lastne vrednosti pa so v matriki C. Nadaljujemo podobno kot pri dokazu izreka 1.2. ∎

1.3 Teorija motenj

Kot pri ostalih problemih, bi tudi tu radi vedeli, koliko so občutljive lastne vrednosti in lastni vektorji. Zanima nas, kaj se z njimi dogaja, ko matriko zmotimo. Vemo, da so lastne vrednosti zvezne funkcije elementov matrike, saj so ničle karakterističnega polinoma, ničle polinoma pa so zvezne funkcije koeficientov polinoma. Če se da matriko diagonalizirati, potem naslednji izrek pove, da je sprememba lastnih vrednosti omejena z občutljivostjo matrike lastnih vektorjev.

Izrek 1.4 (Bauer–Fike) [3] *Predpostavimo, da se da matriko A diagonalizirati kot $A = X\Lambda X^{-1}$, kjer je $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ diagonalna matrika lastnih vrednosti. Potem vse lastne vrednosti matrike $A + \epsilon E$ ležijo v uniji n krogov*

$$K_i = \{z \in \mathbb{C} : |z - \lambda_i| \leq \epsilon \kappa(X) \|E\|\}, \quad i = 1, \ldots, n,$$

kjer za normo lahko vzamemo katerokoli izmed norm $\|.\|_1$, $\|.\|_2$ ali $\|.\|_\infty$.

Dokaz. Naj bo $\lambda(\epsilon)$ lastna vrednost matrike $A + \epsilon E$. Predpostavimo lahko, da se $\lambda(\epsilon)$ razlikuje od vseh lastnih vrednosti $\lambda_1, \ldots, \lambda_n$, saj sicer izrek očitno drži. Matrika $A + \epsilon E - \lambda(\epsilon) I$ je singularna. Zapišemo lahko

$$\begin{aligned} X^{-1}(A + \epsilon E - \lambda(\epsilon) I) X &= \Lambda - \lambda(\epsilon) I + \epsilon X^{-1} E X \\ &= (\Lambda - \lambda(\epsilon) I)\Big(I + \epsilon(\Lambda - \lambda(\epsilon) I)^{-1} X^{-1} E X\Big). \end{aligned}$$

Ker je matrika $\Lambda - \lambda(\epsilon) I$ nesingularna, mora biti $I + \epsilon(\Lambda - \lambda(\epsilon) I)^{-1} X^{-1} E X$ singularna matrika. To pomeni (uporabimo lemo **??**), da je

$$1 \leq \|\epsilon(\Lambda - \lambda(\epsilon) I)^{-1} X^{-1} E X\| \leq \epsilon \|(\Lambda - \lambda(\epsilon) I)^{-1}\| \|X^{-1}\| \|E\| \|X\|.$$

Iz

$$\|(\Lambda - \lambda(\epsilon) I)^{-1}\| = \frac{1}{\min_{i=1,\ldots,n} |\lambda_i - \lambda(\epsilon)|}$$

sledi

$$\min_{i=1,\ldots,n} |\lambda_i - \lambda(\epsilon)| \leq \epsilon \kappa(X) \|E\|.$$

∎

[3] Rezultat sta leta 1960 objavila nemški matematik Friedrich Ludwig Bauer (r. 1924) in Charles Theodore Fike. Bauer je znan predvsem po svojem delu na področju računalništva, kjer je vpeljal podatkovno strukturo sklad, pomembno pa je tudi sodeloval pri razvoju prvih programskih jezikov v 60. letih prejšnjega stoletja.

Opomba 1.1 *Če unija krogov razpade na povezane komponente, potem iz zveznosti lastnih vrednosti sledi, da vsaka komponenta vsebuje natanko toliko lastnih vrednosti, kolikor krogov jo sestavlja.*

V primeru, ko je matrika simetrična, lahko lastne vektorje izberemo tako, da tvorijo ortonormirano bazo. Spektralna občutljivost matrike lastnih vektorjev je potem 1 in iz Bauer–Fikeovega izreka dobimo naslednjo posledico.

Posledica 1.5 *Če sta matriki A in E simetrični, potem pri predpostavkah izreka 1.4 za vsako lastno vrednost $\lambda(\epsilon)$ matrike $A+\epsilon E$ velja*

$$\min_{i=1,\ldots,n} |\lambda(\epsilon)-\lambda_i| \leq \epsilon \|E\|_2.$$

Bauer–Fikeov izrek je ponavadi preveč splošen, saj za motnje vseh lastnih vrednosti uporabi isto zgornjo mejo. V resnici so lahko posamezne lastne vrednosti bolj občutljive od ostalih. Naslednji izrek pove, od česa je odvisna občutljivost enostavne lastne vrednosti.

Izrek 1.6 *Naj bo λ_i enostavna lastna vrednost matrike A z normiranima levim in desnim lastnim vektorjem y_i in x_i. Če je $\lambda_i+\delta\lambda_i$ ustrezna lastna vrednost zmotene matrike $A+\delta A$, potem velja*

$$\lambda_i+\delta\lambda_i = \lambda_i + \frac{y_i^H \delta A x_i}{y_i^H x_i} + \mathcal{O}(\|\delta A\|^2).$$

Izraz ustrezna lastna vrednost v zgornjem izreku pomeni, da si izberemo tisto lastno vrednost zmotene matrike, za katero velja $\lim_{\|\delta A\|\to 0}(\delta\lambda_i)=0$.

Dokaz. Velja $Ax_i=\lambda_i x_i$ in $(A+\delta A)(x_i+\delta x_i)=(\lambda_i+\delta\lambda_i)(x_i+\delta x_i)$, kjer smo z $x_i+\delta x_i$ označili lastni vektor zmotene matrike. Če zanemarimo člene (od tod na koncu dobimo $\mathcal{O}(\|\delta A\|^2)$), ki vsebujejo produkte dveh majhnih popravkov in pomnožimo enačbo z leve z y_i^H, dobimo

$$\delta\lambda_i = \frac{y_i^H \delta A x_i}{y_i^H x_i}.$$

■

Definicija 1.7 *Naj bo λ_i enostavna lastna vrednost, x_i in y_i pa pripadajoča desni in levi lastni vektor. Če definiramo*

$$s_i := \frac{y_i^H x_i}{\|x_i\|_2 \|y_i\|_2},$$

potem je $|s_i|^{-1}$ občutljivost *enostavne lastne vrednosti λ_i. Če je λ_i večkratna lastna vrednost, je njena občutljivost neskončna.*

Če je matrika A simetrična, potem imajo vse enostavne lastne vrednosti najmanjšo možno občutljivost 1, saj so levi lastni vektorji enaki desnim.

Zgled 1.3 *Matrika*

$$A_1 = \begin{bmatrix} -149 & -50 & -154 \\ 537 & 180 & 546 \\ -27 & -9 & -25 \end{bmatrix}$$

ima eksaktne lastne vrednosti $1, 2, 3$. *V Matlabu z ukazom* `eig(A)`, *ki uporablja obratno stabilen algoritem, ter z računanjem v dvojni natančnosti, dobimo*

$$\begin{aligned} \widehat{\lambda}_1 &= 1.000000000010722 \\ \widehat{\lambda}_2 &= 1.999999999991790 \\ \widehat{\lambda}_3 &= 2.999999999997399. \end{aligned}$$

Matrika Godunova[4]

$$A_2 = \begin{bmatrix} 289 & 2064 & 336 & 128 & 80 & 32 & 16 \\ 1152 & 30 & 1312 & 512 & 288 & 128 & 32 \\ -29 & -2000 & 756 & 384 & 1008 & 224 & 48 \\ 512 & 128 & 640 & 0 & 640 & 512 & 128 \\ 1053 & 2256 & -504 & -384 & -756 & 800 & 208 \\ -287 & -16 & 1712 & -128 & 1968 & -30 & 2032 \\ -2176 & -287 & -1565 & -512 & -541 & -1152 & -289 \end{bmatrix}$$

ima eksaktne lastne vrednosti $-4, -2, -1, 0, 1, 2, 4$. *V tem primeru Matlab izračuna naslednje približke za lastne vrednosti:*

$$\begin{aligned} \widehat{\lambda}_1 &= 5.0968 + 1.8932i \\ \widehat{\lambda}_2 &= 5.0968 - 1.8932i \\ \widehat{\lambda}_3 &= 1.2157 + 4.3784i \\ \widehat{\lambda}_4 &= 1.2157 - 4.3784i \\ \widehat{\lambda}_5 &= -5.6738 \\ \widehat{\lambda}_6 &= -3.4756 + 3.4463i \\ \widehat{\lambda}_7 &= -3.4756 - 3.4463i. \end{aligned}$$

V obeh primerih so izračunane lastne vrednosti točne lastne vrednosti malo zmotene matrike, saj Matlab uporablja obratno stabilen algoritem. Medtem, ko so lastne vrednosti matrike A_1 *izračunane zelo natančno, so med izračunanimi in točnimi lastnimi vrednostmi matrike* A_2 *zelo velike razlike. Te so posledica tega, da so občutljivosti lastnih vrednosti matrike* A_2 *velikostnega reda* 10^{13}. *Če to primerjamo z matriko* A_1, *kjer imajo lastne vrednosti občutljivosti velikostnega reda* 10^2, *je očitno, da občutljivost lastne vrednosti pomembno vpliva na natančnost izračunanih približkov.* □

V primeru, ko se da matriko diagonalizirati, lahko kaj povemo tudi o občutljivosti lastnih vektorjev.

Izrek 1.8 *Naj bo* $A = X\Lambda X^{-1} = Y^H \Lambda Y^{-H}$, *kjer je* $X = [x_1 \ \cdots \ x_n]$ *matrika normiranih desnih lastnih vektorjev,* $Y = [y_1 \ \cdots \ y_n]$ *matrika normiranih levih lastnih vektorjev in* $\Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$ *diagonalna matrika lastnih vrednosti. Če je* λ_i *enostavna lastna vrednost in* $\lambda_i + \delta\lambda_i$ *ustrezna lastna vrednost zmotene matrike* $A + \delta A$ *s pripadajočim lastnim vektorjem* $x_i + \delta x_i$, *potem velja*

$$x_i + \delta x_i = x_i + \sum_{\substack{j=1 \\ j \neq i}}^{n} \frac{y_j^H \delta A x_i}{(\lambda_i - \lambda_j) s_j} x_j + \mathcal{O}(\|\delta A\|^2).$$

[4] Po ruskem matematiku Sergeju Konstantinoviču Godunovu (r. 1929).

Dokaz. Zmoteni lastni vektor lahko izrazimo v bazi lastnih vektorjev matrike A kot

$$x_i + \delta x_i = x_i + \sum_{\substack{j=1 \\ j \neq i}}^{n} \alpha_j x_j.$$

V enakost $(A + \delta A)(x_i + \delta x_i) = (\lambda_i + \delta\lambda_i)(x_i + \delta x_i)$ vstavimo zgornji razvoj, jo pomnožimo z leve z y_k^H, kjer je $k \neq i$, nato pa tako kot prej zanemarimo člene, ki vsebujejo produkte dveh majhnih popravkov. Dobimo

$$\alpha_k = \frac{y_k^H \delta A x_i}{(\lambda_i - \lambda_k) s_k},$$

pri čemer smo upoštevali, da je levi lastni vektor y_k ortogonalen na vse desne lastne vektorje x_j, kjer je $k \neq j$. ∎

Lema 1.9 *Če je λ_i enostavna lastna vrednost, potem je $s_i \neq 0$.*

Dokaz. Brez škode za splošnost lahko privzamemo, da je $i = 1$. Naj bo sedaj $s_1 = 0$, $\|x_1\|_2 = \|y_1\|_2 = 1$ pa naj bosta ustrezni levi in desni lastni vektor. U naj bo taka unitarna matrika, da je $Ue_1 = x_1$. Potem je matrika $B = U^H A U$ oblike (glej dokaz izreka 1.2)

$$B = \begin{array}{c} \\ 1 \\ n-1 \end{array} \begin{array}{c} \begin{array}{cc} 1 & n-1 \end{array} \\ \begin{bmatrix} \lambda & \times \cdots \times \\ 0 & C \end{bmatrix} \end{array}.$$

Iz enakosti $Ax_1 = \lambda_1 x_1$, $y_1^H A = \lambda_1 y_1^H$ in $s_1 = y_1^H x_1 = 0$ dobimo

$$\begin{aligned} U^H A U e_1 &= U^H \lambda_1 x_1 = \lambda_1 e_1, \\ (y_1^H U) U^H A U &= \lambda_1 (y_1^H U), \qquad (1.1) \\ y_1^H U e_1 &= 0. \qquad (1.2) \end{aligned}$$

Iz (1.2) sledi, da je $y_1^H U$ oblike $[0\ z_1^H]$, ko pa to vstavimo v (1.1), dobimo $[0\ z_1^H]B = \lambda_1 [0\ z_1^H]$ oziroma $z_1^H C = \lambda_1 z_1^H$. Ker je λ_1 lastna vrednost matrike C, ima matrika A vsaj dvojno lastno vrednost λ_1. ∎

V primeru večkratne lastne vrednosti lahko vektorja x_i in y_i določimo tako, da bosta ortogonalna, ni pa to nujno res za poljubno izbrana lastna vektorja večkratne lastne vrednosti.

Izrek 1.10 *Naj bo $A = X\Lambda X^{-1} = Y^H \Lambda Y^{-H}$, kjer je $X = [x_1 \ \cdots \ x_n]$ matrika normiranih desnih lastnih vektorjev, $Y = [y_1 \ \cdots \ y_n]$ matrika normiranih levih lastnih vektorjev in $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ diagonalna matrika lastnih vrednosti. Potem velja*

$$X^{-1} = \begin{bmatrix} \frac{1}{s_1} y_1^H \\ \vdots \\ \frac{1}{s_n} y_n^H \end{bmatrix}.$$

Dokaz. $Y^H A = \Lambda Y^H$, po drugi strani pa $X^{-1}A = \Lambda X^{-1}$. Od tod sledi, da so stolpci matrike X^{-T} levi lastni vektorji. Torej je

$$X^{-1} = \begin{bmatrix} c_1 y_1^H \\ \vdots \\ c_n y_n^H \end{bmatrix}$$

za neke konstante $c_1, \ldots, c_n$. Zaradi $XX^{-1} = I$ mora veljati $c_i = \frac{1}{s_i}$. ∎

Lema 1.11 *Matrika ne more imeti natanko ene zelo občutljive lastne vrednosti.*

Dokaz. Predpostavimo lahko, da so vse lastne vrednosti enostavne, saj v nasprotnem primeru že imamo zelo občutljiv par. Naj bo $A = X\Lambda X^{-1} = Y^H \Lambda Y^{-H}$, kjer je $X = [x_1 \ \cdots \ x_n]$ matrika normiranih desnih lastnih vektorjev, $Y = [y_1 \ \cdots \ y_n]$ matrika normiranih levih lastnih vektorjev in $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ diagonalna matrika lastnih vrednosti. Po izreku 1.10 je

$$X \begin{bmatrix} \frac{1}{s_1} y_1^H \\ \vdots \\ \frac{1}{s_n} y_n^H \end{bmatrix} = I$$

oziroma

$$\sum_{i=1}^{n} \frac{x_i y_i^H}{s_i} = I.$$

Denimo, da je $|s_1|^{-1}$ največja občutljivost. Ocenimo lahko

$$\frac{1}{|s_1|} \leq 1 + \sum_{j=2}^{n} \frac{1}{|s_j|},$$

od tod pa je očitno, da mora biti vsaj ena izmed preostalih občutljivosti tudi zelo velika. ∎

Zgornjo lemo si lahko razlagamo tudi tako, da velika občutljivost lastne vrednosti pomeni, da je blizu večkratne lastne vrednosti, to pa seveda pomeni, da obstaja vsaj še ena taka lastna vrednost.

Pri občutljivosti linearnega sistema smo videli, da je recipročna vrednost občutljivosti matrike enaka oddaljenosti od najbližjega singularnega sistema. Podobno tudi sedaj velja, da je občutljivost enostavne lastne vrednosti povezana z oddaljenostjo od najbližje matrike z večkratno lastno vrednostjo. Dokaz naslednjega izreka lahko najdete npr. v [8].

Izrek 1.12 *Naj bo λ enostavna lastna vrednost matrike A z normiranima levim in desnim lastnim vektorjem y in x in naj bo $|s| < 1$, kjer je $s = y^H x$. Potem obstaja matrika $A + \delta A$ z večkratno lastno vrednostjo λ in*

$$\frac{\|\delta A\|_2}{\|A\|_2} \leq \frac{|s|}{\sqrt{1 - |s|^2}}.$$

Kaj se zgodi z občutljivostjo lastne vrednosti pri podobnostnih transformacijah? Naj bo λ enostavna lastna vrednost matrike A z normiranima levim in desnim lastnim vektorjem y in x.

Naj bo $B = S^{-1}AS$, kjer je S nesingularna matrika. Če je $\widetilde{s}$ občutljivost λ kot lastne vrednosti matrike B, potem lahko izpeljemo zvezo

$$\frac{|s|}{|\widetilde{s}|} = \frac{\|S^H y\|\|S^{-1}x\|}{\|y\|\|x\|}$$

in ocenimo

$$\frac{1}{\kappa(S)}|\widetilde{s}| \leq |s| \leq \kappa(S)|\widetilde{s}|.$$

Občutljivost lastne vrednosti se torej v najboljšem primeru zmanjša za $\kappa(S)$, v najslabšem primeru pa se za isti faktor poveča. Če je S ortogonalna matrika, potem se občutljivost ne spremeni, zato so tovrstne transformacije stabilne.

1.4 Potenčna metoda

Pri prvem algoritmu za računanje lastnih vrednosti in vektorjev ne potrebujemo drugega kot množenje z matriko A. Denimo, da izberemo normiran začetni vektor z_0 in nato za $k = 0, 1, \ldots$ generiramo vektorje po naslednjem predpisu:

$$y_{k+1} = Az_k, \qquad z_{k+1} = \frac{y_{k+1}}{\|y_{k+1}\|}. \tag{1.3}$$

Dobimo zaporedje normiranih vektorjev z_k, za katere se izkaže, da ko gre k proti neskončno, skonvergirajo proti lastnemu vektorju matrike A.

Izrek 1.13 *Naj bo λ_1 dominantna lastna vrednost matrike A, kar pomeni*

$$|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_n|.$$

Potem, za splošen začetni vektor z_0, ko gre k proti neskončnosti, vektorji z_k, izračunani po predpisu (1.3), po smeri konvergirajo proti lastnemu vektorju za λ_1.

Dokaz. Izrek sicer velja za splošno matriko, dokazali pa ga bomo le za primer, ko se da matriko diagonalizirati. Naj velja $A = X\Lambda X^{-1}$, kjer je $X = [x_1 \ \cdots \ x_n]$ in $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$. Začetni vektor z_0 lahko razvijemo po lastnih vektorjih kot

$$z_0 = \sum_{i=1}^{n} \alpha_i x_i.$$

Potem pri pogoju $\alpha_1 \neq 0$ velja

$$z_k = \frac{A^k z_0}{\|A^k z_0\|_\infty} = \frac{\alpha_1 x_1 + \alpha_2 (\frac{\lambda_2}{\lambda_1})^k x_2 + \cdots + \alpha_n (\frac{\lambda_n}{\lambda_1})^k x_n}{\|\alpha_1 x_1 + \alpha_2 (\frac{\lambda_2}{\lambda_1})^k x_2 + \cdots + \alpha_n (\frac{\lambda_n}{\lambda_1})^k x_n\|}$$

in z_k konvergira v smeri proti x_1, ko gre k proti neskončnosti. ∎

Vidimo, da se da vektor z_k izraziti s produktom k-te potence matrike A z vektorjem z_0. Zaradi tega metodo, ki generira vektorje po predpisu (1.3), imenujemo *potenčna metoda*[5].

V zadnjem dokazu smo predpostavili:

[5] Metodo je vpeljal avstrijski matematik Richard von Mises (1883–1953) leta 1929.

a) $\alpha_1 \neq 0$,

b) A ima enostavno dominantno lastno vrednost.

Pri numeričnem računanju je predpostavka $a)$ v praksi vedno izpolnjena, saj zaokrožitvene napake povzročijo, da je $\alpha_1 \neq 0$. Če nimamo na voljo dobrega začetnega približka, je najbolj priporočljivo, da začetni vektor generiramo iz naključnih vrednosti.

Pri točki b) se da pokazati, da izrek velja tudi, ko je λ_1 večkratna lastna vrednost. Metoda se da ustrezno predelati (če si zapomnimo in hkrati gledamo zadnje tri približke) tudi za primera:

- $|\lambda_1| = |\lambda_2| > |\lambda_3| \geq \cdots$ in $\lambda_1 = -\lambda_2$,
- $|\lambda_1| = |\lambda_2| > |\lambda_3| \geq \cdots$ in $\lambda_1 = \overline{\lambda_2}$.

Vektor z_k po smeri konvergira proti lastnemu vektorju za λ_1. Zaradi tega normiranje vektorja v vsakem koraku v bistvu ni potrebno, izvajamo ga le zato, da pri numeričnem računanju ne pride do prekoračitve (v primeru $|\lambda_1| > 1$) ali podkoračitve (v primeru $|\lambda_1| < 1$).

Kako ugotovimo, kdaj je postopek že skonvergiral dovolj blizu rešitve? Ker vektor konvergira po smeri, ne pa tudi po komponentah, pogoj $\|z_{k+1} - z_k\| \leq \epsilon$ ni dober. Potrebujemo kriterij, ki nam za dani približek za lastni vektor pove, kako dober približek je to. Tega pa se ne da ugotoviti drugače kot da za približek za lastni vektor poiščemo še približek za lastno vrednost in potem skupaj pogledamo kako dober približek za lastni par imamo.

Denimo, da imamo približek x za lastni vektor in iščemo lastno vrednost. Najboljši približek je λ, ki minimizira

$$\|Ax - \lambda x\|_2,$$

rešitev pa je (uporabimo normalni sistem za predoločeni sistem $x\lambda = Ax$) *Rayleighov*[6] *kvocient*

$$\rho(x, A) = \frac{x^H Ax}{x^H x},$$

ki je definiran za $x \neq 0$.

Za Rayleighov kvocient velja $\rho(x, A) = \rho(\alpha x, A)$ za $\alpha \neq 0$. Rayleighov kvocient je torej odvisen le od smeri, ne pa tudi od norme vektorja. Očitno je tudi, da iz $Ax = \lambda x$ sledi $\rho(x, A) = \lambda$.

Pravilen zaustavitveni kriterij za potenčno metodo je, da za vsak vektor z_k izračunamo Rayleighov kvocient $\rho_k = \rho(z_k, A)$, potem pa pogledamo normo ostanka $\|Az_k - \rho_k z_k\|_2$. Če je norma dovolj majhna, je (ρ_k, z_k) dober približek za lastni par.

Vidimo, da računanje Rayleighovega kvocienta in preverjanje konvergence ne vplivata bistveno na časovno zahtevnost. Glavna operacija, ki jo v vsakem koraku algoritma izvedemo enkrat, je množenje z matriko A. Ta ima v primeru splošne matrike zahtevnost $\mathcal{O}(n^2)$, ostale operacije pa imajo zahtevnost $\mathcal{O}(n)$.

Iz dokaza izreka 1.13 sledi, da je konvergenca potenčne metode linearna. Hitrost konvergence je odvisna od razmerja $\left|\frac{\lambda_2}{\lambda_1}\right|$. Če je razmerje blizu 1, bo konvergenca počasna, blizu 0 pa hitrejša.

[6] John William Strutt (1842–1919), tretji baron Rayleigha oziroma lord Rayleigh, je bil znani angleški fizik. Leta 1904 je za raziskave plinov in odkritje argona prejel Nobelovo nagrado za fiziko. Kvocient je uporabljal pri raziskavah o termoakustiki.

Algoritem 1.1 Osnovna varianta potenčne metode. Začetni podatki so matrika A (zadošča funkcija, ki zna za dani vektor x izračunati produkt Ax), normiran vektor z_0 in toleranca ϵ.

$y_1 = Az_0, \quad \rho_0 = z_0^H y_1, \quad k = 0$
dokler $\|y_{k+1} - \rho_k z_k\|_2 \geq \epsilon$
  $k = k + 1$
  $z_k = \frac{1}{\|y_k\|_2} y_k$
  $y_{k+1} = Az_k$
  $\rho_k = z_k^H y_{k+1}$

Tako dobimo dominantno lastno vrednost λ_1. Naj bo x_1 pripadajoči normiran lastni vektor. Za ostale lastne pare lahko naredimo redukcijo:

a) *Hotellingova*[7] *redukcija* za $A = A^T$. Definiramo

$$B = A - \lambda_1 x_1 x_1^T.$$

Hitro lahko preverimo, da velja $Bx_1 = 0$ in $Bx_k = \lambda_k x_k$ za $k \neq 1$. Če uporabimo potenčno metodo na matriki B, bomo tako dobili drugo dominantno lastno vrednost matrike A.

Matrike B nam ni potrebno eksplicitno izračunati. Iz enakosti $Bz = Az - \lambda_1(x_1^T z)x_1$ namreč sledi, da potrebujemo le množenje z matriko A in izračun skalarnega produkta z vektorjem x_1. Na ta način tudi za izračun naslednje lastne vrednosti še vedno zadošča, da poznamo le funkcijo, ki izračuna produkt matrike A z danim vektorjem.

b) *Householderjeva redukcija* za splošno matriko. Poiščemo unitarno matriko U, da je $Ux_1 = e_1$, pri čemer lahko seveda uporabimo Householderjeva zrcaljenja. Potem ima matrika $B = UAU^H$ obliko (glej dokaz izreka 1.2)

$$B = \begin{bmatrix} \lambda_1 & b^T \\ 0 & C \end{bmatrix}.$$

Preostale lastne vrednosti matrike A se ujemajo z lastnimi vrednostmi matrike C.

Tudi v primeru Householderjeve redukcije ekspliciten izračun matrike C ni potreben. Pri potenčni metodi moramo znati izračunati produkt Cw za vektor $w \in \mathbb{C}^{n-1}$. Pri tem si pomagamo z zvezo

$$UAU^H \begin{bmatrix} 0 \\ w \end{bmatrix} = \begin{bmatrix} \lambda_1 & b^T \\ 0 & C \end{bmatrix} \begin{bmatrix} 0 \\ w \end{bmatrix} = \begin{bmatrix} b^T w \\ Cw \end{bmatrix}.$$

Potrebujemo torej množenje z matriko A in dve množenji z ortogonalno matriko Q, ki ju lahko v primeru Householderjevega zrcaljenja izvedemo z zahtevnostjo $\mathcal{O}(n)$.

Če iščemo lastno vrednost nesingularne matrike A, ki je najmanjša po absolutni vrednosti, delamo potenčno metodo za A^{-1}, saj ima A^{-1} lastne vrednosti $\lambda^{-1}, \ldots, \lambda_n^{-1}$. V algoritmu namesto množenja $y_{k+1} = A^{-1}z_k$ rešujemo sistem $Ay_{k+1} = z_k$.

[7] Ameriški ekonomist in statistik Harold Hotelling (1895–1973) je potenčno metodo uporabljal pri kanonični korelacijski analizi.

1.5 Obratna napaka in izračunljive ocene

Denimo, da smo numerično, npr. s potenčno metodo, izračunali približek $(\widehat{\lambda}, \widehat{x})$ za lastni par matrike A. Radi bi ocenili, za koliko se $\widehat{\lambda}$ razlikuje od prave lastne vrednosti.

Definicija 1.14 Po normi relativna obratna napaka *približka* $(\widehat{\lambda}, \widehat{x})$ *za lastni par je*

$$\eta(\widehat{\lambda}, \widehat{x}) = \min\left\{\epsilon > 0 : (A + \delta A)\widehat{x} = \widehat{\lambda}\widehat{x},\ \|\delta A\|_2 \leq \epsilon \|A\|_2\right\}.$$

Če gledamo samo približek za lastno vrednost $\widehat{\lambda}$, definiramo obratno napako kot $\eta(\widehat{\lambda}) = \min_{\widehat{x}\neq 0} \eta(\widehat{\lambda}, \widehat{x})$, podobno za obratno napako približka za lastni vektor vzamemo $\eta(\widehat{x}) = \min_{\widehat{\lambda}} \eta(\widehat{\lambda}, \widehat{x})$.

Lema 1.15 *Velja*

1) $\eta(\widehat{\lambda}, \widehat{x}) = \dfrac{\|A\widehat{x} - \widehat{\lambda}\widehat{x}\|_2}{\|A\|_2\|\widehat{x}\|_2}$,

2) $\eta(\widehat{\lambda}) = \dfrac{\sigma_{\min}(A - \widehat{\lambda}I)}{\|A\|_2}$,

3) $\eta(\widehat{x}) = \dfrac{\|A\widehat{x} - \rho(\widehat{x}, A)\widehat{x}\|_2}{\|A\|_2\|\widehat{x}\|_2}$.

Dokaz. Za točko 1) označimo $r = A\widehat{x} - \widehat{\lambda}\widehat{x}$. Iz $(A + \delta A)\widehat{x} = \widehat{\lambda}\widehat{x}$ sledi $-\delta A\widehat{x} = r$, od tod pa $\|r\|_2 \leq \|\delta A\|_2\|\widehat{x}\|_2$.

Če vzamemo $\delta A = -r\widehat{x}^H/\|\widehat{x}\|_2^2$, potem je $(A + \delta A - \widehat{\lambda}I)\widehat{x} = 0$ in $\|\delta A\|_2 = \|r\|_2/\|\widehat{x}\|_2$, kar pomeni, da je minimum res dosežen in točka 1) drži.

Pri točki 2) upoštevamo dejstvo, da je $\min_{x\neq 0} \|Mx\|_2 = \sigma_{\min}(M)$, pri točki 3) pa lastnost Rayleighovega kvocienta, da je minimum $\|Ax - \tau x\|_2$ dosežen pri $\tau = \rho(x, A)$. ∎

Iz zgornje leme sledi, da je potenčna metoda obratno stabilna, saj vrne tak približek $(\widehat{\lambda}, \widehat{x})$ za lastni par, za katerega velja $\|A\widehat{x} - \widehat{\lambda}\widehat{x}\|_2 \leq \epsilon$.

Če bi radi ocenili, za koliko se približek $\widehat{\lambda}$ razlikuje od točne lastne vrednosti, potrebujemo poleg ocene za obratno napako še oceno za občutljivost lastne vrednosti. Za oceno potrebujemo tudi približek za levi lastni vektor. Tega bodisi izračunamo v algoritmu ali pa uporabimo npr. inverzno iteracijo, ki jo bomo spoznali v naslednjem razdelku.

Naj bosta torej $\widehat{x}$ in $\widehat{y}$ približka za desni in levi lastni vektor, ki pripadata $\widehat{\lambda}$. Potem velja ocena

$$|\widehat{\lambda} - \lambda| \leq \frac{\|r\|_2}{|\widehat{s}|},$$

kjer je

$$\widehat{s} = \frac{-\widehat{y}^H\widehat{x}}{\|\widehat{y}\|_2\|\widehat{x}\|_2}.$$

1.6 Inverzna iteracija

Rayleighov kvocient vrne najboljši približek za lastno vrednost, ki ustreza danemu vektorju. Kaj pa obratno? Denimo, da smo izračunali približek za lastno vrednost σ, sedaj pa potrebujemo še lastni vektor. Tu si lahko pomagamo z *inverzno iteracijo*[8], ki je v grobem predstavljena v algoritmu 1.2.

Algoritem 1.2 Osnovna verzija inverzne iteracije. Začetni podatki so matrika A, približek za lastno vrednost σ in normiran vektor z_0.

$k = 0, 1, \ldots$
  reši sistem $(A - \sigma I)y_{k+1} = z_k$
  $z_{k+1} = \frac{1}{\|y_{k+1}\|} y_{k+1}$

Naj za približek σ velja, da mu je najbližja lastna vrednost λ_i in velja $|\lambda_i - \sigma| \ll |\lambda_j - \sigma|$ za $j \neq i$. Inverzna iteracija v bistvu ni nič drugega kot potenčna metoda za matriko $(A - \sigma I)^{-1}$, zato jo imenujemo tudi *inverzna potenčna metoda*. Od tod vemo, da vektor z_k po smeri konvergira proti lastnemu vektorju, ki pripada dominantni lastni vrednosti matrike $(A - \sigma I)^{-1}$.

Lastne vrednosti matrike $(A - \sigma I)^{-1}$ so $(\lambda_j - \sigma)^{-1}$ za $j = 1, \ldots, n$. Ker velja $|\lambda_i - \sigma| \ll |\lambda_j - \sigma|$ za $j \neq i$, je $|(\lambda_i - \sigma)^{-1}| \gg |(\lambda_j - \sigma)^{-1}|$ za $j \neq i$. Boljši, ko je približek σ, dominantnejša je lastna vrednost $(\lambda_i - \sigma)^{-1}$ in hitrejša je konvergenca.

Inverzno iteracijo ponavadi uporabljamo zato, da dobimo lastni vektor za numerično izračunano lastno vrednost. V tem primeru je σ kar lastna vrednost matrike A, izračunana z natančnostjo $\mathcal{O}(u)$. V praksi zato potrebujemo le en do dva koraka inverzne iteracije, da iz poljubnega začetnega vektorja izračunamo pripadajoči lastni vektor.

Če je σ res lastna vrednost matrike A, potem je matrika $A - \sigma I$ singularna in v algoritmu lahko pričakujemo težave pri reševanju sistema s to matriko. Izkaže se, da nam zaokrožitvene napake pomagajo do tega, da v praksi ne pride do deljenja z nič, zato je inverzna iteracija zelo učinkovita metoda za računanje lastnih vektorjev za lastne vrednosti.

1.7 Ortogonalna iteracija

Pravimo, da je podprostor $\mathcal{N}$ *invarianten* za matriko A, če velja $A\mathcal{N} \subset \mathcal{N}$. Naj ima matrika S_1 p linearno neodvisnih stolpcev. Če jo dopolnimo do nesingularne matrike $S = [\, S_1 \quad S_2 \,]$ in je

$$B = S^{-1}AS = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix},$$

potem lahko hitro preverimo, da stolpci S_1 razpenjajo invariantni podprostor natanko takrat, ko je $B_{21} = 0$. V tem primeru so lastne vrednosti matrike A unija lastnih vrednosti matrik B_{11} in B_{22}. Če lastne vrednosti matrike A lahko uredimo po absolutni vrednosti tako, da velja $|\lambda_1| \geq \cdots \geq |\lambda_p| > |\lambda_{p+1}| \geq \cdots \geq |\lambda_n|$, potem stolpci matrike S_1 razpenjajo dominantni invariantni podprostor natanko takrat, ko so lastne vrednosti matrike B_{11} enake $\lambda_1, \ldots, \lambda_p$.

[8] Metodo je leta 1944 vpeljal nemški matematik Helmut Wielandt (1910–2001).

Če se da matriko diagonalizirati, potem bazo za dominantni invariantni podprostor dimenzije p lahko sestavimo iz p lastnih vektorjev, ki pripadajo po absolutni vrednosti p največjim lastnim vrednostim.

Za izračun baze za dominantni invariantni podprostor imamo na voljo *ortogonalno iteracijo*.

Algoritem 1.3 Osnovna verzija ortogonalne iteracije. Začetni podatki so matrika A velikosti $n \times n$ in matrika Z_0 velikosti $n \times p$, $p \leq n$, z ortonormiranimi stolpci.

$k = 0, 1, \ldots$
  $Y_{k+1} = AZ_k$
  izračunaj QR razcep $Y_{k+1} = QR$ in vzemi $Z_{k+1} = Q$

Opazimo lahko, da je pri $p = 1$ to kar potenčna metoda. Vemo, da potenčna metoda skonvergira proti dominantnemu lastnemu vektorju, linearni podprostor, ki ga razpenja ta lastni vektor, pa je očitno dominanten invarianten podprostor dimenzije 1.

Izrek 1.16 *Naj velja $A = X\Lambda X^{-1}$, kjer je $X = [x_1 \ \cdots \ x_n]$ in $\Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$. Lastne vrednosti naj bodo urejene po absolutni vrednosti in naj velja $|\lambda_p| > |\lambda_{p+1}|$. Potem, za splošno izbrano začetno matriko Z_0, matrika Z_k iz ortogonalne iteracije konvergira proti ortonormirani bazi za invariantni podprostor* $\operatorname{Lin}(\{x_1, \ldots, x_p\})$.

Dokaz. Očitno je $\operatorname{Lin}(Z_{k+1}) = \operatorname{Lin}(Y_{k+1}) = \operatorname{Lin}(AZ_k)$, od tod pa sledi $\operatorname{Lin}(Z_k) = \operatorname{Lin}(A^k Z_0)$. Ker je $A^k = X\Lambda^k X^{-1}$, velja

$$A^k Z_0 = X\Lambda^k X^{-1} Z_0 = \lambda_p^k X \begin{bmatrix} (\lambda_1/\lambda_p)^k & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & (\lambda_n/\lambda_p)^k \end{bmatrix} X^{-1} Z_0.$$

Ko gre k proti neskončno, gre $A^k Z_0$ proti $X \cdot \begin{matrix} p \\ n-p \end{matrix} \begin{pmatrix} \times \\ 0 \end{pmatrix}$, to pa pomeni, da $\operatorname{Lin}(Z_k)$ konvergira proti $\operatorname{Lin}(\{x_1, \ldots, x_p\})$. ∎

Podobno v primeru, ko je $|\lambda_r| > |\lambda_{r+1}|$, za prvih $r < p$ stolpcev velja, da $\operatorname{Lin}(Z_k(:, 1:r))$ konvergira proti $\operatorname{Lin}(\{x_1, \ldots, x_r\})$.

Vzemimo kar $p = n$ in poljubno nesingularno matriko Z_0. V tem primeru seveda ne računamo invariantnega podprostora dimenzije n, saj je to kar celotni prostor. Pri predpostavki, da so absolutne vrednosti λ_i paroma različne (to hkrati pomeni, da so vse realne), lahko pokažemo, da matrika $A_k := Z_k^T A Z_k$ konvergira proti Schurovi formi, ko gre k proti neskončno.

A_k in $Z_k^T A Z_k$ sta podobni matriki, saj je matrika Z_k ortogonalna. Naj bo $Z_k = [Z_{k1} \ Z_{k2}]$, kjer ima Z_{k1} p stolpcev. Potem je

$$Z_k^T A Z_k = \begin{bmatrix} Z_{k1}^T A Z_{k1} & Z_{k1}^T A Z_{k2} \\ Z_{k2}^T A Z_{k1} & Z_{k2}^T A Z_{k2} \end{bmatrix}.$$

Ker $\operatorname{Lin}(Z_{k1})$ konvergira proti invariantnemu podprostoru $\operatorname{Lin}(\{x_1, \ldots, x_p\})$, enako velja tudi za $\operatorname{Lin}(AZ_{k1})$, to pa pomeni, da $Z_{k2}^T A Z_{k1}$ konvergira proti 0, saj je $Z_{k2}^T Z_{k1} = 0$. Ker to velja za

vsak $p = 1, \ldots, n$, sledi, da matrika $Z_k^T A Z_k$ res konvergira proti zgornji trikotni matriki, torej proti Schurovi formi.

Poddiagonalni elementi matrike A_k konvergirajo proti 0 z linearno konvergenco, hitrost konvergence (i, j)-tega elementa, kjer je $i > j$, pa je odvisna od razmerja $|\lambda_j|/|\lambda_i|$.

1.8 QR iteracija

V prejšnjem razdelku smo videli, da z ortogonalno iteracijo lahko izračunamo Schurovo formo in s tem vse lastne vrednosti matrike. V tem razdelku pa bomo spoznali algoritem, ki zna to narediti na ekonomičnejši način. Gre za *QR iteracijo*[9], ki je trenutno najboljša numerična metoda za izračun vseh lastnih vrednosti splošne nesimetrične matrike. Osnovna verzija je zapisana v algoritmu 1.4.

Algoritem 1.4 Osnovna verzija QR iteracije. Začetni podatek je matrika A.

$A_0 = A$
$k = 0, 1, \ldots$
  $A_k = Q_k R_k$ (izračunaj QR razcep)
  $A_{k+1} = R_k Q_k$

V vsakem koraku izračunamo QR razcep matrike in faktorja v zamenjanem vrstnem redu zmnožimo v novo matriko. Iz $A_{k+1} = R_k Q_k = Q_k^T A_k Q_k$ sledi, da sta si matriki A_{k+1} in A_k ortogonalno podobni, torej je A_{k+1} ortogonalno podobna začetni matriki A in velja

$$A_{k+1} = Q_k^T \cdots Q_0^T A Q_0 \cdots Q_k.$$

Izkaže se, da je QR iteracija povezana z ortogonalno iteracijo, od tod pa sledi, da A_k konvergira proti Schurovi formi.

Lema 1.17 *Za matriko A_k iz QR iteracije velja $A_k = Z_k^T A Z_k$, kjer je Z_k matrika, ki jo dobimo pri ortogonalni iteraciji iz $Z_0 = I$. V primeru, ko imajo lastne vrednosti paroma različne absolutne vrednosti, A_k konvergira proti Schurovi formi.*

Dokaz. Uporabimo indukcijo po k. Na začetku je $A_0 = Z_0^T A Z_0$, saj je $Z_0 = I$. Denimo, da je $A_k = Z_k^T A Z_k$. Potem je

$$A_k = Z_k^T A Z_k = Z_k^T \underbrace{(\underbrace{Z_{k+1}}_{\text{ort.}} \underbrace{S_{k+1}}_{\text{zg. trik.}})}_{\text{QR razcep } AZ_k} = \underbrace{Z_k^T Z_{k+1}}_{\text{ort.}} \underbrace{S_{k+1}}_{\text{zg. trik.}} = Q_k R_k.$$

Ker je QR razcep matrike enoličen, to pomeni $S_{k+1} = R_k$ in $Z_k^T Z_{k+1} = Q_k$. Sledi

$$A_{k+1} = R_k Q_k = S_{k+1} Z_k^T Z_{k+1} = \underbrace{Z_{k+1}^T A Z_k}_{S_{k+1}} Z_k^T Z_{k+1} = Z_{k+1}^T A Z_{k+1}.$$

[9] Metodo sta neodvisno leta 1961 odkrila angleški računalnikar John G. F. Francis (r. 1934) in ruska matematičarka Vera N. Kublanovskaja (r. 1920). Francis se je leta 1962 nehal ukvarjati z numerično matematiko in se nato do leta 2007 sploh ni zavedal, kakšen vpliv ima njegov algoritem na numerično matematiko. Po oceni, ki sta jo naredila Jack Dongarra in Francis Sullivan leta 2000, spada QR iteracija med 10 algoritmov iz 20. stoletja, ki so najbolj vplivali na razvoj znanosti in tehnike [10].

V primeru, ko ima matrika A tudi kompleksne lastne vrednosti, matrika A_k skonvergira proti realni Schurovi formi.

Če pogledamo zahtevnost enega koraka QR iteracije, vidimo, da ima časovno zahtevnost $\mathcal{O}(n^3)$, saj moramo v vsakem koraku izračunati QR razcep matrike. Hitrost konvergence je odvisna od razmerja med lastnimi vrednostmi. Če se dve lastni vrednosti le malo razlikujeta, potem lahko pričakujemo, da bo metoda potrebovala veliko korakov, preden bo skonvergirala do Schurove forme.

Da pridemo do uporabne verzije QR iteracije, moramo vpeljati še nekaj izboljšav.

1.8.1 Redukcija na Hessenbergovo obliko

En korak osnovne QR iteracije porabi $\mathcal{O}(n^3)$ operacij, kar ni najbolj ekonomično. Zahtevnost enega koraka lahko močno zmanjšamo, če matriko A predhodno reduciramo na zgornjo Hessenbergovo[10] obliko.

Definicija 1.18 *Pravimo, da je matrika A* zgornja Hessenbergova, *če je $a_{ij} = 0$ za $i > j + 1$.*

Zgornja Hessenbergova matrika ima torej le zgornji trikotnik in eno poddiagonalo. Od Schurove forme jo loči le poddiagonala. Izkaže se, da se zgornja Hessenbergova oblika ohranja med QR iteracijo.

Trditev 1.19 *Če je A zgornja Hessenbergova, se oblika med QR iteracijo ohranja.*

Dokaz. Pri QR razcepu matrike $A = \begin{bmatrix} a_1 & \cdots & a_n \end{bmatrix}$ dobimo zgornjo Hessenbergovo matriko Q in zgornjo trikotno matriko R. Pri $Q = \begin{bmatrix} q_1 & \cdots & q_n \end{bmatrix}$ je oblika razvidna iz dejstva, da je q_i linearna kombinacija stolpcev $a_1, \ldots, a_i$. Hitro lahko preverimo, da je produkt zgornje trikotne in zgornje Hessenbergove matrike spet zgornja Hessenbergova matrika. ∎

Vsako realno matriko A lahko z ortogonalno podobnostno transformacijo spremenimo v zgornjo Hessenbergovo matriko. Za splošno matriko uporabimo Householderjeva zrcaljenja, če pa ima matrika A v spodnjem trikotniku že veliko ničel, so lahko Givensove rotacije še bolj učinkovite.

Zgled 1.4 *Na zgledu matrike velikosti 5×5 poglejmo, kako matriko z ortogonalnimi podobnostnimi transformacijami spremenimo v zgornjo Hessenbergovo matriko. Naj bo*

$$A = \begin{bmatrix} \times & \times & \times & \times & \times \\ (\times) & \times & \times & \times & \times \\ (\times) & \times & \times & \times & \times \\ (\times) & \times & \times & \times & \times \\ (\times) & \times & \times & \times & \times \end{bmatrix}.$$

Elementi v oklepajih označujejo elemente vektorja, ki določa Householderjevo zrcaljenje v naslednjem koraku. Zrcaljenje določimo tako, da se označeni vektor prezrcali v smer prvega enotskega vektorja.

[10] Nemški matematik Karl Adolf Hessenberg (1904–1959).

Najprej poiščemo tako ortogonalno matriko Q_1, da je

$$Q_1A = \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \end{bmatrix}, \quad A_1 = Q_1AQ_1^T = \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ 0 & (\times) & \times & \times & \times \\ 0 & (\times) & \times & \times & \times \\ 0 & (\times) & \times & \times & \times \end{bmatrix}.$$

Nato poiščemo ortogonalno matriko Q_2, da je

$$Q_2A_1 = \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & \times & \times & \times \end{bmatrix}, \quad A_2 = Q_2A_1Q_2^T = \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & 0 & (\times) & \times & \times \\ 0 & 0 & (\times) & \times & \times \end{bmatrix},$$

na koncu pa še ortogonalno matriko Q_3, da je

$$Q_3A_2 = \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & \times & \times \end{bmatrix}, \quad H = Q_3A_2Q_3^T = \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & \times & \times \end{bmatrix}.$$

Tako dobimo zgornjo Hessenbergovo matriko $H = Q_3Q_2Q_1A(Q_3Q_2Q_1)^T$. □

Algoritem za splošno matriko ima naslednjo obliko.

Algoritem 1.5 Redukcija na zgornjo Hessenbergovo obliko preko Householderjevih zrcaljenj. Začetni podatek je $n \times n$ matrika A. Algoritem vrne zgornjo Hessenbergovo matriko H in po potrebi tudi ortogonalno matriko Q, da je $A = Q^THQ$.

$Q = I$ (*)
$i = 1, \ldots, n-2$
  določi $w_i \in \mathbb{R}^{n-i}$ za Householderjevo zrcaljenje P_i, ki prezrcali $A(i+1:n, i)$ v $\pm ke_1$
  $A(i+1:n, i:n) = P_iA(i+1:n, i:n)$
  $A(1:n, i+1:n) = A(1:n, i+1:n)P_i$
  $Q(i+1:n, i:n) = P_iQ(i+1:n, i:n)$ (*)

Korake označene z (*) izvedemo le v primeru, če potrebujemo tudi prehodno matriko Q.

Število operacij je $\frac{10}{3}n^3 + \mathcal{O}(n^2)$ oziroma $\frac{14}{3}n^3 + \mathcal{O}(n^2)$ če potrebujemo tudi matriko Q.

Če na začetku matriko A reduciramo na Hessenbergovo obliko, porabimo potem za en korak QR iteracije le še $\mathcal{O}(n^2)$ namesto $\mathcal{O}(n^3)$ operacij. Matrika Q_k iz QR razcepa zgornje Hessenbergove matrike A_k je namreč produkt $n-1$ Givensovih rotacij, za eno množenje matrike z Givensovo rotacijo pa vemo, da ima zahtevnost $\mathcal{O}(n)$.

Definicija 1.20 *Hessenbergova matrika H je* nerazcepna, *če so vsi njeni subdiagonalni elementi $h_{i+1,i}$ neničelni.*

Če je H razcepna, kot je npr.

$$H = \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & \times & \times \end{bmatrix},$$

potem problem lastnih vrednosti razpade na dva ločena problema. Zaradi tega lahko vedno predpostavimo, da je H nerazcepna.

Pri numeričnem računanju subdiagonalni element $a_{i+1,i}^{(k)}$ matrike A_k postavimo na 0, kadar je dovolj majhen v primerjavi s sosednjima diagonalnima elementoma. To pomeni, da zadošča kriteriju

$$|a_{i+1,i}^{(k)}| < \epsilon(|a_{ii}^{(k)}| + |a_{i+1,i+1}^{(k)}|),$$

kjer je $\epsilon = \mathcal{O}(u)$ izbrana toleranca.

1.8.2 Premiki

Z redukcijo na Hessenbergovo obliko smo zmanjšali zahtevnost posameznega koraka QR iteracije, samo število potrebnih korakov pa se ni zmanjšalo, saj je hitrost konvergence odvisna od razmerja med lastnimi vrednostmi.

Konvergenco lahko pospešimo z vpeljavo premikov, kot je predstavljeno v algoritmu 1.6.

Algoritem 1.6 QR iteracija s premiki. Začetni podatek je matrika A.

$A_0 = A$
$k = 0, 1, \ldots$
  izberi premik σ_k
  $A_k - \sigma_k I = Q_k R_k$ (izračunaj QR razcep)
  $A_{k+1} = R_k Q_k + \sigma_k I$

Naslednja lema nam zagotavlja, da je tudi po vpeljavi premika matrika A_k še vedno ortogonalno podobna začetni matriki A.

Lema 1.21 *Matriki A_k in A_{k+1} pri QR iteraciji s premiki sta ortogonalno podobni.*

Dokaz.

$$A_{k+1} = R_k Q_k + \sigma_k I = Q_k^T (Q_k R_k + \sigma_k I) Q_k = Q_k^T A_k Q_k. \qquad \blacksquare$$

Za hitro konvergenco moramo za premik izbrati čim boljši približek za lastno vrednost. Če bi za premik izbrali kar lastno vrednost, potem iz naslednje leme sledi, da bi se v enem koraku QR iteracije iz matrike izločila ta lastna vrednost in bi računanje lahko nadaljevali na manjši matriki.

Lema 1.22 *Naj bo σ lastna vrednost nerazcepne zgornje Hessenbergove matrike A. Če je QR razcep $A - \sigma I = QR$ in $B = RQ + \sigma I$, potem je $b_{n,n-1} = 0$ in $b_{nn} = \sigma$.*

Dokaz. Ker je A nerazcepna, je prvih $n-1$ stolpcev matrike $A-\sigma I$ linearno neodvisnih. V razcepu $A-\sigma I = QR$ zato velja $r_{ii} \neq 0$ za $i = 1, \ldots, n-1$. Ker je $A-\sigma I$ singularna, mora biti $r_{nn} = 0$. To pomeni, da je zadnja vrstica v matriki RQ enaka 0, torej v matriki $B = RQ + \sigma I$ velja $b_{n,n-1} = 0$ in $b_{nn} = \sigma$.

Preostale lastne vrednosti lahko potem izračunamo iz matrike $B(1:n-1, 1:n-1)$. ■

Za premik potrebujemo čim boljši približek za lastno vrednost matrike A. Uporabljata se naslednji izbiri:

a) *Enojni premik*: za σ_k izberemo $a_{nn}^{(k)}$.

Motivacija, da za premik izberemo element v spodnjem desnem kotu je, da naj bi bil to dober približek za po absolutni vrednosti najmanjšo lastno vrednost λ_n matrike A.

Naj bo y_n levi lastni vektor za λ_n. Z malce računanja lahko pokažemo, da pri QR algoritmu brez premikov velja

$$A^k = \widetilde{Q}_{k-1}\widetilde{R}_{k-1},$$

kjer je $\widetilde{Q}_{k-1} = Q_0 \cdots Q_{k-1}$ in $\widetilde{R}_{k-1} = R_{k-1} \cdots R_0$. Od tod sledi $A^{-k} = \widetilde{R}_{k-1}^{-1}\widetilde{Q}_{k-1}^T$. Iz zadnje enakosti vidimo, da je zadnja vrstica matrike $\widetilde{Q}_{k-1}^T$ proporcionalna $e_n^T A^{-k}$, to je zadnji vrstici matrike A^{-k}. Razen v izjemnem in pri numeričnem računanju malo verjetnem primeru, ko e_n nima nobene komponente v smeri y_n, bo vrstica $e_n^T A^{-k}$ po smeri konvergirala proti y_n^T. Od tod sledi, da zadnji stolpec matrike $\widetilde{Q}_{k-1}$ konvergira proti levemu lastnemu vektorju y_n. Iz zveze $A_k = \widetilde{Q}_{k-1}^T A \widetilde{Q}_{k-1}$ je tako razvidno, da $a_{nn}^{(k)} = e_n^T A_k e_n = (\widetilde{Q}_{k-1}e_n)^T A \widetilde{Q}_{k-1} e_n$ konvergira proti lastni vrednosti λ_n.

Pri tej izbiri imamo kvadratično konvergenco v bližini enostavne realne lastne vrednosti, za matrike, ki imajo tudi kompleksne lastne vrednosti, pa premik ni dober.

b) *Dvojni* oz. *Francisov premik*: vzamemo podmatriko

$$A_k(n-1:n, n-1:n) = \begin{bmatrix} a_{n-1,n-1}^{(k)} & a_{n-1,n}^{(k)} \\ a_{n,n-1}^{(k)} & a_{nn}^{(k)} \end{bmatrix},$$

ki ima lastni vrednosti $\sigma_1^{(k)}, \sigma_2^{(k)}$ (lahko sta tudi kompleksni). Sedaj naredimo dva premika v enem koraku:

$$\begin{aligned} &A_k - \sigma_1^{(k)} I = Q_k R_k \text{ (izračunaj QR razcep)} \\ &A_k' = R_k Q_k + \sigma_1^{(k)} I \\ &A_k' - \sigma_2^{(k)} I = Q_k' R_k' \text{ (izračunaj QR razcep)} \\ &A_{k+1} = R_k' Q_k' + \sigma_2^{(k)} I. \end{aligned}$$

Izkaže se, da lahko en korak QR z dvojnim premikom izvedemo brez kompleksne aritmetike, saj velja:

$$\begin{aligned} Q_k Q_k' R_k' R_k &= Q_k (A_k' - \sigma_2^{(k)} I) R_k \\ &= Q_k Q_k^H (A_k - \sigma_2^{(k)} I) Q_k R_k = (A_k - \sigma_2^{(k)} I) Q_k R_k \\ &= (A_k - \sigma_2^{(k)} I)(A_k - \sigma_1^{(k)} I) \end{aligned}$$

$$= \quad A_k^2 - (\sigma_1^{(k)} + \sigma_2^{(k)})A_k + \sigma_1^{(k)}\sigma_2^{(k)} I =: N_k$$

in $(Q_k Q_k')(R_k' R_k)$ je QR razcep realne matrike N_k. Ker po lemi 1.21 velja tudi

$$A_{k+1} = Q_k'^H A_k' Q_k' = Q_k'^H Q_k^H A_k Q_k Q_k',$$

potrebujemo le realni QR razcep realne matrike N_k.

Na prvi pogled nam zgornja ugotovitev ne pomaga dosti, saj v formuli za N_k nastopa matrika A_k^2, za izračun le te pa potrebujemo $\mathcal{O}(n^3)$ operacij. Kot bomo videli v naslednjem razdelku, pa se izkaže, da v resnici potrebujemo le prvi stolpec matrike N_k.

1.9 Implicitna QR metoda

Izrek 1.23 (Implicitni Q) *Če je $Q = [q_1 \ \cdots \ q_n]$ taka ortogonalna matrika, da je $Q^T A Q = H$ nerazcepna Hessenbergova matrika, potem so stolpci $q_2, \ldots, q_n$ do predznaka natančno določeni s q_1.*

Dokaz. Denimo, da je $V^T A V = G$, kjer je $V = [v_1 \ \cdots \ v_n]$ ortogonalna matrika, G nerazcepna zgornja Hessenbergova matrika in $q_1 = v_1$.

Potem je $W = V^T Q$ ortogonalna matrika. Če zapišemo $W = [w_1 \ \cdots \ w_n]$, potem je $w_1 = e_1$. Velja

$$GW = GV^T Q = V^T A Q = V^T Q H = WH.$$

Iz te zveze sledi $Gw_i = \sum_{j=1}^{i+1} h_{ji} w_j$ oziroma

$$h_{i+1,i} w_{i+1} = Gw_i - \sum_{j=1}^{i} h_{ji} w_j.$$

Ker je $w_1 = e_1$ in ima Gw_i en neničelni element več od w_i, sledi $w_i \in \mathrm{Lin}(\{e_1, \ldots, e_i\})$. To pomeni, da je W zgornja trikotna matrika. Ker pa je W hkrati ortogonalna, je edina možnost $W = \mathrm{diag}(1, \pm 1, \ldots, \pm 1)$, torej $v_i = \pm q_i$ za $i = 2, \ldots, n$. ∎

Posledica je, da če v QR algoritmu $A_k = Q_k R_k$, $A_{k+1} = R_k Q_k = Q_k^T A_k Q_k$, poznamo prvi stolpec matrike Q_k, potem lahko matriko A_{k+1} izračunamo brez računanja celotnega QR razcepa matrike A_k. Tako dobimo *implicitno QR iteracijo*.

Najprej poglejmo implicitno QR iteracijo z enojnim premikom. Vemo, da je prvi stolpec Q_k enak normiranemu prvemu stolpcu $A_k - \sigma_k I$. Če uspemo poiskati tako ortogonalno matriko Q_k, da bo njen prvi stolpec normiran prvi stolpec matrike $A_k - \sigma_k I$ in bo $Q_k^T A_k Q_k$ zgornja Hessenbergova matrika, potem je po izreku o implicitnem Q matrika $Q_k^T A_k Q_k$ enaka matriki iz naslednjega koraka QR metode.

Matriko Q_k poiščemo kot produkt Givensovih rotacij $Q_k = R_{12} R_{23} \cdots R_{n-1,n}$. Prva rotacija R_{12} je že določena s prvim stolpcem $A_k - \sigma_k I$, ostale pa določimo tako, da bo $Q_k^T A_k Q_k$ zgornja Hessenbergova matrika. Če je namreč

$$R_{12} = \begin{bmatrix} c_1 & s_1 & & & \\ -s_1 & c_1 & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{bmatrix},$$

potem je

$$Q_k = R_{12}R_{23}\cdots R_{n-1,n} = \begin{bmatrix} c_1 & \times & \cdots & \cdots & \times \\ -s_1 & \times & \cdots & \cdots & \times \\ & \times & & & \vdots \\ & & \ddots & & \vdots \\ & & & \times & \times \end{bmatrix}.$$

Algoritem lahko označimo kot *premikanje grbe*. Poglejmo si ga na primeru matrike velikosti 5×5. Po prvem koraku dobimo

$$R_{12}^T A_k R_{12} = \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ + & \times & \times & \times & \times \\ & & \times & \times & \times \\ & & & \times & \times \end{bmatrix}.$$

Novi neničelni element, označen s $+$, je grba, ki jo z naslednjimi rotacijami pomikamo navzdol ob diagonali. Tako po vrsti poiščemo R_{23}, R_{34} in R_{45}, da je

$$R_{23}^T R_{12}^T A_k R_{12} R_{23} = \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & + & \times & \times & \times \\ & & & \times & \times \end{bmatrix},$$

$$R_{34}^T R_{23}^T R_{12}^T A_k R_{12} R_{23} R_{34} = \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & & \times & \times & \times \\ & & + & \times & \times \end{bmatrix}$$

in

$$R_{45}^T R_{34}^T R_{23}^T R_{12}^T A_k R_{12} R_{23} R_{34} R_{45} = \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & & \times & \times & \times \\ & & & \times & \times \end{bmatrix}.$$

Dobimo zgornjo Hessenbergovo matriko, ki je po izreku o implicitnem Q enaka matriki A_{k+1}.

Še bolj kot pri enojnem premiku nam izrek o implicitnem Q pride prav pri dvojnem premiku. Vemo, da je $A_{k+1} = U_k^T A_k U_k$, kjer je U_k ortogonalna matrika iz QR razcepa matrike $N_k = A_k^2 - (\sigma_1^{(k)} + \sigma_2^{(k)})A_k + \sigma_1^{(k)}\sigma_2^{(k)} I$. Dovolj je poznati le prvi stolpec matrike N_k. Le ta ima obliko

$$\begin{bmatrix} a_{11}^2 + a_{12}a_{21} - sa_{11} + t \\ a_{21}(a_{11} + a_{22} - s) \\ a_{21}a_{32} \\ 0 \\ \vdots \\ 0 \end{bmatrix},$$

kjer sta

$$s = a_{n-1,n-1} + a_{nn}$$

$$t = a_{n-1,n-1}a_{nn} - a_{n-1,n}a_{n,n-1}.$$

Sedaj najprej poiščemo Householderjevo zrcaljenje oblike

$$P_1 = \begin{bmatrix} \times & \times & \times & & & \\ \times & \times & \times & & & \\ \times & \times & \times & & & \\ & & & 1 & & \\ & & & & \ddots & \\ & & & & & 1 \end{bmatrix},$$

ki ima prvi stolpec enak normiranemu prvemu stolpcu matrike N_k. Po množenju s P_1 dobimo grbo velikosti 2×2, ki jo premikamo navzdol s Householderjevimi zrcaljenji. Poglejmo si, kako to naredimo v primeru matrike velikosti 6×6.

$$P_1 A_k P_1 = \begin{bmatrix} \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ + & \times & \times & \times & \times & \times \\ + & + & \times & \times & \times & \times \\ & & & \times & \times & \times \\ & & & & \times & \times \end{bmatrix},$$

$$P_2 = \begin{bmatrix} 1 & & & & & \\ & \times & \times & \times & & \\ & \times & \times & \times & & \\ & \times & \times & \times & & \\ & & & & 1 & \\ & & & & & 1 \end{bmatrix}, \quad P_2 P_1 A_k P_1 P_2 = \begin{bmatrix} \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ & \times & \times & \times & \times & \times \\ & + & \times & \times & \times & \times \\ & + & + & \times & \times & \times \\ & & & & \times & \times \end{bmatrix},$$

$$P_3 = \begin{bmatrix} 1 & & & & & \\ & 1 & & & & \\ & & \times & \times & \times & \\ & & \times & \times & \times & \\ & & \times & \times & \times & \\ & & & & & 1 \end{bmatrix}, \quad P_3 P_2 P_1 A_k P_1 P_2 P_3 = \begin{bmatrix} \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ & \times & \times & \times & \times & \times \\ & & \times & \times & \times & \times \\ & & + & \times & \times & \times \\ & & + & + & \times & \times \end{bmatrix},$$

$$P_4 = \begin{bmatrix} 1 & & & & & \\ & 1 & & & & \\ & & 1 & & & \\ & & & \times & \times & \times \\ & & & \times & \times & \times \\ & & & \times & \times & \times \end{bmatrix}, \quad P_4 P_3 P_2 P_1 A_k P_1 P_2 P_3 P_4 = \begin{bmatrix} \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ & \times & \times & \times & \times & \times \\ & & \times & \times & \times & \times \\ & & & \times & \times & \times \\ & & & + & \times & \times \end{bmatrix},$$

$$P_5 = \begin{bmatrix} 1 & & & & & \\ & 1 & & & & \\ & & 1 & & & \\ & & & 1 & & \\ & & & & \times & \times \\ & & & & \times & \times \end{bmatrix}, \quad P_5 \cdots P_1 A_k P_1 \cdots P_5 = \begin{bmatrix} \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ & \times & \times & \times & \times & \times \\ & & \times & \times & \times & \times \\ & & & \times & \times & \times \\ & & & & \times & \times \end{bmatrix}.$$

V vsakem koraku grbo velikosti 2×2 premaknemo za eno mesto navzdol, dokler je v zadnjem koraku ne izločimo iz matrike in nam ostane A_{k+1}. S pomočjo izreka o implicitnem Q lahko

tako en korak QR z dvojnim premikom izvedemo v $\mathcal{O}(n^2)$ operacijah. En korak stane $10n^2$ operacij, če računamo le A_{k+1}, in še dodatnih $10n^2$ operacij, če posodobimo še matriko Q.

QR iteracija je obratno stabilna. Za izračunano kvazi zgornjo trikotno matriko $\widehat{T}$ velja, da je ortogonalno podobna relativno malo zmoteni matriki A. To pomeni, da obstaja taka ortogonalna matrika Q, da je $A + E = Q\widehat{T}Q^T$, kjer je $\|E\| \approx \|A\|_2 u$. Podobno za izračunano matriko $\widehat{Q}$ velja, da je skoraj ortogonalna, oziroma $\|\widehat{Q}^T\widehat{Q} - I\|_2 \approx u$.

Matlab

Za izračun lastnih vrednosti in vektorjev imamo na voljo ukaz `eig`, ki uporabi implicitno QR iteracijo. Za nesimetrične matrike uporabi dvojne premike, za simetrične pa Wilkinsonove premike (glej razdelek 2.3).

- `l = eig(A)`: vrne vektor z lastnimi vrednostmi matrike A.
- `[X,D] = eig(A)`: vrne matriko lastnih vektorjev X in diagonalno matriko lastnih vrednosti D, da je $AX = XD$.

Schurovo formo dobimo z naslednjimi ukazi:

- `[Q,R] = schur(A)`: za realno matriko A vrne ortogonalno matriko Q in kvazi zgornjo trikotno matriko R, da je $A = QRQ^T$. Če pa je matrika A kompleksna, potem vrne unitarno matriko U in zgornjo trikotno matriko R, da je $A = QRQ^H$.
- `[Q,R] = schur(A,'complex')`: za realno matriko A vrne vrne unitarno matriko U in zgornjo trikotno matriko R, da je $A = QRQ^H$.

Občutljivosti lastnih vrednosti dobimo z ukazom `condest`. Če uporabimo `c = condest(A)`, potem je $c_i = |s_i|^{-1}$ občutljivost i-te lastne vrednosti.

Dodatna literatura

Za računanje lastnih vrednosti in vektorjev je na voljo obsežna literatura. V slovenščini je na voljo knjiga [8]. Kar se tiče tuje literature, lahko vse algoritme, ki smo jih navedli v tem poglavju, skupaj s potrebno analizo, najdete v [9]. Uporabna sta tudi učbenika [7] in [13].

Poglavje 2

Simetrični problem lastnih vrednosti

2.1 Uvod

V praksi je pogosto potrebno izračunati lastne vrednosti in vektorje za simetrične matrike. V tem primeru lahko izkoristimo lastnosti, ki jih imajo simetrične matrike, in pridemo do učinkovitejših in natančnejših algoritmov.

Naj bo torej A simetrična matrika velikosti $n \times n$. Vemo, da so vse lastne vrednosti realne in da je Schurova forma za simetrično matriko kar diagonalna matrika. Lastne vrednosti lahko uredimo tako, da velja

$$\lambda_n \leq \lambda_{n-1} \leq \cdots \leq \lambda_1,$$

pripadajoče lastne vektorje $x_1, \ldots, x_n$ pa lahko izberemo tako, da tvorijo ortonormirano bazo. Hitro vidimo, da velja

$$\|A\|_2 = \max(|\lambda_1|, |\lambda_n|).$$

Ker imamo opravka le z realnimi lastnimi vektorji, tudi *Rayleighov kvocient* računamo le za realne vektorje. Za $x \neq 0$ je

$$\rho(x, A) = \frac{x^T A x}{x^T x}.$$

Že iz prejšnjega poglavja vemo, da za Rayleighov kvocient veljajo naslednje lastnosti:

a) Za $\alpha \neq 0$ je $\rho(x, A) = \rho(\alpha x, A)$.

b) Za lastni vektor x_i je $\rho(x_i, A) = \lambda_i$.

c) Če je x približek za lastni vektor, je $\rho(x, A)$ najboljša aproksimacija za lastno vrednost v smislu, da je $\min_{\sigma \in \mathbb{R}} \|Ax - \sigma x\|_2$ dosežen pri $\sigma = \rho(x, A)$.

Za simetrične matrike velja še naslednja lastnost.

Lema 2.1 *Če je A simetrična matrika z lastnimi vrednostmi $\lambda_n \leq \cdots \leq \lambda_1$, potem za vsak $x \neq 0$ velja*

$$\lambda_n \leq \rho(x, A) \leq \lambda_1.$$

Dokaz. Če vektor x razvijemo po bazi lastnih vektorjev kot $x = \sum_{i=1}^{n} \alpha_i x_i$, dobimo

$$\rho(x, A) = \frac{\sum_{i=1}^{n} \alpha_i^2 \lambda_i}{\sum_{i=1}^{n} \alpha_i^2},$$

kar lahko očitno ocenimo navzgor in navzdol z λ_1 oziroma λ_n. ■

Iz zgornje leme sledi, da bi lahko največjo in najmanjšo lastno vrednost izrazili z maksimumom oziroma minimumom Rayleighovega kvocienta po vseh neničelnih vektorjih. Kot pravi naslednji izrek, lahko podobno izrazimo tudi vse preostale lastne vrednosti. Rezultat je temelj za številne pomembne teoretične rezultate.

Izrek 2.2 (Courant–Fischerjev minimaks izrek) [1] *Če je A simetrična matrika z lastnimi vrednostmi $\lambda_n \leq \cdots \leq \lambda_1$, potem za $i = 1, \ldots, n$ velja*

$$\lambda_i = \min_{\substack{S \subset \mathbb{R}^n \\ \dim(S) = n-i+1}} \max_{\substack{x \in S \\ x \neq 0}} \rho(x, A) = \max_{\substack{R \subset \mathbb{R}^n \\ \dim(R) = i}} \min_{\substack{x \in R \\ x \neq 0}} \rho(x, A). \tag{2.1}$$

Dokaz. Za poljubna podprostora $S, R \subset \mathbb{R}^n$, $\dim(R) = i$ in $\dim(S) = n - i + 1$, obstaja $x_{RS} \in R \cap S$, $x_{RS} \neq 0$, saj je $\dim(R) + \dim(S) = n + 1$. Očitno velja

$$\min_{\substack{x \in R \\ x \neq 0}} \rho(x, A) \leq \rho(x_{RS}, A) \leq \max_{\substack{x \in S \\ x \neq 0}} \rho(x, A).$$

Ker to velja za vsak par R, S velja tudi za par $\widetilde{R}, \widetilde{S}$, pri katerem je dosežen minimum oz. maksimum v izrazu (2.1). Torej

$$\max_{\substack{R \subset \mathbb{R}^n \\ \dim(R) = i}} \min_{\substack{x \in R \\ x \neq 0}} \rho(x, A) = \min_{\substack{x \in \widetilde{R} \\ x \neq 0}} \rho(x, A) \leq \rho(x_{\widetilde{R}\widetilde{S}}, A) = \max_{\substack{x \in \widetilde{S} \\ x \neq 0}} \rho(x, A) \leq \min_{\substack{S \subset \mathbb{R}^n \\ \dim(S) = n-i+1}} \max_{\substack{x \in S \\ x \neq 0}} \rho(x, A). \tag{2.2}$$

Po drugi strani pa za par $\widehat{R} = \mathrm{Lin}(x_1, \ldots, x_i)$ in $\widehat{S} = \mathrm{Lin}(x_i, \ldots, x_n)$ velja

$$\min_{\substack{x \in \widehat{R} \\ x \neq 0}} \rho(x, A) = \lambda_i = \max_{\substack{x \in \widehat{S} \\ x \neq 0}} \rho(x, A).$$

Od tod dobimo

$$\max_{\substack{R \subset \mathbb{R}^n \\ \dim(R) = i}} \min_{\substack{x \in R \\ x \neq 0}} \rho(x, A) \geq \min_{\substack{x \in \widehat{R} \\ x \neq 0}} \rho(x, A) = \lambda_i = \max_{\substack{x \in \widehat{S} \\ x \neq 0}} \rho(x, A) \geq \min_{\substack{S \subset \mathbb{R}^n \\ \dim(S) = n-i+1}} \max_{\substack{x \in S \\ x \neq 0}} \rho(x, A), \tag{2.3}$$

iz (2.2) in (2.3) pa sledi (2.1). ■

Posledica 2.3 *Če sta A in E simetrični matriki in so $\lambda_n \leq \cdots \leq \lambda_1$ lastne vrednosti matrike A, $\widehat{\lambda}_n \leq \cdots \leq \widehat{\lambda}_1$ pa lastne vrednosti matrike $A + E$, potem za $i = 1, \ldots, n$ velja*

$$\lambda_i + \lambda_n(E) \leq \widehat{\lambda}_i \leq \lambda_i + \lambda_1(E).$$

[1] Avstrijski matematik Ernst Sigismund Fischer (1875–1954) je izrek leta 1905 pokazal za matrike, nemški matematik Richard Courant (1888–1972) pa ga je leta 1920 razširil za neskončno razsežne operatorje. Courant je leta 1933 zapustil Nemčijo in se preselil v ZDA, kjer je leta 1936 ustanovil matematični inštitut, ki se od leta 1964 dalje imenuje Courantov inštitut matematičnih znanosti in velja za enega najboljših matematičnih inštitutov na svetu.

Dokaz. Iz $\rho(x, A+E) = \rho(x, A) + \rho(x, E)$ ocenimo

$$\rho(x, A) + \lambda_n(E) \leq \rho(x, A+E) \leq \rho(x, A) + \lambda_1(E)$$

in uporabimo Courant–Fischerjev minimaks izrek. ■

Od tod sledi naslednji, t.i. Weylov[2] izrek. Če se vam zdi znan, to ni nič nenavadnega, saj smo ga kot posledico Bauer–Fikeovega izreka zapisali že v posledici 1.5.

Posledica 2.4 (Weylov izrek) *Če sta A in E simetrični matriki in so $\lambda_n \leq \cdots \leq \lambda_1$ lastne vrednosti matrike A, $\widehat{\lambda}_n \leq \cdots \leq \widehat{\lambda}_1$ pa lastne vrednosti matrike $A+E$, potem za $i = 1, \ldots, n$ velja*

$$|\lambda_i - \widehat{\lambda}_i| \leq \|E\|_2.$$

Izrek 2.5 (Cauchyjev izrek o prepletanju) *Če je A simetrična matrika in je A_r njena vodilna podmatrika velikosti $r \times r$ za $r = 1, \ldots, n$, potem za $k = 1, \ldots, n-1$ velja*

$$\lambda_{k+1}(A_{k+1}) \leq \lambda_k(A_k) \leq \lambda_k(A_{k+1}) \leq \cdots \leq \lambda_2(A_{k+1}) \leq \lambda_1(A_k) \leq \lambda_1(A_{k+1}).$$

Dokaz. Dovolj je dokazati primer $k = n-1$. Če je $x' \in \mathbb{R}^{n-1}$ in $x = \begin{bmatrix} x' \\ 0 \end{bmatrix} \in \mathbb{R}^n$, potem je $\rho(x', A_{n-1}) = \rho(x, A)$.

Po Courant–Fischerjevem minimaks izreku velja

$$\lambda_k(A_{n-1}) = \min_{\substack{S' \subset \mathbb{R}^{n-1} \\ \dim(S')=n-k}} \max_{\substack{x' \in S' \\ x' \neq 0}} \rho(x', A_{n-1}).$$

Vsak podprostor lahko podamo z njegovim ortogonalnim komplementom, zato lahko pišemo

$$\lambda_k(A_{n-1}) = \min_{p'_1, \ldots, p'_{k-1} \in \mathbb{R}^{n-1}} \max_{\substack{x' \in \mathbb{R}^{n-1},\ x' \neq 0 \\ x' \perp p'_i,\ i=1,\ldots,k-1}} \rho(x', A_{n-1}).$$

Pogoj, da morajo biti vektorji $p'_1, \ldots, p'_{k-1}$ inearno neodvisni, smo izpustimo, saj je minimum očitno dosežen pri linearno neodvisnih vektorjih. Sedaj lahko pišemo

$$\begin{aligned} \lambda_k(A_{n-1}) &= \min_{p_1, \ldots, p_{k-1} \in \mathbb{R}^n} \max_{\substack{x \in \mathbb{R}^n,\ x \neq 0,\ x \perp e_n \\ x \perp p_i,\ i=1,\ldots,k-1}} \rho(x, A) \\ &\leq \min_{p_1, \ldots, p_{k-1} \in \mathbb{R}^n} \max_{\substack{x \in \mathbb{R}^n,\ x \neq 0 \\ x \perp p_i,\ i=1,\ldots,k-1}} \rho(x, A) = \lambda_k(A). \end{aligned}$$

Tako smo pokazali $\lambda_k(A_{n-1}) \leq \lambda_k(A)$. Po drugi strani pa velja

$$\begin{aligned} \lambda_k(A_{n-1}) &= \min_{p_1, \ldots, p_{k-1} \in \mathbb{R}^n} \max_{\substack{x \in \mathbb{R}^n,\ x \neq 0,\ x \perp e_n \\ x \perp p_i,\ i=1,\ldots,k-1}} \rho(x, A) \\ &= \min_{\substack{p_1, \ldots, p_{k-1}, p_k \in \mathbb{R}^n \\ p_k = e_n}} \max_{\substack{x \in \mathbb{R}^n,\ x \neq 0 \\ x \perp p_i,\ i=1,\ldots,k}} \rho(x, A) \\ &\geq \min_{p_1, \ldots, p_{k-1}, p_k \in \mathbb{R}^n} \max_{\substack{x \in \mathbb{R}^n,\ x \neq 0 \\ x \perp p_i,\ i=1,\ldots,k}} \rho(x, A) = \lambda_{k+1}(A). \end{aligned}$$

■

Naslednji izrek pove, da lahko iz norme ostanka $Ax - \beta x$ približka β za lastno vrednost in približka x za lastni vektor dobimo zelo dobro oceno, kako natančno β aproksimira točno lastno vrednost matrike A.

[2] Nemški matematik in teoretični fizik Hermann Weyl (1885–1955) spada med najvplivnejše matematike 20. stoletja.

Izrek 2.6 *Če je A simetrična matrika, $\|x\|_2 = 1$ in β približek za lastno vrednost, potem ima matrika A vsaj eno lastno vrednost λ_i, ki zadošča $|\beta - \lambda_i| \leq \|Ax - \beta x\|_2$.*

Dokaz. Naj bo $r = Ax - \beta x$. Dokaz je podoben kot pri Bauer–Fikeovemu izreku. Predpostavimo lahko, da se β razlikuje od vseh lastnih vrednosti, kar pomeni, da je $A - \beta I$ nesingularna. Matrika A se da diagonalizirati kot $A = XDX^T$, torej $A - \beta I = X(D - \beta I)X^T$. Sedaj je $x = (A - \beta I)^{-1}r$, kar pomeni $1 \leq \|(D - \beta I)^{-1}\|_2 \|r\|_2$, odtod pa dobimo

$$\min_i |\lambda_i - \beta| \leq \|r\|_2.$$

■

Definicija 2.7 *Vsaki simetrični matriki A lahko priredimo trojico celih števil (ν, ζ, π), kjer je ν število negativnih, ζ število ničelnih in π število pozitivnih lastnih vrednosti matrike A. Omenjeno trojico imenujemo* inercija *matrike A.*

Če je matrika X nesingularna, potem pravimo, da sta matriki A in X^TAX *kongruentni*. Izkaže se, da imajo kongruentne simetrične matrike isto inercijo.

Izrek 2.8 (Sylvester) [3] *Naj bo matrika A simetrična. Za poljubno nesingularno matriko X imata A in X^TAX isto inercijo.*

Dokaz. Naj bo matrika A velikosti $n \times n$. Denimo, da se inerciji matrik A in X^TAX razlikujeta tako, da ima A ν negativnih, X^TAX pa $\nu' < \nu$ negativnih lastnih vrednosti.

Naj bo $\mathcal{N}$ invariantni podprostor dimenzije ν, ki ga razpenjajo vsi lastni vektorji matrike A, ki pripadajo negativnim lastnim vrednostim. Podobno naj bo $\mathcal{P}$ podprostor dimenzije $n - \nu'$, ki ga razpenjajo vsi lastni vektorji matrike X^TAX, ki pripadajo nenegativnim lastnim vrednostim. Zaradi nesingularnosti matrike X je podprostor $X\mathcal{P}$ tudi dimenzije $n - \nu'$.

Ker je $n - \nu' + \nu > n$, obstaja neničelni vektor $x \in \mathcal{N} \cap X\mathcal{P}$. Zanj zaradi $x \in \mathcal{N}$ velja $x^TAx < 0$, po drugi strani pa zaradi $x \in X\mathcal{P}$ obstaja tak $y \in \mathcal{P}$, da je $x = Xy$, torej $x^TAx = y^T(X^TAX)y \geq 0$. Prišli smo do protislovja, kar pomeni, da je $\nu = \nu'$.

Podobno pokažemo, da velja tudi $\pi = \pi'$, od tod pa že sledi, da se inerciji ujemata. ■

S pomočjo Sylvestrovega izreka lahko dokažemo relativni Weylov izrek, s katerim lahko natančneje omejimo relativne spremembe lastnih vrednosti simetrične matrike A, če jo z ene strani pomnožimo z matriko X, z druge pa z X^T, kjer je X skoraj ortogonalna matrika.

Izrek 2.9 (Relativni Weylov izrek) *Naj bo A simetrična matrika z lastnimi vrednostmi $\lambda_n \leq \cdots \leq \lambda_1$. Če je X nesingularna matrika in so lastne vrednosti matrike X^TAX enake $\widehat{\lambda}_n \leq \cdots \leq \widehat{\lambda}_1$, potem za $i = 1, \ldots, n$ velja*

$$|\widehat{\lambda}_i - \lambda_i| \leq |\lambda_i|\epsilon,$$

kjer je $\epsilon = \|X^TX - I\|_2$.

[3] Angleški matematik James Joseph Sylvester (1814–1897) je znan tudi po tem, da je vpeljal izraza diskriminanta in graf (za matematični objekt, sestavljen iz točk in povezav).

Dokaz. Vemo, da je i-ta lastna vrednost matrike $A - \lambda_i I$ enaka 0. Po Sylvestrovem izreku o ohranjanju inercije mora tudi i-ta lastna vrednost matrike $X^T(A - \lambda_i I)X$ biti enaka 0. Če zapišemo

$$X^T(A - \lambda_i I)X = X^T AX - \lambda_i I + \lambda_i(I - X^T X),$$

potem iz Weylovega izreka sledi, da se i-ta lastna vrednost $X^T AX - \lambda_i I$, ki je enaka $\widehat{\lambda}_i - \lambda_i$, razlikuje od i-te lastne vrednosti $A - \lambda_i I$, ki je enaka 0, za manj kot $\|\lambda_i(I - X^T X)\|_2$, to pa je ravno $|\widehat{\lambda}_i - \lambda_i| \leq |\lambda_i|\epsilon$. ∎

2.2 Rayleighova iteracija

Če inverzno iteracijo kombiniramo z Rayleighovim kvocientom, dobimo *Rayleighovo iteracijo*, ki je predstavljena v algoritmu 2.1. Namesto fiksnega premika σ pri inverzni iteraciji, sedaj uporabljamo Rayleighov kvocient, ki je najboljši približek za lastno vrednost danega vektorja.

Algoritem 2.1 Rayleighova iteracija. Začetni podatek sta matrika A in neničelni vektor z_0.

$k = 0, 1, \ldots$
  $\sigma_k = \rho(z_k, A)$
  reši $(A - \sigma_k I)y_{k+1} = z_k$
  $z_{k+1} = \frac{y_{k+1}}{\|y_{k+1}\|_2}$

Metoda ni omejena le na simetrične matrike, lahko jo uporabimo tudi za nesimetrične. Pokazati se da, da je konvergenca Rayleighove iteracije v bližini enostavne lastne vrednosti kvadratična, v primeru simetrične matrike pa celo kubična.

Lema 2.10 *Naj bo normiran vektor z_0 približek za lastni vektor x_1. Če za simetrično matriko A z lastnimi vrednostmi $|\lambda_n| \leq \cdots \leq |\lambda_2| < |\lambda_1|$ izvedemo en korak potenčne metode z začetnim vektorjem z_0, potem velja*

$$\|z_1 \pm x_1\|_2 \leq \frac{|\lambda_2|}{|\lambda_1|}\|z_0 - x_1\|_2,$$

kjer predznak $\pm$ izberemo tako, da je norma razlike manjša.

Dokaz. Brez škode za splošnost lahko predpostavimo, da je $\lambda_1 > 0$. Če vektor z_0 razvijemo po lastnih vektorjih, dobimo $z_0 = \sum_{i=1}^n \alpha_i x_i$, kjer je $\alpha_1 \approx 1$. Od tod sledi $z_1 = \sum_{i=1}^n \beta_i x_i$, kjer je

$$\beta_1 = \frac{\alpha_1\lambda_1}{\left(\sum_{i=1}^n \alpha_i^2\lambda_i^2\right)^{1/2}} = \frac{1}{\left(1 + \sum_{i=2}^n \left(\frac{\alpha_i}{\alpha_1}\right)^2\left(\frac{\lambda_i}{\lambda_1}\right)^2\right)^{1/2}}.$$

Velja $\|z_0 - x_1\|_2^2 = 2(1 - \alpha_1)$ in podobno $\|z_1 \pm x_1\|_2^2 = 2(1 - \beta_1)$. Sedaj lahko ocenimo

$$\beta_1 \approx 1 - \frac{1}{2}\sum_{i=2}^n \left(\frac{\alpha_i}{\alpha_1}\right)^2\left(\frac{\lambda_i}{\lambda_1}\right)^2,$$

torej

$$1-\beta_1 \approx \frac{1}{2}\sum_{i=2}^{n}\left(\frac{\alpha_i}{\alpha_1}\right)^2\left(\frac{\lambda_i}{\lambda_1}\right)^2 \leq \frac{1}{2}\left(\frac{\lambda_2}{\lambda_1}\right)^2\sum_{i=2}^{n}\alpha_i^2 = \frac{1}{2}\left(\frac{\lambda_2}{\lambda_1}\right)^2(1-\alpha_1^2).$$

Tako dobimo

$$\|z_1 \pm x_1\|_2^2 = 2(1-\beta_1) \leq \left(\frac{\lambda_2}{\lambda_1}\right)^2(1+\alpha_1)(1-\alpha_1) \leq \left(\frac{\lambda_2}{\lambda_1}\right)^2 2(1-\alpha_1) = \left(\frac{\lambda_2}{\lambda_1}\right)^2\|z_0-x_1\|_2^2$$

in lema je dokazana. ∎

Posledica 2.11 *Naj bo A simetrična matrika in $|\lambda_i-\sigma| \ll |\lambda_j-\sigma|$ za $j \neq i$, $j=1,\ldots,n$. Če izvedemo en korak inverzne iteracije z začetnim vektorjem z_0, potem velja*

$$\|z_1-x_i\|_2 = \mathcal{O}(|\lambda_i-\sigma|\cdot\|z_0-x_1\|_2).$$

Lema 2.12 *Naj bo A simetrična matrika in naj bo normiran vektor z približek za lastni vektor x_k. Potem velja*

$$|\lambda_k-\rho(z,A)| \leq 2\|A\|_2\|z-x_k\|_2^2.$$

Dokaz. Brez škode za splošnost lahko predpostavimo, da je $k=1$. Če vektor z razvijemo po lastnih vektorjih kot $z=\sum_{i=1}^{n}\alpha_i x_i$, potem je $\|z-x_1\|_2^2 = 2(1-\alpha_1)$. Za razliko Rayleighovega kvocienta dobimo

$$\lambda_1-\rho(z,A) = \lambda_1\sum_{i=1}^{n}\alpha_i^2 - \sum_{i=1}^{n}\lambda_i\alpha_i^2 = \sum_{i=2}^{n}(\lambda_1-\lambda_i)\alpha_i^2$$

in ocenimo

$$|\lambda_1-\rho(z,A)| \leq 2\|A\|_2\sum_{i=2}^{n}\alpha_i^2 = 2\|A\|_2(1-\alpha_1^2) \leq 4\|A\|_2(1-\alpha_1) = 2\|A\|_2\|z-x_1\|_2^2.$$ ∎

Izrek 2.13 *Naj bo A simetrična matrika. Potem ima Rayleighova iteracija v bližini enostavne lastne vrednosti kubično konvergenco.*

Dokaz. Naj Rayleighova iteracija konvergira k enostavni lastni vrednosti λ_k s pripadajočim lastnim vektorjem x_k. Po enem koraku Rayleighove iteracije po posledici 2.11 velja

$$\|z_1-x_k\|_2 = \mathcal{O}(|\lambda_k-\sigma_0|\cdot\|z_0-x_k\|_2).$$

Ker po lemi 2.12 velja tudi

$$|\lambda_k-\rho(z_0,A)| \leq 2\|A\|_2\|z_0-x_k\|_2^2,$$

oceni skupaj vrneta $\|z_1-x_k\|_2 = \mathcal{O}(\|z_0-x_k\|_2^3)$ in konvergenca je res kubična. ∎

Zgled 2.1 *Naj bo $A=\begin{bmatrix}\lambda_1 & 0\\ 0 & \lambda_2\end{bmatrix}$, $\lambda_1>\lambda_2$ in $z_r=\begin{bmatrix}c_r\\ s_r\end{bmatrix}$, $\|z_r\|_2^2 = c_r^2+s_r^2 = 1$. Pri enem koraku Rayleighove iteracije dobimo*

$$\sigma_r = z_r^T A z_r = c_r^2\lambda_1 + s_r^2\lambda_2.$$

Iz sistema

$$\begin{bmatrix} \lambda_1 - c_r^2\lambda_1 - s_r^2\lambda_2 & 0 \\ 0 & \lambda_2 - c_r^2\lambda_1 - s_r^2\lambda_2 \end{bmatrix} y_{r+1} = z_r$$

dobimo

$$y_{r+1} = \frac{1}{(\lambda_1 - \lambda_2)c_r^2 s_r^2} \begin{bmatrix} c_r^3 \\ -s_r^3 \end{bmatrix},$$

od tod pa

$$z_{r+1} = \frac{1}{\sqrt{c_r^6 + s_r^6}} \begin{bmatrix} c_r^3 \\ -s_r^3 \end{bmatrix}.$$

V primeru $s_r \neq c_r$ imamo tako očitno kubično konvergenco proti e_1 oziroma e_2. □

2.3 QR iteracija za simetrični lastni problem

V primeru simetrične matrike je zgornja Hessenbergova matrika tridiagonalna, kar nam omogoča, da pridemo do učinkovitejše verzije QR iteracije. Za začetno redukcijo na tridiagonalno obliko zaradi simetrije porabimo približno polovico toliko operacij kot za redukcijo nesimetrične matrike na Hessenbergovo obliko, a to še vedno pomeni $\mathcal{O}(n^3)$ operacij.

Med samo QR iteracijo je razlika večja. Zaradi tridiagonalne oblike lahko en korak QR iteracije sedaj izvedemo z zahtevnostjo $\mathcal{O}(n)$, medtem ko imamo pri zgornji Hessenbergovi obliki, ki nastopa pri nesimetričnem primeru, $\mathcal{O}(n^2)$ operacij.

Pri QR iteraciji torej najprej poiščemo ortogonalno matriko Q, da je $T = QAQ^T$ tridiagonalna, potem pa delamo iteracijo z enojnim premikom:

$T_0 = T$
$k = 0, 1, \ldots$
  izberi premik σ_k
  $T_k - \sigma_k I = Q_k R_k$ (izračunaj QR razcep)
  $T_{k+1} = R_k Q_k + \sigma_k I$

Naj bo

$$T_k = \begin{bmatrix} a_1^{(k)} & b_1^{(k)} & & & \\ b_1^{(k)} & a_2^{(k)} & b_2^{(k)} & & \\ & \ddots & \ddots & \ddots & \\ & & b_{n-2}^{(k)} & a_{n-1}^{(k)} & b_{n-1}^{(k)} \\ & & & b_{n-1}^{(k)} & a_n^{(k)} \end{bmatrix}.$$

Kako izberemo premik:

- *Rayleighov premik*: vzamemo $\sigma_k = a_n^{(k)} = \rho(e_n, T_k)$. V tem primeru imamo za skoraj vse matrike zagotovljeno kubično konvergenco, a vseeno obstajajo primeri, ko metoda ne konvergira, če začetni približek ni dovolj dober.

  Kubično konvergenco v bližini enostavnih lastnih vrednosti nam zagotavlja izrek 2.14, ki povezuje Rayleighovo iteracijo in QR iteracijo.

- *Wilkinsonov premik*: za σ_k vzamemo tisto lastno vrednost matrike $\begin{bmatrix} a_{n-1}^{(k)} & b_{n-1}^{(k)} \\ b_{n-1}^{(k)} & a_n^{(k)} \end{bmatrix}$, ki je bližja $a_n^{(k)}$. Sedaj imamo za vse matrike dokazano vsaj linearno konvergenco, v praksi pa imamo za skoraj vse matrike kubično konvergenco (a brez dokaza).

Zgled 2.2 *Primer matrike, za katero QR iteracija z Rayleighovim premikom ne konvergira, je*

$$T_0 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

V tem primeru je $\sigma_0 = 0$ *in* $T_1 = T_0$. □

Izrek 2.14 *Če je T simetrična in delamo QR iteracijo z Rayleighovimi premiki, potem so premiki σ_k enaki Rayleighovim kvocientom, ki jih dobimo pri Rayleighovi iteraciji za matriko T, če za začetni vektor vzamemo $z_0 = e_n$.*

Dokaz. Najprej vpeljemo matrike $\overline{Q}_k = Q_0 Q_1 \cdots Q_k$ in $\overline{R}_k = R_k R_{k-1} \cdots R_0$, kjer so Q_i in R_i matrike iz QR iteracije. Pokažimo, da za vsak $k \geq 0$ velja

a) $T_{k+1} = \overline{Q}_k^T T \overline{Q}_k$,

b) $(T - \sigma_0 I) \cdots (T - \sigma_k I) = \overline{Q}_k \overline{R}_k$.

Točka a) sledi iz zveze $T_{k+1} = Q_k^T T_k Q_k$.

Pri točki b) uporabimo indukcijo. Za $k = 0$ očitno velja, saj je $T_0 - \sigma_0 I = Q_0 R_0$. Pokažimo, da če velja za $k-1$, potem velja tudi za k. Res, produkt $\overline{Q}_k \overline{R}_k$ lahko pišemo kot

$$\begin{aligned} \overline{Q}_k \overline{R}_k &= \overline{Q}_{k-1} Q_k R_k \overline{R}_{k-1} = \overline{Q}_{k-1} (T_k - \sigma_k I) \overline{R}_{k-1} \\ &= \overline{Q}_{k-1} \overline{Q}_{k-1}^T (T - \sigma_k I) \overline{Q}_{k-1} \overline{R}_{k-1} = (T - \sigma_k I) \overline{Q}_{k-1} \overline{R}_{k-1} \\ &= (T - \sigma_0 I) \cdots (T - \sigma_{k-1} I)(T - \sigma_k I). \end{aligned}$$

Pri izpeljavi smo upoštevali točko a) in dejstvo, da matrike oblike $T - \sigma_i I$ med seboj komutirajo.

Sedaj lahko dokažemo, da za vsak k velja $\rho(e_n, T_k) = \rho(z_k, T)$. To je očitno res za $k = 0$. Vektor z_{k+1} iz Rayleighove iteracije zadošča enačbi $(T - \sigma_k I) z_{k+1} = \alpha_k z_k$, kjer skalar α_k izberemo tako, da bo $\|z_{k+1}\|_2 = 1$. Od tod rekurzivno sledi

$$(T - \sigma_0 I) \cdots (T - \sigma_k I) z_{k+1} = \beta_k e_n$$

za primerno izbran skalar β_k. Po točki b) to lahko zapišemo kot

$$\overline{R}_k^T \overline{Q}_k^T z_{k+1} = \beta_k e_n,$$

kjer smo upoštevali, da zaradi simetrije matrike T velja $\overline{Q}_k \overline{R}_k = \overline{R}_k^T \overline{Q}_k^T$. To pa pomeni, da je

$$z_{k+1} = \beta_k \overline{Q}_k \overline{R}_k^{-T} e_n.$$

Ker je matrika $\overline{R}_k^{-T}$ spodnja trikotna, ima vektor $\overline{R}_k^{-T} e_n$ isto smer kot enotski vektor e_n, od koder sledi $z_{k+1} = \overline{Q}_k e_n$. Sedaj iz točke a) sledi

$$\rho(z_{k+1}, T) = \rho(\overline{Q}_k e_n, T) = \rho(e_n, \overline{Q}_k^T T \overline{Q}_k) = \rho(e_n, T_{k+1})$$

in izrek je dokazan. ■

Tako kot za nesimetrične matrike, je tudi simetrična varianta QR iteracije obratno stabilna. Za numerično izračunano diagonalno matriko $\widehat{D}$ obstaja taka ortogonalna matrika Q, da je $A + E = Q\widehat{D}Q^T$, kjer je $\|E\|_2 \approx \|A\|_2 u$. Weylov izrek nam potem zagotavlja, da za izračunane lastne vrednosti velja $|\widehat{\lambda}_i - \lambda_i| \leq \mathcal{O}(\|A\|)u$.

2.4 Sturmovo zaporedje

Naj bo

$$T = \begin{bmatrix} a_1 & b_1 & & & \\ b_1 & a_2 & b_2 & & \\ & \ddots & \ddots & \ddots & \\ & & b_{n-2} & a_{n-1} & b_{n-1} \\ & & & b_{n-1} & a_n \end{bmatrix}$$

nerazcepna tridiagonalna simetrična matrika (torej $b_i \neq 0$ za vsak i). Če s T_r označimo njeno vodilno $r \times r$ podmatriko in definiramo $f_r(\lambda) = \det(T_r - \lambda I)$, potem z razvijanjem po zadnji vrstici pridemo do rekurzivne formule

$$f_{r+1}(\lambda) = (a_{r+1} - \lambda) f_r(\lambda) - b_r^2 f_{r-1}(\lambda) \tag{2.4}$$

za $r = 0, \ldots, n-1$, ki se začne z $f_0(\lambda) \equiv 1$ in $f_1(\lambda) = a_1 - \lambda$.

Izrek 2.15 *Polinomi $f_0, \ldots, f_n$ tvorijo Sturmovo[4] zaporedje, kar pomeni, da zadoščajo naslednjim trem točkam:*

1) *$f_0(\lambda) \neq 0$ za vsak λ.*
2) *Če je $f_r(\lambda_0) = 0$ za $r < n$, potem je $f_{r-1}(\lambda_0) f_{r+1}(\lambda_0) < 0$.*
3) *Če je $f_n(\lambda_0) = 0$, potem je $f_{n-1}(\lambda_0) f_n'(\lambda_0) < 0$.*

Dokaz. Točka 1) je očitna. Pri točki 2) iz rekurzivne formule sledi $f_{r+1}(\lambda_0) = -b_r^2 f_{r-1}(\lambda_0)$. Obe vrednosti sta neničelni, saj bi sicer veljalo $f_i(\lambda_0) = 0$ za $i = 0, \ldots, n$, to pa je v protislovju s točko 1).

Pri točki 3) definiramo $\Delta_r(\lambda_0) = f_r(\lambda_0) f_{r-1}'(\lambda_0) - f_{r-1}(\lambda_0) f_r'(\lambda_0)$ za $r = 1, \ldots, n$. Z računanjem lahko hitro preverimo, da velja

$$\Delta_{r+1}(\lambda_0) = f_r^2(\lambda_0) + b_r^2 \Delta_r(\lambda_0).$$

Ker je $\Delta_1(\lambda_0) = 1 > 0$, je $\Delta_r(\lambda_0) > 0$ za vsak r. Torej tudi $\Delta_n(\lambda_0) = -f_{n-1}(\lambda_0) f_n'(\lambda_0) > 0$. ■

[4] Francoski matematik Jacques Charles François Sturm (1803–1855).

Posledica 2.16 *Nerazcepna tridiagonalna simetrična matrika ima enostavne lastne vrednosti.*

Dokaz. To sledi iz točke 3) zadnjega izreka, saj v primeru $f_n(\lambda_0) = 0$ velja $f_n'(\lambda_0) \neq 0$. ∎

Pri fiksnem λ_0 označimo z $u(\lambda_0)$ število ujemanj predznaka v zaporedju $f_0(\lambda_0), \ldots, f_n(\lambda_0)$. Pri tem vsako notranjo ničlo štejemo za eno ujemanje, ničlo na koncu pa ne. Primeri:

$$\begin{aligned} u(++--+) &= 2, \\ u(++0-+) &= 2, \\ u(++0-+0) &= 2. \end{aligned}$$

Izrek 2.17 *Število $u(\lambda_0)$ je enako številu lastnih vrednosti matrike T, ki so strogo večje od λ_0.*

Dokaz. Naj λ teče od $-\infty$ do ∞. Pri $\lambda = -\infty$ imamo očitno zaporedje $+++\cdots$, pri $\lambda = \infty$ pa $+-+-\cdots$. Tako je $u(-\infty) = n$ in $u(\infty) = 0$.

Pokazali bomo, da se število $u(\lambda)$ lahko spremeni le, če prečkamo ničlo polinoma f_n, ne pa tudi, če prečkamo ničlo polinoma f_r, $r < n$.

Naj bo $f_r(\lambda_0) = 0$, $r < n$. Potem je iz tabele

	$\lambda_0 - \epsilon$	λ_0	$\lambda_0 + \epsilon$
f_{r-1}	$\pm$	$\pm$	$\pm$
f_r	$\pm$	0	$\mp$
f_{r+1}	$\mp$	$\mp$	$\mp$

razvidno, da pri zadosti majhnem $\epsilon > 0$ velja $u(\lambda_0 - \epsilon) = u(\lambda_0 + \epsilon)$.

V primeru $f_n(\lambda_0) = 0$ pa iz tabele

	$\lambda_0 - \epsilon$	λ_0	$\lambda_0 + \epsilon$
f_{n-1}	$\pm$	$\pm$	$\pm$
f_n	$\pm$	0	$\mp$

vidimo, da pri zadosti majhnem $\epsilon > 0$ velja $u(\lambda_0 - \epsilon) = u(\lambda_0 + \epsilon) + 1$. ∎

Sedaj lahko z bisekcijo ali kakšno drugo metodo poiščemo k-to lastno vrednost. Iščemo točko λ_k, za katero velja $u(\lambda_k - \epsilon) = k$ in $u(\lambda_k + \epsilon) = k - 1$ za dovolj majhen $\epsilon > 0$. Če nimamo boljše ocene za začetni interval, lahko vzamemo $[-\|T\|, \|T\|]$ za poljubno normo matrike T.

Metoda je uporabna tudi, če nas zanimajo samo lastne vrednosti na določenem intervalu ali pa nekaj lastnih vrednosti. Tako je npr. razlika $u(\beta) - u(\alpha)$ enaka številu lastnih vrednosti matrike T, ki so na intervalu $(\alpha, \beta]$.

Za izračun $u(\lambda)$ preko tričlenske rekurzivne formule potrebujemo $4n + \mathcal{O}(1)$ operacij, pri čemer smo predpostavili, da kvadrate obdiagonalnih elementov izračunamo vnaprej. Izkaže pa se, da lahko naredimo še bolje. Če definiramo kvociente $d_i(\lambda) = f_i(\lambda)/f_{i-1}(\lambda)$, potem zanje velja $d_1(\lambda) = a_1 - \lambda$ in

$$d_{r+1}(\lambda) = a_{r+1} - \lambda - \frac{b_r^2}{d_r(\lambda)}. \tag{2.5}$$

Za izračun te formule potrebujemo le $3n + \mathcal{O}(1)$ operacij, namesto zaporednih ujemanj predznakov pa zdaj preštejemo, koliko vrednosti $d_1(\lambda), \ldots, d_n(\lambda)$ je pozitivnih.

Matrika $T - \lambda I$ je tridiagonalna. Denimo, da za to matriko obstaja razcep LDL^T, kjer je L spodnja trikotna bidiagonalna matrika z enicami na diagonali oblike

$$L = \begin{bmatrix} 1 & & & \\ l_1 & 1 & & \\ & \ddots & \ddots & \\ & & l_{n-1} & 1 \end{bmatrix},$$

matrika $D = \operatorname{diag}(d_1, \ldots, d_n)$ pa je diagonalna. S primerjanjem elementov matrik $T - \lambda I$ in LDL^T pridemo do naslednjega algoritma za izračun matrik L in D.

Algoritem 2.2 LDL^T razcep matrike $T - \lambda I$.

$d_1 = a_1 - \lambda$
$i = 1, \ldots, n-1$
    $l_i = b_i / d_i$
    $d_{i+1} = a_{i+1} - \lambda - l_{i-1}^2 d_{i-1}$

Iz zgornjega algoritma dobimo formulo (2.5), če formulo za l_i vstavimo v formulo za d_{i+1}. Ker ne pivotiramo, se lahko v (2.5) pojavi deljenje z 0. Če računamo v aritmetiki, ki zadošča standardom IEEE, to ni nobena težava. Sploh je računanje po formuli (2.5) zelo stabilno, saj velja naslednji izrek, katerega dokaz lahko najdete npr. v [8].

Lema 2.18 *Če ne pride do prekoračitve ali podkoračitve, se predznaki numerično izračunanih diagonalnih elementov $\widehat{d}_1, \ldots, \widehat{d}_n$ ujemajo s predznaki točnih $d_1, \ldots, d_n$ iz LDL^T razcepa matrike $T + \delta T$, kjer je $\delta a_k = 0$ in $|\delta b_k| \leq (5/2)|b_k|u$ za $k = 1, \ldots, n$.*

2.5 Deli in vladaj

Ta metoda lahko za nerazcepno simetrično tridiagonalno matriko velikosti $n \times n$, kjer je $n > 25$, izračuna lastne vrednosti in lastne vektorje hitreje od QR iteracije. Do pojave metode RRR, ki jo bomo spoznali v razdelku 2.7, je bila to najhitrejša metoda za reševanje simetričnega problema lastnih vrednosti.[5]

Naj bo T nerazcepna tridiagonalna simetrična matrike oblike

$$T = \begin{bmatrix} a_1 & b_1 & & \\ b_1 & \ddots & \ddots & \\ & \ddots & \ddots & b_{n-1} \\ & & b_{n-1} & a_n \end{bmatrix}.$$

Za $m \approx n/2$ razdelimo matriko T kot

$$T = \begin{bmatrix} T_1 & 0 \\ 0 & T_2 \end{bmatrix} + b_m v v^T,$$

[5] Osnovno idejo je predstavil Nizozemec Jan Cuppen leta 1981, a se je izkazalo, da ni dovolj stabilna. To sta popravila Ming Gu in Stanley C. Eisenstat z Univerze Yale, ki sta leta 1992 objavila stabilno različico.

kjer je

$$T_1 = \begin{bmatrix} a_1 & b_1 & & \\ b_1 & \ddots & \ddots & \\ & \ddots & a_{m-1} & b_{m-1} \\ & & b_{m-1} & a_m - b_m \end{bmatrix}, \quad T_2 = \begin{bmatrix} a_{m+1} - b_m & b_{m+1} & & \\ b_{m+1} & a_{m+2} & \ddots & \\ & \ddots & \ddots & b_{n-1} \\ & & b_{n-1} & a_n \end{bmatrix}$$

in

$$v = e_m + e_{m+1} = [0 \ \cdots \ 0 \ 1 \ 1 \ 0 \cdots \ 0]^T.$$

Opazimo lahko, da ima matrika $b_m v v^T$ rang 1. Prav to je tudi razlog, zakaj smo b_m odšteli še od matrik T_1 in T_2 in nismo T samo razdelili na dva dela.

Matriki T_1 in T_2 sta simetrični in tridiagonalni, zato obstajata ortogonalni matriki Q_1, Q_2 in diagonalni matriki D_1, D_2, da je $T_1 = Q_1 D_1 Q_1^T$ in $T_2 = Q_2 D_2 Q_2^T$. Potem je

$$T = \begin{bmatrix} Q_1 & 0 \\ 0 & Q_2 \end{bmatrix} \left(\begin{bmatrix} D_1 & 0 \\ 0 & D_2 \end{bmatrix} + b_m u u^T \right) \begin{bmatrix} Q_1^T & 0 \\ 0 & Q_2^T \end{bmatrix},$$

kjer je vektor u sestavljen iz zadnje vrstice Q_1 in prve vrstice Q_2 kot

$$u = \begin{bmatrix} Q_1^T & 0 \\ 0 & Q_2^T \end{bmatrix} v = \begin{bmatrix} Q_1(m,:)^T \\ Q_2(1,:)^T \end{bmatrix}.$$

Lastne vrednosti T so tako enake lastnim vrednostim $D + \rho u u^T$, kjer je $D = \begin{bmatrix} D_1 & 0 \\ 0 & D_2 \end{bmatrix}$ in $\rho = b_m$. Problem smo tako prevedli na računanje lastnih vrednosti diagonalne matrike, ki jo pokvarimo z matriko ranga 1. Če izračunamo lastne vrednosti in vektorje $D + \rho u u^T = Q' \Lambda Q'^T$, potem so na diagonali Λ lastne vrednosti matrike T, stolpci matrike $Q = \begin{bmatrix} Q_1 & 0 \\ 0 & Q_2 \end{bmatrix} Q'$ pa so pripadajoči lastni vektorji.

Algoritem 2.3 Osnovna varianta metode deli in vladaj. Začetni podatek je nerazcepna tridiagonalna simetrična matrika T. Algoritem vrne diagonalno matriko Λ in ortogonalno matriko Q, da je $T = Q \Lambda Q^T$.

$[Q, \Lambda]$=deliinvladaj(T)
če je T velikosti 1×1, potem vrni $Q = 1, \Lambda = T$
sicer:
razdeli $T = \begin{bmatrix} T_1 & 0 \\ 0 & T_2 \end{bmatrix} + b_m v v^T$.
$[Q_1, D_1]$=deliinvladaj(T_1)
$[Q_2, D_2]$=deliinvladaj(T_2)
iz Q_1, Q_2, D_1, D_2 izračunaj $D + \rho u u^T$
izračunaj lastne vrednosti Λ in vektorje Q' za $D + \rho u u^T$
$Q = \begin{bmatrix} Q_1 & 0 \\ 0 & Q_2 \end{bmatrix} Q'$.

Osnovna verzija metode deli in vladaj je zapisana v algoritmu 2.3. Zapisati moramo še, kako lahko ekonomično in natančno izračunamo lastne vrednosti in vektorje matrike $D + \rho u u^T$. Za samo računanje uredimo diagonalne elemente matrike D tako, da je $d_1 \geq d_2 \geq \cdots \geq d_n$. Seveda moramo hkrati ustrezno preurediti tudi elemente vektorja u.

Najprej izločimo vse lastne vrednosti in vektorje, za katere lahko uporabimo naslednjo lemo.

Lema 2.19 *Naj bo* $A = D + \rho uu^T$, *kjer je* $D = \mathrm{diag}(d_1, \ldots, d_n)$, $d_1 \geq d_2 \geq \cdots \geq d_n$ *in* $u = [u_1 \ \cdots \ u_n]^T$.

a) Če je $d_i = d_{i+1}$, *je* d_i *lastna vrednost matrike* A, *lastni vektor pa je* $[0 \ \cdots \ 0 \ -u_{i+1} \ u_i \ 0 \ \cdots \ 0]^T$.

b) Če je $u_i = 0$, *je* d_i *lastna vrednost matrike* A, *lastni vektor pa je* e_i.

V nadaljevanju lahko tako predpostavimo, da so si d_i paroma različni in da so vsi u_i neničelni. Potrebovali bomo še naslednjo lemo.

Lema 2.20 *Za dana vektorja* x *in* y *velja* $\det(I + xy^T) = 1 + y^T x$.

Dokaz. Matrika $I + xy^T$ ima očitno vsaj $(n-1)$-kratno lastno vrednost 1, za katero je lastni vektor poljuben neničelen vektor, pravokoten na y. Ker je vsota lastnih vrednosti enaka sledi matrike, ki je enaka $n + y^T x$, je manjkajoča lastna vrednost ravno $1 + y^T x$. Determinanta, ki je enaka produktu vseh lastnih vrednosti, je potem očitno $1 + y^T x$. ∎

Sdaj predpostavimo, da je λ, ki se razlikuje od vseh diagonalnih elementov $d_1, \ldots, d_n$, lastna vrednost matrike $D + \rho uu^T$. Potem je matrika $D - \lambda I$ nesingularna in iz

$$\det(D + \rho uu^T - \lambda I) = \det\Big((D - \lambda I)(I + \rho(D - \lambda I)^{-1} uu^T)\Big),$$

sledi

$$\det(I + \rho(D - \lambda I)^{-1} uu^T) = 0.$$

Iz leme 2.20 sledi, da je λ ničla *sekularne enačbe* $f(\lambda) = 0$, kjer je

$$\begin{aligned} f(\lambda) &= \det(I + \rho(D - \lambda I)^{-1} uu^T) = 1 + \rho u^T (D - \lambda I)^{-1} u \\ &= 1 + \rho \sum_{i=1}^{n} \frac{u_i^2}{d_i - \lambda}. \end{aligned}$$

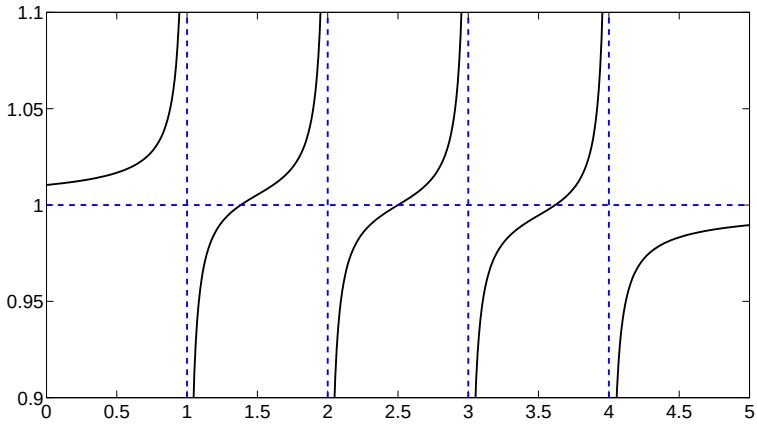
Kako zgleda graf $f(\lambda)$? Asimptota je $y = 1$. Ker je $f'(\lambda) = \rho \sum_{i=1}^{n} \frac{u_i^2}{(d_i - \lambda)^2}$, je za $\rho > 0$ funkcija strogo naraščajoča (med poli), sicer pa padajoča. Ničle ležijo med poli, ena pa desno od zadnjega pola (pri $\rho > 0$) ali levo od prvega pola (pri $\rho < 0$).

Zgled 2.3 *Če vzamemo matriko*

$$T = \begin{bmatrix} 2 & 1 & & \\ 1 & 2.01 & 0.01 & \\ & 0.01 & 3.01 & 1 \\ & & 1 & 3 \end{bmatrix}$$

in $m = 2$, *potem dobimo* $\rho = 0.01$, $d_i = i$ *in* $u_i^2 = 0.5$ *za* $i = 1, \ldots, 4$. *Graf sekularne enačbe je prikazan na sliki 2.1 in opazimo lahko, da so ničle zelo blizu polov.* □

Denimo, da z neko numerično metodo rešimo sekularno enačbo in poiščemo vse lastne vrednosti. Naslednja lema nam pove, kako potem izračunamo pripadajoče lastne vektorje.



Slika 2.1: Tipičen graf sekularne enačbe.

Lema 2.21 *Če je α lastna vrednost $D+\rho uu^T$, je $(D-\alpha I)^{-1}u$ ustrezni lastni vektor.*

Dokaz.

$$\begin{aligned}(D+\rho uu^T)(D-\alpha I)^{-1}u &= (D-\alpha I+\alpha I+\rho uu^T)(D-\alpha I)^{-1}u\\ &= u+\alpha(D-\alpha I)^{-1}u+u(\rho u^T(D-\alpha I)^{-1}u)\\ &= u+\alpha(D-\alpha I)^{-1}u-u\\ &= \alpha(D-\alpha I)^{-1}u,\end{aligned}$$

saj iz $f(\alpha)=1+\rho u^T(D-\alpha I)^{-1}u=0$ sledi $\rho u^T(D-\alpha I)^{-1}u=-1$. ∎

Ker rešujemo diagonalni sistem, lahko vsak lastni vektor izračunamo v času $\mathcal{O}(n)$.

Sedaj pa si podrobno poglejmo, kako rešujemo sekularno enačbo. Kot je razvidno tudi iz slike 2.1, ne moremo uporabiti navadne tangentne metode. Ničle so namreč lahko zelo blizu polov in če ne uporabimo dobrega začetnega približka, metoda ne bo skonvergirala k pravi lastni vrednosti. Namesto aproksimacije funkcije s tangento zato raje uporabimo preprosto racionalno funkcijo, ki se prilega funkciji f, potem pa ničlo te racionalne funkcije vzamemo za nov približek za ničlo funkcije f.

Denimo, da iščemo rešitev na intervalu (d_{i+1},d_i), začetni približek pa je x_r. Sedaj poiščemo racionalno funkcijo oblike

$$h(\lambda)=\frac{c_1}{d_i-\lambda}+\frac{c_2}{d_{i+1}-\lambda}+c_3,$$

za katero velja $h(x_r)=f(x_r)$ in $f'(x_r)=h'(x_r)$. Zaradi stabilnosti razdelimo f na dva dela kot

$$f(\lambda)=1+\rho\sum_{k=1}^{i}\frac{u_k^2}{d_k-\lambda}+\rho\sum_{k=i+1}^{n}\frac{u_k^2}{d_k-\lambda}=:1+\psi_1(\lambda)+\psi_2(\lambda).$$

To naredimo zato, da v vsoti za $\psi_1(\lambda)$ oziroma $\psi_2(\lambda)$ seštevamo enako predznačene člene. Sedaj določimo c_1 in c_1' tako, da za

$$h_1(\lambda)=\frac{c_1}{d_i-\lambda}+c_1'$$

velja $h_1(x_r) = \psi_1(x_r)$ in $h_1'(x_r) = \psi_1'(x_r)$. Podobno določimo c_2 in c_2' tako, da za

$$h_2(\lambda) = \frac{c_2}{d_{i+1} - \lambda} + c_2'$$

velja $h_2(x_r) = \psi_2(x_r)$ in $h_2'(x_r) = \psi_2'(x_r)$. Sedaj je $h(\lambda) = 1 + h_1(\lambda) + h_2(\lambda)$ iskana racionalna funkcija. Enačba $h(\lambda) = 0$ ima dve rešitvi, za x_{r+1} pa vzamemo tisto, ki leži znotraj (d_{i+1}, d_i). Konvergenca je zelo hitra, saj h zelo dobro aproksimira f na intervalu (d_{i+1}, d_i).

Pri numeričnem računanju se je izkazalo, da so lastni vektorji, ki jih izračunamo preko leme 2.21, lahko slabo ortogonalni, zato je bilo potrebno algoritem popraviti. Pri popravku vektor u nadomestimo z bližnjim $\widehat{u}$, za katerega se potem izkaže, da je z ortogonalnostjo lastnih vektorjev vse v redu, vse pa temelji na naslednjem izreku.

Izrek 2.22 (Löwner) [6] *Naj bo $D = \operatorname{diag}(d_1, \ldots, d_n)$, kjer je $d_n < d_{n-1} < \cdots < d_1$, in naj bodo $\alpha_n < \alpha_{n-1} < \cdots < \alpha_1$ dana števila, ki se prepletajo s števili d_i:*

$$d_n < \alpha_n < \cdots < d_1 < \alpha_1.$$

Potem za vektor $\widehat{u}$, podan z

$$|\widehat{u}_i| = \left(\frac{\prod_{j=1}^{n} (\alpha_j - d_i)}{\prod_{j=1, j\neq i}^{n} (d_j - d_i)} \right)^{1/2} \tag{2.6}$$

velja, da so α_i točne lastne vrednosti matrike $\widehat{D} = D + \widehat{u}\widehat{u}^T$.

Dokaz. Po eni strani je karakteristični polinom matrike $\widehat{D}$ enak $\det(\widehat{D} - \lambda I) = \prod_{j=1}^{n} (\alpha_j - \lambda)$, po drugi strani pa zaradi $\widehat{D} - \lambda I = (D - \lambda I)(I + (D - \lambda I)^{-1} \widehat{u}\widehat{u}^T)$ velja

$$\det(\widehat{D} - \lambda I) = \prod_{j=1}^{n} (d_j - \lambda) \left(1 + \sum_{j=1}^{n} \frac{\widehat{u}_j^2}{d_j - \lambda} \right).$$

Ko vstavimo $\lambda = d_i$ in izenačimo izraza, dobimo

$$\prod_{j=1}^{n} (\alpha_j - d_i) = \widehat{u}_i^2 \prod_{\substack{j=1 \\ j\neq i}}^{n} (d_j - d_i).$$

Od tod sledi

$$\widehat{u}_i^2 = (\alpha_i - d_i) \prod_{\substack{j=1 \\ j\neq i}}^{n} \left(\frac{\alpha_j - d_i}{d_j - d_i} \right).$$

Zaradi prepletanja se predznak $\alpha_j - d_i$ ujema s predznakom $d_j - d_i$ za $j \neq i$. Ker velja še $\alpha_i > d_i$, je izraz na desni strani res pozitiven in ga lahko korenimo. ∎

Opomba 2.1 *Zaradi enostavnosti smo v zadnjem izreku privzeli, da je $\rho = 1$. V primeru, ko je $\rho > 0$, lahko u in ρ vedno tako normiramo, da je to res. Na podoben način bi lahko pokazali, da formula (2.6) drži tudi za negativni ρ, ko lahko privzamemo, da je $\rho = -1$ in so $\alpha_1, \ldots, \alpha_n$ točne lastne vrednosti matrike $\widehat{D} = D - \widehat{u}\widehat{u}^T$. V tem primeru se števila prepletajo kot $\alpha_n < d_n < \cdots < \alpha_1 < d_1$.*

[6] Karl Löwner (1893–1968) je bil češki matematik, ki je po nemški zasedbi Prage uspel pobegniti v ZDA, kjer je nadaljeval kariero kot Charles Loewner.

Stabilno računanje lastnih vrednosti in vektorjev matrike $D + \rho uu^T$, kjer je $\rho = \pm 1$, poteka na naslednji način:

a) Iz sekularne enačbe izračunamo lastne vrednosti $\alpha_1, \ldots, \alpha_n$ matrike $D + \rho uu^T$.

b) Po Löwnerjevem izreku izračunamo vektor $\widehat{u}$, da so $\alpha_1, \ldots, \alpha_n$ točne lastne vrednosti matrike $D + \rho\widehat{u}\widehat{u}^T$. Predznake izberemo tako, da se predznak $\widehat{u}_i$ ujema s predznakom u_i za $i = 1, \ldots, n$.

c) Za lastne vektorje vzamemo vektorje $(D - \alpha_i I)^{-1}\widehat{u}$ za $i = 1, \ldots, n$.

Naj bo $T(n)$ število operacij, ki jih porabi algoritem za matriko velikosti n. Potem velja

$$\begin{array}{rcll} T(n) & = & 2T(n/2) & \text{(rekurzivni klic dveh podproblemov)} \\ & & + \mathcal{O}(n^2) & \text{(reševanje sekularne enačbe)} \\ & & + \mathcal{O}(n^2) & \text{(izračun lastnih vektorjev } D + \rho uu^T\text{)} \\ & & + n^3. & \text{(izračun produkta za matriko } Q\text{)} \end{array}$$

Od tod sledi, da je časovna zahtevnost algoritma za izračun vseh lastnih vrednosti in vektorjev enaka $T(n) = (4/3)n^3 + \mathcal{O}(n^2)$, v praksi pa je še manjša, saj lahko velikokrat lastne vektorje in vrednosti izračunamo preko leme 2.19, kar nam močno olajša izračun produkta za matriko Q.

2.6 Jacobijeva metoda

Pri tej metodi matrike predhodno ne reduciramo na tridiagonalno obliko. Ideja je, da matriko A z množenji z Givensovimi rotacijami z leve in z desne poskusimo spraviti čim bližje diagonalni matriki. Ker je Jacobi[7] rotacije uporabljal že dolgo pred Givensom, jih bomo tu imenovali Jacobijeve rotacije.

Denimo, da bi radi v matriki A uničili izvendiagonalni element na mestu pq. Najprej poiščemo rotacijo $R = \begin{bmatrix} c & -s \\ s & c \end{bmatrix}$, da bo veljalo

$$R^T \begin{bmatrix} a_{pp} & a_{pq} \\ a_{pq} & a_{qq} \end{bmatrix} R = \begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} a_{pp} & a_{pq} \\ a_{pq} & a_{qq} \end{bmatrix} \begin{bmatrix} c & -s \\ s & c \end{bmatrix} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}. \tag{2.7}$$

Iz zvez $(a_{qq} - a_{pp})sc + a_{pq}(c^2 - s^2) = 0$ in $c^2 + s^2 = 1$ dobimo

$$\tau := \frac{\cos 2\varphi}{\sin 2\varphi} = \frac{c^2 - s^2}{2sc} = \frac{a_{pp} - a_{qq}}{2a_{pq}}.$$

Če definiramo $t := s/c = \tan\varphi$, potem velja (uporabimo formule za tangens dvojnega kota)

$$t^2 + 2\tau t - 1 = 0.$$

Rešitev je

$$t = \frac{\operatorname{sign}(\tau)}{|\tau| + \sqrt{1 + \tau^2}}, \quad c = \sqrt{\frac{1}{1 + t^2}}, \quad s = ct.$$

Algoritem 2.4 En korak Jacobijeve metode za računanje lastnih vrednosti simetrične matrke. Začetni podatki so simetrična matrika A, indeksa p, q in dosedanji produkt rotacij Q. Algoritem posodobi matriki A in Q.

$[A, Q] = \text{jac}(A, Q, p, q)$
$\tau = \frac{a_{pp}-a_{qq}}{2a_{pq}}$
$t = \frac{\text{sign}(\tau)}{|\tau|+\sqrt{1+\tau^2}}$
$c = \sqrt{\frac{1}{1+t^2}}$
$s = ct$
$A = R_{pq}^T A R_{pq}$
$Q = QR_{pq}$ (če potrebujemo tudi lastne vektorje)

Tako smo dobili formule za izračun Jacobijeve rotacije $\text{jac}(A, p, q)$, ki v A uniči element a_{pq}.

Algoritem za eno rotacijo je predstavljen v Algoritmu 2.6. Po rotaciji (p, q) se v A spremenita vrstici p in q ter stolpca p in q. To pomeni, da se ničle, ki smo jih naredili v prejšnjih korakih, z novimi rotacijami lahko pokvarijo. Izkaže pa se, da se kljub temu norma izvediagonalnih elementov z rotacijami zmanjšuje.

Definicija 2.23 *Za $n \times n$ matriko A definiramo*

$$\text{off}(A) = \sqrt{\sum_{\substack{j,k=1 \\ j \neq k}}^{n} |a_{jk}|^2}.$$

Vidimo, da je $\text{off}(A)$ v bistvu Frobeniusova norma matrike A brez diagonale.

Lema 2.24 *Če A' dobimo iz A z Jacobijevo rotacijo* $\text{jac}(A, p, q)$*, potem velja*

$$\text{off}(A')^2 = \text{off}(A)^2 - 2a_{pq}^2.$$

Dokaz. Iz $A' = R_{pq}^T A R_{pq}$ sledi $\|A'\|_F = \|A\|_F$. Za diagonalne elemente A' velja $a'_{ii} = a_{ii}$ za $i \neq p, q$, za preostala dva elementa pa zaradi (2.7) velja $a'^2_{pp} + a'^2_{qq} = a^2_{pp} + a^2_{qq} + 2a^2_{pq}$. Torej mora za izvendiagonalne elemente veljati $\text{off}(A')^2 = \text{off}(A)^2 - 2a^2_{pq}$. ∎

Z vsako Jacobijevo rotacijo se tako zmanjša $\text{off}(A)$. Denimo, da v vsakem koraku uničimo po absolutni vrednosti največji izvendiagonalni element. Potem lahko hitro preverimo, da velja ocena

$$\text{off}(A')^2 \leq \left(1 - \frac{2}{n(n-1)}\right) \text{off}(A)^2.$$

To pomeni, da norma izvendiagonalnih elementov konvergira proti 0. Postopek ponavljamo, dokler ni $\text{off}(A) \leq \epsilon$.

Kako uničujemo elemente:

[7] Jacobi je najstarejšo numerično metodo za računanje lastnih vrednosti objavil leta 1846. Ob pojavi prvih računalnikov se je izkazalo, da je metodo zelo preprosto implementirati, zato je bila na začetku to glavna metoda za računanje lastnih vrednosti simetričnih matrik. V nadaljevanju jo je izpodrinila hitrejša QR iteracija.

- *klasična varianta*: v vsakem koraku poiščemo po absolutni vrednosti največji izvendiagonalni element in ga uničimo. Sicer imamo po številu rotacij res najhitrejšo konvergenco, a imamo veliko primerjanj, ki povečajo časovno zahtevnost metode. Če si shranjujemo po absolutni vrednosti največje elemente v vsakem stolpcu, potem lahko največji izvendiagonalni element poiščemo v času $\mathcal{O}(n)$, prav toliko pa tudi porabimo v vsakem koraku, da posodobimo seznam največjih elementov. Iskanje torej poveča časovno zahtevnost, a ima isti red kot sicer porabimo za en korak Jacobijeve metode.
- *ciklična varianta*: v vedno enakem vrstnem redu gremo skozi vse elemente. Tu nimamo primerjanj, lahko pa zaradi uničevanja elementov, ki so majhni po absolutni vrednosti porabimo veliko korakov.
- *pragovna varianta*: v vedno enakem vrstnem redu gremo skozi vse elemente, a uničimo le tiste elemente, ki so po absolutni vrednosti čez neko mejo, ki jo zmanjšamo v vsakem sprehodu.

Jacobijeva metoda porabi več operacij kot QR iteracija ali deli in vladaj, njena prednost pa je, da za simetrične pozitivno definitne matrike lastne vrednosti blizu 0 izračuna relativno natančneje od ostalih metod.

2.7 Relativno robustne reprezentacije

Iščemo vse lastne pare tridiagonalne simetrične matrike T. S pomočjo algoritmov, ki smo jih že spoznali, to najhitreje izvedemo tako, da najprej izračunamo samo lastne vrednosti preko QR iteracije, za kar porabimo $\mathcal{O}(n^2)$ operacij. Potem lahko preko inverzne iteracije izračunamo enega po enega še lastne vektorje. Končni algoritem porabi le $\mathcal{O}(n^2)$ operacij, a se v primeru bližjih lastnih vrednosti lahko zgodi, da dobljeni lastni vektorji niso dovolj ortogonalni. V tem primeru je potrebna naknadna ortogonalizacija, ki v najslabšem primeru porabi $\mathcal{O}(n^3)$ operacij. V tem razdelku bomo na grobo predstavili algoritem, ki zna s pomočjo RRR (relativno robustnih reprezentacij) ta problem rešiti z uporabo $\mathcal{O}(n^2)$ operacij.[8]

Najprej poglejmo, kako lahko za nerazcepno tridiagonalno simetrično matriko

$$T = \begin{bmatrix} a_1 & b_1 & & \\ b_1 & \ddots & \ddots & \\ & \ddots & \ddots & b_{n-1} \\ & & b_{n-1} & a_n \end{bmatrix}$$

natančno izračunamo lastni vektor z za znano lastno vrednost λ. Lastni vektor zadošča enačbi $(T - \lambda I)z = 0$. Ker je T nerazcepna, mora veljati $z_1 \neq 0$ in $z_n \neq 0$. Lastni vektor lahko zato npr. določimo tako, da fiksiramo $z_1 = 1$ in iz prve do predzadnje enačbe po vrsti izračunamo preostale komponente. Algoritem je

$$\begin{aligned} &z_1 = 1 \\ &z_2 = -(a_1 - \lambda)/b_1 \\ &k = 2, \ldots, n-1 \\ &\quad z_{k+1} = -(b_{k-1}z_{k-1} + (a_k - \lambda)z_k)/b_k \end{aligned}$$

[8] Algoritem je predstavil indijski matematik Inderjit S. Dhillon v svojem doktoratu leta 1997.

Pri računanju smo opustili zadnjo enačbo, ki ji vektor z avtomatično zadošča. Če pa namesto točne lastne vrednosti λ uporabimo približek $\widetilde{\lambda}$, potem tako izračunani vektor z ne zadošča zadnji enačbi in dobimo ostanek $(T-\widetilde{\lambda}I)z = \delta_n e_n$ za nek δ_n. Podobno, če opustimo k-to enačbo, potem za ostanek velja $(T-\widetilde{\lambda}I)z = \delta_k e_k$. Opustitev k-te enačbe torej ustreza temu, da kot približek za lastni vektor vzamemo $z = (T-\widetilde{\lambda}I)^{-1}e_k$. Vprašanje je, pri katerem indeksu k dobimo najboljši približek za lastni vektor.

Naj bodo $x_1, \ldots, x_n$ ortonormirani lastni vektorji matrike T, $\widetilde{\lambda}$ pa naj bo približek za lastno vrednost λ_j. Iz razvoja $e_k = \sum_{i=1}^{n} (x_i)_k x_i$ za $k = 1, \ldots, n$ sledi

$$z^{(k)} = (T-\widetilde{\lambda}I)^{-1}e_k = \frac{(x_j)_k}{\lambda_j - \widetilde{\lambda}} \left(x_j + \sum_{i \neq j} \frac{(x_i)_k}{(x_j)_k} \cdot \frac{\lambda_j - \widetilde{\lambda}}{\lambda_i - \widetilde{\lambda}} x_i \right).$$

Iz zgornje enačbe vidimo, da bo $z^{(k)}$ dober približek za lastni vektor x_j, če

a) opustimo k-to enačbo, kjer ima lastni vektor x_j maksimalno komponento,

b) je λ_j dobro izolirana od ostalih lastnih vrednosti, kar pomeni $|\lambda_j - \widetilde{\lambda}| \ll |\lambda_i - \widetilde{\lambda}|$ za $i \neq j$.

S pomočjo t.i. zasukanega razcepa lahko ocenimo, katera komponenta lastnega vektorja je po absolutni vrednosti največja. Tako lahko natančno izračunamo vse lastne vektorje.

2.7.1 Zasukani razcep

Za matriko $T - \lambda I$ lahko izračunamo razcepa $T - \lambda I = LD^+L^T = UD^-U^T$, kjer je

$$L = \begin{bmatrix} 1 & & & \\ l_1 & 1 & & \\ & \ddots & \ddots & \\ & & l_{n-1} & 1 \end{bmatrix}, \quad D^+ = \begin{bmatrix} d_1^+ & & & \\ & d_2^+ & & \\ & & \ddots & \\ & & & d_n^+ \end{bmatrix},$$

$$U = \begin{bmatrix} 1 & u_1 & & \\ & \ddots & \ddots & \\ & & 1 & u_{n-1} \\ & & & 1 \end{bmatrix}, \quad D^- = \begin{bmatrix} d_1^- & & & \\ & d_2^- & & \\ & & \ddots & \\ & & & d_n^- \end{bmatrix}.$$

Razcepa izračunamo z algoritmoma 2.5 in 2.6.

Algoritem 2.5 LD^+L^T razcep matrike $T - \lambda I$.

$d_1^+ = a_1$
$k = 1, \ldots, n-1$
  $l_k = b_k / d_k^+$
  $d_{k+1}^+ = a_{k+1} - \lambda - l_k b_k$

Algoritem 2.6 UD^-U^T razcep matrike $T - \lambda I$.

$d_n^- = a_n$
$k = n-1, \ldots, 1$
  $u_k = b_k / d_{k+1}^-$
  $d_k^- = a_k - \lambda - u_k b_k$

Če na zgornjih k vrsticah matrike $T - \lambda I$ naredimo razcep LD^+L^T, na spodnjih $n-k+1$ pa UD^-U^T, potem dobimo *zasukani razcep* $T - \lambda I = N_k D_k N_k^T$, kjer je

$$N_k = \begin{bmatrix} 1 & & & & & & \\ l_1 & \ddots & & & & & \\ & \ddots & 1 & & & & \\ & & l_{k-1} & 1 & u_k & & \\ & & & & 1 & \ddots & \\ & & & & & \ddots & u_{n-1} \\ & & & & & & 1 \end{bmatrix}$$

in $D_k = \operatorname{diag}(d_1^+, \ldots, d_{k-1}^+, \gamma_k, d_{k+1}^-, \ldots, d_n^-)$. Pri tem moramo na novo določiti le element γ_k.

Izrek 2.25 *Za zasukani razcep za $k = 1, \ldots, n$ velja:*

1) $\gamma_k = d_k^+ + d_k^- - a_k$,
2) *če je matrika $T - \lambda I$ nesingularna, potem velja* $1/\gamma_k = e_k^T (T - \lambda I)^{-1} e_k$.

Posledica zgornjega izreka je, da iz razcepov LD^+L^T in UD^-U^T lahko enostavno izračunamo vse zasukane razcepe $N_k D_k N_k^T$ za $k = 1, \ldots, n$.

Posledica 2.26 *Naj bo $N_k D_k N_k^T$ zasukani razcep matrike $T - \widetilde{\lambda} I$. Potem velja*

$$\frac{1}{\gamma_k} = \frac{(x_j)_k^2}{\lambda_j - \widetilde{\lambda}} + \sum_{i \neq j} \frac{(x_i)_k^2}{\lambda_i - \widetilde{\lambda}}.$$

Naj bo lastna vrednost λ_j dobro separirana od ostalih, kar pomeni $|\lambda_j - \widetilde{\lambda}| \ll |\lambda_i - \widetilde{\lambda}|$ za $i \neq j$. Po posledici 2.26 majhna vrednost $|\gamma_k|$ ustreza nadpovprečni vrednosti $|(x_j)_k|$. To nam omogoča, da iz zasukanega razcepa ocenimo, kateri element lastnega vektorja je maksimalen.

Iz zasukanega razcepa lahko enostavno izračunamo približek za lastni vektor. Če je $T - \widetilde{\lambda} I = N_k D_k N_k^T$, potem ima sistem $(T - \widetilde{\lambda} I) z = \gamma_k e_k$ enolično rešitev, ki jo dobimo s postopkom, opisanim v algoritmu 2.7.

Naj bo $\widetilde{\lambda}$ približek za lastno vrednost matrike T. Pripadajoči lastni vektor z izračunamo po algoritmu 2.8, katerega zahtevnost je $12n + \mathcal{O}(1)$ operacij.

Algoritem 2.7 Reševanje $(T-\widetilde{\lambda}I)z=\gamma_k e_k$ preko zasukanega razcepa $T-\widetilde{\lambda}I=N_kD_kN_k^T$.

$z_k=1$
$j=k-1,\ldots,1$
  $z_j=-l_jz_{j+1}$ (v primeru $z_{j+1}=0$ uporabimo $z_j=-b_{j+1}z_{j+2}/b_j$)
$j=k+1,\ldots,n$
  $z_j=-u_{j-1}z_{j-1}$ (v primeru $z_{j-1}=0$ uporabimo $z_j=-b_{j-2}z_{j-2}/b_{j-1}$)

Algoritem 2.8 Izračun lastnega vektorja preko zasukanega razcepa.

1) Izračunaj razcepa $T-\widetilde{\lambda}I=LD^+L^T$ in $T-\widetilde{\lambda}I=UD^-U^T$.
2) Izračunaj $\gamma_1,\ldots,\gamma_n$ za zasukane razcepe in poišči minimalni $|\gamma_k|$.
3) Reši sistem $(T-\widetilde{\lambda}I)z=\gamma_k e_k$ z algoritmom 2.7 in normiraj z.

Opisani postopek lahko izračuna zelo natančne približke za lastne vektorje, če je lastna vrednost dobro separirana od ostalih. Če to ni res, potem lahko z ustreznim premikom σ poskrbimo, da bo v matriki $T-\sigma I$ lastna vrednost $\lambda_j-\sigma$ dobro separirana od ostalih. Tu pa je pomembno, da imamo matriko predstavljeno tako, da se s takim premikom ne pokvari relativna natančnost lastnih vrednosti. Kako to naredimo, bomo obravnavali v naslednjem podrazdelku.

2.7.2 Relativno robustna reprezentacija matrike

Tridiagonalno matriko T lahko predstavimo z vektorjema diagonalnih in obdiagonalnih elementov a in b. Izkaže se, da lahko majhne relativne spremembe a in b povzročijo velike relativne spremembe lastnih vrednosti in vektorjev, zato ta predstavitev ni *relativno robustna.*

Kadar je T pozitivno definitna, jo lahko predstavimo z vektorjema diagonalnih in obdiagonalnih elementov faktorja Choleskega. Pokazati se da, da majhne relativne spremembe teh elementov povzročijo majhne relativne spremembe lastnih parov, zato je takšna predstavitev relativno robustna. Točna definicija relativno robustne reprezentacije je naslednja.

Definicija 2.27 *Množica števil $\{p_i\}$, ki enolično določa matriko T, je* relativno robustna reprezentacija (RRR), *če se lastni pari matrike $T+\delta T$, ki jo določa množica relativno malo zmotenih elementov $\{p_i(1+\epsilon_i)\}$, relativno malo razlikujejo od lastnih parov matrike T. Če reprezentacija z visoko relativno natančnostjo določa le lastne vrednosti $(\lambda_j,\ldots,\lambda_k)$, pravimo da je delna $RRR(j,\ldots,k)$.*

Če je matrika T pozitivno definitna, potem lahko namesto razcepa Choleskega uporabimo LDL^T razcep, ki je prav tako RRR. S tem se izognemo računsko zahtevnim kvadratnim korenom. Če je T nedefinitna, lahko uporabimo razcep $T-\sigma I=LDL^T$ za premik σ, pri čemer ob primerno izbranem σ dosežemo, da je razcep delna RRR za iskane lastne vrednosti.

Denimo, da je $T=LDL^T$ RRR matrike T, radi pa bi izračunali razcepa $L^+D^+L^{+T}$ in $U^-D^-U^{-T}$ za matriko $T-\sigma I$. Obstajata algoritma dstqds in dqds, ki sta pomemben del metode RRR, s katerima lahko ta dva razcepa izračunamo učinkovito in tako, da ohranimo delno RRR, spotoma pa dobimo še vse koeficiente $\gamma_1,\ldots,\gamma_n$ za zasukane razcepe $T-\sigma I=N_kD_kN_k^T$ za $k=1,\ldots,n$. Podrobnosti lahko najdete npr. v [25].

Naj bo $\widetilde{\lambda}$ približek za lastno vrednost matrike LDL^T, ki je RRR. Potem približek za lastni vektor dobimo preko algoritma 2.8, kjer za izračun zasukanih razcepov uporabimo prej omenjena algoritma dstqds in dqds.

Naj bo $\widetilde{\lambda}$ približek za lastno vrednost λ_j matrike $T = LDL^T$. Za lastni vektor z, izračunan preko inverzne iteracije, velja

$$|\sin \angle(z, x_j)| \leq \frac{\mathcal{O}(nu\|T\|)}{\text{gap}(\widetilde{\lambda})},$$

kjer je $\text{gap}(\widetilde{\lambda}) = \min_{i \neq j} |\widetilde{\lambda} - \lambda_i|$. Za lastni vektor, izračunan preko metode RRR, pa velja

$$|\sin \angle(z, x_j)| \leq \frac{\mathcal{O}(nu)}{\text{relgap}(\widetilde{\lambda})}, \quad \text{kjer je} \quad \text{relgap}(\widetilde{\lambda}) = \frac{\text{gap}(\widetilde{\lambda})}{|\widetilde{\lambda}|}.$$

Če je absolutna vrednost $|\widetilde{\lambda}|$ dosti manjša kot $\|T\|$, je metoda RRR lahko veliko natančnejša.

2.7.3 Večkratne relativno robustne reprezentacije

Končni algoritem uporablja večkratne relativno robustne reprezentacije (MRRR). S premiki in preračuni RRR preko dstqds in dqds dosežemo, da bodo lastne vrednosti, za katere računamo lastne vrednosti, dobro relativno separirane od ostalih.

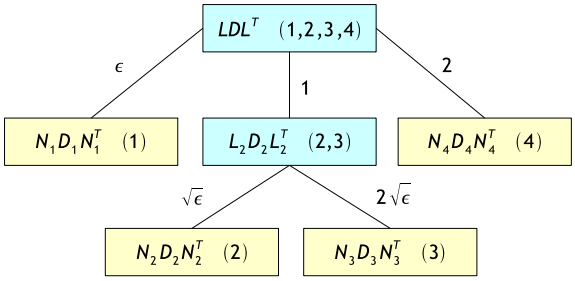
Algoritem 2.9 Metoda večkratnih relativnih robustnih reprezentacij.

1) Izračunaj razcep $T + \tau I = L_0 D_0 L_0^T$, kjer premik τ določiš tako, da bo dobljeni razcep RRR.
2) Relativno natančno izračunaj lastne vrednosti $\widetilde{\lambda}_1, \ldots, \widetilde{\lambda}_n$ matrike $L_0 D_0 L_0^T$.
3) $l = 1, m = n$
4) Razdeli lastne vrednosti $\widetilde{\lambda}_l, \ldots, \widetilde{\lambda}_m$ na izolirane in na gruče.
5) Za vsako izolirano lastno vrednost $\widetilde{\lambda}_j$ izračunaj lastni vektor preko RRR.
6) Za vsako gručo $\widetilde{\lambda}_j, \ldots, \widetilde{\lambda}_{j+k-1}$
   - Preko dstqds izračunaj $L_0 D_0 L_0^T - \tau_s I = L_S D_S L_S^T$, kjer τ_s izbereš tako, da bo v $L_S D_S L_S^T$ vsaj ena lastna vrednost izolirana in da bo $L_S D_S L_S^T$ delna $RRR(j, \ldots, j+k-1)$.
   - Relativno natančno izračunaj lastne vrednosti $\mu_j, \ldots, \mu_{j+k-1}$ matrike $L_s D_s L_s^T$.
   - Določi $\widetilde{\lambda}_i = \mu_i$ za $i = j, \ldots, j+k-1$
   - $l = j, m = j+k-1$, $L_0 = L_s$, $D_0 = D_s$, rekurzivno se vrni na korak 5).

Za lastno vrednost $\widetilde{\lambda}_j$ pravimo, da je relativno izolirana, če velja $\text{relgap}(\widetilde{\lambda}_j) \geq \delta$. Po drugi strani lastne vrednosti $\widetilde{\lambda}_j, \ldots, \widetilde{\lambda}_{j+k-1}$ tvorijo gručo, če velja $\text{reldis}(\widetilde{\lambda}_{j-1}, \widetilde{\lambda}_j) \geq \delta$, $\text{reldis}(\widetilde{\lambda}_{j+k-1}, \widetilde{\lambda}_{j+k}) \geq \delta$ in $\text{reldis}(\widetilde{\lambda}_i, \widetilde{\lambda}_{i+1}) < \delta$ za $i = j, \ldots, j+k-2$, kjer za δ vzamemo npr. 10^{-3} in je

$$\text{reldis}(\lambda, \mu) = \frac{|\lambda - \mu|}{|\lambda|}.$$

Zgled 2.4 *Denimo, da ima matrika lastne vrednosti ϵ, $1+\sqrt{\epsilon}$, $1+2\sqrt{\epsilon}$, 2. Slika 2.2 predstavlja, kako poteka izračun lastnih vektorjev preko MRRR. Vsak list, ki jih je toliko kot je lastnih vrednosti, predstavlja en izračun lastnega vektorja. Vsaka povezava predstavlja prehod na delno RRR premaknjene matrike, kjer uporabimo algoritem dstqds, povezava pa je označena z uporabljenim premikom.* □



Slika 2.2: Zgled predstavitvenega diagrama metode MRRR.

Zahtevnost končne metode za izračun k lastnih parov je $\mathcal{O}(kn)$. Predvideva se, da bo MRRR sčasoma postal osnovna metoda za reševanje simetričnega problema lastnih vrednosti, saj je metoda večinoma hitrejša od ostalih in potrebuje najmanj dodatnega spomina.

Dodatna literatura

Za računanje lastnih vrednosti je na voljo obsežna literatura. V slovenščini lahko skoraj celotno snov najdete v knjigi [8]. V tuji literaturi lahko skoraj vse algoritme, ki smo jih navedli, skupaj s potrebno analizo, najdete v [9], uporabna sta tudi učbenika [7] in [13].

Relativno robustne reprezentacije so lepo opisane v diplomskem delu [25].

Poglavje 3

Posplošitve problema lastnih vrednosti

3.1 Posplošeni problem lastnih vrednosti

Definicija 3.1 *Dani sta kvadratni matriki A in B. Množico vseh matrik oblike $A - \lambda B$, kjer je $\lambda \in \mathbb{C}$, imenujemo* matrični šop *in označimo z (A, B) ali $A - \lambda B$.* Karakteristični polinom *matričnega šopa (A, B) je $p(\lambda) = \det(A - \lambda B)$. Če karakteristični polinom ni identično enak 0, potem je matrični šop* regularen, *sicer pa* singularen.

Če je matrični šop (A, B) regularen in je

$$Ax = \lambda Bx$$

za neničelni vektor x, potem je λ (končna) lastna vrednost *in x* (desni) lastni vektor. *Podobno je neničelni vektor y* levi lastni vektor *za λ, če je $y^H A = \lambda y^H B$.*

Problemu iskanja lastnih vrednosti matričnega šopa pravimo *posplošeni problem lastnih vrednosti*. Očitno je standardni problem lastnih vrednosti poseben primer posplošenega, kjer vzamemo $B = I$. Če je (A, B) regularen matrični šop, potem so njegove končne lastne vrednosti ničle karakterističnega polinoma $\det(A - \lambda B)$, ki je stopnje $m \leq n$. V primeru $m < n$ ima šop še lastno vrednost ∞ z večkratnostjo $n - m$.

Zgled 3.1 *V primeru*

$$A - \lambda B = \begin{bmatrix} 1 & & \\ & 1 & \\ & & 0 \end{bmatrix} - \lambda \begin{bmatrix} 2 & & \\ & 0 & \\ & & 1 \end{bmatrix}$$

dobimo $p(\lambda) = (2\lambda - 1)\lambda$, torej so lastne vrednosti $\lambda_1 = 1/2$, $\lambda_2 = 0$ in $\lambda_3 = \infty$. □

Neskončne lastne vrednosti se pojavijo natanko takrat, ko je matrika B singularna. Vsak neničelni vektor iz $\ker(B)$ je desni lastni vektor za lastno vrednost ∞.

Izrek 3.2 *Za regularen matrični šop (A, B) velja:*

1) *Če je matrika B nesingularna, so vse lastne vrednosti šopa (A, B) končne in enake lastnim vrednostim matrike $B^{-1}A$ ali AB^{-1}.*

2) *Če je matrika B singularna, ima šop (A, B) lastno vrednost ∞ z geometrijsko večkratnostjo* $\dim(\ker(B))$.

3) *Če je matrika A nesingularna, so lastne vrednosti šopa (A, B) recipročne lastne vrednosti matrike $A^{-1}B$ oziroma BA^{-1}, kjer lastna vrednost 0 ustreza neskončni lastni vrednosti šopa (A, B).*

Definicija 3.3 *Če sta matriki U in V nesingularni, potem sta matrična šopa (A, B) in (UAV, UBV)* ekvivalentna.

Izrek 3.4 *Ekvivalentna regularna matrična šopa (A, B) in (UAV, UBV) imata iste lastne vrednosti. Za lastne vektorje velja:*

a) x je desni lastni vektor za (A, B) natanko tedaj, ko je $V^{-1}x$ desni lastni vektor za (UAV, UBV),

b) y je levi lastni vektor za (A, B) natanko tedaj, ko je $U^{-H}y$ levi lastni vektor za (UAV, UBV).

Posplošitev Jordanove forme za regularne matrične šope je Weierstrassova forma. Za vsak regularen matrični šop (A, B) obstajata nesingularni matriki U in V, da je

$$U(A - \lambda B)V = \operatorname{diag}(J_{n_1}(\lambda_1) - \lambda I_{n_1}, \ldots, J_{n_k}(\lambda_k) - \lambda I_{n_k}, N_{m_1}, \ldots, N_{m_l}),$$

kjer je

$$J_{n_i}(\lambda_i) = \begin{bmatrix} \lambda_i & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ & & & \lambda_i \end{bmatrix}, \qquad N_{m_i} = \begin{bmatrix} 1 & -\lambda & & \\ & \ddots & \ddots & \\ & & \ddots & -\lambda \\ & & & 1 \end{bmatrix}.$$

Blok $J_{n_i}(\lambda_i)$ pripada končni lastni vrednosti λ_i, blok N_{m_i} pa neskončni lastni vrednosti. Opazimo lahko, da je $N_{m_i} = I_{m_i} - \lambda J_{m_i}(0)$. Tako kot Jordanova forma je tudi Weierstrassova forma neprimerna za numerično računanje. Stabilnejša je posplošena Schurova forma, o kateri govori naslednji izrek.

Izrek 3.5 *Za vsak regularen matrični šop $A - \lambda B$ obstajata unitarni matriki Q in Z, da je*

$$Q^H(A - \lambda B)Z = S - \lambda T,$$

kjer sta matriki S in T zgoraj trikotni. *Lastne vrednosti so potem kvocienti $\lambda_i = s_{ii}/t_{ii}$ za $t_{ii} \neq 0$ in ∞ v primeru $t_{ii} = 0$.*

Dokaz. Tako, kot pri dokazu izreka 1.2 o obstoju navadne Schurove forme, uporabimo indukcijo. Za matriki velikosti 1×1 je forma trivialna. Denimo, da obstaja za vse pare matrik velikosti $(n-1) \times (n-1)$, matriki A in B pa sta velikosti $n \times n$.

Ker je šop (A, B) regularen, ima vsaj eno lastno vrednost λ, pri čemer je možno tudi $\lambda = \infty$. Naj bo x normiran desni lastni vektor za λ. Iz $Ax = \lambda Bx$ sledi, da sta vektorja Ax in Bx kolinearna. Oba hkrati ne moreta biti enaka nič, saj potem šop ne bi bil regularen. Torej obstaja tak normiran vektor y, da Ax in Bx ležita v podprostoru, ki ga razpenja y.

Naj bo $X = [x\ X_1]$ taka unitarna matrika, da je njen prvi stolpec enak vekorju x, podobno naj velja za matriko $Y = [y\ Y_1]$ in vektor y. Potem velja

$$Y^H A X = \begin{bmatrix} \alpha & a^T \\ 0 & A_1 \end{bmatrix} \quad \text{in} \quad Y^H B X = \begin{bmatrix} \beta & b^T \\ 0 & B_1 \end{bmatrix}.$$

Če je lastna vrednost λ končna, potem je $\lambda = \alpha/\beta$, sicer pa je $\beta = 0$. Ker po indukcijski predpostavki za šop (A_1, B_1) obstaja posplošena Schurova forma, nam to omogoča zapisati posplošeno Schurovo formo za šop (A, B).

Konkretno, po indukcijski predpostavki obstajata unitarni matriki Q_1 in Z_1, da sta matriki $Q_1^H A_1 Z_1$ in $Q_1^H B_1 Z_1$ zgornji trikotni. Potem sta matriki

$$\begin{bmatrix} 1 & 0 \\ 0 & Q_1^H \end{bmatrix} Y^H A X \begin{bmatrix} 1 & 0 \\ 0 & Z_1 \end{bmatrix} \quad \text{in} \quad \begin{bmatrix} 1 & 0 \\ 0 & Q_1^H \end{bmatrix} Y^H B X \begin{bmatrix} 1 & 0 \\ 0 & Z_1 \end{bmatrix}$$

zgoraj trikotni in imamo posplošeni Schurov razcep. ■

Situacija $s_{ii} = t_{ii} = 0$ je možna le, če je matrični šop (A, B) singularen.

Če sta matriki A in B realni, potem obstaja *realna posplošena Schurova forma*, kjer sta matriki Q in Z ortogonalni, S je kvazi zgornja trikotna, T pa zgornja trikotna matrika. Tako se v primeru realnih matrik tudi pri posplošenem problemu lastnih vrednosti lahko izognemo kompleksni aritmetiki.

Kaj lahko povemo o občutljivosti lastnih vrednosti posplošenega problema lastnih vrednosti? Da lahko enakovredno v analizo vključimo še neskončne lastne vrednosti, uporabljamo ločno razdaljo, definirano z

$$\chi(\alpha, \beta) = \frac{|\alpha - \beta|}{\sqrt{1 + |\alpha|^2}\sqrt{1 + |\beta|^2}},$$

za poljubni kompleksni števili α, β. V limiti dobimo

$$\chi(\alpha, \infty) = \frac{1}{\sqrt{1 + |\alpha|^2}}.$$

V [28] najdemo naslednjo oceno z ločno razdaljo.

Izrek 3.6 *Naj bo λ enostavna lastna vrednost šopa (A, B) z normiranim desnim lastnim vektorjem x in levim y. Če je $\widetilde{\lambda}$ ustrezna lastna vrednost zmotenega šopa $(\widetilde{A}, \widetilde{B})$, kjer je $\|A - \widetilde{A}\|_2 \leq \epsilon$ in $\|B - \widetilde{B}\|_2 \leq \epsilon$, potem je*

$$\chi(\lambda, \widetilde{\lambda}) \leq \frac{\epsilon}{|y^H A x|^2 + |y^H B x|^2} + \mathcal{O}(\epsilon^2).$$

3.2 QZ algoritem

Dani sta realni matriki A in B. Za numerično računanje posplošene Schurove forme šopa (A, B) imamo na voljo QZ algoritem, ki je izpeljanka QR metode za nesimetričen problem lastnih vrednosti. Na začetku šop (A, B) z ortogonalnimi ekvivalentnimi transformacijami reduciramo na šop $(\widetilde{A}, \widetilde{B}) = (\widetilde{Q} A \widetilde{Z}, \widetilde{Q} B \widetilde{Z})$, kjer je prva matrika zgornja Hessenbergova, druga pa zgornja trikotna.

Poglejmo, kako naredimo to redukcijo v primeru, ko sta matriki velikosti 4×4. Najprej poiščemo tako ortogonalno matriko P, ki pretvori matriko B v zgornjo trikotno obliko, torej

$$A_1 = PA = \begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{bmatrix}, \quad B_1 = PB = \begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times \end{bmatrix}.$$

Matriko P lahko najenostavneje skonstruiramo s pomočjo Householderjevih zrcaljenj, kjer v bistvu uporabimo algoritem za računanje QR razcepa matrike B. Nato z Givensovimi rotacijami enega po enega uničujemo elemente pod poddiagonalo matrike A in hkrati popravljamo matriko B nazaj na trikotno obliko. Najprej poiščemo rotacijo R_{34}, da je

$$A_2 = R_{34}^T A_1 = \begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ 0 & \times & \times & \times \end{bmatrix}, \quad B_2 = R_{34}^T B_1 = \begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & + & \times \end{bmatrix}.$$

Novi element v matriki B_2 uničimo z Givensovo rotacijo z desne, ki ne spremeni oblike matrike A_2:

$$A_3 = A_2 \widetilde{R}_{34} = \begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ 0 & \times & \times & \times \end{bmatrix}, \quad B_3 = B_2 \widetilde{R}_{34} = \begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times \end{bmatrix}.$$

Podobno uničimo naslednji element v matriki A in nato popravimo matriko B, dobimo

$$A_4 = R_{23}^T A_3 = \begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \end{bmatrix}, \quad B_4 = R_{23}^T B_3 = \begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & + & \times & \times \\ 0 & 0 & 0 & \times \end{bmatrix},$$

$$A_5 = A_4 \widetilde{R}_{23} = \begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \end{bmatrix}, \quad B_5 = B_4 \widetilde{R}_{23} = \begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times \end{bmatrix}.$$

V matriki A gremo na drugi stolpec in uničimo še element na mestu $(4,2)$.

$$A_6 = R_{34}^T A_5 = \begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \end{bmatrix}, \quad B_6 = R_{34}^T B_5 = \begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & + & \times \end{bmatrix},$$

$$\widetilde{A} = A_6 \widetilde{R}_{34} = \begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \end{bmatrix}, \quad \widetilde{B} = B_6 \widetilde{R}_{34} = \begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times \end{bmatrix}.$$

Tako smo z ortogonalnimi transformacijami pretvorili matriki A in B v zgornjo Hessenbergovo in trikotno obliko. Za splošni matriki velikosti $n \times n$ je časovna zahtevnost $8n^3 + \mathcal{O}(n^2)$ operacij za izračun $\widetilde{A}$ in $\widetilde{B}$, še dodatnih $7n^3 + \mathcal{O}(n^2)$ pa potrebujemo za izračun prehodnih ortogonalnih matrik $\widetilde{Q}$ in $\widetilde{Z}$, ki jih dobimo iz produktov uporabljenih zrcaljenj in rotacij.

V nadaljevanju predpostavimo, da smo redukcijo že naredili in je torej matrika A zgornja Hessenbergova, matrika B pa zgornja trikotna. Pri QZ iteraciji v bistvu izvajamo implicitno QR metodo na matriki $C = AB^{-1}$, ki je zgornja Hessenbergova.

Začnemo z $(A_0, B_0) = (A, B)$, potem pa v vsakem koraku posodobimo matrični šop (A_k, B_k) v

$$(A_{k+1}, B_{k+1}) = (Q_k A_k Z_k, Q_k B_k Z_k),$$

kjer ortogonalni matriki Q_k in Z_k določimo tako, da je A_{k+1} zgornja Hessenbergova in B_{k+1} zgornja trikotna, matrika $A_{k+1}B_{k+1}^{-1}$ pa se ujema z matriko, ki bi jo dobili, če bi izvedli en korak QR iteracije za matriko $A_k B_k^{-1}$.

Velja $A_{k+1}B_{k+1}^{-1} = Q_k(A_k B_k^{-1})Q_k^T$. Če je A_{k+1} zgornja Hessenbergova matrika, B_{k+1} zgornja trikotna matrika in se prvi stolpec Q_k ujema s prvim stolpcem ustrezne matrike pri QR iteraciji za matriko $A_k B_k^{-1}$, potem nam izrek o implicitnem Q zagotavlja, da je to ekvivalentno metodi QR na AB^{-1}.

Naj bo $C_k = A_k B_k^{-1}$. Za izračun Francisovega premika σ_1, σ_2 potrebujemo 2×2 podmatriko $R = C(n-1:n, n-1:n)$, ki jo lahko izračunamo s konstantnim številom operacij. Naj bo v_1 prvi stolpec matrike $(C_k - \sigma_1 I)(C_k - \sigma_2 I)$. Tudi to lahko izračunamo s konstantnim številom operacij. Sedaj poiščemo Householderjevo zrcaljenje P_0, ki vektor v_1 prezrcali v smer e_1. Če matriki A_k in B_k z leve pomnožimo s P_0, dobimo grbo, ki jo, podobno kot pri QR iteraciji za nesimetrično matriko, z naslednjimi ortogonalnimi transformacijami premikamo navzdol in na koncu spravimo iz matrik.

Poglejmo, kako grbo premaknemo v primeru, ko imamo matriki velikosti 5×5. Najprej dobimo

$$A_k^{(1)} = P_0 A_k = \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ + & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & \times & \times \end{bmatrix}, \quad B_k^{(1)} = P_0 B_k = \begin{bmatrix} \times & \times & \times & \times & \times \\ + & \times & \times & \times & \times \\ + & + & \times & \times & \times \\ 0 & 0 & 0 & \times & \times \\ 0 & 0 & 0 & 0 & \times \end{bmatrix}.$$

Z dvema zrcaljenjima z desne spravimo B nazaj v trikotno obliko. Najprej z Z_1 uničimo poddiagonalne elemente v tretji vrstici, nato pa z Z_2 še v drugi vrstici. Pri tem se v A pojavijo novi neničelni elementi v četrti vrstici. Dobimo

$$A_k^{(2)} = A_k^{(1)} Z_1 = \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ + & + & \times & \times & \times \\ 0 & 0 & 0 & \times & \times \end{bmatrix}, \quad B_k^{(2)} = B_k^{(1)} Z_1 = \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & \times & \times \\ 0 & 0 & 0 & 0 & \times \end{bmatrix},$$

$$A_k^{(3)} = A_k^{(2)} Z_2 = \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ 0 & 0 & 0 & \times & \times \end{bmatrix}, \quad B_k^{(3)} = B_k^{(2)} Z_2 = \begin{bmatrix} \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & \times & \times \\ 0 & 0 & 0 & 0 & \times \end{bmatrix}.$$

Z zrcaljenjem, ki deluje od druge do četrte vrstice, uničimo elementa v tretji in četrti vrstici

prvega stolpca matrike A. Tako se je grba premaknila za eno mesto navzdol.

$$A_k^{(4)} = P_1 A_k^{(3)} = \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & 0 & 0 & \times & \times \end{bmatrix}, \quad B_k^{(4)} = P_1 B_k^{(3)} = \begin{bmatrix} \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & + & \times & \times & \times \\ 0 & + & + & \times & \times \\ 0 & 0 & 0 & 0 & \times \end{bmatrix}.$$

Postopek ponavljamo, dokler se grba ne izloči iz matrik in nam na koncu ostane par zgornje Hessenbergove in zgornje trikotne matrike.

Ko izvajamo QZ algoritem, lahko izvedemo deflacijo in nadaljujemo z matriko manjše velikosti, če velja $a_{k+1,k} = 0$ oziroma $b_{kk} = 0$. Pri situaciji $a_{k+1,k} = 0$ problem preprosto razdelimo na dva manjša posplošena problema lastnih vrednosti velikosti $k \times k$ in $(n-k) \times (n-k)$.

V primeru $b_{kk} = 0$ z ustreznimi ortogonalnimi transformacijami problem prevedemo na podobnega, kjer velja $a_{n,n-1} = 0$ in $b_{nn} = 0$, potem pa lahko nadaljujemo samo z vodilnima podmatrikama velikosti $(n-1) \times (n-1)$. V tem primeru izločimo lastno vrednost ∞.

S QZ algoritmom na koncu izračunamo posplošeno Schurovo formo. Če nas zanimajo samo lastne vrednosti, potem ni potrebno shranjevati produktov ortogonalnih transformacij. Zadoščata nam končni trikotni matriki.

Naj bo σ izračunana lastna vrednost šopa (A, B), bodisi z metodo QZ ali s kakšno drugo metodo, za katero potrebujemo pripadajoči lastni vektor. Izračunamo ga lahko s s posplošitvijo inverzne iteracije. Metode, ki je zapisana v algoritmu 3.1, ne bomo podrobno izpeljevali, saj gre za poseben primer inverzne iteracije za nelinearni problem lastnih vrednosti iz algoritma 3.3. Hitro lahko preverimo, da je v primeru, ko je matrika B nesingularna, metoda ekvivalentna inverzni iteraciji za matriko $B^{-1}A$.

Algoritem 3.1 Inverzna iteracija za posplošeni problem lastnih vrednosti

izberi začetni vektor q_0
$k = 1, 2, \ldots$
    reši $(A - \sigma B)z_k = Bq_{k-1}$
    $q_k = z_k / \|z_k\|$

3.3 Definiten matrični šop

Denimo, da iščemo lastne vrednosti matričnega para (A, B), kjer sta obe matriki simetrični, matrika B pa je še pozitivno definitna. V tem primeru pravimo, da je matrični šop *definiten*[1].

Zaradi izgube simetrije takega problema ni primerno reševati preko matrike $C = AB^{-1}$. Bolje je, če za matriko B uporabimo razcepa Choleskega $B = VV^T$, kar vodi do

$$\begin{aligned} Ax &= \lambda VV^T x, \\ V^{-1}Ax &= \lambda V^T x, \\ V^{-1}AV^{-T}V^T x &= \lambda V^T x. \end{aligned}$$

[1] To ni potreben pogoj za definitnost, saj točna definicija definitnega šopa zahteva le, da sta matriki A in B simetrični in je $(x^TAx)^2 + (x^TBX)^2 > 0$ za vsak neničelni vektor x.

Dobimo simetričen lastni problem $Cy = \lambda y$, kjer je $C = V^{-1}AV^{-T}$ in $y = V^T x$. Od tod sledi, da ima definiten matrični šop same realne lastne vrednosti. V nadaljevanju lahko uporabimo algoritme za simetrične matrike, kar pa ne bi bilo možno, če bi delali direktno z matriko AB^{-1}.

Za definiten matrični šop (A, B) lahko za neničelni vektor x definiramo *posplošeni Rayleighov kvocient*

$$\rho(x, A, B) = \frac{x^T Ax}{x^T Bx}.$$

Podobno, kot za simetričen problem lastnih vrednosti, velja $\lambda_n \leq \rho(x, A, B) \leq \lambda_1$, kar sledi iz zveze

$$\rho(x, A, B) = \rho(B^{1/2}x, B^{-1/2}AB^{-1/2}).$$

Lema 3.7 *Posplošeni Rayleighov kvocient $\rho(x, A, B)$ vrne skalar λ, ki minimizira*

$$\|Ax - \lambda Bx\|_{B^{-1}},$$

kjer je $\|z\|_{B^{-1}}^2 = z^T B^{-1} z$.

Dokaz.

$$\|Ax - \lambda Bx\|_{B^{-1}}^2 = \lambda^2 x^T Bx - 2\lambda x^T Ax + x^T AB^{-1}Ax.$$

Z odvajanjem lahko ugotovimo, da je minimum tega izraza dosežen pri $\lambda = \rho(x, A, B)$. ∎

Za iskanje posameznih lastnih parov lahko uporabimo posplošeno Rayleighovo iteracijo. Dobimo jo na podoben način kot za simetrični problem lastnih vrednosti. Vzamemo inverzno iteracijo iz algoritma 3.1 in v vsakem koraku približek za lastno vrednost nadomestimo s posplošenim Rayleighovim kvocientom. Metoda ima kubično konvergenco v bližini enostavne lastne vrednosti.

Algoritem 3.2 Posplošena Rayleighova iteracija.

izberi $x_0 \neq 0$
$k = 0, 1, \ldots$
  $\rho_k = \frac{x_k^T Ax_k}{x_k^T Bx_k}$
  reši $(A - \rho_k B)y_{k+1} = Bx_k$
  $x_{k+1} = y_{k+1}/\|y_{k+1}\|$

3.4 Nelinearni problem lastnih vrednosti

Pri splošnem nelinearnem problemu lastnih vrednosti imamo dano matriko $T(\lambda)$ velikosti $n \times n$, katere elementi so dovoljkrat zvezno odvedljive funkcije parametra λ. Če obstajata tak skalar $\lambda \in \mathbb{C}$ in neničelni vektor $x \in \mathbb{C}^n$, da je

$$T(\lambda)x = 0,$$

potem je λ *lastna vrednost*, x pa *(desni) lastni vektor*. Podobno je neničelni vektor $y \in \mathbb{C}^n$ *levi lastni vektor*, če je $y^H T(\lambda) = 0$.

Za nelinearni problem lastnih vrednosti $T(\lambda)x = 0$ pravimo, da je *regularen*, če je $\det(T(\lambda)) \not\equiv 0$, sicer pa je *singularen*. V primeru, ko je problem regularen, so lastne vrednosti rešitve karakteristične enačbe $\det(T(\lambda)) = 0$.

Posebni primeri nelinearnega problema lastnih vrednosti so:

- Navadni linearni problem lastnih vrednosti. Če za matriko $A \in \mathbb{C}^{n\times n}$ definiramo
$$T(\lambda) = \lambda I - A,$$
potem je očitno $T(\lambda)x = 0$ natanko tedaj, ko je $Ax = \lambda x$.
- Posplošeni problem lastnih vrednosti (GEP). Če za matriki $A, B \in \mathbb{C}^{n\times n}$ definiramo
$$T(\lambda) = \lambda B - A,$$
potem je $T(\lambda)x = 0$ ekvivalentno $Ax = \lambda Bx$.
Pri GEP se, za razliko od navadnega problema lastnih vrednosti, lahko zgodi, da sta matriki A in B izbrani tako, da je $\det(T(\lambda)) \equiv 0$ in je problem singularen.
- Kvadratni problem lastnih vrednosti (QEP). Za matrike $M, C, K \in \mathbb{C}^{n\times n}$ definiramo
$$T(\lambda) = \lambda^2 M + \lambda C + K.$$
V primeru, ko je matrika M nesingularna, imamo $2n$ lastnih vrednosti, saj je $\det(T(\lambda))$ v tem primeru polinom stopnje $2n$ z vodilnim koeficientom $\det(M)$.
Problem je sicer nelinearen, a ga je možno linearizirati. Ena izmed možnosti je, da ga prevedemo na GEP oblike
$$\left(\lambda \begin{bmatrix} M & 0 \\ 0 & I \end{bmatrix} + \begin{bmatrix} C & K \\ -I & 0 \end{bmatrix}\right) \begin{bmatrix} \lambda x \\ x \end{bmatrix} = 0.$$
- Polinomski problem lastnih vrednosti (PEP) ima obliko
$$T(\lambda) = \lambda^m A_m + \lambda^{m-1} A_{m-1} + \cdots + \lambda A_1 + A_0$$
za dane matrike $A_0, \ldots, A_m \in \mathbb{C}^{n\times n}$. Podobno kot kvadratni problem lastnih vrednosti lahko tudi polinomski problem lastnih vrednosti prevedemo na GEP z matrikama velikosti $(mn) \times (mn)$. Če je A_m nesingularna matrika, potem ima PEP mn končnih lastnih vrednosti.
- Racionalni problem lastnih vrednosti (REP), npr.:
$$T(\lambda) = -K + \lambda M + \sum_{k=1}^{m} \frac{\lambda}{\sigma_k - \lambda} C_k.$$
Vsak REP lahko prevedemo na polinomski problem lastnih vrednosti, če izraz spravimo na skupni imenovalec. Torej bi tudi REP lahko linearizirali v GEP.
- Iracionalni problem lastnih vrednosti, npr.:
$$T(\lambda) = K - \lambda M + i \sum_{k=1}^{m} \sqrt{\lambda - \sigma_k^2} W_k.$$

Zgornji problem, kjer so $M, K, W_1, \ldots, W_m$ simetrične matrike, pri čemer je K nenegativno, M pa pozitivno definitna, $\sigma_1, \ldots, \sigma_m$ pa so nenegativni skalarji, nastopa pri razvoju linearnih pospeševalnikov [20].

Iracionalnega problema lastnih vrednosti ne moremo linearizirati in zanj lahko rečemo, da je pristno nelinearen.

- Pri študiju diferencialnih enačb s časovnimi zamiki oblike

$$\dot{x}(t) = A_0 x(t) + \sum_{k=1}^{m} A_k x(t - h_k),$$

kjer so $h_1, \ldots, h_m$ pozitivni premiki, $A_0, \ldots, A_m$ pa so realne matrike, se pojavijo nelinearni problemi lastnih vrednosti oblike

$$T(\lambda) = -\lambda I + A_0 + \sum_{k=1}^{m} A_j e^{-h_k \lambda}.$$

Poglejmo si dve numerični metodi, ki ju lahko uporabimo za splošen nelinearni problem lastnih vrednosti.

3.4.1 Newtonova metoda in inverzna iteracija

Če izberemo vektor $v \in \mathbb{C}^n$ in dodamo k enačbi $T(\lambda)x = 0$ še pogoj $v^H x = 1$, potem lahko zapišemo NEP v obliki nelinearnega sistema $n+1$ enačb za $n+1$ neznank:

$$F(x, \lambda) := \begin{bmatrix} T(\lambda)x \\ v^H x - 1 \end{bmatrix} = 0.$$

Za reševanje zgornjega sistema uporabimo Newtonovo metodo. Če je (x_k, λ_k) tekoči približek za lastni par, potem po Newtonovi metodi naslednji približek (x_{k+1}, λ_{k+1}) dobimo kot

$$\begin{bmatrix} x_{k+1} \\ \lambda_{k+1} \end{bmatrix} = \begin{bmatrix} x_k \\ \lambda_k \end{bmatrix} - JF(x_k, \lambda_k)^{-1} F(x_k, \lambda_k),$$

kjer je Jacobijeva matrika enaka

$$JF(x_k, \lambda_k) = \begin{bmatrix} T(\lambda_k) & T'(\lambda_k)x_k \\ v^H & 0 \end{bmatrix}.$$

Tako pridemo do sistema

$$\begin{bmatrix} T(\lambda_k) & T'(\lambda_k)x_k \\ v^H & 0 \end{bmatrix} \begin{bmatrix} x_{k+1} - x_k \\ \lambda_{k+1} - \lambda_k \end{bmatrix} = - \begin{bmatrix} T(\lambda_k)x_k \\ v^H x_k - 1 \end{bmatrix},$$

oziroma

$$\begin{aligned} T(\lambda_k)x_{k+1} &= -(\lambda_{k+1} - \lambda_k) T'(\lambda_k) x_k, \\ v^H x_{k+1} &= 1. \end{aligned} \tag{3.1}$$

Iz zveze (3.1) sledi

$$x_{k+1} = -(\lambda_{k+1} - \lambda_k) T(\lambda_k)^{-1} T'(\lambda_k) x_k. \tag{3.2}$$

To pomeni, da iz zgornje enačbe lahko določimo smer vektorja x_{k+1}. Več kot smer tudi ne potrebujemo, saj lahko potem x_{k+1} enostavno pomnožimo s pravim skalarjem, da bo $v^H x_{k+1} = 1$. Za naslednji korak potrebujemo še nov približek za lastno vrednost λ_{k+1}. Če definiramo

$$u_{k+1} := T(\lambda_k)^{-1} T'(\lambda_k) x_k$$

in enačbo (3.2) skalarno pomnožimo z v, potem dobimo

$$v^H x_{k+1} = -(\lambda_{k+1} - \lambda_k) v^H u_{k+1}.$$

Ob predpostavki, da je prejšnji približek za lastni vektor bil normiran, torej $v^H x_k = 1$, dobimo

$$\lambda_{k+1} = \lambda_k - \frac{v^H x_k}{v^H u_{k+1}}.$$

Tako pridemo do prvega algoritma. Čeprav gre za Newtonovo metodo, metodo zaradi podobnostjo s podobnim algoritmom za navadni problem lastnih vrednosti imenujemo *inverzna iteracija*.

Algoritem 3.3 Inverzna iteracija za nelinarni problem lastnih vrednosti.

izberi vektor v in tak začetni približek (λ_0, x_0) za lastni par, da je $v^H x_0 = 1$
$k = 0, 1, \ldots$

a) reši linearni sistem $T(\lambda_k) u_{k+1} = T'(\lambda_k) x_k$

b) $\lambda_{k+1} = \lambda_k - \dfrac{v^H x_k}{v^H u_{k+1}}$

c) $x_{k+1} = \dfrac{1}{v^H u_{k+1}} u_{k+1}$

3.4.2 Zaporedne linearne aproksimacije

Naj bo preslikava $T : \mathbb{C} \to \mathbb{C}^{n \times n}$ spet dvakrat zvezno odvedljiva. Denimo, da že imamo približek λ_k za lastno vrednost. Iščemo popravek $\Delta\lambda_k$ in neničelni vektor x, da bo

$$T(\lambda_k - \Delta\lambda_k) x = 0.$$

Če zgornjo enačbo razvijemo v vrsto, dobimo

$$\left(T(\lambda_k) - \Delta\lambda_k T'(\lambda_k) + \mathcal{O}(|\Delta\lambda_k|^2)\right) x = 0.$$

Če zanemarimo kvadratne člene, dobimo, da je popravek $\Delta\lambda_k$ lastna vrednost posplošenega problema lastnih vrednosti

$$T(\lambda_k) x = \Delta\lambda_k T'(\lambda_k) x. \tag{3.3}$$

Pri metodi zaporednih linearnih aproksimacij za $\Delta\lambda_k$ vzamemo po absolutni vrednosti najmanjšo lastno vrednost (3.3). V bližini lastne vrednosti je metoda podobna inverzni iteraciji, saj dobimo

$$\frac{1}{\Delta\lambda_k} x = T(\lambda_k)^{-1} T'(\lambda_k) x.$$

Dela v enem koraku je več kot pri prejšnji metodi, saj moramo v vsakem koraku rešiti GEP, je pa konvergenca zato ponavadi hitrejša.

Algoritem 3.4 Zaporedne linearne aproksimacije.

izberi začetni približek λ_0 za lastno vrednost
$k = 0, 1, \ldots$
za $\Delta\lambda_k$ vzemi po absolutni vrednosti najmanjšo lastno vrednost GEP $T(\lambda_k)x = \theta T'(\lambda_k)x$
$\lambda_{k+1} = \lambda_k - \Delta\lambda_k$

3.5 Polinomski (kvadratni) problem lastnih vrednosti

Pri polinomskem problemu lastnih vrednosti (PEP) je dan matrični polinom $P(\lambda) = A_0 + \lambda A_1 + \cdots + \lambda^m A_m$, kjer so $A_0, \ldots, A_m$ matrike velikosti $n \times n$. Iščemo tak skalar λ_0 in neničelni vektor x_0, da je $P(\lambda_0)x_0 = 0$.

PEP P je regularen, če njegov karakteristični polinom, definiran z $g(\lambda) := \det(P(\lambda))$, ni identično enak 0. Ničle polinoma g so končne lastne vrednosti PEP P. Če jih je manj kot mn, jih do mn dopolnimo z neskončnimi lastnimi vrednostmi. Neskončne lastne vrednosti ustrezajo ničelnim lastnim vrednostim *vzvratnega polinoma* $P_R(\lambda) := \lambda^m P(1/\lambda) = \lambda^m A_0 + \lambda^{m-1} A_1 + \cdots + A_m$, pojavijo pa se le, če je matrika A_m nesingularna.

Regularni PEP P ima tako mn (končnih ali neskončnih) lastnih vrednosti.

Zgled 3.2 *Naslednji primer kvadratnega problema lastnih vrednosti je povzet iz [29].*

Če vzamemo $Q(\lambda) = \lambda^2 M + \lambda C + K$, *kjer so*

$$M = \begin{bmatrix} 0 & 6 & 0 \\ 0 & 6 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad C = \begin{bmatrix} 1 & -6 & 0 \\ 2 & -7 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad K = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

potem je $\det Q(\lambda) = -6\lambda^5 + 11\lambda^4 - 12\lambda^3 + 12\lambda^2 - 6\lambda + 1$ *in problem je regularen. Lastni pari so*

k	1	2	3	4	5	6
λ_k	$1/3$	$1/2$	1	i	$-i$	∞
x_k	$\begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$	$\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$

Pet lastnih vrednosti je končnih, ena pa neskončna. Opazimo, da imata različni lastni vrednosti lahko isti lastni vektor, kar pri navadnem in posplošenem lastnem problemu seveda ni možno. □

Lahko se zgodi, da si največ m različnih lastnih vrednosti deli isti lastni vektor. Namreč, če je x lastni vektor, potem je vsaj ena izmed rešitev enačbe $x^H P(\lambda)x = 0$, ki ima največ m rešitev, enaka lastni vrednosti.

Dan je QEP $Q(\lambda) = \lambda^2 M + \lambda C + K$. Standardni postopek za numerično reševanje QEP je *linearizacija*, kjer problem prevedemo na navadni ali posplošeni problem lastnih vrednosti reda $2n$. Možnih linearizacij je več, primera sta:

a) prevedba na navadni nesimetrični problem lastnih vrednosti (če je M nesingularna)

$$Au = \lambda u,$$

kjer sta

$$A = \begin{bmatrix} 0 & I \\ -M^{-1}K & -M^{-1}C \end{bmatrix}, \quad u = \begin{bmatrix} x \\ \lambda x \end{bmatrix},$$

b) prevedba na posplošeni simetrični problem lastnih vrednosti

$$Az = \lambda Bz,$$

kjer so

$$A = \begin{bmatrix} C & K \\ K & 0 \end{bmatrix}, \quad z = \begin{bmatrix} x \\ \lambda x \end{bmatrix}, \quad B = \begin{bmatrix} -M & 0 \\ 0 & K \end{bmatrix}.$$

Podobno lahko splošni polinomski problem lastnih vrednosti $P(\lambda) = \lambda^m A_m + \cdots + \lambda A_1 + A_0$ linearizirамo kot GEP z matrikami velikosti $mn \times mn$. Primer je t.i. *prva spremljevalna forma*, ki ima obliko

$$C_1(\lambda) = \lambda \begin{bmatrix} I & & & \\ & I & & \\ & & \ddots & \\ & & & A_m \end{bmatrix} + \begin{bmatrix} 0 & -I & & \\ & \ddots & \ddots & \\ & & 0 & -I \\ A_0 & A_1 & \cdots & A_{m-1} \end{bmatrix}.$$

V tem primeru je (λ, x) lastni par PEP natanko tedaj, ko je $\left(\lambda, \begin{bmatrix} x \\ \lambda x \\ \vdots \\ \lambda^{m-1}x \end{bmatrix}\right)$ lastni par GEP.

3.5.1 Hermitski kvadratni problem lastnih vrednosti

V tem podrazdelku bomo obravnavali poseben razred kvadratnih problemov lastnih vrednosti, ki se v praksi velikokrat pojavi. Matrike M, K, C so hermitske, matrika M pa je še pozitivno definitna. Za poljuben neničelen vektor $x \in \mathbb{C}^n$ definiramo

$$m(x) = x^H M x, \quad k(x) = x^H K x, \quad c(x) = x^H C x.$$

Definicija 3.8 *Dan je kvadratni problem lastnih vrednosti $Q(\lambda) = \lambda^2 M + \lambda C + K$, kjer so matrike M, K, C hermitske, matrika M pa je pozitivno definitna. Problem Q je*

- hiperboličen, *če za vsak $x \neq 0$ velja $c(x)^2 > 4m(x)k(x)$,*
- eliptičen, *če za vsak $x \neq 0$ velja $c(x)^2 < 4m(x)k(x)$,*
- nadkritično dušen, *če je hiperboličen in je matrika C pozitivno definitna, matrika K pa nenegativno definitna.*

Če je x lastni vektor, potem je vsaj ena izmed rešitev kvadratne enačbe $x^H Q(\lambda) x = 0$ enaka lastni vrednosti, ki pripada x. Od tod sledi, da so vse lastne vrednosti hiperboličnega QEP realne, medtem ko eliptičen QEP nima nobene realne lastne vrednosti. Podobno vidimo, da so vse lastne vrednosti nadkritično dušenega QEP negativne.

Dan je hiperboličen QEP $Q(\lambda) = \lambda^2 M + \lambda C + K$. Vse lastne vrednosti so realne in zanje velja, da jih lahko uredimo tako, da velja

$$\lambda_{2n} \leq \cdots \leq \lambda_{n+1} < \lambda_n \leq \cdots \leq \lambda_1,$$

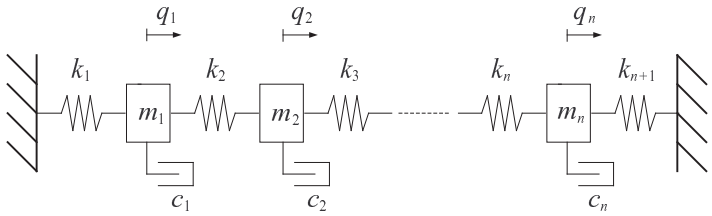
kjer je med λ_{n+1} in λ_n strog razmik.

Definiramo lahko t.i. *primarni in sekundarni Rayleighov funkcional*

$$p_1(x) = \frac{-c(x) + d(x)}{2m(x)}, \qquad p_2(x) = \frac{-c(x) - d(x)}{2m(x)},$$

kjer je $d(x) = \sqrt{c(x)^2 - 4m(x)k(x)}$. Za vsak $x \neq 0$ velja $p_2(x) < p_1(x)$. Če je x lastni vektor, potem se izkaže, da se natanko eden izmed Rayleighov funkcionalov $p_1(x)$ in $p_2(x)$ ujema z lastno vrednostjo. Na podlagi tega lastne vrednosti razdelimo na primarne $\lambda_1, \ldots, \lambda_n$ in sekundarne $\lambda_{n+1}, \ldots, \lambda_{2n}$. Lastni vektorji, ki pripadajo primarnim lastnim vrednostim, so linearno neodvisni in razpenjajo cel prostor, enako pa velja za lastne vektorje, ki pripadajo sekundarnim lastnim vrednostim.

Zgled 3.3 *Za zgled uporabe kvadratnega problema lastnh vrednosti vzemimo nihanje dušenega sistema mas in vzmeti.*



Če predpostavimo, da je dušenje linearno proporcionalno hitrosti in definiramo $q_0 = q_{n+1} = 0$, potem iz Newtonovega zakona dobimo enačbe

$$m_i \ddot{q}_i(t) = -k_i \left(q_i(t) - q_{i-1}(t)\right) - k_{i+1} \left(q_i(t) - q_{i+1}(t)\right) - c_i \dot{q}_i(t)$$

za $i = 1, \ldots, n$, ki tvorijo sistem diferencialnih enačb 2. reda s konstantnimi koeficienti

$$M\ddot{q}(t) + C\dot{q}(t) + Kq(t) = 0,$$

kjer je

$$M = \begin{bmatrix} m_1 & & \\ & \ddots & \\ & & m_n \end{bmatrix}, \quad C = \begin{bmatrix} c_1 & & \\ & \ddots & \\ & & c_n \end{bmatrix},$$

$$K = \begin{bmatrix} k_1 + k_2 & -k_2 & & \\ -k_2 & \ddots & \ddots & \\ & \ddots & \ddots & -k_n \\ & & -k_n & k_n + k_{n+1} \end{bmatrix},$$

pri čemer je M masna matrika, C matrika dušenja, K pa togostna matrika.

V primeru, ko so vse lastne vrednosti enostavne, ima splošna rešitev homogene diferencialne enačbe

$$M\ddot{q}(t) + C\dot{q}(t) + Kq(t) = 0,$$

obliko

$$q(t) = \sum_{k=1}^{2n} \alpha_k e^{\lambda_k t} x_k,$$

kjer so $\alpha_1, \ldots, \alpha_{2n}$ poljubne konstante, $\lambda_1, \ldots, \lambda_n$ so lastne vrednosti, $x_1, \ldots, x_{2n}$ pa lastni vektorji kvadratnega problema lastnih vrednosti

$$\lambda^2 Mx + \lambda Cx + Kx = 0.$$

Konstante $\alpha_1, \ldots, \alpha_{2n}$ določimo iz začetnih odmikov $q(0)$ in hitrosti $\dot{q}(0)$.

Sistem je stabilen, če velja $\mathrm{Re}(\lambda_k) < 0$ za vse k oziroma šibko stabilen, če velja $\mathrm{Re}(\lambda_k) \leq 0$ za vse k, če pa je $\mathrm{Re}(\lambda_k) = 0$ je λ_k enostavna lastna vrednost.

V primeru vsiljenega nihanja imamo enačbo

$$M\ddot{q}(t) + C\dot{q}(t) + Kq(t) = f(t).$$

Če je desna stran oblike $f(t) = e^{i\omega_0 t} f_0$, kar ustreza vsiljenemu nihanju, potem je partikularna rešitev

$$q_p(t) = e^{i\omega_0 t} \sum_{k=1}^{2n} \frac{y_k^H f_0}{i\omega_0 - \lambda_k} x_k,$$

kjer so $y_1, \ldots, y_{2n}$ levi lastni vektorji. Če se $i\omega_0$ približa eni izmed lastnih vrednosti λ_k, se lahko pojavi resonanca. *Odvisno od problema je to lahko zaželjeno (npr. v električnih krogih) ali pa zelo moteče (npr. vpliv potresa na zgradbo).* □

3.6 Računanje singularnega razcepa

Za matriko $A \in \mathbb{R}^{m \times n}$ bi radi izračunali singularni razcep $A = U\Sigma V^T$. Ker velja

$$A^T A = V\Sigma^T \Sigma V^T,$$

dobimo preprosto idejo:

1) izračunamo $A^T A$,
2) rešimo lastni problem $A^T A = V\Lambda V^T$, odtod dobimo V,
3) za Σ vzamemo $m \times n$ matriko, ki ima v zgornjem bloku kvadratni koren Λ,
4) rešimo sistem $U\Sigma = AV$ za matriko U.

Kot kaže naslednji zgled, računanje na zgornji način ni numerično stabilno in lahko pride do velikih relativnih napak, če je razmerje med največjo in najmanjšo singularno vrednostjo zelo veliko, to pa je ekvivalentno temu, da je matrika zelo občutljiva.

Zgled 3.4 *Če v dvojni natančnosti vzamemo* $A = \begin{bmatrix} 1 & 1 \\ 0 & 10^{-8} \end{bmatrix}$, *potem je* $A^TA = \begin{bmatrix} 1 & 1 \\ 1 & 1+10^{-16} \end{bmatrix}$, *kar se v plavajoči vejici zaokroži v* $\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$. *Numerično izračunani singularni vrednosti sta tako* $\sqrt{2}$ *in* 0, *točni pa* $\sqrt{2} - 2.22 \cdot 10^{-16}$ *in* $7.07 \cdot 10^{-9}$. *Prva singularna vrednost je izračunana z visoko relativno natančnostjo, pri drugi singularni vrednosti pa pride do velike relativne napake.* □

Eksplicitno računanje matrike A^TA tako ni najboljša ideja in se mu poskušamo izogniti. Pri računanju singularnega razcepa (razen pri Jacobijevi metodi) matriko najprej reduciramo na bidiagonalno obliko. Postopek je:

1) poišči ortogonalni matriki U_1 in V_1, da bo $A = U_1BV_1^T$ in B bidiagonalna matrika,
2) izračunaj singularni razcep za B: $B = U_2\Sigma V_2^T$,
3) singularni razcep za A je $A = (U_1U_2)\Sigma(V_1V_2)^T$.

Redukcijo na bidiagonalno obliko naredimo z množenji s Householderjevimi zrcaljenji izmenično z leve in z desne. V primeru matrike velikosti 6×4 dobimo:

$$A = \begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{bmatrix}, \quad P_1A = \begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \end{bmatrix},$$

$$P_1AP_1' = \begin{bmatrix} \times & \times & 0 & 0 \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \end{bmatrix}, \quad P_2P_1AP_1' = \begin{bmatrix} \times & \times & 0 & 0 \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & \times & \times \end{bmatrix},$$

$$P_2P_1AP_1'P_2' = \begin{bmatrix} \times & \times & 0 & 0 \\ 0 & \times & \times & 0 \\ 0 & 0 & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & \times & \times \end{bmatrix}, \quad P_4P_3P_2P_1AP_1'P_2' = \begin{bmatrix} \times & \times & 0 & 0 \\ 0 & \times & \times & 0 \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} = B.$$

Zgornji algoritem za splošno matriko velikosti $m \times n$ potrebuje $8mn^2 - 8n^3/3$ operacij. Če je m dosti večji kot n, potem se splača najprej izračunati QR razcep matrike A, kjer dobimo $m \times m$ ortogonalno matriko Q in $n \times n$ zgornjo trikotno matriko R_1, da je

$$Q^TA = \begin{bmatrix} R_1 \\ 0 \end{bmatrix}.$$

V nadaljevanju potem bidiagonaliziramo matriko R_1, da dobimo

$$U_R^TR_1V = B_1,$$

kjer je B_1 bidiagonalna $n \times n$ matrika. Na koncu je

$$\begin{bmatrix} U_R^T & 0 \\ 0 & I \end{bmatrix} Q^T A V = \begin{bmatrix} B_1 \\ 0 \end{bmatrix} = B$$

iskana bidiagonalizacija matrike A. Za to redukcijo potrebujemo $4mn^2 + 4n^3$ operacij, torej je cenejša, če je $m \geq 5n/3$.

Lema 3.9 *Naj bo B realna matrika velikosti $m \times n$, $m \geq n$, s singularnimi vrednostmi in vektorji $Bv_i = \sigma_i u_i$, $i = 1, \ldots, n$. Lastne vrednosti matrike*

$$C = \begin{bmatrix} 0 & B^T \\ B & 0 \end{bmatrix}$$

so $\pm\sigma_i$ z lastnimi vektorji $\frac{1}{\sqrt{2}} \begin{bmatrix} v_i \\ \pm u_i \end{bmatrix}$ za $i = 1, \ldots, n$ in dodatnih $m - n$ lastnih vrednosti 0 z lastnimi vektorji $\begin{bmatrix} 0 \\ u_j \end{bmatrix}$ za $j = n+1, \ldots, m$,

Dokaz. Vemo, da za $i = 1, \ldots, n$ velja $Bv_i = \sigma u_i$ in $B^T u_i = \sigma_i v_i$. Od tod sledi

$$C \begin{bmatrix} v_i \\ \pm u_i \end{bmatrix} = \pm\sigma_i \begin{bmatrix} v_i \\ \pm u_i \end{bmatrix}.$$

Podobno za $j = n+1, \ldots, m$ velja $B^T u_j = 0$, torej

$$C \begin{bmatrix} 0 \\ \pm u_j \end{bmatrix} = 0.$$

■

Hitro se lahko prepričamo, da za izračun singularnega razcepa bidiagonalne matrike B potrebujemo le njenih prvih n vrstic. Torej lahko pred nadaljevanjem predpostavimo, da je B bidiagonalna kvadratna matrika velikosti $n \times n$.

Tako lahko računamo singularni razcep brez računanja $B^T B$ a z dvakrat večjo matriko. Tudi tu pa imamo lahko še vedno težave z majhnimi singularnimi vrednostmi.

Če je B bidiagonalna matrika in vzamemo permutacijsko matriko

$$P = [\, e_1 \quad e_{n+1} \quad e_2 \quad e_{n+2} \quad \cdots \quad e_n \quad e_{2n} \,],$$

ki predstavlja t.i. perfektno mešanje[2], potem ima $P^T C P$ obliko

$$\begin{bmatrix} 0 & a_1 & & & & & \\ a_1 & 0 & b_1 & & & & \\ & b_1 & 0 & a_2 & & & \\ & & a_2 & 0 & \ddots & & \\ & & & \ddots & \ddots & b_{n-1} & \\ & & & & b_{n-1} & 0 & a_n \\ & & & & & a_n & 0 \end{bmatrix}. \tag{3.4}$$

[2]Ta permutacija ustreza mešanju kart, ko jih razdelimo na dva enaka kupčka in potem po vrsti z vsakega kupčka jemljemo po eno karto. Matematiki so dokazali (glej [30]), da je potrebno, če hočemo, da bodo karte res dobro premešane, postopek ponoviti vsaj šestkrat!

Sedaj lahko uporabimo poljubno metodo za simetrično tridiagonalno matriko. Imamo sicer dvakrat večjo matriko, a ker vemo, da lastne vrednosti nastopajo v plus-minus parih, se da nekaj dela prihraniti.

Posledica formule (3.4) je, da v primeru, ko so vsi $a_i \neq 0$ in $b_i \neq 0$, matrika B nima večkratnih singularnih vrednosti. V primeru, ko je $a_i = 0$ ali $b_i = 0$, lahko naredimo deflacijo, zato lahko predpostavimo, da so vsi diagonalni in bidiagonalni elementi neničelni.

Iz leme 3.9 sledi naslednja ocena za motnjo singularnih vrednosti.

Posledica 3.10 *Za singularne vrednosti $A + E$ velja*

$$|\sigma_k(A+E) - \sigma_k(A)| \leq \|E\|_2$$

za $k = 1, \ldots, n$.

Sedaj lahko pokažemo, zakaj računanje singularnih vrednosti preko eksplicitno izračunanega produkta A^TA ni obratno stabilno. Od obratno stabilnega algoritma za računanje singularnih vrednosti matrike A pričakujemo, da bo za izračunane približke veljalo $\widetilde{\sigma}_k = \sigma_k(A+E)$, kjer je $\|E\| = \mathcal{O}(u\|A\|)$, od tod pa po posledici 3.10 sledi $|\widetilde{\sigma}_k - \sigma_k| = \mathcal{O}(u\|A\|)$.

Če računamo preko A^TA z obratno stabilnim algoritmom za računanje lastnih vrednosti simetrične matrike, potem velja

$$|\widetilde{\sigma}_k^2 - \sigma_k^2| = \mathcal{O}(u\|A^TA\|) = \mathcal{O}(u\|A\|^2).$$

Po korenjenju dobimo

$$|\widetilde{\sigma}_k - \sigma_k| = \mathcal{O}\left(u\|A\|^2 \frac{1}{\sigma_k}\right).$$

Za večje singularne vrednosti je to v redu, za manjše pa ne. Tako lahko pri σ_n pričakujemo napako velikosti $\mathcal{O}(u\|A\|\kappa_2(A))$.

V nadaljevanju si bomo pogledali nekaj specialnih metod za računanje singularnega razcepa bidiagonalne matrike B.

3.7 QR iteracija za računanje singularnega razcepa

Naj bo

$$B = \begin{bmatrix} a_1 & b_1 & & \\ & \ddots & \ddots & \\ & & a_{n-1} & b_{n-1} \\ & & & a_n \end{bmatrix}.$$

Vemo že, da bi lahko singularne vrednosti izračunali kot kvadratne korene lastnih vrednosti matrike B^TB, ki je simetrična in tridiagonalna. Za računanje lastnih vrednosti simetrične tridiagonalne matrike pa lahko uporabimo QR iteracijo z Wilkinsonovimi premiki.

Denimo, da je $C = B^TB$. Po enem koraku QR iteracije, kjer zaradi nenegativne definitnosti matrike C izberemo premik σ^2, dobimo simetrično tridiagonalno matriko $\widetilde{C} = Q^TCQ$, kjer je Q

ortogonalna matrika. Prvi stolpec matrike Q, ki je enak normiranemu prvemu stolpcu matrike $C - \sigma^2 I$, po izreku 1.23 do predznaka natančno določa preostale stolpce.

Pri implicitni QR iteraciji matrike C ne izračunamo eksplicitno, temveč poiščemo taki ortogonalni matriki P in Q, da je matrika $\widetilde{B} = PBQ$ bidiagonalna in je prvi stolpec matrike Q enak normiranemu prvemu stolpcu matrike $C - \sigma^2 I$. Ker je potem matrika $\widetilde{B}^T\widetilde{B} = Q^T B^T B Q$ tridiagonalna, se po izreku 1.23 ujema z matriko $\widetilde{C}$. Tako v bistvu naredimo en korak QR iteracije za matriko $C = B^T B$, v resnici pa posodobimo samo matriko B.

Za Wilkinsonov premik potrebujemo lastne vrednosti desne spodnje 2×2 podmatrike $B^T B$

$$\begin{bmatrix} a_{n-1}^2 + b_{n-2}^2 & a_{n-1}b_{n-1} \\ a_{n-1}b_{n-1} & a_n^2 + b_{n-1}^2 \end{bmatrix},$$

za premik σ^2 pa vzamemo tisto lastno vrednost, ki je bližja $a_n^2 + b_{n-1}^2$. Nato izračunamo Givensovo rotacijo $R = \begin{bmatrix} c & -s \\ s & c \end{bmatrix}$, da bo (glede na prvi stolpec $B^T B$)

$$R \begin{bmatrix} a_1^2 - \sigma^2 \\ a_1 b_2 \end{bmatrix} = \begin{bmatrix} \times \\ 0 \end{bmatrix}$$

in vzamemo $R'_{12} = \begin{bmatrix} R & 0 \\ 0 & I \end{bmatrix}$. Ko izračunamo BR'_{12} (to je ekvivalentno $R'^T_{12} B^T B R'_{12}$ pri QR za $B^T B$), dobimo (v primeru 5×5)

$$BR'_{12} = \begin{bmatrix} \times & \times & & & \\ + & \times & \times & & \\ & & \times & \times & \\ & & & \times & \times \\ & & & & \times \end{bmatrix}.$$

Dodatni neničelni element $+$ uničimo tako, da ga z množenjem z ustreznimi rotacijami poganjamo navzdol z leve na desno. Tako dobimo

$$R_{12}BR'_{12} = \begin{bmatrix} \times & \times & + & & \\ 0 & \times & \times & & \\ & & \times & \times & \\ & & & \times & \times \\ & & & & \times \end{bmatrix}, \quad R_{12}BR'_{12}R'_{23} = \begin{bmatrix} \times & \times & 0 & & \\ & \times & \times & & \\ & + & \times & \times & \\ & & & \times & \times \\ & & & & \times \end{bmatrix}.$$

Na koncu je $(R_{45}R_{34}R_{23}R_{12})B(R'_{12}R'_{23}R'_{34}R'_{45})$ bidiagonalna matrika za naslednji korak QR metode.

Za en korak enostranske QR iteracije porabimo $30n + \mathcal{O}(1)$ osnovnih operacij in $2n$ kvadratnih korenov. Kvadratni koren je zahtevnejši od osnovnih operacij in stane tudi 10 ali več osnovnih operacij.

Kriterija, ki ju uporabljamo za deflacijo, sta $|b_i| \leq \epsilon(|a_i| + |a_{i+1}|)$ in $|a_i| \leq \epsilon\|B\|$.

3.8 dqds metoda

Če je T simetrična pozitivno definitna matrika, lahko delamo naslednji postopek, ki ga imenujemo *LR iteracija*:

Algoritem 3.5 LR iteracija brez premikov za simetrično pozitivno definitno matriko.

$T_0 = T$
$i = 0, 1, 2, \ldots$:
  $T_i = V_i^T V_i$ (razcep Choleskega)
  $T_{i+1} = V_i V_i^T$

Izkaže se, da T_i konvergira proti diagonalni matriki lastnih vrednosti matrike T. Za začetek je očitno, da sta matriki T_i in T_{i+1} podobni, saj velja

$$T_{i+1} = V_i V_i^T = V_i V_i^T V_i V_i^{-1} = V_i T_i V_i^{-1}.$$

Torej, če T_i konvergira proti diagonalni matriki, ima ta na diagonali lastne vrednosti T. Če je T tridiagonalna matrika, se da hitro preveriti, da se tridiagonalnost med LR algoritmom ohranja.

Ker V_i ni ortogonalna matrika, je LR iteracija na prvi pogled potencialno nestabilna, a je skrb odveč, saj nam stabilnost in konvergenco zagotavlja naslednji izrek, ki povezuje QR iteracijo in LR iteracijo.

Lema 3.11 *Dva koraka LR iteracije ustrezata enemu koraku QR iteracije brez premikov.*

Dokaz. En korak QR algoritma je:

$$T_0 = QR, \quad T' = RQ.$$

Dva koraka LR iteracije sta

$$T_0 = V_0^T V_0, \quad T_1 = V_0 V_0^T = V_1^T V_1, \quad T_2 = V_1 V_1^T.$$

Pokazati moramo, da je $T' = T_2$.

Ker je T_0 simetrična je

$$T_0^2 = T_0^T T_0 = (QR)^T QR = R^T R.$$

To je razcep Choleskega za s.p.d. matriko T_0^2. Po drugi strani je

$$T_0^2 = T_0 T_0 = V_0^T V_0 V_0^T V_0 = V_0^T V_1^T V_1 V_0 = (V_1 V_0)^T (V_1 V_0).$$

Ker je to tudi razcep Choleskega, ta pa je enoličen, velja $R = V_1 V_0$. To pa pomeni

$$\begin{aligned} T' &= RQ = RQRR^{-1} = RT_0R^{-1} = (V_1V_0)(V_0^TV_0)V_0^{-1}V_1^{-1} \\ &= V_1V_0V_0^TV_1^{-1} = V_1V_1^TV_1V_1^{-1} = V_1V_1^T = T_2. \end{aligned}$$

■

Namesto enega koraka QR iteracije lahko tako naredimo dva koraka LR iteracije, kar je ekonomičneje. Konvergenco pospešimo s premiki, pri čemer pa moramo sedaj paziti, da bo premik manjši od najmanjše lastne vrednosti, saj sicer izgubimo pozitivno definitnost. Metoda je predstavljena v algoritmu 3.6.

Tudi matriki T_{i+1} in T_i iz algoritma 3.6 sta podobni, saj velja

$$T_{i+1} = V_i V_i^T + \tau_i^2 I = (V_i V_i^T + \tau_i^2 I) V_i V_I^{-1} = V_i T_i V_i^{-1}.$$

Algoritem 3.6 LR iteracija s premiki za simetrično pozitivno definitno matriko.

$T_0 = T$
$i = 0, 1, 2, \ldots$:
  izberi premik $\tau_i^2 > 0$, tako da je $\tau_i^2 \leq \lambda_n(T)$
  $T_i - \tau_i^2 I = V_i^T V_i$ (razcep Choleskega)
  $T_{i+1} = V_i V_i^T + \tau_i^2 I$

Pri *dqds metodi* delamo LR iteracijo s premikom za $B^T B$, pri čemer pa te matrike ne računamo eksplicitno. Z ekonomičnim algoritmom iz bidiagonalne matrike B_i dobimo bidiagonalno B_{i+1}, pri tem pa velja, da če bi naredili en korak LR iteracije s premikom za $B_i^T B_i$, bi bil rezultat matrika $B_{i+1}^T B_{i+1}$.

Označimo

$$B_i = \begin{bmatrix} a_1 & b_1 & & \\ & \ddots & \ddots & \\ & & a_{n-1} & b_{n-1} \\ & & & a_n \end{bmatrix} \quad \text{in} \quad B_{i+1} = \begin{bmatrix} \widetilde{a}_1 & \widetilde{b}_1 & & \\ & \ddots & \ddots & \\ & & \widetilde{a}_{n-1} & \widetilde{b}_{n-1} \\ & & & \widetilde{a}_n \end{bmatrix}.$$

S primerjavo elementov $T_i = B_i^T B_i + \tau_i^2 I$ in $T_{i+1} = B_{i+1}^T B_{i+1} + \tau_{i+1}^2 I = B_i B_i^T + \tau_i^2 I$ pridemo do osnovnega *dqds* algoritma, kjer je $\delta = \tau_{i+1}^2 - \tau_i^2 \geq 0$ in privzamemo $b_0 = \widetilde{b}_0 = 0$.

Algoritem 3.7 Osnovna dqds metoda.

$k = 1, \ldots, n-1$
  $\widetilde{a}_k^2 = a_k^2 + b_k^2 - \widetilde{b}_{k-1}^2 - \delta$
  $\widetilde{b}_k^2 = a_{k+1}^2 b_k^2 / \widetilde{a}_k^2$
$\widetilde{a}_n^2 = a_n^2 - \widetilde{b}_{n-1}^2 - \delta$

Ves čas lahko računamo s kvadrati in korenimo le na koncu. Če označimo $c_k = a_k^2$ in $d_k = b_k^2$, dobimo iz algoritma 3.7 novo različico, imenovano qds algoritem, ki porabi manj operacij.

Algoritem 3.8 qds metoda.

$k = 1, \ldots, n-1$
  $\widetilde{c}_k = c_k + d_k - \widetilde{d}_{k-1} - \delta$
  $\widetilde{d}_k = c_{k+1} d_k / \widetilde{c}_k$
$\widetilde{c}_n = c_n - \widetilde{d}_{n-1} - \delta$

Algoritem 3.8 porabi le $5n + \mathcal{O}(1)$ operacij za en korak LR algoritma. Ta algoritm lahko še izboljšamo, da bo stabilnejši (ob enakem številu operacij) in tako končno pridemo do *dqds* algoritma.

Če definiramo $g_k = c_k - \widetilde{d}_{k-1} - \delta$, potem velja

$$g_k = c_k - \frac{c_k d_{k-1}}{\widetilde{c}_{k-1}} - \delta = c_k \frac{\widetilde{c}_{k-1} - d_{k-1}}{\widetilde{c}_{k-1}} - \delta = c_k \frac{c_{k-1} - \widetilde{d}_{k-2} - \delta}{\widetilde{c}_{k-1}} - \delta = \frac{c_k}{\widetilde{c}_{k-1}} g_{k-1} - \delta.$$

Končni *dqds* algoritem je

Algoritem 3.9 dqds metoda.

$g = c_1 - \delta$
$k = 1, \ldots, n-1$
  $\widetilde{c}_k = g + d_k$
  $t = c_{k+1}/\widetilde{c}_k$
  $\widetilde{d}_k = d_k \cdot t$
  $g = g \cdot t - \delta$
$\widetilde{c}_n = g$

Za razliko od algoritma 3.8 je v algoritmu 3.9 eno množenje namesto seštevanja. Pokazati se da, da nam ravno to zagotavlja visoko obratno stabilnost. Dokaz naslednjega izreka lahko najdete v [8].

Izrek 3.12 *Če je B bidiagonalna nesingularna matrika s singularnimi vrednostmi $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n > 0$, potem za izračunane singularne vrednosti $\widetilde{\sigma}_1 \geq \cdots \geq \widetilde{\sigma}_n > 0$ po dqds metodi velja (če ne uporabljamo premikov)*

$$\frac{|\widetilde{\sigma}_i - \sigma_i|}{\sigma_i} \leq (10n-5)u + O(u^2).$$

3.9 Enostranska Jacobijeva metoda za singularni razcep

Iz Jacobijeve metode za simetrično matriko, ki smo jo predstavili v razdelku 2.6, lahko razvijemo enostransko metodo za računanje singularnega razcepa. Tudi sedaj delamo s polno matriko A.

Predstavljamo si, da izvajamo Jacobijevo metodo za računanje lastnih vrednosti matrike A^TA, pri čemer matrike A^TA ne računamo eksplicitno. Pri enem koraku Jacobijeve metode bi iz matrike A^TA dobili $R_{pq}^TA^TAR_{pq}$. Namesto tega pri enostranski Jacobijevi metodi posodobimo le matriko A in v enem koraku iz nje dobimo matriko AR_{pq}, pri čemer ustrezno rotacijo določimo iz elementov matrike A^TA na mestih (p,p), (p,q) in (q,q). Postopek je predstavljen v algoritmu 3.10.

Po izvedenem koraku iz algoritma 3.10 se v matriki A posodobita p-ti in q-ti stolpec. Za celotni algoritem prideta v poštev le ciklična in pragovna varianta, saj bi morali pri klasični izračunati vse elemente matrike $B = A^TA$, da bi lahko poiskali maksimalni element, to pa bi bilo bistveno bolj zahtevno kot je sicer en korak algoritma 3.10. Tako v vsakem koraku za izračun ustrezne rotacije potrebujemo le tri elemente matrike A^TA, kar lahko izračunamo v linearni časovni zahtevnosti. Končamo, ko velja $|b_{pq}| \leq \epsilon\sqrt{b_{pp}b_{qq}}$ za vse $p \neq q$.

Matrika A na koncu skonvergira proti matriki z ortogonalnimi stolpci, saj je v limiti A^TA diagonalna matrika. Ko končamo z iteracijami, lahko kot približek za singularne vrednosti vzamemo kar norme stolpcev matrike A oziroma $\sigma_i = \|A(:,i)\|_2$ za $i = 1, \ldots, n$, pri čemer singularne vrednosti sedaj niso urejene po velikosti. Če smo med postopkom shranjevali produkte rotacij, potem je $V = Q$ in imamo matriko desnih singularnih vektorjev. Za neničelne singularne vre-

Algoritem 3.10 En korak enostranske Jacobijeve metode za računanje singularnega razcepa. Začetni podatki so matrika A, indeksa p, q in dosedanji produkt rotacij Q. Algoritem posodobi matriki A in Q.

$[A, Q] = \text{oneside\_jac}(A, Q, p, q)$
  izračunaj $b_{pp} = (A^TA)_{pp}$, $b_{pq} = (A^TA)_{pq}$, $b_{qq} = (A^TA)_{qq}$
  če velja $|b_{pq}| > \epsilon\sqrt{b_{pp}b_{qq}}$, potem
    $\tau = \frac{b_{pp}-b_{qq}}{2b_{pq}}$
    $t = \frac{\text{sign}(\tau)}{|\tau|+\sqrt{1+\tau^2}}$
    $c = \frac{1}{1+t^2}$
    $s = ct$
    $A = AR_{pq}(\varphi)$
    $Q = QR_{pq}(\varphi)$ (če potrebujemo tudi singularne vektorje).

dnosti lahko potem stolpce matrike U dobimo iz formule $u_i = \frac{1}{\sigma_i}A(:, i)$, matriko pa potem dopolnimo do ortogonalne matrike.

V nadaljevanju bomo pokazali, da lahko za številne matrike z Jacobijevo metodo z veliko relativno natančnostjo izračunamo tudi majhne singularne vrednosti.

Posledica 3.13 (Relativni Weylov izrek za singularne vrednosti) *Naj bodo $\sigma_1 \geq \cdots \geq \sigma_n$ singularne vrednosti matrike A in $\widetilde{\sigma}_1 \geq \cdots \geq \widetilde{\sigma}_n$ singularne vrednosti matrike $\widetilde{A} = Y^TAX$. Potem za $i = 1, \ldots, n$ velja*

$$|\widetilde{\sigma}_i - \sigma_i| \leq \sigma_i\epsilon,$$

kjer je $\epsilon = \max(\|X^TX - I\|_2, \|Y^TY - I\|_2)$.

Dokaz. Če definiramo matriki

$$C = \begin{bmatrix} 0 & A^T \\ A & 0 \end{bmatrix}, \quad Z = \begin{bmatrix} X & 0 \\ 0 & Y \end{bmatrix},$$

potem velja

$$Z^TCZ = \begin{bmatrix} 0 & \widetilde{A}^T \\ \widetilde{A} & 0 \end{bmatrix}.$$

Od tod za $i = 1, \ldots, n$ iz leme 3.9 in relativnega Weylovega izreka za lastne vrednosti simetrične matrike sledi $|\widetilde{\sigma}_i - \sigma_i| \leq \sigma_i\epsilon$, kjer je

$$\epsilon = \left\|Z^TZ - I\right\|_2 = \left\|\begin{bmatrix} X^TX - I & 0 \\ 0 & Y^TY - I \end{bmatrix}\right\|_2 = \max(\|X^TX - I\|_2, \|Y^TY - I\|_2). \qquad \blacksquare$$

Potrebujemo še nekaj pomožnih rezultatov, ki nam pomagajo omejiti obratno napako, ko matriko množimo z osnovnimi ortogonalnimi transformacijami.

Izrek 3.14 *Naj bo R točna Givensova rotacija (Householderjevo zrcaljenje) in $\widehat{R}$ njena aproksimacija v plavajoči vejici. Potem je*

$$\begin{aligned} fl(\widehat{R}A) &= R(A+E), \quad \|E\|_2 = \mathcal{O}(u)\|A\|_2 \\ fl(A\widehat{R}) &= (A+F)R, \quad \|F\|_2 = \mathcal{O}(u)\|A\|_2. \end{aligned}$$

Dokaz. Izrek bomo dokazali le za primer Givensove rotacije. Dokaz za Householderjevo zrcaljenje lahko najdete npr. v [15].

S podrobno analizo zaokrožitvenih napak (ki je opravljena npr. v [32]), se da pokazati, da za izračunana parametra Givensove rotacije velja $\widehat{c} = c(1+\delta_1)$ in $\widehat{s} = s(1+\delta_2)$, kjer je $|\delta_1|, |\delta_2| = \mathcal{O}(u)$. Od tod sledi $\|\widehat{R} - R\| = \mathcal{O}(u)$, pomembno pa je tudi, da ima matrika $\widehat{R} - R$ lahko največ štiri neničelne elemente.

Naj bo

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}.$$

Z analizo zaokrožitvenih napak dobimo

$$\begin{bmatrix} \widehat{y}_1 \\ \widehat{y}_2 \end{bmatrix} = \begin{bmatrix} \widehat{c}x_1(1+\epsilon_1) + \widehat{s}x_2(1+\epsilon_2) \\ -\widehat{s}x_1(1+\epsilon_3) + \widehat{c}x_2(1+\epsilon_4) \end{bmatrix},$$

kjer je $|\epsilon_i| \leq 2u$ za $i = 1, \ldots, 4$. Iz

$$\begin{bmatrix} \widehat{y}_1 \\ \widehat{y}_2 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} + \begin{bmatrix} c\delta_1 & s\delta_2 \\ -s\delta_2 & c\delta_1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} \widehat{c}\epsilon_1 & \widehat{s}\epsilon_2 \\ -\widehat{s}\epsilon_3 & \widehat{c}\epsilon_4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

sledi

$$\left\| \begin{bmatrix} \widehat{y}_1 \\ \widehat{y}_2 \end{bmatrix} - \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \right\|_2 = \mathcal{O}(u) \left\| \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \right\|_2.$$

Denimo, da R predstavlja rotacijo v p-ti in q-ti vrstici. Potem pri računanju produkta $\widehat{R}A$ lahko pride do napak le v p-ti in q-ti vrstici. Za p-to in q-to vrstico matrike RA potem dobimo

$$\|fl(\widehat{R}A) - RA\|_2 \leq \|fl(\widehat{R}A) - RA\|_F = \mathcal{O}(u)\|A([p\ q],:)\|_F \leq \mathcal{O}(u)\sqrt{2}\|A\|_2. \qquad \blacksquare$$

Izrek 3.15 *Vzemimo zaporedje ortogonalnih transformacij (rotacij ali zrcaljenj) $P_1, \ldots, P_j$ in $Q_1, \ldots, Q_j$. Za numerično izračunano matriko $B = P_j \cdots P_1 A Q_1 \cdots Q_j$ velja*

$$\widetilde{B} = P_j \cdots P_1 (A + E_j) Q_1 \cdots Q_j,$$

kjer je $\|E_j\|_2 = 2j\mathcal{O}(u)\|A\|_2$.

Dokaz. Uporabimo indukcijo. Za $j = 0$ očitno drži.

Predpostavimo, da izrek velja za $j-1$ množenj z ortogonalnimi transformacijami. Naj bo $B_{j-1} = P_{j-1} \cdots P_1 A Q_1 \cdots Q_{j-1}$. Če definiramo $P^{(k)} = P_k \cdots P_1$ in $Q^{(k)} = Q_1 \cdots Q_k$, potem je predpostavka, da velja $\widehat{B}_{j-1} = P^{(j-1)}(A + E_{j-1})Q^{(j-1)}$, kjer je $\|E_{j-1}\|_2 = 2(j-1)\mathcal{O}(u)\|A\|_2$.

Pri analizi računanja $B_j = P_j B_{j-1} Q_j$ dobimo $\widehat{B}_{j-1/2} = P_j(\widehat{B}_{j-1} + F_1)$ in $\widehat{B}_j = (\widehat{B}_{j-1/2} + F_2)Q_j$, kjer je $\|F_1\|_2, \|F_2\|_2 = \mathcal{O}(u)\|A\|_2$. Tako pridemo do

$$\begin{aligned} \widehat{B}_j &= (P_j(\widehat{B}_{j-1} + F_1) + F_2)Q_j \\ &= P_j \widehat{B}_{j-1} Q_j + P_j F_1 Q_j + F_2 Q_j \\ &= P_j \left(P^{(j-1)}(A + E_{j-1})Q^{(j-1)} \right) Q_j + P_j F_1 Q_j + F_2 Q_j \\ &= P^{(j)} \left[A + E_{j-1} + P^{(j-1)^T} F_1 Q^{(j-1)^T} + P^{(j)^T} F_2 Q^{(j-1)^T} \right] Q^{(j)}. \end{aligned}$$

Sledi $E_j = E_{j-1} + P^{(j-1)^T} F_1 Q^{(j-1)^T} + P^{(j)^T} F_2 Q^{(j-1)^T}$ in ocena $\|E_j\|_2 \leq 2j\mathcal{O}(u)\|A\|_2$. $\blacksquare$

Izrek 3.16 *Naj bo $A = DX$ $n \times n$ matrika, kjer je D nesingularna diagonalna matrika, X pa nesingularna matrika. Če $\widehat{A}_m$ dobimo po m korakih enostranske Jacobijeve metode in so $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n$ singularne vrednosti A, $\widehat{\sigma}_1 \geq \cdots \widehat{\sigma}_n$ pa singularne vrednosti $\widehat{A}_m$, potem velja*

$$\frac{|\widehat{\sigma}_i - \sigma_i|}{\sigma_i} \leq O(mu)\kappa_2(X).$$

Dokaz. Uporabimo indukcijo. Najprej poglejmo, kaj dobimo v primeru $m = 1$, ko matriko A množimo le z eno Givensovo rotacijo R_1.

Naj bo $F_1 = \widehat{A}_1 - AR_1$. Če uporabimo izrek 3.14 za i-to vrstico matrike $A_1 = AR_1$, potem dobimo

$$\|F_1(i,:)\|_2 = \|\widehat{A}_1(i,:) - A(i,:)R_1\|_2 = \mathcal{O}(u)\|A(i,:)\|_2 = \mathcal{O}(u)\|d_{ii}X(i,:)\|_2.$$

To pomeni

$$\left\| \frac{1}{d_{ii}} F_1(i,:) \right\|_2 = \mathcal{O}(u)\|X(i,:)\|_2,$$

in $\|D^{-1}F_1\|_2 = \mathcal{O}(u)\|X\|_2$. Iz zveze

$$\widehat{A}_1 = AR_1 + F_1 = AR_1(I + R_1^T A^{-1} F_1) = AR_1(I + R_1^T X^{-1} D^{-1} F_1)$$

sledi $\widehat{A}_1 = AR_1(I + E_1)$, kjer lahko ocenimo $\|E_1\|_2 = \mathcal{O}(u)\|X^{-1}\|_2\|X\|_2$.

Denimo, da za $A_{m-1} = AR_1 \cdots R_{m-1}$ za numerično izračunano matriko velja $\widehat{A}_{m-1} = A_{m-1}(I + E_{m-1})$, kjer je $\|E_{m-1}\|_2 = \mathcal{O}((m-1)u)\kappa_2(X)$. Vemo, da velja $\widehat{A}_m = \widehat{A}_{m-1}R_m(I + E)$, kjer je $\|E\|_2 = \mathcal{O}(u)\|A\|_2$. Če razpišemo

$$\begin{aligned} \widehat{A}_m &= A_m R_1 \cdots R_{m-1}(I + E_{m-1})R_m(I + E) \\ &= A_m R_1 \cdots R_m(I + R_m^T E_{m-1} R_m + E + R_m^T E_{m-1} R_m E), \end{aligned}$$

dobimo

$$E_m = R_m^T E_{m-1} R_m + E + R_m^T E_{m-1} R_m E.$$

Če zanemarimo $R_m^T E_{m-1} R_m E$, dobimo oceno $\|E_m\|_2 = \mathcal{O}(mu)\kappa_2(X)$.

Sedaj uporabimo relativni Weylov izrek in dobimo končno oceno $|\widehat{\sigma}_i - \sigma_i| \leq \sigma_i \epsilon$, kjer je

$$\epsilon = \|(I + E_m)^T(I + E_m) - I\|_2 \leq 3\|E_m\|_2 \leq 3\mathcal{O}(mu)\kappa_2(X),$$

za $i = 1, \ldots, n$. ∎

Izrek 3.17 *Če enostransko Jacobijevo metodo zaustavimo, ko za vse $j \neq k$ velja*

$$|b_{jk}| \leq \epsilon \sqrt{b_{jj} b_{kk}},$$

kjer je $B = A^T A$, in so $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n$ singularne vrednosti A, $\alpha_1^2 \geq \ldots \geq \alpha_n^2$ pa diagonalni elementi B, potem je

$$|\sigma_i - \alpha_i| \leq n\epsilon|\alpha_i|.$$

Dokaz. Pišemo lahko $B = D\widetilde{B}D$, kjer je $D = \mathrm{diag}(\sqrt{b_{11}}\ldots,\sqrt{b_{nn}})$ in $|\widetilde{b}_{jk}| \leq \epsilon$ za vse $j \neq k$. Matrika $\widetilde{B}$ je simetrična in pozitivno definitna, zato obstaja razcep Choleskega $\widetilde{B} = LL^T$, kjer je L spodnja trikotna matrika. Tako dobimo $B = A^TA = DLL^TD$.

Ker na koncu kot približke za singularne vrednosti vzamemo diagonalne elemente matrike B, bi radi ocenili razliko med singularnimi vrednostmi matrik A in D. Iz $A^TA = DLL^TD$ sledi

$$L^{-1}D^{-1}A^TAD^{-1}L^{-T} = (AD^{-1}L^{-T})^T(AD^{-1}L^{-T}) = I,$$

to pa pomeni, da je matrika $AD^{-1}L^{-T}$ ortogonalna. Torej obstaja taka ortogonalna matrika Q, da je $L^TD = QA$. Ker imata matriki A in QA iste singularne vrednosti, iz relativnega Weylovega izreka sledi $|\sigma_i - \alpha_i| \leq \alpha_i\theta$, kjer je

$$\theta = \|L^TL - I\|_2 = \|LL^T - I\|_2 \leq \|\widetilde{G} - I\|_F \leq n\epsilon.$$

■

Jacobijeva metoda torej izračuna singularne vrednosti (in vektorje) z visoko relativno natančnostjo, če lahko zapišemo $A = DX$, kjer je X dobro pogojena matrika, D pa nesingularna diagonalna matrika.

Jacobijeva metoda tako ponavadi deluje bolje od drugih v primeru, ko ima D elemente, ki se po velikosti bistveno razlikujejo. Za zgled si oglejmo naslednji primer iz [8].

Zgled 3.5 *Naj bo*

$$A = \begin{bmatrix} \delta & 1 & 1 & 1 \\ \delta & \delta & 0 & 0 \\ \delta & 0 & \delta & 0 \\ \delta & 0 & 0 & \delta \end{bmatrix},$$

kjer je $\delta = 10^{-20}$. *Velja*

$$A = \begin{bmatrix} 1 & & & \\ & \delta & & \\ & & \delta & \\ & & & \delta \end{bmatrix} \cdot \begin{bmatrix} \delta & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}.$$

Po prvem koraku bidiagonalizacije dobimo

$$\begin{bmatrix} -2\delta & -1/2 - \delta/2 & -1/2 - \delta/2 & -1/2 - \delta/2 \\ 0 & -1/2 + 5\delta/6 & -1/2 - \delta/6 & -1/2 - \delta/6 \\ 0 & -1/2 - \delta/6 & -1/2 + 5\delta/6 & -1/2 - \delta/6 \\ 0 & -1/2 - \delta/6 & -1/2 - \delta/6 & -1/2 + 5\delta/6 \end{bmatrix},$$

kar se zaokroži v

$$\begin{bmatrix} -2\delta & -1/2 & -1/2 & -1/2 \\ 0 & -1/2 & -1/2 & -1/2 \\ 0 & -1/2 & -1/2 & -1/2 \\ 0 & -1/2 & -1/2 & -1/2 \end{bmatrix}$$

in v naslednjem koraku dobimo

$$\begin{bmatrix} -2\delta & \sqrt{3}/2 & 0 & 0 \\ 0 & 3/2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

Singularne vrednosti dobljene bidiagonalne matrike so $\sqrt{3}, \sqrt{3}\delta, 0, 0$, *točne singularne vrednosti pa so* $\sqrt{3}, \sqrt{3}\delta, \delta, \delta$. □

Za bidiagonalno matriko je dqds obratno stabilen algoritem, problem pa je, če imamo na začetku polno matriko. Natančnost lahko izgubimo pri redukciji na bidiagonalno obliko, pri Jacobijevi metodi pa se to ne zgodi, saj matrike na začetku ne reduciramo.

3.9.1 Lastni razcep simetrične pozitivno definitne matrike

Lastni razcep simetrične pozitivno definitne matrike A lahko izračunamo na naslednji način:

a) izračunamo razcep Choleskega $A = LL^T$,

b) izračunamo singularni razcep $L = U\Sigma V^T$,

c) $A = U\Sigma^2 U^T$.

Če za točko b) uporabimo enostransko Jacobijevo metodo in je $L = DX$, kjer je D diagonalna matrika in je matrika X dobro pogojena, potem imajo izračunane singularne vrednosti relativno napako omejeno z $\mathcal{O}(u)\kappa(X)$.

Če upoštevamo še obratno stabilnost razcepa Choleskega v točki a), celotni postopek vrne singularne vrednosti z relativno napako $\mathcal{O}(u)\kappa(X)^2$.

V primeru, ko je matrika X dobro pogojena, lahko s tem postopkom vse lastne vrednosti simetrične pozitivno definitne matrike A izračunamo z visoko relativno natančnostjo. Za zgled si oglejmo naslednji primer iz [8].

Zgled 3.6 *Vzemimo matriko*

$$A = \begin{bmatrix} 1 & \sqrt{\delta} & \sqrt{\delta} \\ \sqrt{\delta} & 1 & 10\delta \\ \sqrt{\delta} & 10\delta & 100\delta \end{bmatrix},$$

kjer je $\delta = 10^{-20}$. *Če* A *reduciramo na tridiagonalno matriko, je točen rezultat*

$$T = \begin{bmatrix} 1 & \sqrt{2\delta} & \\ \sqrt{2\delta} & 1/2 + 60\delta & 1/2 - 50\delta \\ & 1/2 - 50\delta & 1/2 + 40\delta \end{bmatrix}.$$

Pri numeričnem računanju v dvojni natančnosti dobimo sicer simetrično matriko

$$\widetilde{T} = \begin{bmatrix} 1 & \sqrt{2\delta} & \\ \sqrt{2\delta} & 1/2 & 1/2 \\ & 1/2 & 1/2 \end{bmatrix},$$

ki pa ni pozitivno definitna, pri čemer smo izgubili relativno natančnost najmanjše lastne vrednosti.

Če naredimo najprej razcep Choleskega in nato uporabimo enostransko Jacobijevo metodo, potem izračunamo vse lastne vrednosti s polno natančnostjo: $\lambda_1 = 1 + \sqrt{\delta}$, $\lambda_2 = 1 - \sqrt{\delta}$, $\lambda_3 = 99\delta$. □

Matlab

Ukaz `eig` lahko uporabimo tudi za posplošene probleme lastnih vrednosti. Tako `[X,D] = eig(A,B)` vrne matriko lastnih vektorjev X in diagonalno matriko lastnih vrednosti D, da je $AX = BXD$.

Za izračun posplošene Schurove forme je na voljo ukaz `qz`. Tako `[S, T, Q, Z] = qz(A,B)` za realni matriki A, B vrne kvazi zgornje trikotni matriki S, T in ortogonalni matriki Q, Z, da je $QAZ = S$ in $QBZ = T$.

Za polinomski problem lastnih vrednosti je na voljo ukaz `polyeig`. Tako npr. `polyeig(A,B,C)` vrne lastne vrednosti kvadratnega problema lastnih vrednosti $(\lambda^2 A + \lambda B + C)x = 0$.

Dodatna literatura

Za računanje singularnega razcepa je na voljo obsežna literatura. V slovenščini lahko skoraj celotno snov najdete v knjigi [8]. Kar se tiče tuje literature, lahko skoraj vse algoritme, ki smo jih navedli, skupaj s potrebno analizo, najdete v [9]. Uporabna sta tudi učbenika [7] in [13].

Več o kvadratnem problemu lastnih vrednosti lahko najdete v preglednem članku [29].

Več o regularizaciji s pomočjo singularnega razcepa lahko najdete npr. v [12].

Poglavje 4

Enakomerna aproksimacija

4.1 Aproksimacija

Denimo, da imamo podano funkcijo f, ki je zvezna na intervalu $[a, b]$. Radi bi jo čim bolje aproksimirali s kakšno preprosto funkcijo g, ki bi bila lažje izračunljiva in bi se jo dalo po potrebi preprosto odvajati oziroma integrirati.

Ponavadi za aproksimacijo uporabljamo polinome, odvisno od oblike funkcije f in namena aproksimacije pa uporabljamo tudi druge funkcije, npr. trigonometrične polinome, racionalne funkcije in zlepke. Polinomi imajo zelo lepe lastnosti, saj je računanje vrednosti enostavno, prav tako pa jih je enostavno odvajati in integrirati. Pomembno je tudi, da množica polinomov stopnje kvečjemu k tvori vektorski prostor, ki ga bomo označili s $\mathbb{P}_k$. Kadar iščemo aproksimacijsko funkcijo, ki se jo da zapisati kot linearno kombinacijo baznih funkcij, govorimo o linearni aproksimaciji.

Kakovost aproksimacije merimo z normo ostanka $\|f - g\|$. Različne norme pripeljejo do različnih aproksimacijskih funkcij. Najpogostejše izbire so:

a) *diskretna aproksimacija po metodi najmanjših kvadratov*, kjer iščemo g, ki minimizira

$$\sum_{i=1}^{m} |f(x_i) - g(x_i)|^2,$$

b) *zvezna aproksimacija po metodi najmanjših kvadratov*, kjer iščemo g, ki minimizira

$$\|f - g\|_2 := \left(\int_a^b |f(x) - g(x)|^2 dx \right)^{1/2},$$

c) *diskretna enakomerna aproksimacija*, kjer iščemo g, ki minimizira

$$\max_{i=1,\ldots,m} |f(x_i) - g(x_i)|,$$

d) *zvezna enakomerna aproksimacija*, kjer iščemo g, ki minimizira

$$\|f - g\|_\infty := \max_{x \in [a,b]} |f(x) - g(x)|.$$

Z aproksimacijo po metodi najmanjših kvadratov smo se že srečali v **??**. poglavju. Izkazalo se je, da v primeru linearnega modela problem lahko prevedemo na reševanje linearnega sistema in tako dobimo aproksimacijsko funkcijo z direktno metodo. V tem poglavju si bomo podrobneje pogledali zvezno enakomerno aproksimacijo. Izkaže se, da polinoma najboljše enakomerne aproksimacije ne moremo izračunati s končno formulo.

Enakomerno aproksimacijo srečamo med drugim pri računanju vrednosti elementarnih funkcij. Če npr. od računalnika zahtevamo, da nam vrne vrednost funkcije sin pri izbranem argumentu x, bomo kot rezultat dobili numerično izračunano vrednost $g(x)$, kjer je g polinom (ali racionalna funkcija) nizke stopnje, izbran tako, da se izračunani $g(x)$ od $\sin(x)$ ne razlikuje za več kot osnovno zaokrožitveno napako. Za elementarne funkcije zadošča, če poznamo tak polinom na nekem zaprtem intervalu $[a, b]$, saj lahko izračun $f(x)$ prevedemo na ta interval. Npr., za izračun $f(x) = \sin(x)$ očitno zadošča, če znamo aproksimirati vrednosti funkcije na intervalu $[0, \pi]$.

4.2 Enakomerna aproksimacija s polinomi

Vsako zvezno funkcijo se da na zaprtem intervalu poljubno dobro aproksimirati s polinomom. To nam zagotavlja znani Weierstrassov izrek, katerega dokaz lahko najdete npr. v [18].

Izrek 4.1 (Weierstrass) *Naj bo f zvezna funkcija na intervalu $[a, b]$. Za poljuben $\epsilon > 0$ obstaja polinom p, da je $|f(x) - p(x)| \leq \epsilon$ za vsak $x \in [a, b]$.*

Seveda nam zgornji izrek nič ne pove o stopnji polinoma p, ki dovolj dobro aproksimira funkcijo f. Če se omejimo le na polinome stopnje kvečjemu n, potem lahko definiramo

$$\operatorname{dist}(f, \mathbb{P}_n) = \min_{p \in \mathbb{P}_n} \|f - p\|_\infty.$$

Weierstrassov izrek nam pove, da za vsako zvezno funkcijo f velja $\lim_{n\to\infty} \operatorname{dist}(f, \mathbb{P}_n) = 0$, nas pa zanima, kako poiščemo polinom najboljše enakomerne aproksimacije stopnje kvečjemu n.

Izkaže se, da je zadosten pogoj za to, da je p polinom najboljše enakomerne aproksimacije za f, ta, da na intervalu $[a, b]$ razlika $f - p$ alternirajoče zavzame maksimum po absolutni vrednosti v $n + 2$ točkah.

Izrek 4.2 *Naj bo f zvezna funkcija na intervalu $[a, b]$. Če za polinom $p \in \mathbb{P}_n$ na $[a, b]$ obstaja $n + 2$ takih točk $x_0 < x_1 < \ldots < x_{n+1}$, da za razliko $r = f - p$ velja, da v točkah $x_0, \ldots, x_{n+1}$ doseže $|r|$ svoj maksimum, pri čemer predznak r med točkami z zaporednim indeksom alternira, potem je p polinom najboljše enakomerne aproksimacije za f na $[a, b]$.*

Dokaz. Denimo, da p ni polinom najboljše aproksimacije. Potem obstaja tak polinom $q \in \mathbb{P}_n$, da velja

$$|f(x_i) - q(x_i)| < |f(x_i) - p(x_i)|, \quad i = 0, \ldots, n + 1.$$

Iz zgornje neenakosti sledi, da je predznak $(f(x_i) - p(x_i)) - (f(x_i) - q(x_i))$ enak predznaku $f(x_i) - p(x_i)$ za $i = 0, \ldots, n + 1$. Toda,

$$(f - p) - (f - q) = q - p$$

je polinom stopnje kvečjemu n, ki ne more alternirati v $n+2$ točkah, saj bi potem imel $n+1$ ničel. Torej tak polinom q ne obstaja in je p res polinom najboljše enakomerne aproksimacije. ■

Dokazati se da tudi, da tak polinom res obstaja, dokaz lahko najdete npr. v [18]. To dejstvo se potem uporabi v Remezovem[1] postopku za iskanje polinoma najboljše enakomerne aproksimacije, ki je opisan v algoritmu 4.1. Postopek skonstruira zaporedje polinomov, ki konvergirajo proti polinomu $p \in \mathbb{P}_n$, ki zadošča predpostavkam izreka 4.2 in je tako polinom najboljše enakomerne aproksimacije.

Algoritem 4.1 Remezov postopek. Začetni podatki so funkcija f, interval $[a,b]$, stopnja n in toleranca ϵ. Algoritem vrne polinom najboljše enakomerne aproksimacije stopnje kvečjemu n za f na $[a,b]$.

---

izberi delilne točke $x_0 < x_1 < \ldots < x_{n+1}$ na intervalu $[a,b]$
ponavljaj

a) določi polinom $p \in \mathbb{P}_n$, za katerega velja $f(x_i) - p(x_i) = (-1)^i r$ za nek r
b) določi točko y, kjer $|f-p|$ doseže maksimum na $[a,b]$
c) če je $|f(y) - p(y)| - |r| < \epsilon$, končaj in vrni p
č) poišči delilno točko z za zamenjavo:

- $x_k \le y \le x_{k+1}$: če je predznak $f-p$ v x_k in y enak, potem $z = x_k$, sicer $z = x_{k+1}$;
- $a \le y \le x_0$: če je predznak $f-p$ v x_0 in y enak, potem $z = x_0$, sicer $z = x_{n+1}$;
- $x_{n+1} \le y \le b$: če je predznak $f-p$ v x_{n+1} in y enak, potem $z = x_{n+1}$, sicer $z = x_0$

d) zamenjaj točko z z y in po potrebi preštevilči delilne točke po velikosti

---

Poglejmo si podrobneje, kako izvedemo posamezne korake algoritma 4.1. Pri točki a) moramo rešiti linearni sistem. Če npr. polinom p zapišemo v standardni bazi kot $p(x) = a_0 + \cdots + a_n x^n$, dobimo sistem

$$\begin{aligned} a_0 + a_1 x_0 + \cdots + a_n x_0^n - (-1)^0 r &= f(x_0) \\ &\vdots \\ a_0 + a_1 x_{n+1} + \cdots + a_n x_{n+1}^n - (-1)^{n+1} r &= f(x_{n+1}) \end{aligned}$$

za koeficiente $a_0, \ldots, a_n$ in r.

Pri točki b) moramo uporabiti eno izmed numeričnih metod za iskanje lokalnih ekstremov. Če lahko f odvajamo, potem lahko npr. z Newtonovo metodo pridemo do približkov za lokalne ekstreme.

Naslednja lema nam zagotavlja, da se v primeru izpolnjene točke c) polinom p od polinoma najboljše enakomerne aproksimacije na celem intervalu $[a,b]$ razlikuje za manj kot ϵ.

Lema 4.3 *Naj bo f zvezna funkcija na $[a,b]$. Če za polinom $p \in \mathbb{P}_n$ velja, da razlika $f-p$ alternira v*

[1] Evgenij Jakovlevič Remez (1896-1975) je bil ukrajinski matematik, rojen v Belorusiji. Postopek konstrukcije polinoma najboljše enakomerne aproksimacije je objavil leta 1936 v francoščini kot Eugene Remes, zato je postopek znan tudi kot Remesov. Tako kot se danes veliko ukrajinskih in ruskih matematičnih člankov objavi v angleščini, da so dostopni širšemu občinstvu, tako so v obdobju med prvo in drugo svetovno vojno za to uporabljali francoščino.

$n+2$ točkah $x_0 < x_1 < \ldots < x_{n+1}$ z intervala $[a,b]$, potem velja

$$\min_{i=0,\ldots,n+1} |f(x_i) - p(x_i)| \leq \operatorname{dist}(f, \mathbb{P}_n) \leq \|f-p\|_\infty.$$

Dokaz. Desna neenakost $\operatorname{dist}(f, \mathbb{P}_n) \leq \|f-p\|_\infty$ očitno velja za vsak polinom $p \in \mathbb{P}_n$.

Naj bo q polinom najboljše enakomerne aproksimacije za f na $[a,b]$. Denimo, da leva neenakost ne velja in je torej $|f(x_i) - p(x_i)| > \operatorname{dist}(f, \mathbb{P}_n)$ za $i = 0, \ldots, n+1$. Od tod, podobno kot v dokazu izreka 4.2, sledi, da je v vseh točkah x_i, $i = 0, \ldots, n+1$, predznak $(f(x_i) - p(x_i)) - (f(x_i) - q(x_i))$ enak predznaku $f(x_i) - p(x_i)$. To pa pomeni, da polinom q ne more obstajati, kar je seveda protislovje. ∎

Zgled 4.1 *Poglejmo si preprost primer uporabe Remezovega postopka. Funkcijo e^x bi radi na intervalu $[0,1]$ po metodi najboljše enakomerne aproksimacije čim bolje aproksimirali s premico oblike $y = a_0 + a_1 x$.*

Denimo, da smo na začetku izbrali točke $x_0 = 0$, $x_1 = 1/2$, $x_2 = 1$. Sedaj nastavimo sistem $f(x_i) - p(x_i) = (-1)^i r$ za $i = 0,1,2$. Enačbam

$$\begin{aligned} 1 - a_0 &= r \\ e^{1/2} - a_0 - 1/2 a_1 &= -r \\ e - a_0 - a_1 &= r \end{aligned}$$

ustreza rešitev $a_0 = \frac{1}{4}(1 - e^{1/2})(3 + e^{1/2})$, $a_1 = e - 1$, $r = \frac{1}{4}(e^{1/2} - 1)^2$.

Z odvajanjem $f - p$ ugotovimo, da ima $|f - p|$ maksimum pri $y = \ln(e-1) = 0.54132$. Ker velja $f(y) - p(y) > 0$, v naslednjem koraku zamenjamo točko x_1 z y. Nove točke za naslednji korak so tako $x_0 = 0$, $x_1 = \ln(e-1)$, $x_2 = 1$.

Z novimi točkami ponovimo postopek od začetka in dobimo $a_0 = 1/2(e + (1-e)\ln(e-1))$, $a_1 = e - 1$, $r = 1 - a_0$. Ker se v naslednjem koraku izkaže, da $|f - p|$ doseže maksimum na $[0,1]$ ravno v delilnih točkah, se postopek konča in našli smo polinom najboljše enakomerne aproksimacije. □

Zgornji zgled ni povsem tipičen, saj se ponavadi Remezov postopek ne konča v končnem številu korakov. Običajno postopek končamo, ko je pogoj v točki c) izpolnjen za izbrano toleranco ϵ, saj $|r|$ konvergira proti $\operatorname{dist}(f, \mathbb{P}_n)$. Pokazati se da (glej npr. [18]), da je konvergenca linearna.

Omenimo še, da obstaja tudi različica Remezovega postopka, kjer v vsakem koraku zamenjamo vse točke s točkami, kjer ima razlika $f - p$ lokalne ekstreme. Pri tej metodi imamo sicer več dela z iskanjem ekstremov, ima pa zato v bližini rešitve kvadratično konvergenco.

4.3 Ekonomizacija Čebiševa

Remezov postopek ni preprost in velikokrat bi nam zadoščalo, če bi znali funkcijo f na kak enostavnejši način aproksimirati s polinomom nizke stopnje p, za katerega bi veljalo $\|f - p\|_\infty \leq \epsilon$ za izbrano konstanto $\epsilon > 0$.

Tako npr. vemo, da lahko vsako dovolj gladko funkcijo aproksimiramo s pomočjo Taylorjevega polinoma, ki ga dobimo tako, da odrežemo razvoj v Taylorjevo vrsto. Iz ocene za napako

Taylorjeve vrste lahko ocenimo, koliko členov potrebujemo, da bo skupna napaka na celem intervalu $[a, b]$ dovolj majhna. Ker za Taylorjev polinom velja, da dobro aproksimira le v okolici točke, okrog katere smo ga razvili, je smiselno vzeti razvoj v Taylorjevo vrsto okrog točke $(a + b)/2$.

Izkaže se, da so polinomi, ki jih tako dobimo, ponavadi občutno višjih stopenj kot pa bi jih lahko dobili preko algoritmov za najboljšo enakomerno aproksimacijo. Stopnjo polinoma lahko zmanjšamo s pomočjo ekonomizacije Čebiševa, ki temelji na lastnostih polinomov Čebiševa[2].

Definicija 4.4 *Polinomi Čebiševa so definirani s* $T_0(x) = 1$, $T_1(x) = x$ *in z rekurzivno zvezo*

$$T_{m+1}(x) = 2xT_m(x) - T_{m-1}(x)$$

za $m \geq 1$.

Preverimo lahko, da za polinome Čebiševa velja:

a) $T_m(x) = 2^{m-1}x^m + \mathcal{O}(x^{m-1})$,

b) $T_m(x) = \begin{cases} \cos(m \arccos x), & |x| \leq 1, \\ \cosh(m \operatorname{arccosh} x), & |x| \geq 1, \end{cases}$

c) $|T_m(x)| \leq 1$ za $|x| \leq 1$.

Naslednji seznam prikazuje nekaj prvih polinomov Čebiševa in kako se z njimi izražajo polinomi iz standardne baze:

$$\begin{array}{lcllcl} T_0(x) & = & 1 & 1 & = & T_0 \\ T_1(x) & = & x & x & = & T_1 \\ T_2(x) & = & 2x^2 - 1 & x^2 & = & \frac{1}{2}(T_0 + T_2) \\ T_3(x) & = & 4x^3 - 3x & x^3 & = & \frac{1}{4}(3T_1 + T_3) \\ T_4(x) & = & 8x^4 - 8x^2 + 1 & x^4 & = & \frac{1}{8}(3T_0 + 4T_2 + T_3) \\ T_5(x) & = & 16x^5 - 20x^3 + 5x & x^5 & = & \frac{1}{16}(10T_1 + 5T_3 + T_5). \end{array}$$

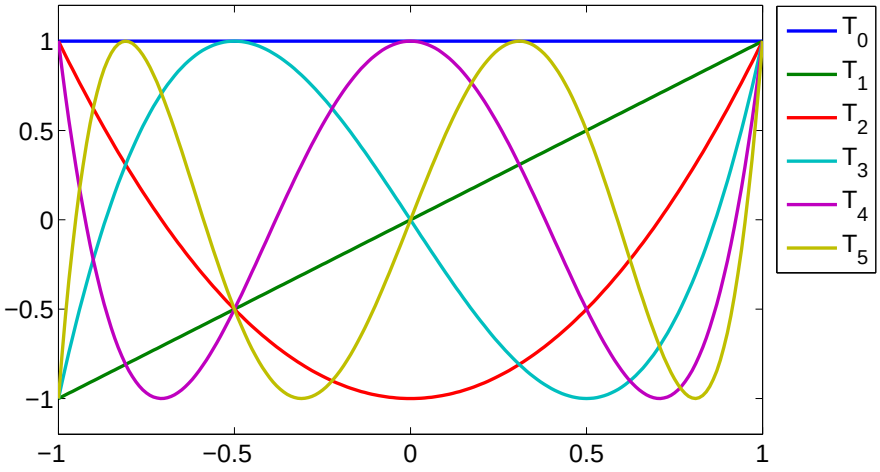
Razlog, da so polinomi Čebiševa zelo pomembni pri najboljši enakomerni aproksimaciji, se skriva v naslednji lemi.

Lema 4.5 *Velja:*

a) Polinom stopnje kvečjemu $n - 1$, *ki najbolje enakomerno aproksimira* x^n *na intervalu* $[-1, 1]$ *je podan s* $q(x) = x^n - 2^{1-n}T_n(x)$.

b) Izmed vseh polinomov stopnje n *z vodilnim koeficientom* 1 *ima* $2^{1-n}T_n$ *najmanjšo neskončno normo na intervalu* $[-1, 1]$.

[2]Polinomi se imenujejo po ruskem matematiku Pafnutiju Lvoviču Čebiševu (1821-1894), s črko T pa se označujejo zaradi transkripcije njegovega priimka v Tchebychef. Čebišev je prvi razvil splošno teorijo ortogonalnih polinomov, med drugim pa je znan tudi po pomembnih rezultatih iz aproksimacije in teorije verjetnosti.



Slika 4.1: Nekaj prvih polinomov Čebiševa na intervalu $[-1,1]$.

Dokaz. Velja $2^{1-n}T_n(x) = x^n - q(x)$. Ker $2^{1-n}T_n$ na $[-1,1]$ zavzame maksimum absolutne vrednosti v $n+1$ točkah z alternirajočim predznakom, je po izreku 4.2 potem q polinom najboljše enakomerne aproksimacije stopnje kvečjemu $n-1$ za x^n na $[-1,1]$.

Poljuben polinom p stopnje n z vodilnim koeficientom 1 lahko zapišemo kot $p(x) = x^n - r(x)$, kjer je r polinom stopnje kvečjemu $n-1$. Očitno bo norma $\|p\|_\infty$ minimalna natanko takrat, ko bo r polinom najboljše enakomerne aproksimacije za x^n na $[-1,1]$, to pa pomeni $r = q$ in $p = 2^{1-n}T_n$. ∎

Denimo, da na intervalu $[-1,1]$ za polinom p stopnje n iščemo polinom najboljše enakomerne aproksimacije stopnje kvečjemu $n-1$. Rešitev lahko direktno izračunamo s pomočjo leme 4.5. Če polinom p zapišemo v bazi polinomov Čebiševa kot

$$p = \sum_{i=0}^{n} a_i T_i,$$

potem je polinom najboljše aproksimacije stopnje $n-1$ kar

$$q = \sum_{i=0}^{n-1} a_i T_i.$$

Za razliko $p - q = a_n T_n$ namreč velja, da ima na intervalu $[-1,1]$ $n+1$ ekstremov, v katerih predznak alternira. Po izreku 4.2 je potem q polinom najboljše enakomerne aproksimacije iz $\mathbb{P}_{n-1}$ za p na $[-1,1]$.

Na zgornji ugotovitvi temelji postopek ekonomizacije Čebiševa, zapisan v algoritmu 4.2, s katerim lahko polinom aproksimiramo s polinomom nižje stopnje in tako zmanjšamo število operacij, potrebnih za izračun vrednosti aproksimacijskega polinoma. Če nismo na intervalu $[-1,1]$ to ni nobena težava, saj uporabimo linearno substitucijo in problem prevedemo na $[-1,1]$.

V primeru, ko stopnjo polinoma zmanjšamo za ena, torej ko je $k = n-1$, je p_{n-1}^{econ}, ki ga vrne algoritem 4.2, kar polinom najboljše enakomerne aproksimacije stopnje $n-1$ za začetni polinom p_n. V primeru $k < n$ pa si lahko predstavljamo, da ekonomizacija po vrsti niža stopnjo poli-

Algoritem 4.2 Ekonomizacija Čebiševa. Začetni podatki so polinom p_n stopnje n na intervalu $[a,b]$ in $k<n$. Algoritem vrne polinom p_k^{econ} stopnje k in tak $\delta>0$, da velja $\|p_n-p_k^{\text{econ}}\|\le\delta$.

a) Polinom p_n s substitucijo preslikaj v polinom q_n na intervalu $[-1,1]$, da je

$$q_n(\xi)=p_n\left(a+\frac{b-a}{2}(\xi+1)\right).$$

b) Polinom q_n razvij po polinomih Čebiševa kot $q_n(\xi)=d_0+d_1T_1(\xi)+\cdots+d_nT_n(\xi)$.

c) Razvoj q_n odreži pri $k<n$ v $q_k(\xi)=d_0+d_1T_1(\xi)+\cdots+d_kT_k(\xi)$ in za oceno razlike vzemi $\delta=|d_{k+1}|+\cdots+|d_n|$.

č) Polinom q_k izrazi spet v standardni bazi kot $q_k(\xi)=b_0+b_1\xi+\cdots+b_k\xi^k$.

d) Polinom q_k z obratno substitucijo preslikaj nazaj na intervalu $[a,b]$ v

$$p_k^{\text{econ}}(x)=q_k\left(-1+2\frac{x-a}{b-a}\right).$$

noma za eno. Če vpeljemo še vmesne polinome $p_{k+1}^{\text{econ}},\ldots,p_{n-1}^{\text{econ}}$, potem je vedno p_m^{econ} polinom najboljše enakomerne aproksimacije stopnje m za polinom p_{m+1}^{econ} za $m=n-1,n-2,\ldots,k$.

Vemo, da na $[a,b]$ velja $\|p_n-p_k^{\text{econ}}\|_\infty\le\delta$. Denimo, da smo začetni polinom p_n skonstruirali tako, da velja $\|f-p_n\|_\infty\le\epsilon$. Od tod sledi

$$\|f-p_k^{\text{econ}}\|_\infty\le\|f-p_n\|_\infty+\|p_n(x)-p_k^{\text{econ}}\|_\infty\le\epsilon+\delta.$$

Zgled 4.2 *Funkcijo $f(x)=e^x$ bi radi na intervalu $[-1/2,1/2]$ aproksimirali s polinomom stopnje 3 oziroma 4. Če f aproksimiramo s prvimi 6 členi razvoja v Taylorjevo vrsto okrog 0, dobimo*

$$p_5(x)=1+x+\frac{1}{2}x^2+\frac{1}{6}x^3+\frac{1}{24}x^4+\frac{1}{120}x^5.$$

Napako Taylorjevega polinoma lahko ocenimo z

$$|f(x)-p_5(x)|\le\frac{e^{1/2}}{720}=3.6\cdot10^{-5}.$$

Če p_5 preslikamo na $[-1,1]$ s substitucijo $q_5(\xi)=p_5(x/2)$, dobimo

$$q_5(\xi)=1+\frac{1}{2}\xi+\frac{1}{8}\xi^2+\frac{1}{48}\xi^3+\frac{1}{384}\xi^4+\frac{1}{3840}\xi^5.$$

Polinom q_5 se v polinomih Čebiševa izraža kot

$$q_5=\frac{1089}{1024}T_0+\frac{3169}{6144}T_1+\frac{49}{768}T_2+\frac{65}{12288}T_3+\frac{1}{3072}T_4+\frac{1}{61440}T_5.$$

Ker iščemo polinom stopnje 4, odrežemo zadnji člen in dobimo

$$q_4=\frac{1089}{1024}T_0+\frac{3169}{6144}T_1+\frac{49}{768}T_2+\frac{65}{12288}T_3+\frac{1}{3072}T_4.$$

Ko polinom preslikamo nazaj na $[-1/2, 1/2]$ in ga spet izrazimo v standardni bazi, dobimo

$$p_4^{\text{econ}} = 1 + \frac{6143}{6144}x + \frac{1}{2}x^2 + \frac{65}{384}x^3 + \frac{1}{24}x^4.$$

Podobno, če bi odrezali še člen T_4, bi dobili polinom stopnje 3

$$p_3^{\text{econ}} = \frac{3071}{3072} + \frac{6143}{6144}x + \frac{49}{96}x^2 + \frac{65}{384}x^3.$$

Vemo, da velja $\|p_5 - p_4^{\text{econ}}\| \leq 1/61400 = 1.6 \cdot 10^{-5}$ oziroma $\|p_5 - p_3^{\text{econ}}\| \leq 1/61400 + 1/3072 = 3.4 \cdot 10^{-4}$. Od tod lahko dobimo skupno oceno $\|f - p_4^{\text{econ}}\| \leq 1.6 \cdot 10^{-5} + 3.6 \cdot 10^{-5} = 5.2 \cdot 10^{-5}$ oziroma $\|f - p_3^{\text{econ}}\| \leq 3.4 \cdot 10^{-4} + 3.6 \cdot 10^{-5} = 3.8 \cdot 10^{-4}$.

Če bi izračunali še polinoma najboljše enakomerne aproksimacije stopnje 3 in 4 bi dobili $\operatorname{dist}(f, \mathbb{P}_4) = 1.6 \cdot 10^{-5}$ in $\operatorname{dist}(f, \mathbb{P}_3) = 3.3 \cdot 10^{-4}$. Od tod lahko ugotovimo: $\|f - p_4^{\text{econ}}\| \leq 3.3 \cdot \operatorname{dist}(f, \mathbb{P}_4)$ in $\|f - p_3^{\text{econ}}\| \leq 1.2 \cdot \operatorname{dist}(f, \mathbb{P}_3)$, kar pomeni, da smo brez Remezovega postopka dobili zelo dobra približka za polinoma najboljše enakomerne aproksimacije. Polinom stopnje 3 je skoraj optimalen, pri polinomu stopnje 4 pa bi lahko dobili še boljši približek, če bi na začetku uporabili več členov razvoja v Taylorjevo vrsto. □

Iz samega postopka Čebiševe ekonomizacije tudi ugotovimo, da je namesto ekvidistantnih točk bolje za začetno delitev pri Remezovem postopku uporabiti točke, kjer polinom Čebiševa T_{n+1}, preslikan na $[a, b]$, alternirajoče doseže svoje ekstreme, to pa so

$$x_k = \frac{a+b}{2} + \frac{b-a}{2} \cos\left(\frac{k}{n+1}\right)$$

za $k = 0, \ldots, n+1$.

Dodatna literatura

V slovenščini lahko več podrobnosti o problemu najboljše enakomerne aproksimacije najdete v knjigi [18], o ekonomizaciji Čebiševa pa imate na voljo literaturo v angleščini, npr. [6].

Poglavje 5

Interpolacija

5.1 Uvod

Podane imamo vrednosti funkcije f v $n+1$ paroma različnih točkah $x_0, \ldots, x_n$, iščemo pa preprostejšo *interpolacijsko funkcijo* g, ki se v točkah x_i ujema s funkcijo f, torej $g(x_i) = f(x_i)$ za $i = 0, \ldots, n$. Interpolacijsko funkcijo potrebujemo npr. takrat, kadar imamo funkcijo f tabelirano, zanima pa nas vrednost funkcije v točki x, ki v tabeli ni vsebovana. Kot približek za vrednost $f(x)$ potem vzamemo vrednost interpolacijske funkcije $g(x)$. Preprost primer za interpolacijsko funkcijo g je npr. kosoma linearna funkcija, kjer vrednost v točki $x_i \leq x \leq x_{i+1}$ dobimo s konveksno kombinacijo vrednosti funkcije f v x_i in x_{i+1}, oziroma

$$g(x) = \frac{1}{x_{i+1} - x_i}\Big((x_{i+1} - x)f(x_i) + (x - x_i)f(x_{i+1})\Big).$$

Ponavadi za interpolacijsko funkcijo izberemo polinom. Poazali bomo, da obstaja natanko en polinom stopnje n ali manj, ki se v $n+1$ paroma različnih točkah $x_0, \ldots, x_n$ ujema s funkcijo f. Tak polinom imenujemo *interpolacijski polinom*. Skoraj nikoli ne iščemo zapisa interpolacijskega polinoma v standardni bazi. Največkrat zadošča, da znamo učinkovito izračunati njegovo vrednost v dani točki.

Interpolacijo so v preteklosti uporabljali večinoma za določanje vrednosti tabeliranih funkcij, danes pa se v glavnem uporablja kot orodje pri numeričnem odvajanju, integriranju in reševanju diferencialnih enačb.

Namesto polinomov lahko, odvisno od funkcije in potreb, uporabljamo tudi druge funkcije, npr. trigonometrične polinome, racionalne funkcije in zlepke. Polinomi imajo zelo lepe lastnosti, saj sta konstrukcija in računanje vrednosti enostavni, prav tako pa jih je zelo preprosto odvajati in integrirati.

5.2 Interpolacijski polinom

Denimo, da imamo paroma različne točke $x_0, \ldots, x_n$ in vrednosti $y_0, \ldots, y_n$. Iščemo polinom I_n stopnje n ali manj, za katerega velja $I_n(x_i) = y_i$ za $i = 0, \ldots, n$.

Naivna varianta je, da polinom iščemo v standarni bazi in za koeficiente polinoma

$$p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$$

nastavimo linearni sistem

$$\begin{array}{ccccccccccc} a_0 & + & a_1 x_0 & + & a_2 x_0^2 & + & \cdots & + & a_n x_0^n & = & y_0 \\ a_0 & + & a_1 x_1 & + & a_2 x_1^2 & + & \cdots & + & a_n x_1^n & = & y_1 \\ & \vdots & & \vdots & & \vdots & & \vdots & & \vdots & \\ a_0 & + & a_1 x_n & + & a_2 x_n^2 & + & \cdots & + & a_n x_n^n & = & y_n. \end{array}$$

Matrika zgornjega sistema je t.i. *Vandermondova matrika*[1]

$$V(x_0, x_1, \ldots, x_n) = \begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{bmatrix}.$$

Zanjo velja, da je v primeru paroma različnih točk nesingularna, saj se izkaže, da velja

$$\det(V(x_0, x_1, \ldots, x_n)) = \prod_{i>j} (x_i - x_j).$$

Izkaže se, da je računanje interpolacijskega polinoma preko reševanja zgornjega linearnega sistema z Vandermondovo matriko po eni strani zelo zamudno, po drugi strani pa je sistem lahko zelo slabo pogojen. Če npr. interpoliramo na intervalu $[0, 1]$ in vzamemo ekvidistantne točke $x_0 = 0, x_1 = 1/n, \ldots, x_n = 1$, potem ima Vandermondova matrika v primeru $n = 5$ pogojenostno število $4.9 \cdot 10^3$, pri $n = 10$ je $1.2 \cdot 10^8$, pri $n = 20$ pa že kar $9.7 \cdot 10^{16}$.

V resnici znamo interpolacijski polinom izračunati na ekonomičnejši način. To izkoriščajo tudi hitre metode za reševanje sistemov z Vandermondovimi matrikami, saj lahko reševanje sistemov te oblike prevedemo na iskanje koeficientov interpolacijskega polinoma.

Izrek 5.1 *Za paroma različne točke $x_0, \ldots, x_n$ in vrednosti $y_0, \ldots, y_n$ obstaja natanko en polinom I_n stopnje n ali manj, za katerega velja $I_n(x_i) = y_i$ za $i = 0, \ldots, n$.*

Dokaz. Eksistenco pokažemo s konstrukcijo. Za $i = 0, \ldots, n$ definirajmo polinome

$$L_{n,i}(x) = \frac{(x - x_0) \cdots (x - x_{i-1})(x - x_{i+1}) \cdots (x - x_n)}{(x_i - x_0) \cdots (x_i - x_{i-1})(x_i - x_{i+1}) \cdots (x_i - x_n)} = \prod_{\substack{k=0 \\ k \neq i}}^{n} \frac{x - x_k}{x_i - x_k},$$

ki so stopnje n, imenujemo pa jih *Lagrangeevi*[2] *koeficienti*. Če v definicijo po vrsti vstavimo točke $x_0, \ldots, x_n$, lahko hitro preverimo, da velja

$$L_{n,i}(x_j) = \delta_{ij} := \begin{cases} 1 & i = j \\ 0 & i \neq j. \end{cases}$$

[1] Imenovana je v čast francoskega matematika Alexandra-Théophila Vandermonda (1735—1796).

[2] Italijansko-francoski matematik in astronom Joseph Louis Lagrange (1736–1813) spada med najpomembnejše znanstvenike 18. stoletja. Med drugim je znan po razvoju variacijskega računa, Lagrangeevi mehaniki in metodi variacije konstant za reševanje diferencialnih enačb.

Vsak izmed Lagrangevih koeficientov $L_{n,i}$ ima tako v eni izmed točk $x_0, \ldots, x_n$ vrednost 1, v preostalih pa 0. Skupaj tvorijo bazo za vse polinome stopnje n ali manj. Če sedaj definiramo

$$I_n(x) = \sum_{k=0}^{n} y_k L_{n,k}(x),$$

je to očitno polinom stopnje n ali manj, prav tako pa velja $I_n(x_i) = y_i$ za $i = 0, \ldots, n$.

Pokažimo še enoličnost. Denimo, da imamo še en interpolacijski polinom $\widetilde{I}_n$ stopnje manjše ali enake n, za katerega prav tako velja $\widetilde{I}_n(x_i) = f(x_i)$ za $i = 0, \ldots, n$. Razlika $I_n - \widetilde{I}_n$ je potem tudi polinom stopnje manjše ali enake n, ki pa ima vsaj $n+1$ ničel v točkah $x_0, \ldots, x_n$. To pa je seveda možno le, če je polinom $I_n - \widetilde{I}_n$ identično enak 0 in prišli smo do protislovja. ■

Interpolacijski polinom I_n se s funkcijo f ujema v točkah $x_0, \ldots, x_n$, drugje pa ne nujno. Lahko pa ocenimo razliko, če je funkcija dovoljkrat zvezno odvedljiva. V oceni natopa polinom

$$\omega(x) = (x - x_0)(x - x_1) \cdots (x - x_n),$$

v katerem nastopajo vse interpolacijske točke.

Izrek 5.2 *Če je f $(n+1)$-krat zvezno odvedljiva funkcija na intervalu $[a, b]$, ki vsebuje vse paroma različne točke $x_0, \ldots, x_n$, potem za vsak $x \in [a, b]$ obstaja tak $\xi \in (a, b)$, da velja*

$$f(x) - I_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}\omega(x) \tag{5.1}$$

in $\min(x, x_0, \ldots, x_n) < \xi < \max(x, x_0, \ldots, x_n)$.

Dokaz. Pokazati moramo, da za vsak $x \in [a, b]$ obstaja ustrezen ξ. V primeru $x = x_i$ lahko vzamemo kar poljuben ξ, saj je v enakosti (5.1) na obeh straneh vrednost 0. Zato lahko predpostavimo, da velja $x \neq x_i$ za $i = 0, \ldots, n$, in definiramo funkcijo

$$F(z) := f(z) - I_n(z) - R \cdot \omega(z). \tag{5.2}$$

Funkcija F je očitno $(n+1)$-krat zvezno odvedljiva in velja $F(x_i) = 0$ za $i = 0, \ldots, n$. Za našo točko $x \in [a, b]$, za katero smo predpostavili, da je različna od $x_0, \ldots, x_n$, lahko konstanto R določimo tako, da bo $F(x) = 0$.

Pri tako izbrani konstanti R ima funkcija F vsaj $n+2$ različnih ničel na intervalu $[a, b]$. Po Rolleovem izreku ima zato prvi odvod F' vsaj $n+1$ različnih ničel na (a, b), drugi odvod F'' ima vsaj n ničel, ..., in končno, $(n+1)$-vi odvod $F^{(n+1)}$ ima vsaj eno ničlo na (a, b), to pa je ravno točka ξ, ki jo iščemo.

Iz enačbe (5.2) sledi $0 = F^{(n+1)}(\xi) = f^{(n+1)}(\xi) - R(n+1)!$, torej

$$R = \frac{f^{(n+1)}(\xi)}{(n+1)!}$$

in za izbrani x smo poiskali tak ξ, da velja (5.1). ■

Lagrangeeva oblika ni najprimernejša za konstrukcijo in računanje vrednosti interpolacijskega polinoma. Prva težava je veliko število operacij, druga težava pa je, da moramo že na začetku

določiti stopnjo polinoma. Boljša je npr. zaporedna linearna interpolacija, pa tudi deljene diference.

Označimo z $I_{i,i+1,\ldots,i+k}$ interpolacijski polinom stopnje manjše ali enake k za funkcijo f na točkah $x_i, x_{i+1}, \ldots, x_{i+k}$. Denimo, da poznamo interpolacijska polinoma $I_{i,i+1,\ldots,i+k-1}$ in $I_{i+1,\ldots,i+k}$. Množici točk, na katerih interpolirata funkcijo f, se ujemata v točkah $x_{i+1}, \ldots, x_{i+k-1}$ in razlikujeta v točkah x_i in x_{i+k}. Hitro se lahko prepričamo, da lahko polinom stopnje kvečjemu $k+1$, ki interpolira f v vseh točkah $x_i, \ldots, x_{i+k}$, zapišemo v obliki

$$I_{i,i+1,\ldots,i+k}(x) = \frac{I_{i,i+1,\ldots,i+k-1}(x)(x - x_{i+k}) - I_{i+1,i+2,\ldots,i+k}(x)(x - x_i)}{x_i - x_{i+k}} \tag{5.3}$$

oziroma

$$I_{i,i+1,\ldots,i+k}(x) = \frac{1}{x_{i+k} - x_i} \begin{vmatrix} x - x_i & I_{i,i+1,\ldots,i+k-1}(x) \\ x - x_{i+k} & I_{i+1,i+2,\ldots,i+k}(x) \end{vmatrix}.$$

Pri *metodi zaporednih linearnih interpolacij* začnemo z interpolacijskimi polinomi - konstantami, ki interpolirajo funkcijo f le v eni izmed točk $x_0, \ldots, x_n$, potem pa z uporabo formule (5.3) povečujemo stopnje interpolacijskih polinomov, dokler ne pridemo na koncu do polinoma, ki interpolira f v vseh točkah $x_0, \ldots, x_n$. Obstaja več podobnih postopkov, opisali pa bomo *Nevilleovo*[3] *shemo*, ki se jo da preprosto implementirati v računalniku. Predstavimo jo z naslednjo trikotno shemo za primer $n = 3$:

x_i	$x - x_i$	y_i	$I_{..}(x)$	$I_{...}(x)$	$I_{....}(x)$
x_0	$x - x_0$	y_0			
			$I_{01}(x)$		
x_1	$x - x_1$	y_1		$I_{012}(x)$	
			$I_{12}(x)$		$I_{0123}(x)$
x_2	$x - x_2$	y_2		$I_{123}(x)$	
			$I_{23}(x)$		
x_3	$x - x_3$	y_3			

Zgled 5.1 *Poiskati je potrebno vrednost interpolacijskega polinoma za podatke* $x : 0, 2, 4$ *in* $f(x) : 2, 4, 8$ *v točki* $x = 1$. *Uporabimo Nevilleovo shemo:*

x_i	$x - x_i$	y_i	$I_{..}(x)$	$I_{...}(x)$
$x_0 = 0$	$x - x_0 = 1$	$y_0 = 2$		
			$I_{01}(x) = 3$	
$x_1 = 2$	$x - x_1 = -1$	$y_1 = 4$		$I_{012}(x) = \frac{11}{4}$
			$I_{12}(x) = 2$	
$x_2 = 4$	$x - x_2 = -3$	$y_2 = 8$		

[3] Angleški matematik Eric Harold Neville (1889–1961).

Do vrednosti smo prišli z naslednjimi računi:

$$\begin{aligned} I_{01}(x) &= \tfrac{1}{x_1-x_0}\begin{vmatrix} x-x_0 & I_0(x) \\ x-x_1 & I_1(x) \end{vmatrix} = \tfrac{1}{2}\begin{vmatrix} 1 & 2 \\ -1 & 4 \end{vmatrix} = 3, \\ I_{12}(x) &= \tfrac{1}{x_2-x_1}\begin{vmatrix} x-x_1 & I_1(x) \\ x-x_2 & I_2(x) \end{vmatrix} = \tfrac{1}{2}\begin{vmatrix} -1 & 4 \\ -3 & 8 \end{vmatrix} = 2, \\ I_{012}(x) &= \tfrac{1}{x_2-x_0}\begin{vmatrix} x-x_0 & I_{01}(x) \\ x-x_2 & I_{12}(x) \end{vmatrix} = \tfrac{1}{4}\begin{vmatrix} 1 & 3 \\ -3 & 2 \end{vmatrix} = \tfrac{11}{4}. \end{aligned}$$ □

5.3 Deljene diference

Še enostavnejši zapis interpolacijskega polinoma nam omogočajo deljene diference. Kot bomo videli v nadaljevanju, lahko z njimi posplošimo interpolacijo tudi na primere, ko interpolacijske točke niso paroma različne. Večkratna točka potem pomeni, da se polinom in funkcija ujemata ne samo v vrednosti, temveč še v ustreznem številu odvodov. Za začetek definirajmo deljeno diferenco na paroma različnih točkah.

Definicija 5.3 *Deljena diferenca $f[x_0, x_1, \ldots, x_k]$ je vodilni koeficient (pri x^k) interpolacijskega polinoma stopnje k, ki se ujema s funkcijo f v paroma različnih točkah $x_0, x_1, \ldots, x_k$.*

Izrek 5.4 *Za deljene diference velja:*

a)

$$P_n(x) = f[x_0] + f[x_0, x_1](x - x_0) + \cdots + f[x_0, x_1, \ldots, x_n](x - x_0)\cdots(x - x_{n-1}) \tag{5.4}$$

je interpolacijski polinom, ki se v točkah $x_0, \ldots, x_n$ ujema s funkcijo f.

b) $f[x_0, x_1, \ldots, x_k]$ je simetrična funkcija svojih argumentov.

c) $(\alpha f + \beta g)[x_0, \ldots, x_k] = \alpha f[x_0, \ldots, x_k] + \beta g[x_0, \ldots, x_k]$.

d) Velja rekurzivna formula

$$f[x_0, \ldots, x_k] = \frac{f[x_1, \ldots, x_k] - f[x_0, \ldots, x_{k-1}]}{x_k - x_0}. \tag{5.5}$$

Dokaz.

a) Uporabimo indukcijo. Pri $n = 0$ se očitno $f[x_0]$ ujema z $f(x_0)$. Naj bo sedaj

$$P_i(x) = f[x_0] + f[x_0, x_1](x - x_0) + \cdots + f[x_0, x_1, \ldots, x_i](x - x_0)\cdots(x - x_{i-1})$$

interpolacijski polinom na točkah $x_0, \ldots, x_i$. Hitro se lahko prepričamo, da obstaja taka konstanta c, da lahko polinom, ki interpolira funkcijo f v točkah $x_0, \ldots, x_{i+1}$ zapišemo v obliki

$$P_{i+1}(x) = P_i(x) + c(x - x_0)\cdots(x - x_i).$$

Za $k = 0, \ldots, i$ namreč neodvisno od c velja $P_{i+1}(x_k) = P_i(x_k) = f(x_k)$. Od tod sledi, da lahko c določimo tako, da bo $P_{i+1}(x_{i+1}) = f(x_{i+1})$. Ker pa je c vodilni koeficient polinoma P_{i+1}, je po definiciji deljene diference $c = f[x_0, \ldots, x_{i+1}]$.

b), c) Očitno.

d) Naj bo p_0 interpolacijski polinom, ki se ujema z f v točkah $x_0, \ldots, x_{k-1}$ in p_1 polinom, ki se ujema z f v točkah $x_1, \ldots, x_k$. Interpolacijski polinom p, ki se z f ujema v vseh točkah $x_0, \ldots, x_k$, ima po formuli (5.3) obliko

$$p(x) = \frac{x - x_k}{x_0 - x_k} p_0(x) + \frac{x - x_0}{x_k - x_0} p_1(x).$$

S primerjanjem vodilnih koeficientov polinomov p, p_0 in p_1 pridemo do zveze (5.5). ■

Obliko (5.4) imenujemo *Newtonov interpolacijski polinom*.

Za deljeno diferenco v eni točki smo že ugotovili, da velja $f[x_0] = f(x_0)$. Za dve točki dobimo po formuli (5.5) deljeno diferenco

$$f[x_0, x_1] = \frac{f(x_1) - f(x_0)}{x_1 - x_0},$$

kar se ujema s smernim koeficientom premice, ki gre skozi točki $(x_0, f(x_0))$ in $(x_, f(x_1))$.

Kaj se zgodi, če vzamemo interpolacijski polinom na dveh točkah in v limiti pošljemo eno točko proti drugi? Iz sekante

$$p(x) = f[x_0] + f[x_0, x_1](x - x_0) = f(x_0) + \frac{f(x_1) - f(x_0)}{x_1 - x_0}(x - x_0)$$

v limiti, ko gre x_1 proti x_0, dobimo tangento

$$p(x) = f(x_0) + f'(x_0)(x - x_0).$$

To pomeni, da lahko definiciji interpolacijskega polinoma in deljene diference razširimo na primere, ko se točke ponavljajo. Mislimo si, da so bile na začetku vse točke paroma različne, potem pa jih v limiti po potrebi združimo. Večkratnost interpolacijske točke pomeni, v koliko odvodih naj se interpolacijski polinom ujema s funkcijo. Tako npr. pri točkah $x_1, x_2, x_2, x_2, x_3, x_3$ iščemo polinom p stopnje manjše ali enake 5, za katerega velja $p(x_1) = f(x_1)$, $p(x_2) = f(x_2)$, $p'(x_2) = f'(x_2)$, $p''(x_2) = f''(x_2)$, $p(x_3) = f(x_3)$ in $p'(x_3) = f'(x_3)$.

Če dopuščamo tudi ponavljanje točk, potem za deljene diference velja rekurzivna zveza

$$f[x_0, x_1, \ldots, x_k] = \begin{cases} \dfrac{f^k(x_0)}{k!}, & x_0 = x_1 = \cdots = x_k, \\ \dfrac{f[x_1, \ldots, x_k] - f[x_0, \ldots, x_{k-1}]}{x_k - x_0}, & \text{sicer.} \end{cases}$$

Če je funkcija f v točki x_0 k-krat zvezno odvedljiva, potem iz interpolacijskega polinoma

$$p(x) = f[x_0] + f[x_0, x_1](x - x_0) + \cdots + f[x_0, x_1, \ldots, x_k](x - x_0) \cdots (x - x_{k-1})$$

v limiti, ko pošljemo vse točke proti x_0, dobimo Taylorjev polinom razvoja f okoli točke x_0:

$$T_k(x) = f(x_0) + \frac{f'(x_0)}{1!}(x - x_0) + \cdots + \frac{f^{(k)}(x_0)}{k!}(x - x_0)^k.$$

Deljene diference računamo v trikotni shemi iz katere na koncu lahko hitro preberemo enačbo polinoma oz. izračunamo njegovo vrednost v iskani točki. Konstrukcija je razvidna iz naslednjega zgleda.

Zgled 5.2 *Zapiši enačbo polinoma stopnje kvečjemu 5, za katerega velja:* $p(0)=1$, $p'(0)=2$, $p''(0)=3$, $p(1)=-1$, $p'(1)=3$ *in* $p(2)=4$.

x_i	$f[.]$	$f[.,.]$	$f[.,.,.]$	$f[.,.,.,.]$	$f[.,.,.,.,.]$	$f[.,.,.,.,.,.]$
0	1					
		2				
0	1		$\frac{3}{2}$			
		2		$-\frac{11}{2}$		
0	1		-4		$\frac{29}{2}$	
		-2		9		$-\frac{79}{8}$
1	-1		5		$-\frac{21}{4}$	
		3		$-\frac{3}{2}$		
1	-1		2			
		5				
2	4					

Vrednosti v tabeli, ki so označene z rdečo, dobimo direktno iz začetnih podatkih, saj so v teh deljenih diferencah vse točke enake. Ostale vrednosti izračunamo po rekurzivni zvezi. Tako npr. vrednost $f[0,0,0]=3/2$ *dobimo iz formule* $f[0,0,0]=f''(0)/2$, *vrednost* $f[0,0,0,1]=29/2$ *pa izračunamo iz* $f[0,0,0,1]=(f[0,0,1]-f[0,0,0])/(1-0)$.

Koeficiente interpolacijskega polinoma preberemo iz zgornje diagonale in dobimo

$$p(x)=1+2x+\frac{3}{2}x^2-\frac{11}{2}x^3+\frac{29}{2}x^3(x-1)-\frac{79}{8}x^3(x-1)^2.$$

Namesto tega bi lahko koeficiente vzeli tudi iz spodnje diagonale. V tem primeru interpolacijske točke nastopajo v obratnem vrstnem redu. Interpolacijski polinom je seveda enak kot prej, saj je enoličen, le zapisan je v drugačni bazi. Če vzamemo spodnjo diagonalo, dobimo

$$p(x)=4+5(x-2)+2(x-2)(x-1)-\frac{3}{2}(x-2)(x-1)^2$$
$$-\frac{21}{4}(x-2)(x-1)x-\frac{79}{8}(x-2)(x-1)^2x^2.$$ □

Izrek 5.5 (Hermite–Genocchijeva formula) [4] *Za k-krat zvezno odvedljivo funkcijo f velja*

$$f[x_0,\ldots,x_k]=\int_0^1 dt_1\int_0^{t_1}dt_2\int\cdots\int_0^{t_{k-1}}f^{(k)}\Big(t_k(x_k-x_{k-1})+\cdots+t_1(x_1-x_0)+x_0\Big)dt_k. \quad (5.6)$$

Dokaz. V primeru, ko so vse točke enake, lahko integral (5.6) direktno izračunamo in je enak

$$\int_0^1 dt_1\int_0^{t_1}dt_2\int\cdots\int_0^{t_{k-1}}f^{(k)}(x_0)dt_k=f^{(k)}(x_0)\int_0^1 dt_1\int_0^{t_1}dt_2\int\cdots\int_0^{t_{k-1}}dt_k=\frac{1}{k!}f^{(k)}(x_0).$$

V glavnem delu dokaza lahko tako predpostavimo, da se vsaj dve izmed točk $x_0,\ldots,x_k$ razlikujeta. Veljavnost formule preverimo z indukcijo. V primeru $k=1$, pri predpostavki $x_0\neq x_1$, res dobimo

$$\int_0^1 f'(t_1(x_1-x_0)+x_0)dt_1=\frac{1}{x_1-x_0}f(t_1(x_1-x_0)+x_0)\Big|_{t_1=0}^{t_1=1}=\frac{f(x_1)-f(x_0)}{x_1-x_0}=f[x_0,x_1].$$

[4] Italijanski matematik Angelo Genocchi (1817–1889).

Sedaj predpostavimo, da izrek velja, če imamo k ali manj točk in ga poskusimo dokazati za primer $k+1$ točk. Če definiramo novo spremenljivko $\xi = t_k(x_k - x_{k-1}) + \cdots + t_1(x_1 - x_0) + x_0$, potem lahko zapišemo integral (5.6) kot

$$\int_0^1 dt_1 \int_0^{t_1} dt_2 \int \cdots \int_0^{t_{k-1}} f^{(k)}(\xi) dt_k.$$

S substitucijo spremenljivk dobimo

$$d\xi = (x_k - x_{k-1}) dt_k$$

in

$$\int_0^{t_{k-1}} f^{(k)}(\xi) dt_k = \frac{1}{x_k - x_{k-1}} \int_{\xi_0}^{\xi_1} f^{(k)}(\xi) d\xi = \frac{f^{(k-1)}(\xi_1) - f^{(k-1)}(\xi_0)}{x_k - x_{k-1}},$$

kjer sta meji enaki (ξ_0 ustreza $t_k = 0$, ξ_1 pa $t_k = t_{k-1}$)

$$\begin{aligned} \xi_0 &= t_{k-1}(x_{k-1} - x_{k-2}) + t_{k-2}(x_{k-2} - x_{k-3}) + \cdots + t_1(x_1 - x_0) + x_0, \\ \xi_1 &= t_{k-1}(x_k - x_{k-2}) + t_{k-2}(x_{k-2} - x_{k-3}) + \cdots + t_1(x_1 - x_0) + x_0. \end{aligned}$$

Po indukcijski predpostavki sledi

$$\int_0^1 dt_1 \int_0^{t_1} dt_2 \int \cdots \int_0^{t_{k-2}} f^{(k-1)}(\xi_1) dt_{k-1} = f[x_0, x_1, \ldots, x_{k-2}, x_k]$$

in

$$\int_0^1 dt_1 \int_0^{t_1} dt_2 \int \cdots \int_0^{t_{k-2}} f^{(k-1)}(\xi_0) dt_{k-1} = f[x_0, x_1, \ldots, x_{k-2}, x_{k-1}],$$

ker pa po rekurzivni zvezi (5.5) velja

$$f[x_0, x_1, \ldots, x_k] = \frac{f[x_0, x_1, \ldots, x_{k-2}, x_k] - f[x_0, x_1, \ldots, x_{k-2}, x_{k-1}]}{x_k - x_{k-1}},$$

je dokaz končan. ■

Naslednjo zvezo dobimo tako, da za integral (5.7) uporabimo izrek o povprečni vrednosti.

Posledica 5.6 *Za k-krat zvezno odvedljivo funkcijo f velja*

$$f[x_0, \ldots, x_k] = \frac{1}{k!} f^{(k)}(\xi),$$

kjer je

$$\min_{i=0,\ldots,k}(x_i) \le \xi \le \max_{i=0,\ldots,k}(x_i).$$

Izrek 5.7 *Za $(n+1)$-krat zvezno odvedljivo funkcijo f in interpolacijski polinom I_n na točkah $x_0, \ldots, x_n$ velja*

$$f(x) = I_n(x) + f[x_0, \ldots, x_n, x](x - x_0) \cdots (x - x_n). \tag{5.7}$$

Dokaz. Naj polinom q interpolira funkcijo f v točkah $x_0, \ldots, x_n, t$. V Newtonovi obliki lahko q zapišemo kot

$$q(x) = I_n(x) + f[x_0, \ldots, x_n, t](x - x_0) \cdots (x - x_n).$$

Očitno pri izbranem t potem velja $q(t) = I_n(t) + f[x_0, \ldots, x_n, t](t - x_0) \cdots (t - x_n)$. Ker to velja za vsak t, dobimo formulo (5.7), ki jo je bilo potrebno dokazati. ■

Posledica 5.8 *Za $(n+1)$-krat zvezno odvedljivo funkcijo f in interpolacijski polinom I_n na točkah $x_0, \ldots, x_n$ velja*

$$f(x) - I_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}\omega(x),$$

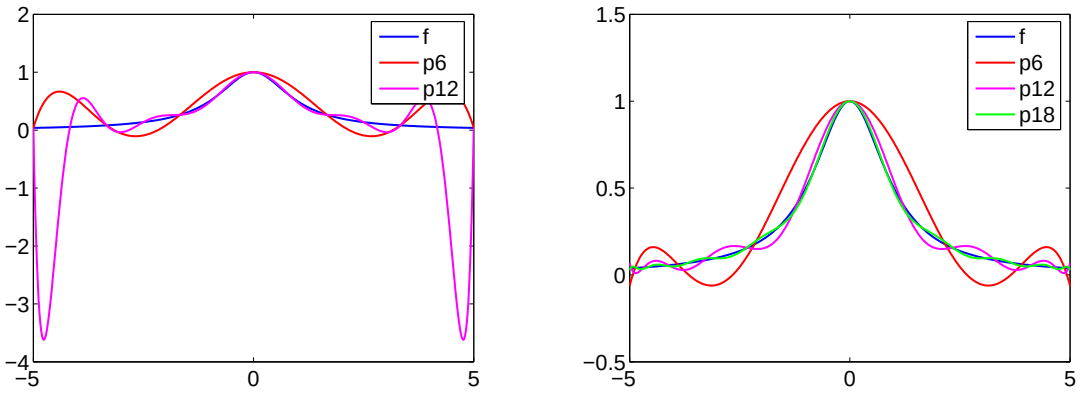
kjer je $\min(x, x_0, \ldots, x_n) \leq \xi \leq \max(x, x_0, \ldots, x_n)$. ■

Zgornja ocena se ujema z oceno, ki smo jo zapisali v izreku 5.2 za paroma različne točke. Prednost nove ocene je, da pride v poštev tudi, kadar vse točke niso medsebojno različne.

5.4 Interpolacija s kosoma polinomskimi funkcijami

Če želimo, da bi se na danem intervalu interpolacijski polinom čim bolj prilegal izbrani funkciji, je prva ideja ta, da povečamo stopnjo interpolacijskega polinoma. To naredimo tako, da povečamo število interpolacijskih točk. Kot kaže naslednji zgled, ki je znan pod imenom *Rungejev*[5] *protiprimer*, ni nujno, da večanje stopnje interpolacijskega polinoma res izboljša aproksimacijo funkcije s polinomom.

Zgled 5.3 *Denimo, da interpoliramo funkcijo $f(x) = 1/(1+x^2)$ na intervalu $[-5, 5]$. Če uporabimo ekvidistantne točke, potem z večanjem stopnje interpolacijski polinom vedno slabše aproksimira f, kar kaže slika 5.1 (leva slika).*



Slika 5.1: Rungejev protiprimer. Če funkcijo $f(x) = 1/(1+x^2)$ na intervalu $[-5, 5]$ interpoliramo v ekvidistantnih točkah (levo), razlika med funkcijo in interpolacijskim polinomom z večanjem n narašča. Če pa interpoliramo na Čebiševih točkah (desno), potem z večanjem števila točk interpolacijski polinom vedno bolje aproksimira f.

Težava se v tem primeru pojavi zaradi ekvidistantnih točk. Če za interpolacijske točke vzamemo Čebiševe točke, ki so v tem primeru definirane kot $x_j = 5\cos(j\pi/n)$ za $j = 0, \ldots, n$, potem z večanjem stopnje interpolacijski polinom vedno bolje aproksimira funkcijo f, kar se vidi na sliki 5.1 (desna slika). □

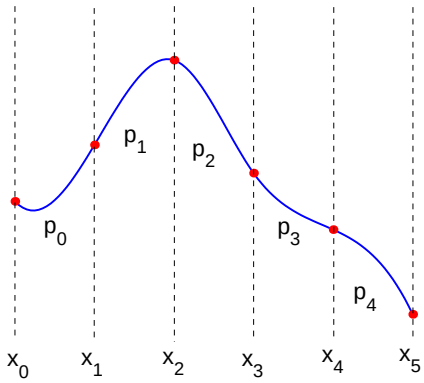
[5] Nemški matematik Carl David Tolmé Runge (1856—1927) je znan predvsem po Runge–Kutta metodi za numerično reševanje diferencialnih enačb.

Tu je potrebno še omeniti, da za razliko med interpolacijo z ekvidistantnimi in Čebiševimi točkami v zadnjem zgledu niso odgovorne zaokrožitvene napake oz. numerično računanje nasploh, saj pride do razlik pri eksaktnem računanju.

Ponavadi se Čebiševe točke sicer res obnašajo bolje kot ekvidistantne točke, a je dokazano, da za vsako zaporedje predpisanih interpolacijskih točk obstaja funkcija, pri kateri se zgodi, da z večanjem stopnje interpolacijskega polinoma prihaja do vse večjih razlik med to funkcijo in interpolacijskim polinomom.

To ni v nasprotju z Weierstrassovim izrekom iz prejšnjega poglavja. Ta pravi, da je možno zvezno funkcijo poljubno dobro aproksimirati s polinomom dovolj visoke stopnje, a pri tem ni predpisano, v katerih točkah se mora polinom ujemati s funkcijo. Če za te točke predpišemo, da morajo biti npr. ekvidistantne, potem izreka seveda ne moremo več uporabiti.

Rungejev protiprimer kaže, da večanje stopnje interpolacijskega polinoma ni nujno dober način za boljšo aproksimacijo funkcije. Rešitev je, da sicer povečamo število interpolacijskih točk, a izberemo drugačno vrsto interpolacijske funkcije. Namesto enega polinoma visoke stopnje, interpolacijsko funkcijo sestavimo (zlepimo) iz polinomov nizkih stopenj. Tako dobimo kosoma polinomsko funkcijo oziroma *zlepek*. Ker imamo še vedno opravka s polinomi, sta konstrukcija zlepka in izračun vrednosti še vedno preprosta. Primer zlepka je prikazan na sliki 5.2.



Slika 5.2: Zgled polinomskega zlepka. Na vsakem intervalu $[x_i, x_{i+1}]$ imamo definiran polinom p_i za $i = 0, \ldots, 4$, skupaj pa sestavljajo zvezno funkcijo na intervalu $[x_0, x_5]$.

Denimo, da iščemo interpolacijsko funkcijo, ki bo v točkah $x_0, \ldots, x_{n+1}$ po vrsti imela vrednosti $y_0, \ldots, y_{n+1}$. Pri polinomskem zlepku interval $[x_0, x_{n+1}]$ razdelimo na podintervale $[x_i, x_{i+1}]$ za $i = 0, \ldots, n$. Na vsakem intervalu $[x_i, x_{i+1}]$ vzamemo polinom p_i nizke stopnje, za katerega velja $p_i(x_i) = y_i$ in $p_i(x_{i+1}) = y_{i+1}$. Želimo, da bo sestavljena krivulja čim bolj gladka in da bo čim bolje opisovala obliko začetnih podatkov. Ker gre krivulja čez točke (x_i, y_i) za $i = 1, \ldots, n$, je očitno najmanj zvezna, saj v notranjih točkah velja

$$p_i(x_{i+1}) = p_{i+1}(x_{i+1}) = y_{i+1}.$$

Osnovna varianta je *kosoma linearna interpolacija*, kjer na vsakem intervalu $[x_i, x_{i+1}]$ podatke interpoliramo z linearno funkcijo

$$p_i(x) = y_i + \frac{y_{i+1} - y_i}{x_{i+1} - x_i}(x - x_i).$$

Dobljena kosoma linearna funkcija je očitno zvezna, ni pa zvezno odvedljiva.

Če želimo, da bi bil zlepek tudi zvezno odvedljiv, moramo povečati stopnjo polinomov. Naslednji na vrsti so kvadratni zlepki, kjer sicer lahko dosežemo, da bo zlepek zvezno odvedljiv, a so preobčutljivi na motnje v začetnih podatkih. Izkaže se, da so najbolj primerni naslednji na vrsti, torej *kubični zlepki*, kjer je p_i kubičen polinom na $[x_i, x_{i+1}]$. Za kubični interpolacijski zlepek ponavadi zahtevamo še, da je vsaj zvezno odvedljiv.

Posamezni kubični polinom p_i je natanko določen z vrednostmi in odvodi v krajiščih:

$$p_i(x_i) = y_i, \qquad p_i(x_{i+1}) = y_{i+1}$$

$$p_i'(x_i) = d_i, \qquad p_i'(x_{i+1}) = d_{i+1}.$$

Če označimo $h_i = x_{i+1} - x_i$, $\delta_i = (y_{i+1} - y_i)/h_i$, in uporabimo deljene diference, dobimo

$$\begin{array}{c|cccc}
x_i & y_i & & & \\
 & & d_i & & \\
x_i & y_i & & \frac{\delta_i - d_i}{h_i} & \\
 & & \delta_i & & \frac{d_i + d_{i+1} - 2\delta_i}{h_i^2} \\
 & & & \frac{d_{i+1} - \delta_i}{h_i} & \\
x_{i+1} & y_{i+1} & & & \\
 & & d_{i+1} & & \\
x_{i+1} & y_{i+1} & & &
\end{array}$$

in lahko zapišemo polinom p_i v obliki

$$p_i(x) = y_i + d_i(x - x_i) + \frac{\delta_i - d_i}{h_i}(x - x_i)^2 + \frac{d_i + d_{i+1} - 2\delta_i}{h_i^2}(x - x_i)^2(x - x_{i+1}).$$

Kubični zlepek je tako določen s parametri $d_0, \ldots, d_{n+1}$, ki so zaenkrat še prosti, določiti pa jih moramo tako, da bo imel zlepek željene lastnosti. Obstaja veliko možnih izbir, najpogostejše pa so:

- Pri *Hermiteovem kubičnem zlepku* izberemo kar $d_i = y_i'$, kjer je y_i' odvod funkcije, ki jo interpoliramo, v točki x_i za $i = 0, \ldots, n+1$. Na ta način se zlepek in funkcija v interpolacijskih točkah ujemata ne samo v vrednostih temveč tudi v prvih odvodih.
- Pri *dvakrat zvezno odvedljivem kubičnem zlepku* parametre $d_0, \ldots, d_{n+1}$ določimo tako, da je zlepek dvakrat zvezno odvedljiv. S kratkim računom lahko preverimo, da za druge odvode v interpolacijskih točkah velja

  $$p_i''(x_{i+1}) = \frac{1}{h_i}(2d_i + 4d_{i+1} - 6\delta_i)$$

  in

  $$p_{i+1}''(x_{i+1}) = \frac{1}{h_{i+1}}(-4d_{i+1} - 2d_{i+2} + 6\delta_{i+1}).$$

  Da bo zlepek dvakrat zvezno odvedljiv mora torej veljati

  $$p_i''(x_{i+1}) = p_{i+1}''(x_{i+1})$$

za $i = 0, \ldots, n-1$. Tako dobimo sistem n linearnih enačb za $n+2$ parametrov:

$$h_{i+1}d_i + 2(h_i + h_{i+1})d_{i+1} + h_i d_{i+2} = 3h_{i+1}\delta_i + 3h_i\delta_i.$$

Enačb je manj kot parametrov, kar pomeni, da imamo v splošni rešitvi še dve prostostni stopnji. Najbolj pogoste rešitve, kako izbrati še dve manjkajoči enačbi, da dobimo enolično rešitev, so:

- *Kompletni zlepek*: če poznamo vrednosti odvodov funkcije, ki jo interpoliramo, v začetni in končni točki, potem lahko vzamemo $d_0 = y'_0$ in $d_{n+1} = y'_{n+1}$.
- *Naravni zlepek*: s pogojema $p''_0(x_0) = 0$ in $p''_n(x_{n+1}) = 0$ dosežemo, da ima zlepek v začetni in končni točki prevoj.
- *Zlepek brez vozlov*: zahtevamo, da je zlepek na $[x_0, x_2]$ in $[x_{n-1}, x_{n+1}]$ kubični polinom oziroma, da velja $p'''_0(x_1) = p'''_1(x_1)$ in $p'''_{n-1}(x_n) = p'''_n(x_n)$. Tako se v bistvu znebimo vozlov x_1 in x_n.

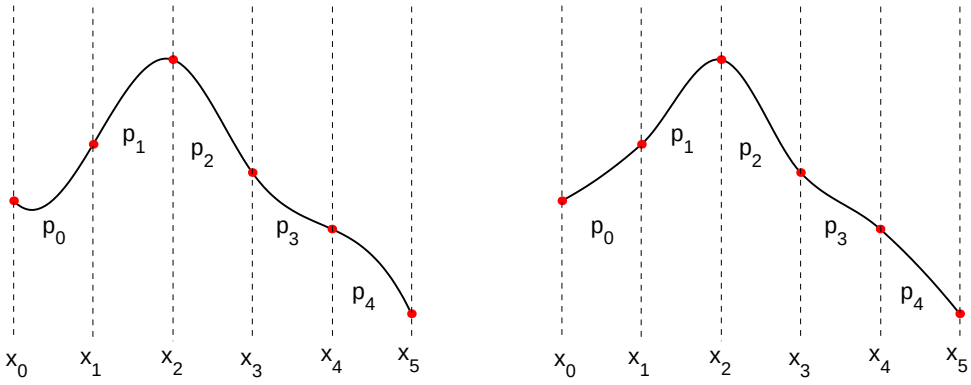
V vseh treh primerih moramo na koncu rešiti linearni sistem, iz katerega dobimo vrednosti parametrov $d_0, \ldots, d_n$. Ker je sistem tridiagonalen, lahko to naredimo zelo ekonomično v časovni zahtevnosti $\mathcal{O}(n)$.

- Pri *kubičnem zlepku, ki ohranja obliko*, naklone d_i določimo tako, da za $1/d_i$ vzamemo povprečje recipročnih vrednosti smernih koeficientov premic skozi (x_{i-1}, x_i) in (x_i, x_{i+1}). Če sta smerna koeficienta δ_i in δ_{i+1} enako predznačena, potem pri predpostavki $h_i = h_{i+1}$ dobimo

$$\frac{1}{d_i} = \frac{1}{2}\left(\frac{1}{\delta_i} + \frac{1}{\delta_{i+1}}\right).$$

Če sta širini intervalov $[x_{i-1}, x_i]$ in $[x_i, x_{i+1}]$ različni, je potrebno formulo za d_i popraviti. V primeru, ko sta smerna koeficienta δ_i in δ_{i+1} nasprotno predznačena, vzamemo $d_i = 0$.

Dobljeni kubični zlepek je le enkrat zvezno odvedljiv, preskok v drugem odvodu je lepo razviden na sliki 5.3. Na levi strani je dvakrat zvezno odvedljiv (naravni) zlepek, na desni strani pa zlepek, ki ohranja obliko. Ohranjanje oblike pomeni, da na vsakem intervalu $[x_i, x_{i+1}]$ vrednosti zlepka ležijo med $f(x_i)$ in $f(x_{i+1})$. Da to pri dvakrat zvezno odvedljivem zlepku ni nujno res, se vidi na sliki 5.3 na intervalu $[x_0, x_1]$.



Slika 5.3: Primerjava kubičnega zlepka, ki ohranja obliko (desno), z dvakrat zvezno odvedljivim naravnim kubičnim zlepkom (levo).

Pokažimo, da v primeru Hermiteovega kubičnega zlepka res velja, da, z večanjem števila interpolacijskih točk, zlepek vedno bolj aproksimira funkcijo, ki jo interpolira. Pri večanju števila točk moramo seveda paziti na to, da se manjša maksimalna širina podintervalov

$$h := \max_{i=0,\ldots,n} h_i.$$

Izrek 5.9 *Naj bo f štirikrat zvezno odvedljiva funkcija na intervalu $[a,b]$. Če interval razdelimo s točkami $a = x_0 < x_1 < \ldots < x_n < x_{n+1} = b$, potem za Hermiteov kubični zlepek p za vsak $x \in [a,b]$ velja ocena*

$$|f(x) - p(x)| \leq \frac{h^4}{384} M_4, \tag{5.8}$$

kjer je $h = \max_{i=0,\ldots,n}(x_{i+1} - x_i)$ in $M_4 = \max_{x\in[a,b]} |f^{(4)}(x)|$.

Dokaz. Na podintervalu $[x_i, x_{i+1}]$ funkcijo f interpoliramo s kubičnim polinomom p_i, za katerega velja $p_i(x_i) = f(x_i)$, $p_i'(x_i) = f(x_i)$, $p_i(x_{i+1}) = f(x_{i+1})$ in $p_i(x_{i+1}) = f(x_{i+1})$. Za razliko vemo, da velja

$$f(x) - p_i(x) = f[x_i, x_i, x_{i+1}, x_{i+1}, x](x - x_i)^2(x - x_{i+1})^2.$$

Če je $h_i = x_{i+1} - x_i$, lahko ocenimo

$$|f(x) - p_i(x)| \leq \frac{M_4}{4!} \cdot \frac{h_i^4}{16}.$$

Od tod sledi ocena (5.8). ∎

Podobne ocene se da izpeljati tudi za ostale zlepke. Tako npr. v [18] lahko najdete izpeljavo, da za kompletni kubični zlepek p in štirikrat zvezno odvedljivo funkcijo f pri predpostavkah izreka 5.9 veljajo ocene

$$\begin{aligned} |f(x) - p(x)| &\leq \frac{5h^4}{384} M_4, \\ |f'(x) - p'(x)| &\leq \frac{h^3}{24} M_4, \\ |f''(x) - p''(x)| &\leq \frac{3h^2}{8} M_4. \end{aligned}$$

Ne samo, da kompletni kubični zlepek dobro aproksimira funkcijo f, iz zadnjih dveh ocen sledi, da tudi prvi in drugi odvod zlepka dobro aproksimirata prvi in drugi odvod funkcije f.

5.5 Beziérove krivulje

Pri interpolaciji v ravnini imamo dane točke (x_i, y_i) za $i = 0, \ldots, n$, kjer x in y koordinate niso nujno paroma različne oziroma monotono urejene, iščemo pa krivuljo, ki bi šla skozi te točke v navedenem vrstnem redu. Ena izmed možnih rešitev je, da izberemo parametre $t_0 < t_1 < \ldots < t_n$, potem pa poiščemo interpolacijska polinoma p_x in p_y stopnje manjše ali enake n, za katera velja $p_x(t_i) = x_i$ in $p_y(t_i) = y_i$ za $i = 0, \ldots, n$. Polinomska krivulja

$p(t) = (p_x(t), p_y(t))$ gre potem skozi vse izbrane točke. Očitno imamo tu veliko svobode, saj lahko vrednosti parametrov $t_0 < \ldots < t_n$ izberemo poljubno, vsaka izbira pa vpliva na obliko interpolacijske krivulje.

Izkaže se, da boljše lastnosti dobimo, če tudi tu uporabimo zlepke in skupaj sestavljamo polinomske odseke nižjih stopenj med dvema zaporednima točkama.

Pri tem so pomembno orodje *Beziérove krivulje*[6]. Vsaka Beziérova krivulja je določena s kontrolnimi točkami $p_i = (x_i, y_i)$ za $i = 0, \ldots, n$. Formula za točke na krivulji je

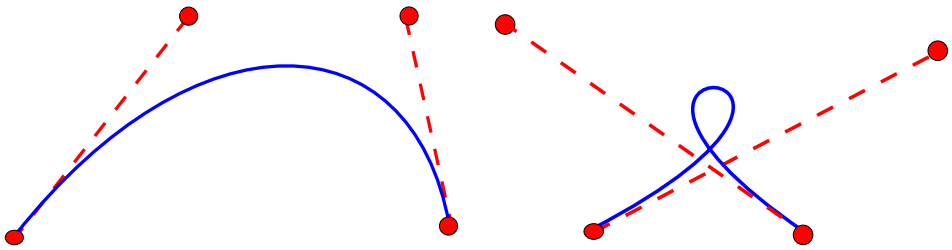
$$P(t) = \sum_{k=0}^{n} \binom{n}{k} t^k (1-t)^{n-k} p_k,$$

kjer je $t \in [0, 1]$. V formuli nastopajo *Bernsteinovi*[7] *polinomi*

$$B_{n,k}(t) = \binom{n}{k} t^k (1-t)^{n-k}$$

za $k = 0, \ldots, n$, za katere iz $(t + (1-t))^n = 1$ sledi, da velja $\sum_{k=0}^{n} B_{n,k}(t) = 1$ za vsak $t \in [0, 1]$. Izkaže se tudi (glej [18] za podrobnosti), da Bernsteinovi polinomi $B_{n,0}, \ldots, B_{n,n}$ tvorijo bazo za polinome stopnje manjše ali enake n.

Najpogosteje se uporabljajo kubične Beziérove krivulje. Dva zgleda sta predstavljena na sliki 5.4.



Slika 5.4: Kubični Beziérovi krivulji.

Nekaj lastnosti Beziérovih krivulj je:

- $P(0) = p_0$ in $P(1) = p_n$.
- Krivulja leži znotraj konveksne ogrinjače točk $p_0, \ldots, p_n$.
- Naklon pri $t = 0$ je enak naklonu premice skozi p_0 in p_1, podobno se naklon pri $t = 1$ ujema z naklonom premice skozi p_{n-1} in p_n.

Pri interpolaciji v ravini sestavljamo skupaj kubične Beziérove krivulje v (Beziérjev) zlepek. Pogoj za geometrijsko zveznost dobljenega zlepka je, da zadnji dve točki prve krivulje in prvi dve točki druge krivulje ležijo na isti premici.

[6] Neodvisno sta jih odkrila francoski matematik Paul de Casteljau (r. 1930) v avtomobilski tovarni Citroën leta 1959 in francoski inženir Pierre Étienne Bézier (1910–1999) v konkurenčni tovarni Renault leta 1962. Krivulje so glavno orodje za računalniško podprto načrtovanje in izdelavo (CAD/CAM), zato je razvoj na začetku potekal večinoma v letalski in avtomobilski industriji.

[7] Ruski matematik Sergej Natanovič Bernstein (1880–1968).

Beziérovi kubični zlepki se npr. uporabljajo za zapis računalniških pisav v vektorski obliki. Zapis ne porabi veliko prostora in omogoča, da se lahko velikost pisave brez izgube kvalitete poljubno poveča.

Omenimo še nekaj pojmov, ki se pojavijo, kadar pri interpolaciji ravninskih krivulj uporabljamo zlepke:

- *Geometrijska zveznost*: krivulji se dotikata, kar pomeni, da se zadnja točka prve krivulje ujema z začetno točko druge krivulje.
- *Geometrijska zveznost odvodov (G^1)*: tangenti prve in druge krivulje imata v skupni točki enako smer.
- *Parametrična zveznost odvodov (C^1)*: prva odvoda prve in druge krivulje v skupni točki se ujemata (tako v smeri kot v velikosti, pri čemer je velikost odvisna od parametrizacije).

Matlab

Na voljo so naslednje funkcije za delo s polinomi in za konstrukcijo interpolacijskih polinomov in zlepkov:

- `polyval(p,x)`: vrednost polinoma, podanega s koeficienti v p v točki x
- `roots(p)`: ničle polinoma, podanega s koeficienti v p
- `poly(r)`: vrne koeficiente polinoma z danimi ničlami v r
- `polyfit(x,y,n)`: vrne koeficiente polinoma p stopnje n, ki po metodi najmanjših kvadratov najbolje aproksimira točke (x_i, y_i), kar pomeni da je vsota

$$\sum_{i=0}^{n} (p(x_i) - y_i)^2$$

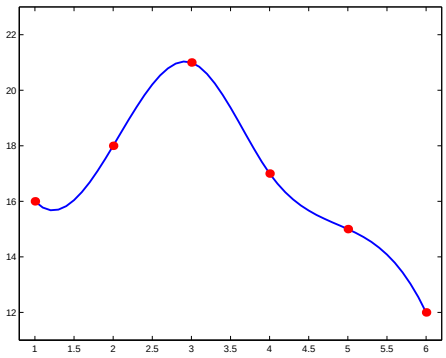
  minimalna. Če izberemo $m \geq n$, potem dobimo interpolacijski polinom.

- `yi=interp1(x,y,xi)`: vrne vrednost kosoma polinomske interpolacijske funkcije v točki `xi`. Kot četrti argument lahko podamo vrsto interpolacijske funkcije, na voljo so
  - `'nearest'`: kosoma konstantna interpolacija,
  - `'linear'`: kosoma linearna interpolacija,
  - `'spline'`: dvakrat zvezno odvedljiv kubični zlepek,
  - `'cubic'`: kubični zlepek, ki ohranja obliko.

Za interpolacijo v dveh dimenzijah sta na voljo ukaza `interp2` in `griddata`.

Zgled 5.4 *Naslednji ukazi v Matlabu zgenerirajo dvakrat zvezno odvedljiv kubični zlepek in narišejo njegov graf, ki je prikazan na desni strani.*

```
x=[1 2 3 4 5 6];
y=[16 18 21 17 15 12];
t=1:0.1:6;
y3=interp1(x,y,t,'spline');
plot(t,y3,'b-','LineWidth',2);
hold on
plot(x,y,'r.','MarkerSize',27);
axis([0.8 6.2 11 23])
hold off
```



Dodatna literatura

Za interpolacijo je na voljo obsežna literatura, saj je obravnavana skoraj v vseh učbenikih numerične analize. V slovenščini imate na voljo knjigo [18], osnove interpolacije pa najdete tudi v knjigah [3] in [34]. Od tuje literature omenimo npr. knjigi [6] in [17].

5.6 Numerično odvajanje

Iščemo odvod funkcije, ki je podana s tabelo vrednosti v točkah $x_0, \ldots, x_n$. Ideja je, da za približek vzamemo odvod interpolacijskega polinoma. Če je funkcija dovoljkrat zvezno odvedljiva, potem vemo, da je

$$f(x) = I_n(x) + \frac{\omega(x)}{(n+1)!} f^{(n+1)}(\xi),$$

kjer je $\omega(x) = (x - x_0) \cdots (x - x_n)$. Z odvajanjem zgornje zveze dobimo

$$f'(x) = I_n'(x) + \underbrace{\frac{\omega'(x)}{(n+1)!} f^{(n+1)}(\xi) + \frac{\omega(x)}{(n+1)!} \cdot \frac{df^{(n+1)}(\xi)}{dx}}_{\text{napaka}}.$$

Izraz za napako ni najlepši, saj ne poznamo odvisnosti ξ od x. To nam preprečuje, da bi lahko ocenili maksimalno napako. Če pa računamo odvod v eni izmed točk $x_0, \ldots, x_n$, zadnji člen odpade in dobimo

$$f'(x_k) = I_n'(x_k) + \frac{\omega'(x_k)}{(n+1)!} f^{(n+1)}(\xi). \tag{5.9}$$

V zgornji formuli je $I_n'(x_k)$ odvod interpolacijskega polinoma v točki x_k. Formulo za $I_n'(x_k)$ lahko izpeljemo preko Lagrangeevih koeficientov. Iz $I_n(x) = \sum_{i=0}^{n} f(x_i) L_{n,i}(x)$ sledi $I_n'(x_k) = \sum_{i=0}^{n} f(x_i) L_{n,i}'(x_k)$. Z odvajanjem Lagrangeevega koeficienta

$$L_{n,i}(x) = \frac{(x - x_0) \cdots (x - x_{i-1})(x - x_{i+1}) \cdots (x - x_n)}{(x_i - x_0) \cdots (x_i - x_{i-1})(x_i - x_{i+1}) \cdots (x_i - x_n)}$$

dobimo

a) $i \neq k$: $L'_{n,i}(x_k) = \dfrac{\omega'(x_k)}{(x_k - x_i)\omega'(x_i)}$,

b) $i = k$: $L'_{n,k}(x_k) = \sum\limits_{\substack{j=0 \\ j\neq k}}^{n} \dfrac{1}{x_k - x_j}$.

S pomočjo teh formul lahko zapišemo

$$I'_n(x_k) = \omega'(x_k) \sum_{\substack{j=0 \\ j\neq k}}^{n} \frac{f(x_j)}{(x_k - x_j)\omega'(x_j)} + f(x_k) \sum_{\substack{j=0 \\ j\neq k}}^{n} \frac{1}{x_k - x_j}.$$

V primeru, ko so točke ekvidistantne in velja $x_i = x_0 + ih$ za $i = 0, 1, \ldots, n$, se formula (6.1) še poenostavi. Z malce računanja lahko izpeljemo formulo

$$f'(x_k) = \frac{1}{h}\left(\frac{(-1)^k}{\binom{n}{k}} \sum_{\substack{j=0 \\ j\neq k}}^{n} \frac{(-1)^j \binom{n}{j} f(x_j)}{k-j} + f(x_k) \sum_{\substack{j=0 \\ j\neq k}}^{n} \frac{1}{k-j} \right) + \frac{(-1)^{n-k} h^n}{(n+1)\binom{n}{k}} f^{(n+1)}(\xi).$$

Nekaj prvih formul, ki jih dobimo z vstavljanjem n in k, je:

- $n = 1$:

$$\begin{aligned} f'(x_0) &= \frac{1}{h}(f(x_1) - f(x_0)) - \frac{1}{2}hf''(\xi_0) \\ f'(x_1) &= \frac{1}{h}(f(x_1) - f(x_0)) + \frac{1}{2}hf''(\xi_1) \end{aligned}$$

- $n = 2$:

$$\begin{aligned} f'(x_0) &= \frac{1}{2h}(-3f(x_0) + 4f(x_1) - f(x_2)) + \frac{1}{3}h^2 f'''(\xi_0) \\ f'(x_1) &= \frac{1}{2h}(-f(x_0) + f(x_2)) - \frac{1}{6}h^2 f'''(\xi_1) \quad \text{(simetrična diferenca)} \\ f'(x_2) &= \frac{1}{2h}(f(x_0) - 4f(x_1) + 3f(x_2)) + \frac{1}{3}h^2 f'''(\xi_2) \end{aligned}$$

Če primerjamo formuli pri $n = 1$ in simetrično diferenco, potem v vseh treh formulah vrednost odvoda aproksimiramo z vrednostnima funkcije v dveh točkah. Pri simetrični diferenci zaradi simetrije pridobimo red h^2 namesto h, višji red pa imamo v bistvu zaradi tega, ker smo odvajali interpolacijski polinom na treh točkah namesto na dveh.

5.7 Drugi načini izpeljave

Formule za numerično odvajanje lahko izpeljujemo tudi iz razvoja v Taylorjevo vrsto. Naj bodo točke ekvidistantne in $y_i = f(x_i)$. Iz razvojev

$$\begin{aligned} y_0 &= y_1 - hy'_1 + \frac{1}{2}h^2 y''_1 - \frac{1}{6}h^3 y'''_1 + \frac{1}{24}h^4 f^{(4)}(\xi_0) \\ y_1 &= y_1 \\ y_2 &= y_1 + hy'_1 + \frac{1}{2}h^2 y''_1 + \frac{1}{6}h^3 y'''_1 + \frac{1}{24}h^4 f^{(4)}(\xi_2) \end{aligned}$$

s seštevanjem dobimo formulo $f''(x_1) = \frac{1}{h^2}(y_0 - 2y_1 + y_2) - \frac{1}{24}h^2(f^{(4)}(\xi_0) + f^{(4)}(\xi_2))$. Namesto $f^{(4)}(\xi_0) + f^{(4)}(\xi_2)$ lahko pišemo $2f^{(4)}(\xi)$ in dobimo končno formulo

$$f''(x_1) = \frac{1}{h^2}(y_0 - 2y_1 + y_2) - \frac{1}{12}h^2 f^{(4)}(\xi). \tag{5.10}$$

Formula (6.2) je simetrična diferenca za drugi odvod. Skupaj s simetrično diferenco za prvi odvod se npr. uporabljata za numerično reševanje robnih problemov preko diferenčne metode. Formuli aproksimirata drugi oz. prvi odvod z natančnostjo $\mathcal{O}(h^2)$.

Alternativni način izpeljave je *metoda nedoločenih koeficientov*. Pri tej metodi nastavimo sistem

$$f'(x_k) = \sum_{j=0}^{n} \alpha_j f(x_j) + R(f)$$

in določimo koeficiente $\alpha_0, \ldots, \alpha_n$ tako, da je formula točna za polinome čim višje stopnje. Napako dobimo iz polinoma najnižje stopnje, za katerega formula ni točna.

Zgled 5.5 *Izpeljimo formulo $f''(x_1) = af(x_0) + bf(x_1) + cf(x_2) + R(f)$.*

Za bazo izberemo $1, x - x_1, (x - x_1)^2, \ldots$, saj tako dobimo lepši sistem:

$$\left.\begin{array}{rcccl} 1 & : & 0 & = & a + b + c \\ x - x_1 & : & 0 & = & -ha + hc \\ (x - x_1)^2 & : & 2 & = & h^2 a + h^2 c \end{array}\right\} \Rightarrow a = \frac{1}{h^2},\ b = \frac{-2}{h^2},\ c = \frac{1}{h^2}.$$

Napaka ima obliko $R(f) = Ch^p f^{(r)}(\xi)$, kjer je C konstanta, p in r pa sta stopnji, ki ju moramo določiti. Odvod r ustreza najnižji stopnji polinoma, za katerega formula ni točna.

Napako dobimo tako, da po vrsti v formulo vstavljamo $f(x) = x^r$ in poiščemo najnižjo stopnjo, za katero formula ni točna. Ker imamo v formuli tri točke, je formula točna za polinome stopnje 2 ali manj, zato najprej vstavimo $r = 3$, ker pa se izkaže, da je formula točna, nadaljujemo z $r = 4$.

$$\left.\begin{array}{rcccl} (x - x_1)^3 & : & 0 & = & -h^3 a + h^3 c \\ (x - x_1)^4 & : & 0 & \neq & h^4 a + h^4 c \end{array}\right\} \Rightarrow r = 4,\ p = 2.$$

Ker za $f(x) = (x - x_1)^4$ velja $f^{(4)}(\xi) = 4!$, sledi, da je napaka enaka $R(f) = -\frac{1}{12}h^2 f^{(4)}(\xi)$. □

5.8 Celotna napaka

Naslednji zgled prikazuje težave, ki se pojavijo pri numeričnem računanju odvodov.

Zgled 5.6 *Če po formulah*

$$\begin{aligned} f'(0) &= \frac{1}{2h}(f(h) - f(-h)) + \mathcal{O}(h^2) \\ f''(0) &= \frac{1}{h^2}(f(-h) - 2f(0) + f(h)) + \mathcal{O}(h^2) \end{aligned}$$

računamo $f'(0)$ in $f''(0)$ za $f(x) = e^x$, potem v enojni natančnosti dobimo:

h	$f'(0)$	$f''(0)$
0.4	1.0268809	1.0134048
0.04	1.0002673	1.0001659
0.004	1.0000094	1.0021030
0.0004	1.0000169	0.7450581
0.00004	1.0006130	$3.7252907 \cdot 10^1$
0.000004	1.0058284	$3.7252903 \cdot 10^3$

Napaka se z manjšanjem h nekaj časa manjša, potem pa začne naraščati, čeprav za obe formuli velja, da imata napako reda $\mathcal{O}(h^2)$. □

Do težav, predstavljenih v prejšnjem zgledu, pride zaradi neodstranljive napake. Oglejmo si situacijo na zgledu formule

$$f''(x_1) = \frac{1}{h^2}(y_0 - 2y_1 + y_2) - \frac{1}{12}h^2 f^{(4)}(\xi). \tag{5.11}$$

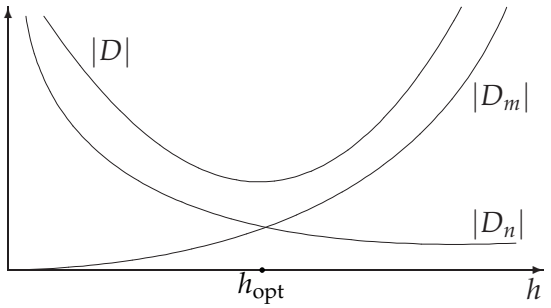
Vemo, da za napako metode D_m velja $D_m = -\frac{1}{12}h^2 f^{(4)}(\xi)$. Od tod lahko ocenimo

$$|D_m| \leq \frac{h^2}{12}\|f^{(4)}\|_\infty.$$

Poleg napake metode pa se pojavi tudi neodstranljiva napaka D_n, saj namesto s točnimi vrednostmi $f(x_i)$ računamo s približki $\widetilde{y}_i$, za katere predpostavimo $|f(x_i) - \widetilde{y}_i| \leq \epsilon$. Če nič drugega, pride do neodstranljive napake že zaradi tega, ker namesto točnih vrednosti uporabljamo najbližja predstavljiva števila. Ocenimo lahko

$$|D_n| \leq \frac{4\epsilon}{h^2}.$$

Pri samem računanju se pojavi še zaokrožitvena napaka D_z, ki pa smo jo v tokratni analizi zanemarili.



Slika 5.5: Ocene za neodstranljivo napako D_n, napako metode D_m in celotno napako D v odvisnosti od h.

Ocena za celotno napako (za formulo (6.3)) je

$$|D| \leq |D_m| + |D_n| \leq \frac{h^2}{12}\|f^{(4)}\|_\infty + \frac{4\epsilon}{h^2}.$$

Če poznamo oceni za $\|f^{(4)}\|_\infty$ in ϵ, lahko iz ocen za $|D_m|$ in $|D_n|$ določimo optimalni h, kjer bo ocena za skupno napako najmanjša.

Numerično odvajanje torej ni numerično dobro pogojen problem, saj se pri premajhnem h zgodi, da se z manjšanjem h napaka poveča.

Dodatna literatura

Za numerično odvajanje imate v slovenščini na voljo knjige [18], [3] in [34], od tuje literature pa omenimo npr. knjigi [6] in [17].

Poglavje 6

Numerično odvajanje in integriranje

6.1 Numerično odvajanje

Denimo, da iščemo odvod funkcije f, ki je podana s tabelo vrednosti v točkah $x_0, \ldots, x_n$. Ideja je, da za približek vzamemo odvod interpolacijskega polinoma. Če je funkcija f dovoljkrat zvezno odvedljiva in je I_n njen interpolacijski polinom na točkah $x_0, \ldots, x_n$, potem vemo, da za napako velja

$$f(x) = I_n(x) + \frac{\omega(x)}{(n+1)!} f^{(n+1)}(\xi),$$

kjer je $\omega(x) = (x - x_0) \cdots (x - x_n)$ in $\min(x, x_0, \ldots, x_n) \leq \xi \leq \max(x, x_0, \ldots, x_n)$. Z odvajanjem zgornje zveze dobimo

$$f'(x) = I_n'(x) + \underbrace{\frac{\omega'(x)}{(n+1)!} f^{(n+1)}(\xi) + \frac{\omega(x)}{(n+1)!} \cdot \frac{df^{(n+1)}(\xi)}{dx}}_{\text{napaka}}.$$

Izraz za napako ni najlepši, saj ne poznamo odvisnosti ξ od x, vemo le, da za vsako točko x obstaja ustrezna točka ξ. To nam preprečuje, da bi lahko ocenili maksimalno napako. Če pa računamo odvod v eni izmed točk $x_0, \ldots, x_n$, potem zadnji člen odpade in ostane

$$f'(x_k) = I_n'(x_k) + \frac{\omega'(x_k)}{(n+1)!} f^{(n+1)}(\xi), \tag{6.1}$$

za $k = 0, \ldots, n$. kjer je $I_n'(x_k)$ odvod interpolacijskega polinoma v točki x_k za $k = 0, \ldots, n$. Formulo za $I_n'(x_k)$ lahko izpeljemo preko Lagrangeevih koeficientov. Iz $I_n(x) = \sum_{i=0}^{n} f(x_i) L_{n,i}(x)$ sledi $I_n'(x_k) = \sum_{i=0}^{n} f(x_i) L_{n,i}'(x_k)$. Z odvajanjem Lagrangeevega koeficienta

$$L_{n,i}(x) = \frac{(x - x_0) \cdots (x - x_{i-1})(x - x_{i+1}) \cdots (x - x_n)}{(x_i - x_0) \cdots (x_i - x_{i-1})(x_i - x_{i+1}) \cdots (x_i - x_n)}$$

dobimo

a) $i \neq k$: $\quad L_{n,i}'(x_k) = \dfrac{\omega'(x_k)}{(x_k - x_i)\omega'(x_i)}$,

b) $i = k$: $\quad L'_{n,k}(x_k) = \sum\limits_{\substack{j=0 \\ j\neq k}}^{n} \dfrac{1}{x_k - x_j}$.

S pomočjo teh formul lahko zapišemo

$$I'_n(x_k) = \omega'(x_k) \sum_{\substack{j=0 \\ j\neq k}}^{n} \frac{f(x_j)}{(x_k - x_j)\omega'(x_j)} + f(x_k) \sum_{\substack{j=0 \\ j\neq k}}^{n} \frac{1}{x_k - x_j}.$$

V primeru, ko so točke ekvidistantne in velja $x_i = x_0 + ih$ za $i = 0, 1, \ldots, n$, se formula (6.1) še poenostavi. Z malce računanja lahko izpeljemo splošno formulo

$$f'(x_k) = \frac{1}{h}\left(\frac{(-1)^k}{\binom{n}{k}} \sum_{\substack{j=0 \\ j\neq k}}^{n} \frac{(-1)^j \binom{n}{j} f(x_j)}{k-j} + f(x_k) \sum_{\substack{j=0 \\ j\neq k}}^{n} \frac{1}{k-j} \right) + \frac{(-1)^{n-k} h^n}{(n+1)\binom{n}{k}} f^{(n+1)}(\xi).$$

Nekaj prvih formul, ki jih dobimo z vstavljanjem n in k, je:

- $n = 1$:

$$\begin{aligned} f'(x_0) &= \frac{1}{h}(f(x_1) - f(x_0)) - \frac{1}{2}hf''(\xi_0), \\ f'(x_1) &= \frac{1}{h}(f(x_1) - f(x_0)) + \frac{1}{2}hf''(\xi_1). \end{aligned}$$

- $n = 2$:

$$\begin{aligned} f'(x_0) &= \frac{1}{2h}(-3f(x_0) + 4f(x_1) - f(x_2)) + \frac{1}{3}h^2 f'''(\xi_0), \\ f'(x_1) &= \frac{1}{2h}(-f(x_0) + f(x_2)) - \frac{1}{6}h^2 f'''(\xi_1), \quad \text{(simetrična diferenca)} \\ f'(x_2) &= \frac{1}{2h}(f(x_0) - 4f(x_1) + 3f(x_2)) + \frac{1}{3}h^2 f'''(\xi_2). \end{aligned}$$

Če primerjamo formuli pri $n = 1$ in simetrično diferenco, potem v vseh treh formulah vrednost odvoda aproksimiramo z vrednostnima funkcije v dveh točkah. Pri simetrični diferenci zaradi simetrije pridobimo red h^2 namesto h, višji red pa imamo v bistvu zaradi tega, ker smo odvajali interpolacijski polinom na treh točkah namesto na dveh.

Formule za numerično odvajanje lahko izpeljujemo tudi iz razvoja v Taylorjevo vrsto. Naj bodo točke ekvidistantne in $y_i = f(x_i)$. Iz razvojev

$$\begin{aligned} y_0 &= y_1 - hy'_1 + \frac{1}{2}h^2 y''_1 - \frac{1}{6}h^3 y'''_1 + \frac{1}{24}h^4 f^{(4)}(\xi_0) \\ y_1 &= y_1 \\ y_2 &= y_1 + hy'_1 + \frac{1}{2}h^2 y''_1 + \frac{1}{6}h^3 y'''_1 + \frac{1}{24}h^4 f^{(4)}(\xi_2) \end{aligned}$$

s seštevanjem dobimo formulo $f''(x_1) = \dfrac{1}{h^2}(y_0 - 2y_1 + y_2) - \dfrac{1}{24}h^2(f^{(4)}(\xi_0) + f^{(4)}(\xi_2))$. Namesto $f^{(4)}(\xi_0) + f^{(4)}(\xi_2)$ lahko pišemo $2f^{(4)}(\xi)$ v neki točki ξ in tako dobimo končno formulo

$$f''(x_1) = \frac{1}{h^2}(y_0 - 2y_1 + y_2) - \frac{1}{12}h^2 f^{(4)}(\xi). \tag{6.2}$$

Formula (6.2) je simetrična diferenca za drugi odvod. Skupaj s simetrično diferenco za prvi odvod se npr. uporabljata za numerično reševanje robnih problemov preko diferenčne metode. Formuli aproksimirata drugi oz. prvi odvod z natančnostjo $\mathcal{O}(h^2)$.

Formule lahko izpeljemo tudi preko *metode nedoločenih koeficientov*. Tu nastavimo sistem

$$f'(x_k) = \sum_{j=0}^{n} \alpha_j f(x_j) + R(f)$$

in določimo koeficiente $\alpha_0, \ldots, \alpha_n$ tako, da je formula točna za polinome čim višje stopnje. Napako dobimo iz polinoma najnižje stopnje, za katerega formula ni točna.

Zgled 6.1 *Preko metode nedoločenih koeficientov izpeljimo formulo* $f''(x_1) = af(x_0) + bf(x_1) + cf(x_2) + R(f)$ *za ekvidistantne točke* x_0, $x_1 = x_0 + h$ *in* $x_2 = x_0 + 2h$.

Ker iščemo formulo, ki računa odvod v točki x_1, *je najbolje za polinomsko bazo izbrati kar* $1, x - x_1, (x - x_1)^2, \ldots$, *saj tako dobimo lepši linearni sistem za nedoločene koeficiente* a, b *in* c:

$$\left.\begin{array}{rcccl} 1 & : & 0 & = & a + b + c \\ x - x_1 & : & 0 & = & -ha + hc \\ (x - x_1)^2 & : & 2 & = & h^2 a + h^2 c \end{array}\right\} \Rightarrow a = \frac{1}{h^2},\ b = \frac{-2}{h^2},\ c = \frac{1}{h^2}.$$

Napaka ima obliko $R(f) = Ch^p f^{(r)}(\xi)$, *kjer je* C *konstanta,* p *in* r *pa sta stopnji, ki ju moramo določiti. Odvod* r *ustreza najnižji stopnji polinoma, za katerega formula ni točna.*

Napako dobimo tako, da po vrsti v formulo vstavljamo $f(x) = x^r$ *in poiščemo najnižjo stopnjo, za katero formula ni točna. Ker imamo v formuli tri točke, že iz izpeljave sledi, da je formula točna za polinome stopnje 2 ali manj. Zaradi tega najprej vstavimo* $r = 3$, *ker pa se izkaže, da je formula točna, nadaljujemo z* $r = 4$.

$$\left.\begin{array}{rcccl} (x - x_1)^3 & : & 0 & = & -h^3 a + h^3 c \\ (x - x_1)^4 & : & 0 & \neq & h^4 a + h^4 c \end{array}\right\} \Rightarrow r = 4,\ p = 2.$$

Ker za $f(x) = (x - x_1)^4$ *velja* $f^{(4)}(\xi) = 4!$, *sledi, da je napaka enaka* $R(f) = -\frac{1}{12} h^2 f^{(4)}(\xi)$. □

Naslednji zgled prikazuje težave, ki se pojavijo pri numeričnem računanju odvodov. Teorija pravi, da bi morali pri manjšem razmiku med točkami dobiti boljše približke za vrednost odvoda, a se to pri numeričnem računanju ne zgodi.

Zgled 6.2 *Če po formulah za simetrični diferenci*

$$\begin{aligned} f'(0) &= \frac{1}{2h}(f(h) - f(-h)) + \mathcal{O}(h^2), \\ f''(0) &= \frac{1}{h^2}(f(-h) - 2f(0) + f(h)) + \mathcal{O}(h^2) \end{aligned}$$

računamo $f'(0)$ *in* $f''(0)$ *za* $f(x) = e^x$, *potem v enojni natančnosti dobimo naslednje vrednosti:*

h	$f'(0)$	$f''(0)$
0.4	1.0268809	1.0134048
0.04	1.0002673	1.0001659
0.004	1.0000094	1.0021030
0.0004	1.0000169	0.7450581
0.00004	1.0006130	$3.7252907 \cdot 10^1$
0.000004	1.0058284	$3.7252903 \cdot 10^3$

Napaka se z manjšanjem h nekaj časa manjša, potem pa začne naraščati, čeprav za obe formuli velja, da imata napako reda $\mathcal{O}(h^2)$. □

Do težav, predstavljenih v zadnjem zgledu, pride zaradi neodstranljive napake. Oglejmo si situacijo na primeru simetrične diference za drugi odvod

$$f''(x_1) = \frac{1}{h^2}(y_0 - 2y_1 + y_2) - \frac{1}{12}h^2 f^{(4)}(\xi). \tag{6.3}$$

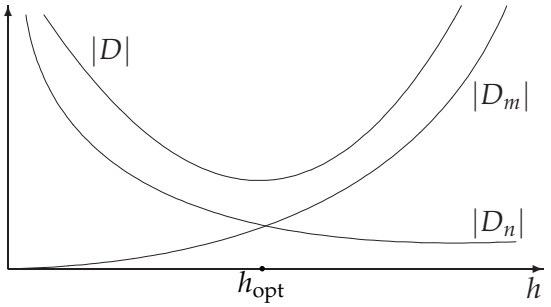
Vemo, da za napako metode D_m velja $D_m = -\frac{1}{12}h^2 f^{(4)}(\xi)$. Od tod lahko ocenimo

$$|D_m| \leq \frac{h^2}{12}\|f^{(4)}\|_\infty,$$

kar očitno pomeni, da gre v limiti, ko h pošljemo proti 0, napaka metode tudi proti 0. Poleg napake metode pa se pojavi tudi neodstranljiva napaka D_n, saj namesto s točnimi vrednostmi $f(x_i)$ računamo s približki $\widetilde{y}_i$, za katere predpostavimo $|f(x_i) - \widetilde{y}_i| \leq \epsilon$. Če nič drugega, pride do neodstranljive napake že zaradi tega, ker pri numeričnem računanju namesto s točnimi vrednostmi v resnici računamo z najbližjimi predstavljivimi števili. V našem primeru lahko neodstranljivo napako za formulo (6.3) ocenimo z

$$|D_n| \leq \frac{4\epsilon}{h^2}.$$

Ker je ϵ neodvisen od h, je neodstranljiva napaka, ko gre h proti 0, lahko poljubno velika.



Slika 6.1: Ocene za neodstranljivo napako D_n, napako metode D_m in celotno napako D v odvisnosti od h.

Pri samem računanju se pojavi še zaokrožitvena napaka D_z, ki pa smo jo v tokratni analizi zanemarili. Ocena za celotno napako (za formulo (6.3)) je tako

$$|D| \leq |D_m| + |D_n| \leq \frac{h^2}{12}\|f^{(4)}\|_\infty + \frac{4\epsilon}{h^2}.$$

Če poznamo oceni za $\|f^{(4)}\|_\infty$ in ϵ, lahko iz ocen za $|D_m|$ in $|D_n|$ določimo optimalni h, kjer bo ocena za skupno napako najmanjša.

Numerično odvajanje torej ni numerično dobro pogojen problem, saj se pri premajhnem h zgodi, da se z manjšanjem h napaka poveča.

6.2 Kvadraturne formule

Radi bi izračunali določeni integral

$$I(f) = \int_a^b f(x)dx.$$

Pri tem predpostavimo, da je funkcija f taka, da integral obstaja, npr., da je kosoma zvezna.

Ideja je, da namesto funkcije f, ki se je v splošnem primeru ne da analitično integrirati, integriramo drugo funkcijo, ki se čim bolj prilega f in za katero obstaja določeni integral. Prva izbira so interpolacijski polinomi.

Če za paroma različne interpolacijske točke izberemo $x_0, \ldots, x_n$ in predpostavimo, da je funkcija f dovoljkrat zvezno odvedljiva, potem lahko s pomočjo Lagrangeevih koeficientov zapišemo

$$f(x) = \sum_{i=0}^{n} f(x_i)L_{n,i}(x) + \frac{f^{(n+1)}(\xi)}{(n+1)!}\omega(x).$$

Če to integriramo, dobimo

$$\int_a^b f(x)dx = \sum_{i=0}^{n} f(x_i) \int_a^b L_{n,i}(x)dx + \int_a^b \frac{f^{(n+1)}(\xi)}{(n+1)!}\omega(x)dx.$$

Od tod sledi, da integral $I(f)$ lahko aproksimiramo s *kvadraturno formulo* oblike

$$\int_a^b f(x)dx = \sum_{i=0}^{n} A_i f(x_i) + R(f), \tag{6.4}$$

kjer paroma različne točke $x_0, \ldots, x_n$ imenujemo *vozli*, koeficiente $A_0, \ldots, A_n$, za katere velja

$$A_i = \int_a^b L_{n,i}(x)dx$$

za $i = 0, \ldots, n$, imenujemo *uteži*, $R(f)$ pa je napaka, ki je enaka

$$R(f) = \int_a^b \frac{f^{(n+1)}(\xi)}{(n+1)!}\omega(x)dx.$$

Formuli pravimo kvadraturna zato, ker je integral $I(f)$ enak ploščini lika, ki ga določa graf funkcije f na intervalu $[a, b]$.

Alternativno bi lahko izpeljali kvadraturno formulo (6.4) tudi tako, da bi najprej izbrali vozle $x_0, \ldots, x_n$, potem pa določili uteži $A_0, \ldots, A_n$ tako, da je formula točna za polinome čim višje stopnje. Pri tem bi podobno kot pri numeričnem odvajanju lahko uporabili metodo nedoločenih koeficientov.

Za kvadraturno formulo (6.4) pravimo, da je reda k, če je točna za vse polinome stopnje manjše ali enake k, ni pa točna za polinome stopnje $k + 1$. V obeh zgornjih primerih dobimo kvadraturno formulo, ki je očitno reda vsaj n, saj je pravilna za polinome stopnje manjše ali enake n, ne glede na to, kako smo izbrali $n + 1$ vozlov. Kot bomo videli kasneje, pa lahko s primerno izbiro vozlov dosežemo, da bo formula točna tudi za polinome višjih stopenj.

6.3 Newton–Cotesova pravila

Pri *Newton-Cotesovih*[1] *pravilih* so vozli ekvidistantni, torej $a = x_0$, $b = x_n$, $h = (b-a)/n$ in $x_i = x_0 + ih$ za $i = 0, \ldots, n$. Vrednosti funkcije f v vozlih označimo z $y_i = f(x_i)$ za $i = 0, \ldots, n$. Ločimo dva tipa Newton–Cotesovih pravil:

a) *zaprti tip*: upoštevamo tudi krajišča: $\int_a^b f(x)dx = \sum_{k=0}^{n} A_k y_k + R_n(f)$;

b) *odprti tip*: brez krajišč: $\int_a^b f(x)dx = \sum_{k=1}^{n-1} B_k y_k + R_n(f)$.

Poglejmo nekaj osnovnih zaprtih Newton–Cotesovih pravil.

- Pri $n = 1$ dobimo *trapezno pravilo*

$$\int_{x_0}^{x_1} f(x)dx = \frac{h}{2}(y_0 + y_1) - \frac{h^3}{12}f''(\xi). \tag{6.5}$$

  Kot se da sklepati že iz imena, funkcijo interpoliramo s premico in površino trapeza (glej sliko 6.2) vzamemo za približek za integral funkcije.

  Koeficiente lahko izračunamo preko integralov Lagrangeevih koeficientov. Tako dobimo

$$\begin{aligned} A_0 &= \int_{x_0}^{x_1} \frac{x - x_1}{x_0 - x_1}dx = \frac{h}{2}, \\ A_1 &= \int_{x_0}^{x_1} \frac{x - x_0}{x_1 - x_0}dx = \frac{h}{2}. \end{aligned}$$

  Za napako velja

$$R_1(f) = \int_{x_0}^{x_1} \frac{f''(\xi_x)}{2}(x - x_0)(x - x_1)dx = \frac{f''(\xi)}{2}\int_{x_0}^{x_1}(x - x_0)(x - x_1)dx = -\frac{h^3}{12}f''(\xi).$$

  Pri izpeljavi napake smo uporabili izrek o povprečni vrednosti, saj ima izraz $(x - x_0)(x - x_1)$ konstanten predznaka na intervalu $[x_0, x_1]$. Trapezno pravilo je točno za polinome stopnje manjše ali enake 1, kar je očitno že iz same interpolacije s premico.

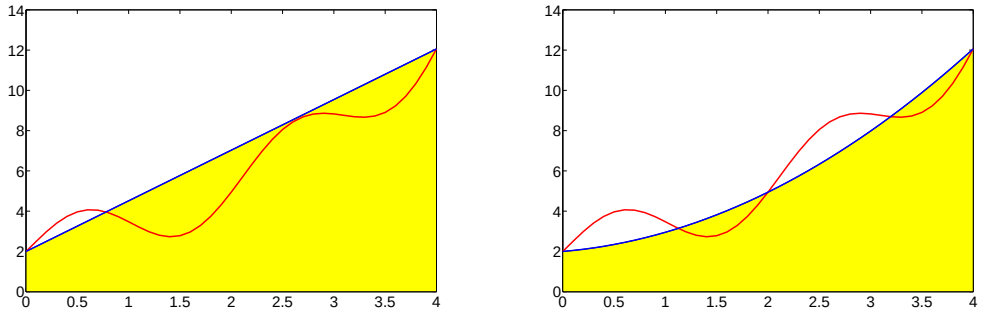
- Pri $n = 2$ dobimo *Simpsonovo*[2] *pravilo*

$$\int_{x_0}^{x_2} f(x)dx = \frac{h}{3}(y_0 + 4y_1 + y_2) - \frac{h^5}{90}f^{(4)}(\xi).$$

  Kot je razvidno tudi iz slike 6.2, funkcijo interpoliramo s parabolo in integral parabole vzamemo za približek za integral funkcije. Podobno kot pri trapezni metodi lahko tudi

[1] Angleški matematik Roger Cotes (1682–1716) je vpeljal merjenje kotov v radianih namesto v stopinjah, z Newtonom pa je sodeloval pri drugi izdaji Newtonove knjige *Principia*.

[2] Pravilo se imenuje po angleškem matematiku Thomasu Simpsonu (1710–1761), ki pa je zanj izvedel od Newtona [24]. Pravilo je že 100 let prej uporabljal nemški matematik in astronom Johannes Kepler (1571–1630).



Slika 6.2: Primerjava trapeznega pravila (levo) in Simpsonovega pravila (desno) na integralu $\int_0^4 (x^2 - x + 2 + 3\sin(2x)\cos x)dx$.

sedaj uteži določimo z integriranjem Lagrangeevih koeficientov. Tako dobimo

$$\begin{aligned} A_0 &= \int_{x_0}^{x_2} \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)}dx = \frac{h}{3}, \\ A_1 &= \int_{x_0}^{x_2} \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)}dx = \frac{4h}{3}, \\ A_2 &= \int_{x_0}^{x_2} \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}dx = \frac{h}{3}. \end{aligned}$$

Iz interpolacije s parabolo je očitno, da je pravilo točno za polinome stopnje 2 ali manj. Izkaže pa se, da je pravilo točno tudi za kubične polinome, saj za $f(x) = x^3$ velja

$$\int_{x_0}^{x_2} \frac{f^{(3)}(\xi)}{6}(x-x_0)(x-x_1)(x-x_2)dx = 0.$$

Podobno za vsa Newton–Cotesova pravila z lihim številom točk (sodi n) zaradi simetrije velja, da so točne tudi za polinome stopnje $n+1$.

Tu sicer pri izpeljavi napake ne moremo več uporabiti izreka o povprečni vrednosti, a na srečo za Newton–Cotesova pravila velja, da napako za dovoljkrat zvezno odvedljivo funkcijo lahko ugotovimo iz napake polinoma x^{n+1} (oziroma x^{n+2} za sodi n). To sledi iz Peanovega izreka, ki je predstavljen v razdelku 6.6.

- Pri $n = 3$ dobimo *3/8 pravilo*

$$\int_{x_0}^{x_3} f(x)dx = \frac{3h}{8}(y_0 + 3y_1 + 3y_2 + y_3) - \frac{3h^5}{80}f^{(4)}(\xi).$$

Čeprav vsebuje en vozel več, ima 3/8 pravilo enak red natančnosti kot Simpsonovo pravilo. Zaradi tega se 3/8 pravila večinoma ne uporablja.

- Pri $n = 4$ dobimo *Booleovo*[3] *pravilo*

$$\int_{x_0}^{x_4} f(x)dx = \frac{2h}{45}(7y_0 + 32y_1 + 12y_2 + 32y_3 + 7y_4) - \frac{8h^7}{945}f^{(6)}(\xi).$$

[3] Angleški matematik Goerge Boole (1815–1864) je znan predvsem po Booleovi mreži in njegovem delu na področju formalne logike.

Tako kot pri Simpsonovi metodi, tudi pri Booleovi metodi zaradi simetrije pridobimo en red natančnosti.

Nekaj osnovnih odprtih Newton–Cotesovih pravil je:

- Pri $n = 2$ dobimo *sredinsko pravilo*

$$\int_{x_0}^{x_2} f(x)dx = 2hy_1 + \frac{h^3}{3}f''(\xi).$$

Čeprav uporabimo vrednost v eni sami točki, je zaradi simetrije pravilo točno tudi za polinome stopnje 1.

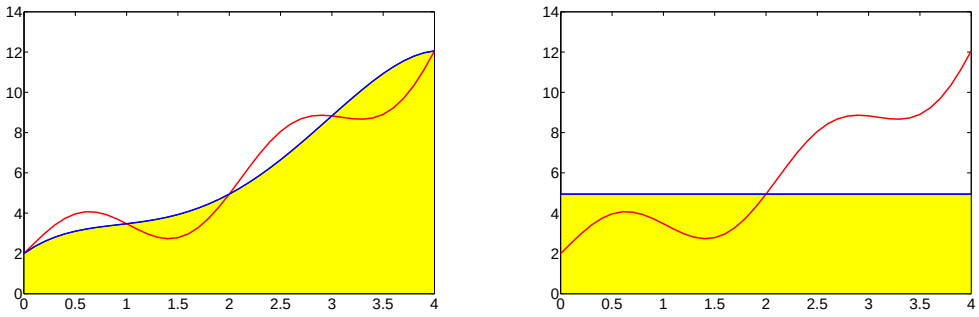
- Pri $n = 3$ dobimo pravilo

$$\int_{x_0}^{x_3} f(x)dx = \frac{3h}{2}(y_1 + y_2) + \frac{3h^3}{4}f''(\xi).$$

- Pri $n = 4$ dobimo *Milneovo*[4] *pravilo*

$$\int_{x_0}^{x_4} f(x)dx = \frac{4h}{3}(2y_1 - y_2 + 2y_3) + \frac{28h^5}{90}f^{(4)}(\xi).$$

To je Newton–Cotesovo pravilo z najmanjšim številom vozlov, ki nima samih uteži s pozitivnim predznakom. Izkaže se, da pri povečevanju števila točk ravno negativne uteži povzročijo, da Newton–Cotesove formule za velike n niso primerne za numerično računanje.



Slika 6.3: Primerjava Booleovega pravila (levo) in sredinskega pravila (desno) na integralu $\int_0^4 (x^2 - x + 2 + 3\sin(2x)\cos x)dx$.

6.4 Neodstranljiva napaka

Pri numeričnem odvajanju smo imeli pri numeričnem računanju težave, ko je šel razmik med točkami h proti 0, saj se je izkazalo, da zaradi neodstranljive napake ne moremo odvoda izračunati poljubno natančno. Glavni problem pri formulah za numerično odvajanje je, da imamo h v

[4] Ameriški matematik William Edmund Milne (1890–1971).

imenovalcu in ko gre potem h proti 0, zgoraj v števcu pa je majhna napaka, lahko neodstranljiva napaka naraste preko vseh meja.

Kako pa je s tem pri kvadraturnih formulah? Ker se sedaj h pojavi v števcu, je na prvi pogled vse v redu, a se izkaže, da imamo tudi tukaj lahko težave. Vzemimo splošno kvadraturno pravilo

$$\int_a^b f(x)dx = \sum_{i=0}^{n} A_i f(x_i) + R(f)$$

in predpostavimo, da je pri izračunu $f(x_i)$ absolutna napaka omejena z $\epsilon > 0$. Ocena za neodstranljivo napako je potem

$$|D_n| \leq \epsilon \sum_{i=0}^{n} |A_i|.$$

Ker so uteži enake integralom Lagrangeevih koeficientov, za vsoto Lagrangeevih koeficientov pa vemo, da je enaka 1, velja

$$\sum_{i=0}^{n} A_i = b - a.$$

Če so vse uteži pozitivne, od tod sledi $|D_n| \leq (b-a)\epsilon$ in dobimo lepo oceno za neodstranljivo napako, ki je omejena in neodvisna od n. Na žalost pa tako pri zaprtih (za $n > 8$) kot pri odprtih (za $n > 3$) Newton–Cotesovih pravilih dobimo negativne uteži in vsota $\sum_{i=0}^{n} |A_i|$ je lahko zelo velika. Tako se tudi tukaj pojavi napaka in višanje stopnje polinoma ni dobra odločitev.

Naslednja tabela prikazuje vsote $\sum_{i=0}^{n} |A_i|$, ki jih dobimo pri zaprtih Newton–Cotesovih pravilih, če integriramo na intervalu $[0, 1]$.

n	$\sum_{i=0}^{n} \lvert A_i \rvert$
10	3.06
20	$5.44 \cdot 10^2$
30	$2.12 \cdot 10^5$
40	$1.10 \cdot 10^8$
50	$6.70 \cdot 10^{10}$

Iz tabele je očitno, da pri Newton–Cotesovih pravilih z višanjem stopnje polinoma ne moremo natančno izračunati integrala. Namesto tega je bolje uporabljati sestavljene formule.

6.5 Sestavljene formule

Pri sestavljenih formulah interval, po katerem integriramo, razdelimo na manjše podintervale. Na vsakem podintervalu uporabimo kvadraturno pravilo nizke stopnje, nato pa rezultate seštejemo.

Denimo, da interval $[a, b]$ razdelimo z ekvidistantnimi točkami $a = x_0$, $b = x_n$, $h = (b-a)/n$ in $x_i = x_0 + ih$ za $i = 0, \ldots, n$. Osnovne sestavljene formule so:

- *Trapezna formula*:

$$\int_{x_0}^{x_n} f(x)dx = \underbrace{\frac{h}{2}\left(y_0 + 2y_1 + \cdots + 2y_{n-1} + y_n\right)}_{T_h(f)} - \frac{h^2(x_n - x_0)}{12} f''(\xi).$$

Kratka izpeljava je

$$\int_{x_0}^{x_n} f(x)dx = \sum_{k=0}^{n-1} \int_{x_k}^{x_{k+1}} f(x)dx = \sum_{k=0}^{n-1} \left(\frac{h}{2}(y_k + y_{k+1}) - \frac{h^3}{12} f''(\xi_k) \right).$$

Opazimo, da je globalna napaka za en red nižja kot napaka na vsakem intervalu $[x_i, x_{i+1}]$. Do tega pride, ker moramo sešteti n lokalnih napak, pri čemer upoštevamo, da je $nh = x_n - x_0$.

- *Simpsonova formula*:

$$\int_{x_0}^{x_n} f(x)dx = \underbrace{\frac{h}{3}\left(y_0 + 4y_1 + 2y_2 + 4y_3 + \cdots + 2y_{n-2} + 4y_{n-1} + y_n\right)}_{S_h(f)} - \frac{h^4(x_n - x_0)}{180} f^{(4)}(\xi).$$

- *Sredinska formula*:

$$\int_{x_0}^{x_n} f(x)dx = \underbrace{2h\left(y_1 + y_3 + \cdots + y_{n-1}\right)}_{U_h(f)} + \frac{h^2(x_n - x_0)}{6} f''(\xi).$$

Pri Simpsonovi in sredinski formuli mora biti $n = 2m$.

6.6 Peanov izrek

Napake kvadraturnih pravil in formul za numerično odvajanje lahko ocenimo s Peanovim[5] izrekom. Izrek pride prav tudi v primerih, ko funkcija ni dovoljkrat zvezno odvedljiva.

Izrek 6.1 (Peano) *Naj bo $\mathcal{L}$ linearen funkcional oblike*

$$\begin{aligned} \mathcal{L}(f) &= \int_a^b \left(a_0(x)f(x) + a_1(x)f'(x) + \cdots + a_n(x)f^{(n)}(x) \right) dx \\ &+ \sum_{i=0}^{j_0} b_{i0} f(x_{i0}) + \sum_{i=0}^{j_1} b_{i1} f'(x_{i1}) + \cdots + \sum_{i=0}^{j_n} b_{in} f^{(n)}(x_{in}), \end{aligned}$$

kjer so funkcije a_i odsekoma zvezne na $[a,b]$, točke x_{ij} pa ležijo na intervalu $[a,b]$. Funkcional $\mathcal{L}$ naj bo za vse polinome stopnje n ali manj enak 0. Tedaj za vsako funkcijo f, ki je $(n+1)$-krat zvezno odvedljiva na $[a,b]$, velja

$$\mathcal{L}(f) = \int_a^b f^{(n+1)}(t) K_n(t) dt,$$

kjer je K_n Peanovo jedro

$$K_n(t) = \frac{1}{n!} \mathcal{L}((x-t)_+^n)$$

in

$$(x-t)_+^n = \begin{cases} (x-t)^n, & x \geq t, \\ 0, & x < t. \end{cases}$$

[5] Italijanski matematik Giuseppe Peano (1858–1932) je znan predvsem po Peanovih aksiomih o naravnih številih.

Dokaz. Taylorjev izrek z ostankom v integralski obliki pravi

$$f(x) = f(a) + f'(a)(x-a) + \cdots + \frac{f^{(n)}(a)}{n!}(x-a)^n + \frac{1}{n!}\int_a^x f^{(n+1)}(t)(x-t)^n dt,$$

kar lahko zapišemo kot

$$f(x) = f(a) + f'(a)(x-a) + \cdots + \frac{f^{(n)}(a)}{n!}(x-a)^n + \frac{1}{n!}\int_a^b f^{(n+1)}(t)(x-t)_+^n dt. \tag{6.6}$$

Če na obeh straneh (6.6) uporabimo funkcional $\mathcal{L}$, dobimo

$$\mathcal{L}(f) = \frac{1}{n!}\mathcal{L}\left(\int_a^b f^{(n+1)}(t)(x-t)_+^n dt\right) = \frac{1}{n!}\int_a^b f^{(n+1)}(t)\mathcal{L}((x-t)_+^n)dt,$$

saj lahko zamenjamo vrstni red $\mathcal{L}$ in integriranja. ∎

Posledica 6.2 *Če je Peanovo jedro K_n konstantnega predznaka, potem za $(n+1)$-krat zvezno odvedljivo funkcijo f velja*

$$\mathcal{L}(f) = \frac{f^{(n+1)}(\xi)}{(n+1)!}\mathcal{L}(x^{n+1}).$$

Dokaz. Po Peanovem izreku velja $\mathcal{L}(x^{n+1}) = \int_a^b (n+1)!K_n(t)dt$, torej je

$$\int_a^b K_n(t)dt = \frac{1}{(n+1)!}\mathcal{L}(x^{n+1}).$$

Ker je Peanovo jedro K_n konstantnega predznaka, po izreku o povprečni vrednosti sledi

$$\mathcal{L}(f) = \int_a^b f^{(n+1)}(t)K_n(t)dt = f^{(n+1)}(\xi)\int_a^b K_n(t)dt.$$

∎

Zgled 6.3 *Ocenimo napako trapeznega pravila pri integriranju enkrat zvezno odvedljive funkcije f. Standardne formule za napako iz (6.5) ne moremo uporabiti, saj f ni dvakrat zvezno odvedljiva. Uporabili bomo Peanov izrek in K_0.*

$$\begin{aligned} K_0(t) &= \int_{x_0}^{x_1}(x-t)_+^0 dx - \frac{h}{2}\left((x_0-t)_+^0 + (x_1-t)_+^0\right) \\ &= \int_t^{x_1}(x-t)^0 dx - \frac{h}{2}(0+1) \\ &= x_1 - t - \frac{h}{2} = \frac{x_0+x_1}{2} - t. \end{aligned}$$

Peanovo jedro ni konstantnega predznaka, zato lahko le ocenimo

$$|R(f)| \le \int_{x_0}^{x_1}|f'(t)|\left|\frac{x_0+x_1}{2} - t\right| dt \le \frac{h^2}{4}\|f'\|_\infty.$$

□

6.7 Richardsonova ekstrapolacija

Pri numeričnem odvajanju in integriranju ponavadi poskušamo priti do čim boljšega približka za točen rezultat tako, da zmanjšamo razmik h. Pri tem si ne moremo privoščiti, da bi bil h poljubno majhen, saj potem v primeru numeričnega odvajanja podivja neodstranljiva napaka, pri numeričnem integriranju pa postane izračun predrag.

Denimo, da je $F(h)$ približek, ki ga dobimo z razmikom h, in denimo, da velja

$$F(h) = a_0 + a_1 h^p + \mathcal{O}(h^r) \tag{6.7}$$

za $r > p$, pri čemer sta konstanti a_0 in a_1 neodvisni od h. Točen rezultat, ki ga iščemo, je $F(0) = a_0$. Če izračunamo približka $F(h)$ in $F(h/2)$, potem iz (6.7) in

$$F(h/2) = a_0 + a_1 (h/2)^p + \mathcal{O}(h^r)$$

sledi

$$a_0 = F(h/2) + \frac{F(h/2) - F(h)}{2^p - 1} + \mathcal{O}(h^r). \tag{6.8}$$

Iz formule (6.8) ugotovimo dvoje. Kot prvo, pri predpostavki (6.7) lahko iz približkov $F(h)$ in $F(h/2)$ dobimo natančnejši približek

$$F(h, h/2) = \frac{2^p F(h/2) - F(h)}{2^p - 1},$$

za katerega velja $F(h, h/2) = a_0 + \mathcal{O}(h^r)$. Temu postopku pravimo Richardsonova[6] ekstrapolacija. Druga ugotovitev je, da lahko kot oceno za napako približka $F(h/2)$ vzamemo kar $(F(h/2) - F(h))/(2^p - 1)$.

Podobno bi lahko v primeru, ko za napako vemo, da ima obliko

$$F(h) = a_0 + a_1 h^{p_1} + a_2 h^{p_2} + \mathcal{O}(h^r), \tag{6.9}$$

kjer je $p_1 < p_2 < r$, iz $F(h)$, $F(h/2)$ in $F(h/4)$ izračunali približek $F(h, h/2, h/4)$, ki bi imel napako velikosti $\mathcal{O}(h^r)$. Tovrsten postopek bomo uporabili kasneje pri Rombergovi ekstrapolaciji.

Za zgled uporabimo Richardsonovo ekstrapolacijo na Simpsonovi formuli. Naj bo $S_h(f)$ Simpsonova formula s korakom h in $R_h(f)$ napaka Simpsonove formule pri koraku h. Vemo, da velja

$$I(f) = \int_a^b f(x)dx = S_h(f) + R_h(f) = S_{h/2}(f) + R_{h/2}(f).$$

Za napaki velja

$$R_h(f) = \frac{-(b-a)h^4}{180} f^{(4)}(\xi_1), \quad R_{h/2}(f) = \frac{-(b-a)h^4}{16 \cdot 180} f^{(4)}(\xi_2).$$

Če predpostavimo, da je $f^{(4)}(\xi_1) \approx f^{(4)}(\xi_2)$, dobimo

$$R_h(f) \approx 16 R_{h/2}(f).$$

[6] Lewis Fry Richardson (1881–1953) je bil angleški matematik, fizik in meteorolog. Prvi je uporabljal metodo končnih diferenc za numerično reševanje parcialnih diferencialnih enačb.

Iz $R_{h/2}(f) = I(f) - S_{h/2}(f) = S_h(f) + R_h(f) - S_{h/2}(f) \approx S_h(f) + 16R_{h/2}(f) - S_{h/2}(f)$ dobimo

$$R_{h/2}(f) \approx \frac{S_{h/2}(f) - S_h(f)}{15}.$$

To formulo lahko uporabimo za oceno napake Simpsonove formule. Ocena sicer ni preveč zanesljiva, saj smo izenačili odvode, a v večini primerov vseeno dobimo spodobne rezultate in lahko ocenimo velikostni razred napake. Poleg ocene lahko iz $S_{h/2}(f)$ in $S_h(f)$ z ekstrapolacijo dobimo še boljši približek, saj je

$$I(f) = S_{h/2}(f) + R_{h/2}(f) \approx \frac{16S_{h/2}(f) - S_h(f)}{15}.$$

Podobno lahko naredimo pri trapezni in sredinski formuli, le konstante so drugačne.

6.8 Adaptivne metode

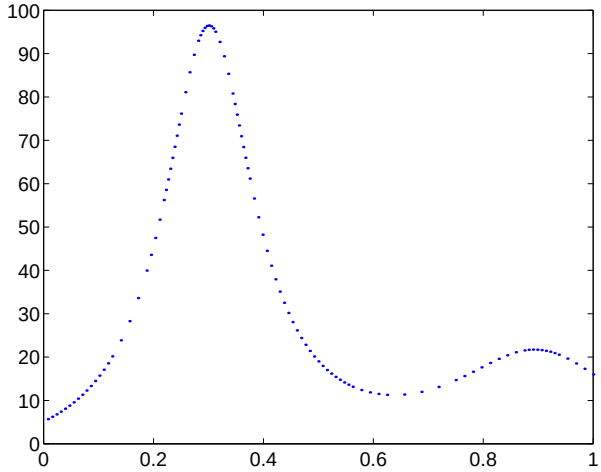
Večina metod, ki se jih v praksi uporablja za numerično integriranje, deluje na adaptivnem principu. To pomeni, da metoda sproti ocenjuje napako in temu prilagaja velikost podintervalov. Pri adaptivnih metodah tako na območjih, kjer je funkcija pohlevna, uporabimo večji razmik h, kjer je njeno obnašanje bolj divje, pa manjši h.

Rekurzivna adaptivna metoda, ki temelji na Simpsonovem pravilu in Richardsonovi ekstrapolaciji, je predstavljena v algoritmu 6.1.

Algoritem 6.1 Adaptivna Simpsonova metoda. Vhodni podatki so funkcija f, interval $[a, b]$ in toleranca δ. Metoda vrne vrednost integrala Q in oceno $\epsilon \leq \delta$ za napako tega približka.

$[Q, \epsilon]$ =AdaptivniSimpson(f, a, b, δ)
  $h = b - a$
  $c = (a + b)/2$
  $d = (a + c)/2, \quad e = (c + b)/2$
  $Q_1 = (h/6)(f(a) + 4f(c) + f(b))$
  $Q_2 = (h/12)(f(a) + 4f(d) + 2f(c) + 4f(e) + f(b))$
  $\epsilon = |Q_1 - Q_2|$
  if $\epsilon \leq \delta$
    $Q = Q_2 + (Q_2 - Q_1)/15$
  else
    $[Q_a, \epsilon_a]$ =AdaptivniSimpson$(f, a, c, \delta/2)$
    $[Q_b, \epsilon_b]$ =AdaptivniSimpson$(f, c, b, \delta/2)$
    $Q = Q_a + Q_b$
    $\epsilon = \epsilon_a + \epsilon_b$
  end

V algoritmu 6.1 prvi približek Q_1 za integral dobimo tako, da na celem intervalu uporabimo Simpsonovo pravilo, drugi približek Q_2 pa dobimo tako, da interval razdelimo na dva enaka dela in na vsakem uporabimo Simpsonovo pravilo. Potem primerjamo Q_1 in Q_2.



Slika 6.4: Primer uporabe adaptivnega kvadraturnega pravila v Matlabu. Vrednosti, označene na grafu, so bile uporabljene za izračun integrala. Vidi se, da točke niso ekvidistantne.

- Če je razlika $|Q_2 - Q_1|$ večja od zahtevane natančnosti δ, potem interval razdelimo na dva dela in na vsakem rekurzivno izračunamo integral z isto metodo, zahtevamo pa, da je sedaj napaka na vsaki polovici pod $\delta/2$.
- Če je $|Q_2 - Q_1| \leq \delta$, je Q_2 približek za integral, izračunan z zahtevano natančnostjo. Ker pa imamo na voljo še Q_1, lahko naredimo še korak Richardsonove ekstrapolacije in tako še izboljšamo rezultat.

Namesto Simpsonovega pravila bi lahko uporabili tudi kakšno drugo kvadraturno pravilo. V grobem pri adaptivnem integriranju vedno vrednost integrala ocenimo z dvema kvadraturnima praviloma. Njuna razlika nam služi kot ocena za napako. Če je razlika prevelika, interval razpolovimo in postopek rekurzivno ponovimo. Da bo postopek ekonomičen, moramo kvadraturni pravili izbrati tako, da se vozli čim bolj prekrivajo in da se vsaj del vozlov prenese v razpolovljeni interval. Tako dosežemo, da je končnih izračunov vrednosti funkcije čim manj, saj pri numeričnem integriranju zahtevnost merimo v tem, koliko vrednosti funkcije moramo izračunati, da pridemo do dovolj dobrega približka.

6.9 Rombergova metoda

Če je funkcija f na intervalu $[a, b]$ dovoljkrat zvezno odvedljiva, potem iz Euler-Maclaurinove[7] sumacijske formule sledi, da se ostanek trapezne formule izraža kot asimptotična vrsta potence h^2. To nam omogoča, da z zaporedno uporabo Richardsonove ekstrapolacije po vrsti uničimo vodilne člene in izboljšamo rezultat, ustrezen postopek pa se imenuje *Rombergova metoda*.

Izrek 6.3 *(Euler-Maclaurin) Če je funkcija f na intervalu $[a, b]$ $(2m + 2)$-krat zvezno odvedljiva, po-*

[7]Formulo sta neodvisno odkrila švicarski matematik Leonhard Euler (1707–1783) in škotski matematik Colin Maclaurin (1698–1746). Euler, eden najpomebnejših matematikov 18. stoletja, znan po številnih rezultatih iz analize, teorije števil in teorije grafov, je sumacijsko formulo uporabljal za računanje vsot počasi konvergentnih vrst, Maclaurin pa jo je uporabljal za računanje integralov.

tem velja sumacijska formula

$$I(f) = \int_a^b f(x)dx = T_h(f) - \sum_{k=1}^{m} \frac{B_{2k}}{(2k)!} h^{2k} \left(f^{(2k-1)}(b) - f^{(2k-1)}(a) \right) + R(f), \tag{6.10}$$

kjer je $T_h(f)$ trapezna formula s korakom h, B_k so Bernoullijeva števila, ostanek pa je enak

$$R(f) = -\frac{B_{2m+2}}{(2m+2)!}(b-a)h^{2m+2} f^{(2m+2)}(\xi)$$

za nek $\xi \in (a, b)$.

Dokaz in podrobno izpeljavo formule (6.10) lahko najdete npr. v [18].

Definicija 6.4 *Bernoullijeva*[8] *števila B_k so določena z razvojem*

$$\frac{x}{e^x - 1} = \sum_{k=0}^{\infty} \frac{B_k}{k!} x^k, \quad |x| < 2\pi.$$

Vsa Bernoullijeva števila so racionalna, vsa z lihimi indeksi razen B_1 pa so enaka 0. Nekaj prvih Bernoullijevih števil je

$$B_0 = 1, \quad B_1 = -\frac{1}{2}, \quad B_2 = \frac{1}{6}, \quad B_4 = -\frac{1}{30}, \quad B_6 = \frac{1}{42}, \quad B_8 = -\frac{1}{30}, \quad B_{10} = \frac{5}{66}.$$

Če zapišemo nekaj prvih členov vsote v (6.10), dobimo

$$I(f) = T_h(f) - \frac{h^2}{12}(f'(b) - f'(a)) + \frac{h^4}{720}(f'''(b) - f'''(a)) + \cdots.$$

Če poznamo nekaj začetnih lihih odvodov funkcije f v robnih točkah intervala, lahko tako z njimi izboljšamo približek $T_h(f)$, ki ga dobimo s trapezno formulo.

Za Rombergovo metodo je bistvena ugotovitev, da lahko (6.10) pišemo v obliki

$$I(f) = T_h(f) + \sum_{k=1}^{m} a_{k,0} h^{2k} + \mathcal{O}(h^{2m+2}),$$

pri čemer so koeficienti $a_{k,0}$ neodvisni od h. To je posplošitev formule (6.9) in z Richardsonovo ekstrapolacijo lahko iz približkov, ki jih dobimo za različne razmike h, dobimo boljši približek. Če uporabimo še razmika $h/2$ in $h/4$, dobimo

$$\begin{aligned}
I(f) &= T_h(f) + a_{1,0}h^2 + a_{2,0}h^4 + a_{3,0}h^6 + \cdots \\
I(f) &= T_{h/2}(f) + a_{1,0}\left(\frac{h}{2}\right)^2 + a_{2,0}\left(\frac{h}{2}\right)^4 + a_{3,0}\left(\frac{h}{2}\right)^6 + \cdots \\
I(f) &= T_{h/4}(f) + a_{1,0}\left(\frac{h}{4}\right)^2 + a_{2,0}\left(\frac{h}{4}\right)^2 + a_{3,0}\left(\frac{h}{4}\right)^6 + \cdots
\end{aligned}$$

[8] Števila je odkril švicarski matematik Jacob Bernoulli (1654–1705). Leta 1842 je Ada Lovelace zapisala algoritem, ki bi s pomočjo analitičnega stroja, ki ga je načrtoval Charles Babbage, računal Bernoullijeva števila. Čeprav stroj ni bil nikoli dokončan, velja Ada Lovelace za pionirko računalniškega programiranja.

Če enačbo s $T_{h/2}(f)$ pomnožimo s 4 in odštejemo od enačbe s $T_h(f)$, se znebimo vodilnega člena h^2 in dobimo natančnejši približek, podobno pa naredimo tudi s približkoma $T_{h/4}(f)$ in $T_{h/2}(f)$. Velja

$$\begin{aligned} I(f) &= T^{(1)}_{h/2}(f) + a_{2,1}h^4 + a_{3,1}h^6 + \cdots \\ I(f) &= T^{(1)}_{h/4}(f) + a_{2,1}\left(\frac{h}{2}\right)^4 + a_{3,1}\left(\frac{h}{2}\right)^6 + \cdots, \end{aligned}$$

kjer je

$$T^{(1)}_{h/2}(f) = \frac{4T_{h/2}(f) - T_h(f)}{3}, \quad T^1_{h/4}(f) = \frac{4T_{h/4}(f) - T_{h/2}(f)}{3}.$$

Postopek sedaj nadaljujemo tako, da se znebimo vodilnega člena h^4. Dobimo

$$I(f) = T^{(2)}_{h/4}(f) + a_{3,2}h^6 + a_{4,2}h^8 + \cdots,$$

kjer je

$$T^{(2)}_{h/4}(f) = \frac{16T^{(1)}_{h/4}(f) - T^{(1)}_{h/2}(f)}{15}.$$

V splošnem postopku tvorimo trikotno shemo

napaka	$\mathcal{O}(h^2)$	$\mathcal{O}(h^4)$	$\mathcal{O}(h^6)$	$\mathcal{O}(h^8)$	$\cdots$
	$T^{(0)}_h(f)$				
	$T^{(0)}_{h/2}(f)$	$T^{(1)}_{h/2}(f)$			
	$T^{(0)}_{h/4}(f)$	$T^{(1)}_{h/4}(f)$	$T^{(2)}_{h/4}(f)$		
	$T^{(0)}_{h/8}(f)$	$T^{(1)}_{h/8}(f)$	$T^{(2)}_{h/8}(f)$	$T^{(3)}_{h/8}(f)$	

s splošno formulo

$$T^{(j)}_{h/2^k}(f) = \frac{4^j T^{(j-1)}_{h/2^k}(f) - T^{(j-1)}_{h/2^{k-1}}(f)}{4^j - 1}$$

za $j = 0, \ldots, m$ in $k = m - j + 1, \ldots, m$.

Zgled 6.4 *Z Rombergovo metodo bomo izračunali* $\int_1^{2.2} \ln x\, dx = 0.5346062$. *Začeli bomo s* $h = 0.6$ *in nato naredili dve razpolavljanji.*

Dobimo:

$$\begin{aligned} T^{(0)}_h &= 0.6\left(\frac{1}{2}\ln 1 + \ln 1.6 + \frac{1}{2}\ln 2.2\right) = 0.5185394 \\ T^{(0)}_{h/2} &= \frac{1}{2}T^{(0)}_h + 0.3(\ln 1.3 + \ln 1.9) = 0.5305351 \\ T^{(0)}_{h/4} &= \frac{1}{2}T^{(0)}_{h/2} + 0.15(\ln 1.15 + \ln 1.45 + \ln 1.75 + \ln 2.05) = 0.5335847. \end{aligned}$$

Pomembno je, da $T_{h/2^k}$ *vedno računamo kot*

$$T_{h/2^k} = \frac{1}{2}T_{h/2^{k-1}} + \frac{h}{2^k}(y_1 + y_3 + \cdots + y_{2^k-1}).$$

Tako vsako funkcijsko vrednost izračunamo le enkrat in imamo z Rombergovo ekstrapolacijo zanemarljivo dodatnega dela v primerjavi z računanjem $T^{(0)}_{h/2^k}$, *rezultat pa je lahko mnogo natančnejši.*

Sedaj z Rombergovo ekstrapolacijo dobimo

$$\begin{aligned} T_{h/2}^{(1)} &= \frac{4T_{h/2}^{(0)} - T_h^{(0)}}{3} = 0.5345337 \\ T_{h/4}^{(1)} &= \frac{4T_{h/4}^{(0)} - T_{h/2}^{(0)}}{3} = 0.5346013 \\ T_{h/4}^{(2)} &= \frac{16T_{h/4}^{(1)} - T_{h/2}^{(1)}}{15} = 0.5346058. \end{aligned}$$

Vidimo, da je ekstrapolirana vrednost $T_{h/4}^{(2)}$ mnogo natančnejša od $T_{h/4}^{(0)}$, pri obeh pa smo uporabili vrednosti funkcije v istih točkah. □

Zgled 6.5 *Če isti postopek poskusimo na integralu $\int_0^1 \sqrt{x}\,dx = 2/3$ z začetnim $h = 0.5$, potem dobimo naslednjo tabelo*

$$\begin{array}{lll} T_h^{(0)}(f) = 0.6035534 & & \\ T_{h/2}^{(0)}(f) = 0.6432831 & T_{h/2}^{(1)}(f) = 0.6565263 & \\ T_{h/4}^{(0)}(f) = 0.6581302 & T_{h/4}^{(1)}(f) = 0.6630793 & T_{h/4}^{(2)}(f) = 0.6635162 \end{array}$$

Ekstrapolacija očitno ne deluje tako dobro kot v prejšnjem zgledu, razlog pa je, da funkcija ni odvedljiva v levi robni točki in zato pogoji izreka 6.3 niso izpolnjeni. □

6.10 Gaussove kvadraturne formule

Integral $\int_a^b f(x)\rho(x)dx$, kjer smo dodali še nenegativno utež ρ, aproksimiramo s kvadraturno formulo

$$\int_a^b f(x)\rho(x)dx = \sum_{i=0}^{n} A_i^{(n)} f(x_i^{(n)}) + R(f). \tag{6.11}$$

V primeru $\rho \equiv 1$ dobimo integral $\int_a^b f(x)dx$, tako da teorija iz tega razdelka pokrije tudi standardni primer.

Če uporabimo isto idejo kot v razdelku 6.2 in namesto f integriramo interpolacijski polinom, ugotovimo da so koeficienti prav tako določeni z vozli, saj velja

$$A_i^{(n)} = \int_a^b L_{n,i}(x)\rho(x)dx,$$

formula pa je točna za polinome stopnje vsaj n. S primerno izbiro vozlov lahko dosežemo, da bo formula (6.11) točna za polinome stopnje vsaj $2n+1$, v ozadju pa so ortogonalni polinomi.

Da je formula (6.11) lahko točna za polinome stopnje $2n+1$, je očitno že iz tega, da v njej nastopa $2n+2$ parametrov (vozlov in uteži). Po metodi nedoločenih koeficientov bi lahko te parametre nastavili tako, da bi bilo pravilo točno za polinome stopnje $2n+1$. Težava pri tem pristopu je, da dobimo nelinearen sistem, ki ga je težko numerično rešiti. Lažje je, če najprej s pomočjo ortogonalnih polinomov določimo vozle, uteži pa lahko izračunamo z integriranjem Lagrangeevih koeficientov.

Za funkciji f in g, definirani na intervalu $[a,b]$, definiramo njun skalarni produkt kot

$$\langle f,g\rangle = \int_a^b f(x)g(x)\rho(x)dx. \tag{6.12}$$

Funkciji sta si ortogonalni, če je $\langle f,g\rangle = 0$. Iz standardne baze polinomov $1, x, x^2, \ldots$ lahko z ortogonalizacijo dobimo ortonormirano bazo $P_0(x), P_1(x), P_2(x), \ldots$, kjer je P_i polinom stopnje i in velja

$$\langle P_i, P_k\rangle = \delta_{ik}.$$

Lema 6.5 *Naj bo P_{n+1} normiran polinom, ki je glede na skalarni produkt (6.12) ortogonalen na vse polinome stopnje manjše ali enake n. Potem so vse ničle polinoma P_{n+1} realne, enostavne in ležijo na intervalu (a,b).*

Dokaz. Lemo dokažemo s protislovjem. Denimo, da so $z_1, \ldots, z_k$ vse ničle polinoma P_{n+1}, ki ležijo na intervalu (a,b), pri čemer morebitne večkratne ničle štejemo le enkrat, in naj velja $k < n+1$. Sedaj definiramo polinom

$$q(x) = (x-z_1)^{j_1}\ldots(x-z_k)^{j_k},$$

kjer za potenco j_i vzamemo 1, če je z_i liha ničla polinoma P_{n+1}, oziroma 2 v primeru sode ničle za $i = 1, \ldots, k$. Za q očitno velja $\langle P_{n+1}, q\rangle \neq 0$, kar je protislovje, saj je v primeru $k < n+1$ stopnja polinoma q manjša kot $n+1$ in bi moral biti P_{n+1} ortogonalen na q. ∎

Po zgornji lemi lahko torej pišemo

$$P_{n+1}(x) = k_{n+1}(x-x_0^{(n)})\cdots(x-x_n^{(n)}),$$

kjer vse ničle $x_0^{(n)}, \ldots, x_n^{(n)}$ ležijo na intervalu (a,b). Pri *Gaussovi kvadraturni formuli* za vozle izberemo ravno točke $x_0^{(n)}, \ldots, x_n^{(n)}$. Pokažimo, da je tako dobljeno pravilo točno za vse polinome stopnje manjše ali enake $2n+1$.

Poljuben polinom p stopnje manjše ali enake $2n+1$ lahko zapišemo kot $p(x) = q(x)\omega(x) + r(x)$, kjer je je $\omega(x) = (x-x_0^{(n)})\cdots(x-x_n^{(n)})$ in sta q in r polinoma stopnje kvečjemu n. Ker sta polinoma q in ω ortogonalna, dobimo:

$$\begin{aligned}\int_a^b p(x)\rho(x)dx &= \int_a^b q(x)\omega(x)\rho(x)dx + \int_a^b r(x)\rho(x)dx \\ &= 0 + \sum_{i=0}^{n} A_i^{(n)} r(x_i^{(n)}) = \sum_{i=0}^{n} A_i^{(n)} p(x_i^{(n)}),\end{aligned}$$

saj v vozlih velja $r(x_i^{(n)}) = p(x_i^{(n)})$. Torej je pravilo res točno za vse polinome stopnje $2n+1$ ali manj.

Naslednja lema pove, da imajo Gaussove kvadraturne formule poleg tega, da imajo maksimalni možni red (glede na število uporabljenih vozlov), še to lepo lastnost, da so vse uteži pozitivne. Zaradi tega pri Gaussovih kvadraturnih formulah pri povečevanju števila točk nimamo težav z neodstranljivo napako.

Lema 6.6 *Uteži Gaussovih kvadraturnih pravil so pozitivne.*

Dokaz. Vzemimo

$$P_i(x) = \frac{\omega^2(x)}{(x - x_i^{(n)})^2}$$

za $i = 0, \ldots, n$. P_i je polinom stopnje $2n$, torej je zanj kvadraturno pravilo točno in velja

$$\int_a^b P_i(x)\rho(x)dx = \sum_{k=0}^{n} A_k P_i(x_k^{(n)}) = A_i P_i(x_i^{(n)}).$$

Ker je $P_i(x_i^{(n)}) > 0$ in je P_i nenegativna funkcija, mora biti $A_i > 0$. ■

Izrek 6.7 *Če je $f \in \mathcal{C}^{(2n+2)}[a, b]$, potem za napako Gaussovega kvadraturnega pravila na $n + 1$ vozlih velja*

$$\int_a^b f(x)\rho(x) = \sum_{i=0}^{n} A_i^{(n)} f(x_i^{(n)}) + \frac{f^{(2n+2)}(\xi)}{(2n+2)!k_{n+1}^2}.$$

Dokaz. Vzamemo Hermiteov interpolacijski polinom H stopnje $2n + 1$ v točkah $x_0^{(n)}, \ldots, x_n^{(n)}$, za katerega velja $H(x_i^{(n)}) = f(x_i^{(n)})$ in $H'(x_i^{(n)}) = f'(x_i^{(n)})$ za $i = 0, \ldots, n$. Iz

$$f(x) = H(x) + \frac{f^{(2n+2)}(\xi)}{(2n+2)!}\omega^2(x)$$

sledi

$$\int_a^b f(x)\rho(x)dx = \int_a^b H(x)\rho(x)dx + \int_a^b \frac{f^{(2n+2)}(\xi)}{(2n+2)!}\omega^2(x)dx.$$

Ker je H polinom stopnje $2n + 1$, velja

$$\int_a^b H(x)\rho(x)dx = \sum_{k=0}^{n} A_k H(x_k^{(n)}) = \sum_{k=0}^{n} A_k f(x_k^{(n)}),$$

za drugi integral pa velja

$$\int_a^b \frac{f^{(2n+2)}(\xi)}{(2n+2)!}\omega^2(x)dx = \int_a^b \frac{f^{(2n+2)}(\xi)}{(2n+2)!} \cdot \frac{P_{n+1}^2(x)}{k_{n+1}^2}dx = \frac{f^{(2n+2)}(\xi)}{(2n+2)!k_{n+1}^2}\int_a^b P_{n+1}^2(x)dx.$$ ■

Koeficiente A_i lahko izračunamo z integriranjem Lagrangeevih koeficientov ali pa uporabimo *Darboux–Cristoffelove*[9] *formule* (izpeljavo lahko najdete npr. v [18]):

$$A_i^{(n)} = \frac{1}{\sum_{j=0}^{n} P_j^2(x_i^{(n)})}, \quad i = 0, \ldots, n. \tag{6.13}$$

Eleganten numerični način računanja uteži in vozlov za Gaussovo kvadraturno formulo je, da problem prevedemo na iskanje lastnih vrednosti in lastnih vektorjev. Najprej pokažimo, da obstajajo taki koeficienti a_k in b_k, da ortonormirani polinomi zadoščajo tričlenski rekurzivni formuli

$$xP_k(x) = b_{k-1}P_{k-1}(x) + a_k P_k(x) + b_k P_{k+1}(x). \tag{6.14}$$

[9] Francoski matematik Jean Gaston Darboux (1842–1917) in nemški matematik Elwin Bruno Christoffel (1829–1900).

Omenili smo že, da za računanje baze ortonormiranih polinomov lahko uporabimo Gram–Schmidtovo ortogonalizacijo. Če poznamo polinome $P_0, \ldots, P_k$, potem lahko P_{k+1} dobimo tako, da polinom xP_k, ki je stopnje $k+1$, ortogonaliziramo glede na $P_0, \ldots, P_k$. Pred dokončnim normiranjem ima tako polinom $\widetilde{P}_{k+1}$ obliko

$$\widetilde{P}_{k+1} = xP_k - \sum_{j=1}^{k} \langle xP_k, P_j \rangle P_j.$$

V vsoti sta lahko neničelna le zadnja dva člena, saj za $j < k-1$ velja $\langle xP_k, P_j \rangle = \langle P_k, xP_j \rangle = 0$. Tako ostane

$$\widetilde{P}_{k+1} = xP_k - \langle xP_k, P_k \rangle P_k - \langle xP_k, P_{k-1} \rangle P_{k-1}.$$

Sedaj definiramo $a_j = \langle xP_j, P_j \rangle$ in $b_j = \|\widetilde{P}_{j+1}\| = \langle \widetilde{P}_{j+1}, \widetilde{P}_{j+1} \rangle^{(1/2)}$ za $j = 1, \ldots, k$. Iz izračuna

$$\langle xP_k, P_{k-1} \rangle = \langle P_k, xP_{k-1} \rangle = \langle P_k, \widetilde{P}_k \rangle = \|\widetilde{P}_k\| = b_{k-1}$$

sledi, da velja

$$b_k P_{k+1} = \widetilde{P}_{k+1} = xP_k - a_k P_k - b_{k-1} P_{k-1},$$

kar je ravno formula (6.14). Na podlagi tega lahko razvijemo postopek za izračun ortogonalnih polinomov, ki je predstavljen v algoritmu 6.2. Stranski produkt algoritma so konstante a_k in b_k iz zveze (6.14).

Algoritem 6.2 Algoritem za izračun ortogonalnih polinomov. Začetni podatki so interval $[a, b]$ in utež ρ, ki definirata skalarni produkt $\langle f, g \rangle = \int_a^b f(x)g(x)\rho(x)dx$. Algoritem vrne ortonormirane polinome $P_0, \ldots, P_{n+1}$.

$b_{-1} = \langle 1, 1 \rangle^{1/2}, \quad P_0 = 1/b_{-1}, \quad P_{-1} = 0$
$j = 0, 1, \ldots, n$
  $\widetilde{P} = xP_j$
  $a_j = \langle \widetilde{P}, P_j \rangle$
  $\widetilde{P} = \widetilde{P} - a_j P_j - b_{j-1} P_{j-1}$
  $b_j = \langle \widetilde{P}, \widetilde{P} \rangle^{1/2}$
  $P_{j+1} = (1/b_j)\widetilde{P}$

Če poznamo koeficiente a_k in b_k, potem so vozli $x_0^{(n)}, \ldots, x_n^{(n)}$ lastne vrednosti simetrične tridiagonalne matrike

$$T_n = \begin{bmatrix} a_0 & b_0 & & & \\ b_0 & a_1 & b_1 & & \\ & \ddots & \ddots & \ddots & \\ & & b_{n-2} & a_{n-1} & b_{n-1} \\ & & & b_{n-1} & a_n \end{bmatrix},$$

uteži pa dobimo iz prvih komponent ortonormiranih lastnih vektorjev

$$A_k^{(n)} = x_{1k}^2 \int_a^b \rho(x)dx,$$

kjer je X matrika ortonormiranih lastnih vektorjev matrike T_n.

Najbolj znane družine ortogonalnih polinomov so:[10]

$[-1,1]$,	$\rho(x)=1$,		(Legendre)
$[-1,1]$,	$\rho(x)=(1-x^2)^{-\frac{1}{2}}$,		(Čebišev 1. vrste)
$[-1,1]$,	$\rho(x)=(1-x^2)^{\frac{1}{2}}$,		(Čebišev 2. vrste)
$[-1,1]$,	$\rho(x)=(1-x)^{\alpha}(1+x)^{\beta}$,	$\alpha,\beta>-1$,	(Jacobi)
$[-1,1]$,	$\rho(x)=(1-x^2)^{\sigma-\frac{1}{2}}$,	$\sigma>\frac{1}{2}$,	(Gegenbauer)
$[0,\infty)$,	$\rho(x)=x^{\sigma}e^{-x}$,	$\sigma>-1$	(Laguerre)
$(-\infty,\infty)$,	$\rho(x)=e^{-x^2}$,		(Hermite).

Za te ortogonalne polinome se da poiskati tabelirane vozle in uteži za ustrezne Gaussove kvadraturne formule.

Zgled 6.6 *Gauss–Legendreovi kvadraturni formuli na dveh in treh točkah sta*

$$\int_{-1}^{1} f(x)dx = f\left(-\sqrt{\frac{1}{3}}\right) + f\left(\sqrt{\frac{1}{3}}\right) + \frac{1}{135}f^{(4)}(\xi),$$

$$\int_{-1}^{1} f(x)dx = \frac{5}{9}f\left(-\sqrt{\frac{3}{5}}\right) + \frac{8}{9}f(0) + \frac{5}{9}f\left(\sqrt{\frac{3}{5}}\right) + \frac{1}{15750}f^{(6)}(\xi).$$

Za primerjavo, pri trapeznem in Simpsonovem pravilu, kjer uporabimo isti interval in isto število vozlov, dobimo

$$\int_{-1}^{1} f(x)dx = f(-1) + f(1) - \frac{2}{3}f^{(2)}(\xi),$$

$$\int_{-1}^{1} f(x)dx = \frac{1}{3}f(-1) + \frac{4}{3}f(0) + \frac{1}{3}f(1) + \frac{1}{90}f^{(4)}(\xi).$$ □

Gaussove kvadraturne formule lahko posplošimo tako, da so nekateri vozli fiksni, ostale pa določimo tako, da bo formula točna za polinome čim višje stopnje. Tako pri Gauss–Lobattovih[11] formulah fiksiramo obe krajišči, pri Gauss–Radaujevih[12] formulah pa eno krajišče.

Gauss–Lobatto: fiksna vozla sta $x_0 = a$ in $x_n = b$, npr.

$$\int_{-1}^{1} f(x)dx = \frac{5}{6}f(-1) + \frac{1}{6}f\left(-\sqrt{\frac{1}{5}}\right) + \frac{1}{6}f\left(\sqrt{\frac{1}{5}}\right) + \frac{5}{6}f(1) + Cf^{(4)}(\xi).$$

Gauss–Radau: fiksen vozel je $x_0 = a$, npr.

$$\int_{-1}^{1} f(x)dx = \frac{2}{9}f(-1) + \frac{16+\sqrt{6}}{18}f\left(\frac{1-\sqrt{6}}{5}\right) + \frac{16-\sqrt{6}}{18}f\left(\frac{1+\sqrt{6}}{5}\right) + Cf^{(5)}(\xi).$$

[10] Francoski matematik Adrien-Marie Legendre (1752–1833) in avstrijski matematik Leopold Gegenbauer (1849–1903).

[11] Nizozemski matematik Rehuel Lobatto (1797-1866).

[12] Francoski astronom in matematik Jean Charles Rodolphe Radau (1835–1911).

6.11 Izlimitirani integrali

Poglejmo si nekaj možnih pristopov pri računanju izlimitiranih integralov, kjer imamo bodisi pol v krajišču ali pa neskončen interval. Pole in morebitne točke nezveznosti v notranjosti intervala moramo poznati že na začetku in potem interval razdeliti na take podintervale, da v nobenem ni pola ali nezveznosti v notranjosti intervala.

Denimo, da računamo

$$I = \int_a^b \frac{g(x)}{(x-a)^p} dx, \quad 0 < p < 1,$$

kjer je funkcija g zvezna na $[a, b]$. Po definiciji je

$$I = \lim_{\epsilon \to 0} \int_{a+\epsilon}^b \frac{g(x)}{(x-a)^p} dx.$$

Ta konvergenca je lahko prepočasna, da bi bila uporabna za praktično računanje. Poleg tega lahko pri formulah, kjer vozli vsebujejo tudi krajišča, pričakujemo velike napake pri računanju vrednosti v levem krajišču, ko bo ϵ blizu 0. Zato iščemo boljše možnosti.

Varianta 1: Integral I razdelimo na $I = I_1 + I_2$, kjer sta

$$I_1 = \int_a^{a+\epsilon} \frac{g(x)}{(x-a)^p} dx, \qquad I_2 = \int_{a+\epsilon}^b \frac{g(x)}{(x-a)^p} dx.$$

Integral I_2 izračunamo s standardnimi metodami, pri računanju integrala I_1 pa funkcijo g razvijemo v Taylorjevo vrsto okoli točke a:

$$g(x) = g(a) + (x-a)g'(a) + \frac{(x-a)^2}{2} g''(a) + \cdots.$$

Tako dobimo

$$I_1 = \epsilon^{1-p} \left(\frac{g(a)}{1-p} + \frac{\epsilon g'(a)}{1!(2-p)} + \frac{\epsilon^2 g''(a)}{2!(3-p)} + \cdots \right).$$

Varianta 2: Funkcijo g razvijemo v Taylorjevo vrsto okoli a:

$$g(x) = P_s(x) + \text{ostanek},$$

kjer je P_s polinom stopnje s. Sedaj I zapišemo kot $I = I_1 + I_2$, kjer sta

$$I_1 = \int_a^b \frac{P_s(x)}{(x-a)^p} dx, \qquad I_2 = \int_a^b \frac{g(x) - P_s(x)}{(x-a)^p} dx.$$

Integral I_1 lahko izračunamo eksplicitno, za računanje integrala I_2 pa uporabimo standardne numerične metode.

Varianta 3: Uporabimo substitucijo, npr.

$$(x-a) = t^m, \quad dx = m t^{m-1} dt, \quad m = \frac{k}{1-p}, \quad k \in \mathbb{N}.$$

Dobimo

$$I = m \int_0^{(b-a)^{1/m}} g(a + t^m) t^{k-1} dt$$

in lahko uporabimo standardne metode, saj smo se znebili pola.

Varianta 4: Pomagamo si z Gaussovimi kvadraturnimi formulami. Npr., če imamo

$$I = \int_{-1}^{1} \frac{g(x)}{\sqrt{1-x^2}}\, dx,$$

potem je utež $\frac{1}{\sqrt{1-x^2}}$ in lahko uporabimo Gauss–Čebiševe kvadraturne formule. Tako lahko npr. pridemo do formule

$$\int_{-1}^{1} \frac{g(x)}{\sqrt{1-x^2}}\, dx = \frac{\pi}{3}\left(g\left(-\frac{\sqrt{3}}{2}\right) + g(0) + g\left(\frac{\sqrt{3}}{2}\right)\right) + Cf^{(6)}(\xi).$$

Pri računanju integrala $I = \int_a^\infty f(x)dx$ na neskončnem intervalu lahko I razdelimo na $I = I_1 + I_2$, kjer sta

$$I_1 = \int_a^b f(x)dx, \qquad I_2 = \int_b^\infty f(x)dx.$$

Integral I_1 izračunamo normalno s standardnimi metodami. Variante za izračun I_2 so

- I_2 zanemarimo, če poznamo kakšno dobro oceno za $|I_2|$.
- Naredimo substitucijo $u = 1/x$ in dobimo integral po končnem intervalu

$$I_2 = -\int_{1/b}^{0} f(1/u)u^{-2}du.$$

Tudi tu si lahko mogoče pomagamo z Gaussovimi kvadraturnimi formulami.

6.12 Večdimenzionalni integrali

Denimo, da računamo integral funkcije f dveh spremenljivk po pravokotniku $\Omega = [a,b] \times [c,d]$. Pišemo lahko

$$\int_\Omega f(x,y)dxdy = \int_a^b dx \int_c^d f(x,y)dy.$$

Če interval $[a,b]$ razdelimo s točkami $x_i = a + ih$, $i = 0,\ldots,n$, $h = (b-a)/n$, interval $[c,d]$ razdelimo s točkami $y_j = c + jk$, $j = 0,\ldots,m$, $k = (d-c)/m$, in za integrale uporabimo trapezno pravilo, dobimo

$$\int_a^b dx \int_c^d f(x,y)dy = \frac{hk}{4}\sum_{i=0}^{n}\sum_{j=0}^{m} A_{ij} f(x_i,y_j) + \mathcal{O}(h^2+k^2),$$

kjer za koeficiente A_{ij} velja $A_{ij} = A_1(i)A_2(j)$ in $A_1(0) = A_1(n) = 1$, $A_1(i) = 2$ za $i \in \{1,\ldots,n-1\}$, ter $A_2(0) = A_2(m) = 1$, $A_2(j) = 2$ za $j \in \{1,\ldots,m-1\}$. V bistvu dobimo tenzorski produkt sestavljenega trapeznega pravila, podobno pa lahko naredimo tudi za ostala sestavljena pravila.

Zgled 6.7 *Če imamo npr. mrežo* 5×4

(x_0, y_3)	(x_1, y_3)	(x_2, y_3)	(x_3, y_3)	(x_4, y_3)
(x_0, y_2)	(x_1, y_2)	(x_2, y_2)	(x_3, y_2)	(x_4, y_2)
(x_0, y_1)	(x_1, y_1)	(x_2, y_1)	(x_3, y_1)	(x_4, y_1)
(x_0, y_0)	(x_1, y_0)	(x_2, y_0)	(x_3, y_0)	(x_4, y_0)

potem so koeficienti pri trapeznem pravilu

$$\frac{hk}{4} \cdot \begin{array}{|c|c|c|c|c|} \hline 1 & 2 & 2 & 2 & 1 \\ \hline 2 & 4 & 4 & 4 & 2 \\ \hline 2 & 4 & 4 & 4 & 2 \\ \hline 1 & 2 & 2 & 2 & 1 \\ \hline \end{array}.$$

Če pa bi npr. uporabili Simpsonovo metodo za mrežo 5×5*, bi dobili*

$$\frac{hk}{9} \cdot \begin{array}{|c|c|c|c|c|} \hline 1 & 4 & 2 & 4 & 1 \\ \hline 2 & 8 & 4 & 8 & 2 \\ \hline 4 & 16 & 8 & 16 & 4 \\ \hline 2 & 8 & 4 & 8 & 2 \\ \hline 1 & 4 & 2 & 4 & 1 \\ \hline \end{array}.$$

Če je $\Omega = [0,1]^d$ in v vsaki dimenziji porabimo n točk, uporabimo pa sestavljeno pravilo reda k, potem za dobljeno tenzorsko pravilo velja, da je napaka reda $\mathcal{O}(N^{-k/d})$, kjer je $N = n^d$ število vseh točk.

Težava je, da pri velikem d prvič potrebujemo zelo veliko točk (n^d), po drugi strani pa napaka prepočasi pada glede na število uporabljenih točk. Zaradi tega se pri velikih n bolje obnaša metoda Monte Carlo, kjer lahko z znosnim številom točk ponavadi dobimo zadovoljiv približek.

6.13 Metoda Monte Carlo

Integral $I(f) = \int_0^1 f(x)dx$ je enak povprečni vrednosti funkcije f na intervalu $[0,1]$. To je osnova za *metodo Monte Carlo*.

Če je X slučajna spremenljivka, enakomerno porazdeljena po $[0,1]$, potem za matematično upanje velja $E(f(X)) = I(f)$. Približek za matematično upanje je

$$I_N(f) = \frac{1}{N} \sum_{i=1}^{N} f(X_i),$$

kjer so X_i neodvisne naključne vrednosti z $[0,1]$.

Za napako $\epsilon_N(f) = I(f) - I_N(f)$ velja

$$\epsilon \approx \sigma(f) N^{-1/2},$$

kjer je $\sigma(f) = \sqrt{D(f)}$ standardna deviacija,

$$D(f) = \int_0^1 (f(x) - I(f))^2 dx$$

pa varianca funkcije f.

Pri računanju integrala po $\Omega = [0,1]^d$ velja enako, le na mestu X_i sedaj izbiramo vektorje iz Ω, kjer je vsak element naključno število med 0 in 1.

Za integral po območju $I^d = [0,1]^d$ dobimo

$$I[f] = E[f(\boldsymbol{X})] = \int_{I^d} f(\boldsymbol{X}) d\boldsymbol{X},$$

kjer je $\boldsymbol{X}$ vektor neodvisnih slučajnih spremenljivk $X_1, \ldots, X_d$, enakomerno porazdeljenih na $[0,1]$. Za napako še vedno velja, da pada z $\mathcal{O}(N^{-1/2})$, neodvisno od d.

Če iz kvadraturne formule reda k s tenzorskimi produkti izpeljemo mrežno kvadraturno formulo za integriranje po I^d, potem pri N točkah napaka pada z $\mathcal{O}(N^{-k/d})$, saj velja $N = n^d$, napaka pa pada kot n^{-k}.

Ostale razlike med mrežnimi kvadraturnimi formulami in metodo Monte Carlo so:

- Mreža pri mrežnih kvadraturnih pravilih je pri velikem d prevelika. Če npr. uporabimo tenzorsko trapezno pravilo, potem potrebujemo vsaj 2^d izračunov vrednosti funkcije.
- Monte Carlo metode so tudi v eni dimenziji primernejše za računanje Fourierovih koeficientov.
- Monte Carlo metoda nima težav z nezveznostmi, ki sicer zmanjšajo red kvadraturnih pravil.

Z računalnikom ne moremo generirati pravih naključnih števil, saj jih vedno generiramo z nekim procesom, ki ga lahko ponovimo. Tako v resnici delamo s psevdonaključnimi števili. Ena izmed preprostih metod za njihovo generiranje je *linearni kongruenčni model*, kjer računamo zaporedje

$$x_{r+1} = ax_r + c \pmod{m},$$

kjer sta a in c določeni pozitivni konstanti, izbiramo pa števila med 0 in $m-1$. Pri tem a, c in m izberemo tako, da je perioda (števila se najkasneje po m korakih začnejo ponavljati) čim večja.

Model ima še to težavo, da če generiramo naključne d-dimenzionalne vektorje, potem točke ležijo na $(d-1)$-dimenzionalnih ravninah, ki jih je približno $m^{1/d}$.

6.14 Numerično integriranje v Matlabu

Na voljo so naslednje funkcije:

- `quad(f,a,b,tol)`: adaptivno Simpsonovo pravilo,
- `quadl(f,a,b,tol)`: adaptivno Gauss–Lobatto–Kronrodovo pravilo na 4+7 točkah,
- `dlbquad(fun,a,b,c,d,tol)`: adaptivno Simpsonovo pravilo na pravokotniku.
- `trapz(x,y)`: sestavljeno trapezno pravilo.

Argumenti so:

- f, fun: funkcija, podana v inline obliki, delovati mora tudi, če je argument vektor,
- a,b,c,d: meje območij ($[a,b]$ oziroma $[a,b]\times[c,d]$),
- tol: relativna natančnost, ko prenehamo rekurzivno deliti podinterval.
- x,y: vektorja točk x_i in $y_i = f(x_i)$.

Primeri uporabe:

- `f=inline('cos(x.^2)'); quad(f,0,1)`
- `f=inline('cos(x.^2)+sin(y.^3)'); dblquad(f,0,1,0,1)`
- `x=linspace(0,pi,100); y=sin(x); trapz(x,y)`

6.15 Numerično seštevanje vrst

Če integral prevedemo na računanje vrste ali pa tudi kako drugače, se srečamo s problemom, ko je potrebno sešteti vrsto

$$S = \sum_{k=1}^{\infty} a_k,$$

za katero vemo, da je konvergentna. Iščemo torej limito zaporedja delnih vsot

$$S_n = \sum_{k=1}^{n} a_k.$$

V številnih primerih lahko z ustrezno transformacijo pohitrimo konvergenco delnih vsot in tako pridemo do dovolj točnega približka z računanjem relativno malo členov vrste.

Prva takšna transformacija je *Aitkenova δ^2-transformacija*. Kadar je zaporedje delnih vsot $S_n = \sum_{k=1}^{n} a_k$ počasi konvergentno in se obnaša podobno kot geometrijska vrsta, potem velikokrat dobimo hitrejše zaporedje z uporabo transformacije

$$T(S_n) = \frac{S_{n+1}S_{n-1} - S_n^2}{S_{n+1} - 2S_n + S_{n-1}}.$$

Transformacijo lahko ponovno uporabimo na novem zaporedju $T(S_n)$. Metoda se obnaša dobro pri konvergentnih alternirajočih zaporedjih.

Posplošitev Aitkenove transformacije je *Wynnova ϵ-metoda*

$$\epsilon_{r+1}(S_n) = \epsilon_{r-1}(S_n) + \frac{1}{\epsilon_r(S_{n+1}) - \epsilon_r(S_n)},$$

kjer vzamemo $\epsilon_0(S_n) = S_n$ in $\epsilon_{-1}(S_n) = 0$. Metoda se obnaša dobro pri konvergentnih alternirajočih zaporedjih.

Pri *Eulerjevi transformaciji* konvergentno alternirajočo vrsto

$$\sum_{k=0}^{\infty}(-1)^k a_k = a_0 - a_1 + a_2 - a_3 + \cdots$$

spremenimo v hitreje konvergentno vrsto z isto vsoto

$$\sum_{k=0}^{\infty}\frac{(-1)^k \Delta^k a_0}{2^{k+1}},$$

kjer je

$$\Delta^k a_0 = \sum_{j=0}^{k}(-1)^j \binom{k}{j} a_{k-j}$$

k-ta prema diferenca a_0.

Pomagamo si lahko tudi z Euler–Maclaurinovo formulo, ki smo jo uporabili pri razvoju Rombergove metode. Pri računanju določenega integrala vrednost le tega aproksimiramo s trapezno formulo. Če pa zamenjamo, lahko vrednost vsote za trapezno formulo aproksimiramo z integralom in preostalimi členi v obliki

$$\sum_{k=1}^{N} f(k) = \int_1^N f(x)dx + \frac{1}{2}(f(N+1)+f(1)) + \sum_{k=1}^{m} B_{2k}\frac{f^{(2k-1)}(N+1) - f^{(2k-1)}(1)}{(2k)!} + R_m,$$

kjer so B_m Bernoullijeva števila. Za napako velja

$$R_m = -N \cdot B_{2m+2}\frac{f^{(2m+2)}(\xi)}{(2m+2)!}.$$

Tako dobimo

$$\sum_{k=1}^{N-1} f(k) = \int_1^N f(x)dx - \frac{1}{2}(f(0)-f(N)) + \frac{1}{12}(f'(N)-f'(0))$$
$$-\frac{1}{720}\left(f^{(3)}(N)-f^{(3)}(0)\right) + \frac{1}{302240}\left(f^{(5)}(N)-f^{(5)}(0)\right) - \frac{1}{1209600}\left(f^{(7)}(N)-f^{(7)}(0)\right).$$

Tako lahko npr. izpeljemo formulo za $\sum_{k=1}^{n} k^4$. Dobimo

$$\sum_{k=1}^{n} k^4 = \frac{1}{5}n^5 + \frac{1}{2}n^4 + \frac{1}{12}(4n^3) - \frac{1}{720}(24n),$$

višji odvodi pa so enaki 0. Ostane

$$\sum_{k=1}^{n} k^4 = \frac{1}{30}n(n+1)(2n+1)(3n^2+2n-1).$$

Dodatna literatura

Numerično integriranje je obravnavano skoraj v vseh učbenikih numerične analize. V slovenščini lahko več o tem preberete v knjigi [18], osnovne informacije so tudi v knjigah [3] in [34], od tuje literature pa omenimo [6] in [17].

Poglavje 7

Diferencialne enačbe

7.1 Uvod

Rešujemo *začetni problem prvega reda* v obliki

$$\begin{aligned} y' &= f(x,y) \\ y(x_0) &= y_0, \end{aligned}$$

kjer je f dana dovolj gladka funkcija, zanima pa nas rešitev na intervalu $[x_0, b]$. Želimo, da je problem dobro definiran, kar pomeni, da rešitev obstaja in je enolična.

Zgled 7.1 *Začetni problem $y' = 1 + y^2$, $y(0) = 0$ ima analitično rešitev $y(x) = \tan x$. Rešitev desno od $x = 0$ vedno bolj strmo narašča, v levo stran pa pada, in pri končni vrednosti x (pri $\pm\pi/2$) pridemo do $\pm\infty$. Torej rešitev obstaja le na intervalu $(-\pi/2, \pi/2)$. Ta zgled kaže, da se lahko zgodi, da rešitev začetnega problema ni definirana na celotnem intervalu $[x_0, b]$.* □

Zgled 7.2 *Začetni problem $y' = y^{2/3}$, $y(0) = 0$ ima dve rešitvi, saj tako trivialna rešitev $y_1(x) = 0$ kot $y_2(x) = x^3/27$ ustrezata tako diferencialni enačbi kot začetnemu pogoju. To je zgled začetnega problema, ki nima enolične rešitve.* □

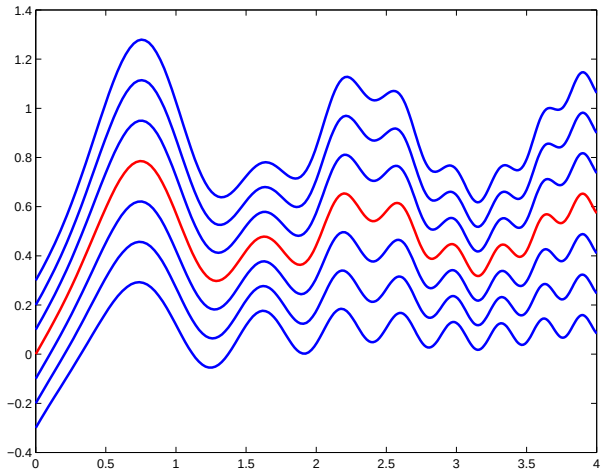
Numerična rešitev bo sestavljena iz zaporedja točk $x_0 < x_1 < x_2 < \ldots$ in pripadajočega zaporedja vrednosti $y_0, y_1, y_2, \ldots$, kjer je y_n izračunani približek za rešitev začetnega problema v x_n, oziroma

$$y_n \approx y(x_n), \quad n = 0, 1 \ldots,$$

kjer je $y(x_n)$ točna vrednost rešitve v x_n. Pri tem sodobne metode delujejo na adaptivnem principu in avtomatično določajo velikosti korakov $h_i = x_{i+1} - x_i$ tako, da napake približkov y_n ostanejo v vnaprej določenih mejah.

Metode za reševanje začetnega problema delimo na:

- *enokoračne metode*: y_{n+1} izračunamo iz y_n.
- *večkoračne metode*: y_{n+1} izračunamo iz $y_n, y_{n-1}, \ldots, y_{n-k+1}$, kjer je $k \geq 1$.



Slika 7.1: Rešitev diferencialne enačbe $y' = \cos(3x^2) + \sin(4x)y$ pri različnih začetnih pogojih $y(0) = -0.3, -0.2, \ldots, 0.3$.

Metode ločimo še na:

- *eksplicitne metode*: imamo direktno formulo za y_{n+1},
- *implicitne metode*: y_{n+1} dobimo tako, da rešimo nelinearno enačbo.

Slika 7.1 prikazuje tipično obnašanje rešitev diferencialne enačbe pri različnih začetnih pogojih. Vsakemu začetnemu pogoju pripada svoja veja. Težava pri reševanju začetnega problema je, da je veja, ki ji želimo slediti, določena z vrednostjo v začetni točki. Ko pa med računanjem zaidemo s prave veje, nimamo več nobene informacije, ki bi nas vrnila nanjo, saj se metode obnašajo tako, kot da bi problem začeli reševati znova iz zadnje točke.

Sistem diferencialnih enačb prvega reda

$$\begin{aligned} y_1' &= f_1(x, y_1, y_2, \ldots, y_k) \\ y_2' &= f_2(x, y_1, y_2, \ldots, y_k) \\ &\vdots \\ y_k' &= f_k(x, y_1, y_2, \ldots, y_k) \end{aligned}$$

z začetnimi pogoji

$$y_1(x_0) = y_{10},\ y_2(x_0) = y_{20},\ \ldots,\ y_k(x_0) = y_{k0}$$

rešujemo tako kot eno enačbo, če ga zapišemo v vektorski obliki

$$Y' = F(x, Y),$$

kjer je $Y = [y_1\ y_2\ \cdots\ y_k]^T$. Zaradi tega lahko vse metode, ki jih bomo spoznali v nadaljevanju, uporabljamo tako za eno samo enačbo kot tudi za sistem diferencialnih enačb.

7.2 Obstoj rešitve, občutljivost in stabilnost

V tem razdelku bomo na kratko ponovili teorijo s področja diferencialnih enačb, ki jo bomo potrebovali v nadaljevanju. Zanima nas, kdaj imamo zagotovljen obstoj enolične rešitve, za samo numerično reševanje pa sta pomembni tudi občutljivost problema in stabilnost rešitve.

Funkcija $f(x,y)$, kjer je $f : \mathbb{R}^{k+1} \to \mathbb{R}^k$, na območju $D = [a,b] \times \Omega \subset \mathbb{R}^{k+1}$ *zadošča Lipschitzovemu pogoju* na y s konstanto L, če za poljubna $(x,y_1),(x,y_2) \in D$ velja

$$\|f(x,y_1) - f(x,y_2)\| \leq L\|y_1 - y_2\|.$$

Zadostni pogoj, da f zadošča Lipschitzovemu pogoju, je, da je f zvezno odvedljiva po y, saj potem lahko vzamemo

$$L = \max_{(x,y)\in D} \|Jf_y(x,y)\|,$$

kjer je Jf_y $k \times k$ Jacobijeva matrika f glede na y:

$$Jf_y(x,y) = \begin{bmatrix} \frac{\partial f_1}{\partial y_1}(x,y) & \cdots & \frac{\partial f_1}{\partial y_k}(x,y) \\ \vdots & & \vdots \\ \frac{\partial f_k}{\partial y_1}(x,y) & \cdots & \frac{\partial f_k}{\partial y_k}(x,y) \end{bmatrix}.$$

V primeru navadne diferencialne enačbe, kjer je $k = 1$, je $Jf_y(x,y)$ kar $f_y(x,y)$.

Če f ustreza Lipschitzovemu pogoju, potem za vsak $(x_0,y_0) \in D$ obstaja podinterval $[a,b]$, ki vsebuje x_0, na katerem rešitev začetnega problema $y' = f(x,y)$, $y(x_0) = y_0$ obstaja in je enolična.

7.2.1 Občutljivost rešitve začetnega problema

Pri občutljivosti nas zanima, kako se spremeni rešitev, če se spremenijo začetni podatki. Tu bomo ločili dva primera.

V prvem imamo motnjo samo v začetnem pogoju, kar ustreza temu, da rešujemo isto diferencialno enačbo, a smo na drugi veji. To se dogaja med numeričnim reševanjem začetnega problema, saj se v vsakem koraku zaradi lokalne napake premaknemo na drugo vejo. Denimo, da f ustreza Lipschitzovemu pogoju. Točna rešitev začetnega problema $y' = f(x,y)$, $y(x_0) = y_0$ naj bo $y(x)$. Če je $\widetilde{y}(x)$ rešitev začetnega problema z zmotenim začetnim pogojem $y' = f(x,y)$, $y(x_0) = \widetilde{y}_0$, potem za poljuben $x \geq x_0$ velja

$$\|\widetilde{y}(x) - y(x)\| \leq e^{L(x-x_0)}\|\widetilde{y}_0 - y_0\|.$$

V drugem primeru obravnavamo tako motnjo v začetnem pogoju kot tudi motnjo v diferencialni enačbi. Tu nas zanima, kaj se zgodi, če npr. zapleteno diferencialno enačbo nadomestimo z lažje izračunljivo aproksimacijo. Če namesto začetnega problema $y' = f(x,y)$, $y(x_0) = y_0$, rešujemo bližnji problem $\widehat{y}' = \widehat{f}(x,\widehat{y})$, $\widehat{f}(x_0) = \widehat{y}_0$, potem velja

$$\|\widehat{y}(x) - y(x)\| \leq e^{L(x-x_0)}\|\widehat{y}_0 - y_0\| + \frac{e^{L(x-x_0)} - 1}{L}\|\widehat{f} - f\|,$$

kjer je $\|\widehat{f} - f\| = \max_{(x,y)\in D} \|\widehat{f}(x,y) - f(x,y)\|$,

7.2.2 Stabilnost rešitve začetnega problema

Ponavadi smo stabilnost omenjali v povezavi z numeričnimi metodami. Pri diferencialnih enačbah pa govorimo o stabilnosti rešitve problema, ki ni povezana z numeričnimi metodami.

Definicija 7.1 *Pravimo, da je rešitev začetnega problema* $y' = f(x,y)$, $y(x_0) = y_0$ stabilna, *če za vsak* $\epsilon > 0$ *obstaja tak* $\delta > 0$, *da za* $\widetilde{y}(x)$, *ki je rešitev* $\widetilde{y}' = f(x,\widetilde{y})$, $\widetilde{y}(x_0) = \widetilde{y}_0$, *kjer je* $\|y_0 - \widetilde{y}_0\| \leq \delta$, *velja* $\|\widetilde{y}(x) - y(x)\| \leq \epsilon$ *za vse* $x \geq x_0$.

Če ima začetni problem stabilno rešitev, lahko pričakujemo, da bo rešitev pri zmotenem začetnem pogoju ostala blizu točne rešitve.

Pravimo, da je stabilna rešitev *asimptotično stabilna*, če gre $\|\widetilde{y}(x) - y(x)\|$ proti 0, ko gre x proti neskončnosti. Pri asimptotični rešitvi lahko pričakujemo, da bo napaka zmotenega začetnega pogoja šla proti 0, ko gre x proti neskončnosti.

Zgled 7.3 *Rešitev diferencialne enačbe* $y'(x) = \lambda y$, $y(0) = y_0$, *kjer je* $\lambda = a + ib \in \mathbb{C}$, *je*

$$y(x) = y_0 e^{\lambda x} = y_0 e^{ax}(\cos(bx) + i\sin(bx)).$$

Stabilnost rešitve je odvisna od a. Rešitev je asimptotično stabilna pri $\mathrm{Re}(\lambda) < 0$, *stabilna pri* $\mathrm{Re}(\lambda) = 0$, *in nestabilna pri* $\mathrm{Re}(\lambda) > 0$. □

Zgled 7.4 *Imamo sistem* $y' = Ay$, $y_0 = y_0$, *kjer je A matrika* $k \times k$. *Če se da A diagonalizirati in so njene lastne vrednosti* $\lambda_1, \ldots, \lambda_k$ *z lastnimi vektorji* $v_1, \ldots, v_k$, *potem je rešitev*

$$y(x) = \sum_{i=1}^{k} \alpha_i e^{\lambda_i x} v_i,$$

kjer so koeficienti α_i *določeni z*

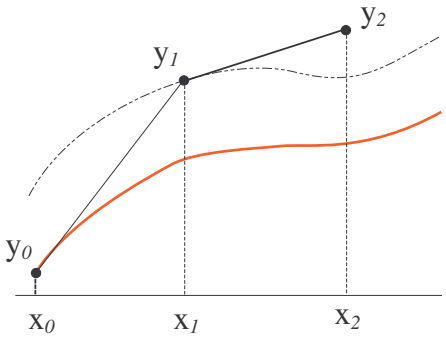
$$y_0 = \sum_{i=1}^{k} \alpha_i v_i.$$

Rešitev je asimptotično stabilna, če za vse lastne vrednosti velja $\mathrm{Re}(\lambda_i) < 0$; *stabilna, če velja* $\mathrm{Re}(\lambda_i) \leq 0$ *za vse lastne vrednosti; in nestabilna, če obstaja lastna vrednost* λ_i, *kjer je* $\mathrm{Re}(\lambda_i) > 0$. □

7.3 Enokoračne metode

7.3.1 Eulerjeva metoda

V prvem koraku reševanja začetnega problema $y' = f(x,y)$, $y(x_0) = y_0$ se moramo iz točke (x_0, y_0) premakniti v točko (x_1, y_1), kjer je y_1 približek za vrednost $y(x_1)$ in $x_1 = x_0 + h$. Poleg vrednosti (x_0, y_0) lahko v tej točki iz diferencialne enačbe izračunamo še smer tangente $y'_0 = f(x_0, y_0)$. Če je h dovolj majhen, ne bomo naredili velike napake, če bomo rešitev na intervalu $[x_0, x_1]$ aproksimirali s premico in se premaknili v smeri tangente.



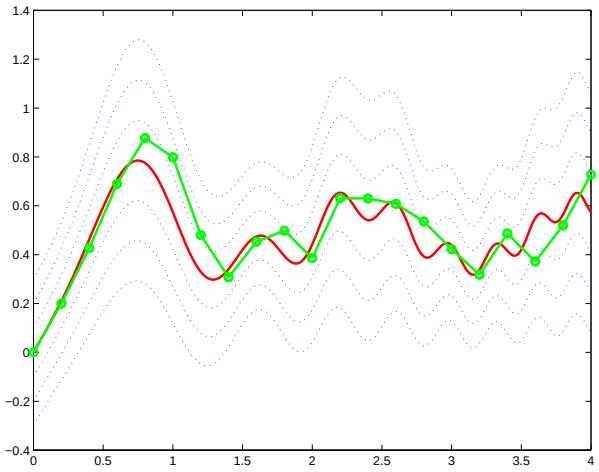
Slika 7.2: Eksplicitna Eulerjeva metoda.

Tako pridemo do najpreprostejše enokoračne metode, ki se imenuje *eksplicitna Eulerjeva metoda*. Nastavek za en korak je

$$\begin{aligned} y_{n+1} &= y_n + hf(x_n, y_n) \\ x_{n+1} &= x_n + h. \end{aligned}$$

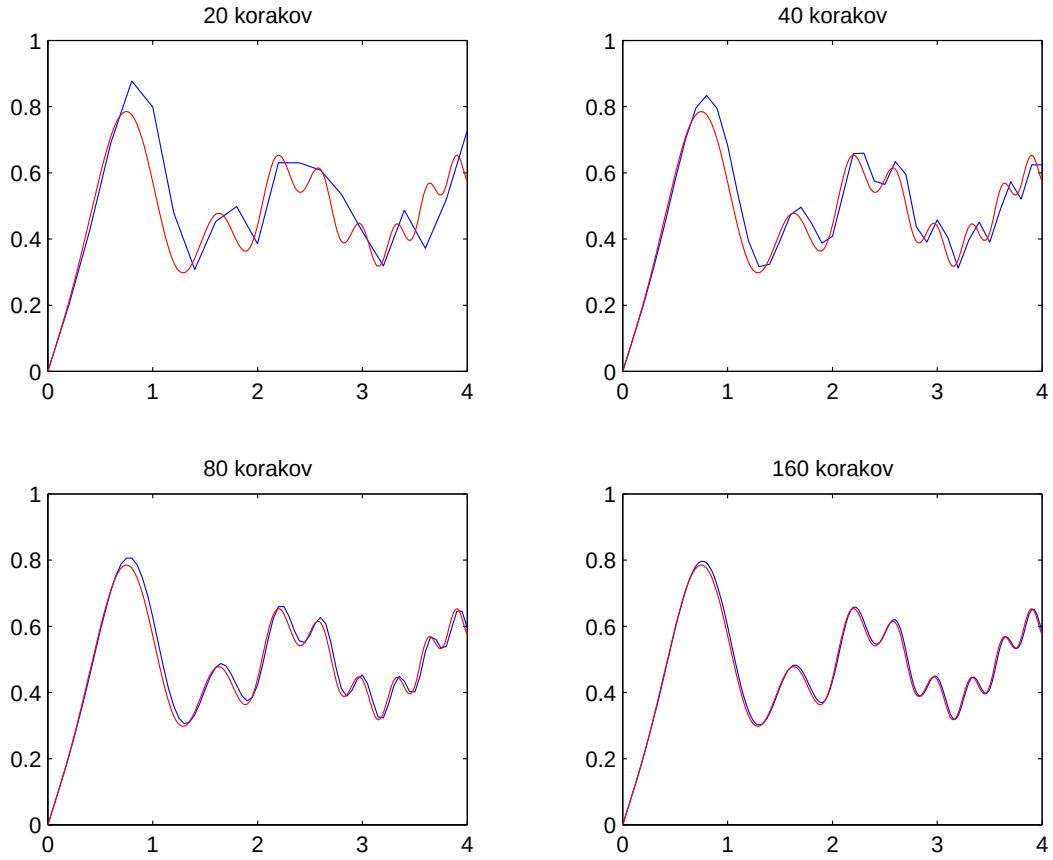
Metoda je prikazana na sliki 7.2. V vsaki točki se premaknemo v smeri tangente. Ponavadi uporabljamo fiksen h, lahko pa tudi velikost h v vsakem koraku zmanjšamo ali povečamo odvisno od obnašanja rešitve.

Zgled 7.5 *Oglejmo si delovanje eksplicitne Eulerjeve metode na primeru diferencialne enačbe* $y' = \cos(3x^2) + \sin(4x)y$, $y(0) = 1$. *Denimo, da nas zanima rešitev na intervalu* $[0, 4]$.



Slika 7.3: Delovanje eksplicitne Eulerjeve metode na enačbi $y' = \cos(3x^2) + \sin(4x)y$, $y(0) = 1$ na intervalu $[0, 4]$ s premikom $h = 0.2$.

Če vzamemo $h = 0.2$ *in uporabimo eksplicitno Eulerjevo metodo, potem je graf numerično dobljene rešitve na sliki 7.3 predstavljen z zeleno barvo, medtem, ko je točna rešitev pobarvana rdeče. Vidimo, da je premik* $h = 0.2$ *prevelik, da bi rešitev lahko sledila vseh lokalnim ekstremom, ki jih ima točna rešitev.*



Slika 7.4: Delovanje eksplicitne Eulerjeve metode na enačbi $y' = \cos(3x^2) + \sin(4x)y$, $y(0) = 1$ na intervalu $[0, 4]$ s premiki $h = 0.2, 0.1, 0.05, 0.025$.

Če zmanjšamo premik h se poveča čas računanja, a kot kaže slika 7.4, se z manjšanjem h povečuje natančnost dobljene rešitve. Ko gre h proti 0, numerična rešitev konvergira proti točni rešitvi. □

Poleg eksplicitne poznamo tudi *implicitno Eulerjevo metodo*, ki je podana z

$$\begin{aligned} y_{n+1} &= y_n + hf(x_{n+1}, y_{n+1}) \\ x_{n+1} &= x_n + h. \end{aligned}$$

Pri implicitni metodi rešitev na intervalu $[x_n, x_{n+1}]$ še vedno aproksimiramo s premico, le da tokrat za smerni koeficient vzamemo vrednost odvoda v točki (x_{n+1}, y_{n+1}). Ker y_{n+1} nastopa na obeh straneh enačbe, je potrebno pri implicitni metodi v vsakem koraku rešiti nelinearni sistem za y_{n+1}.

7.3.2 Taylorjeva vrsta

Pri eksplicitni Eulerjevi metodi smo rešitev aproksimirali s tangento. Če bi imeli na voljo še višje odvode, potem bi lahko uporabili razvoj v Taylorjevo vrsto in rešitev aproksimirali s polinomom stopnje višje od ena.

Za razvoj v Taylorjevo vrsto potrebujemo višje odvode y. Dobimo jih tako, da odvajamo diferencialno enačbo. Če predpostavimo, da je f dovoljkrat zvezno odvedljiva, potem z odvajanjem enačbe $y' = f(x, y)$ dobimo

$$\begin{aligned} y' &= f \\ y'' &= f_x + f_y y' = f_x + f_y f \\ y''' &= f_{xx} + 2f_{xy}f + f_{yy}f^2 + f_y(f_x + f_y f) \\ &\vdots \end{aligned}$$

Preko razvoja v Taylorjevo vrsto lahko potem izračunamo

$$y(x_1) = y(x_0 + h) = y_0 + hy_0' + \frac{h^2}{2}y_0'' + \cdots.$$

Zgled 7.6 *Z metodo razvoja v Taylorjevo vrsto bomo naredili en korak za začetni problem* $y' = xy + 1$, $y(0) = 0$. *Vzeli bomo* $h = 0.2$ *in uporabili prve štiri člene razvoja v Taylorjevo vrsto.*

Z odvajanjem diferencialne enačbe dobimo

$$\begin{aligned} y' = xy + 1 &\implies y'(0) = 1 \\ y'' = xy' + y &\implies y''(0) = 0 \\ y''' = xy'' + 2y' &\implies y'''(0) = 2 \end{aligned}$$

$$y(h) = h + \frac{1}{3}h^3 + \cdots.$$

Pri $h = 0.2$ *tako dobimo*

$$y_1 = 0.2 + 0.00267 = 0.20267.$$

Če iščemo $y(0.4)$, *nadaljujemo iz točke* $(0.2, 0.20267)$ *in ponovimo postopek.* □

Očitno bo napaka izračunane rešitve manjša, če uporabimo več členov razvoja v Taylorjevo vrsto. Na osnovi tega definiramo lokalno napako metode.

Definicija 7.2 *Pravimo, da ima metoda* lokalno napako *reda k, če se pri točni vrednosti* $y_n = y(x_n)$ *izračunani* y_{n+1} *ujema z razvojem* $y(x_n + h)$ *v Taylorjevo vrsto okrog* x_n *do vključno člena* h^k.

Lokalna napaka predstavlja razliko, ki nastopi v enem samem koraku. Pove nam, koliko se naslednji približek razlikuje od vrednosti na veji, kjer smo pričeli z računanjem, torej točki (x_n, y_n).

Metoda iz zadnjega zgleda ima red 3, sicer pa z razvijanjem v Taylorjevo vrsto lahko dobimo metodo poljubnega reda. Eulerjeva metoda (tako eksplicitna kot implicitna) ima red 1.

7.3.3 Runge-Kutta metode

Runge-Kutta metode so še vedno enokoračne. Da določimo čim natančnejši premik, izračunamo vrednost funkcije f v m točkah. Tako dobimo m premikov v smereh tangent, na koncu pa sestavimo premik iz uteženega povprečja dobljenih premikov.

Najprej izračunamo m koeficientov

$$k_i = hf(x_n + \alpha_i h, y_n + \sum_{j=1}^{i} \beta_{ij} k_j), \quad i = 1, \ldots, m,$$

nato pa je naslednji približek enak

$$y_{n+1} = y_n + \sum_{i=1}^{m} \gamma_i k_i.$$

Pri tem je m stopnja Runge-Kutta metode, ki je ne smemo zamenjevati z redom metode. Konstante α_i, β_{ij} in γ_i določimo tako, da se y_{n+1} čim bolje ujema z razvojem $y(x_n + h)$ v Taylorjevo vrsto. Izkaže se, da mora veljati $\alpha_i = \sum_{j=1}^{i} \beta_{ij}$ in $\sum_{i=1}^{m} \gamma_i = 1$.

V primeru, ko je $\beta_{ii} = 0$ za $i = 1, \ldots, m$, je metoda eksplicitna, sicer pa implicitna.

Zgled 7.7 *Dvostopenjska eksplicitna Runge-Kutta metoda ima obliko*

$$\begin{aligned} k_1 &= hf(x_n, y_n) \\ k_2 &= hf(x_n + \alpha h, y_n + \beta k_1) \\ y_{n+1} &= y_n + \gamma_1 k_1 + \gamma_2 k_2. \end{aligned}$$

S pomočjo razvoja funkcije f v Taylorjevo vrsto kot funkcijo dveh spremenljivk dobimo razvoja

$$\begin{aligned} k_1 &= hf \\ k_2 &= hf + \alpha h^2 f_x + \beta h k_1 f_y + \mathcal{O}(h^3). \end{aligned}$$

Od tod sledi, da za naslednji približek y_{n+1} vzamemo

$$y_{n+1} = y_n + (\gamma_1 + \gamma_2) hf + \gamma_2 \alpha h^2 f_x + \gamma_2 \beta h^2 f f_y + \mathcal{O}(h^3),$$

kar primerjamo z

$$y(x_n + h) = y(x_n) + hf + \frac{1}{2} h^2 (f_x + f f_y) + \mathcal{O}(h^3).$$

Sledi

$$\begin{aligned} \gamma_1 + \gamma_2 &= 1 \\ \alpha \gamma_2 &= \tfrac{1}{2} \\ \beta \gamma_2 &= \tfrac{1}{2}. \end{aligned}$$

Sistem ima več rešitev, saj za poljubni $\gamma_2 \neq 0$ dobimo

$$\gamma_1 = 1 - \gamma_2, \quad \alpha = \frac{1}{2\gamma_2}, \quad \beta = \frac{1}{2\gamma_2}.$$

□

Primera ekplicitne dvostopenjske Runge-Kutta metode drugega reda sta:

- *Heunova metoda*

$$\begin{aligned} k_1 &= hf(x_n, y_n) \\ k_2 &= hf(x_n + h, y_n + k_1) \\ y_{n+1} &= y_n + \tfrac{1}{2}(k_1 + k_2), \quad \text{lokalna napaka : } \mathcal{O}(h^3); \end{aligned}$$

- *modificirana Eulerjeva metoda*

$$\begin{aligned} k_1 &= hf(x_n, y_n) \\ k_2 &= hf(x_n + \tfrac{1}{2}h, y_n + \tfrac{1}{2}k_1) \\ y_{n+1} &= y_n + k_2, \quad \text{lokalna napaka : } \mathcal{O}(h^3). \end{aligned}$$

Zelo znana je tudi Runge-Kutta 4-stopenjska metoda reda 4, kjer vzamemo

$$\begin{aligned} k_1 &= hf(x_n, y_n) \\ k_2 &= hf(x_n + \tfrac{1}{2}h, y_n + \tfrac{1}{2}k_1) \\ k_3 &= hf(x_n + \tfrac{1}{2}h, y_n + \tfrac{1}{2}k_2) \\ k_4 &= hf(x_n + h, y_n + k_3) \\ y_{n+1} &= y_n + \tfrac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4), \quad \text{lokalna napaka : } \mathcal{O}(h^5). \end{aligned}$$

Zgled 7.8 *Denimo, da s 4-stopenjsko Runge-Kutta metodo rešujemo* $y' = -y - 5e^x \sin(x)$, $y(0) = 1$, *kjer vzamemo* $h = 0.1$. *Dobimo*

$$\begin{aligned} k_1 &= hf(x_0, y_0) = 0.1 * f(0, 1) = -0.1 \\ k_2 &= hf(x_0 + \tfrac{1}{2}h, y_0 + \tfrac{1}{2}k_1) = 0.1 * f(0.05, 0.95) = -0.12127 \\ k_3 &= hf(x_0 + \tfrac{1}{2}h, y_0 + \tfrac{1}{2}k_2) = 0.1 * f(0.05, 0.93936) = -0.12021 \\ k_4 &= hf(x_0 + h, y_0 + k_3) = 0.1 * f(0.1, 0.87979) = -0.14315 \end{aligned}$$

$$y_1 = y_0 + \tfrac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) = 0.87898. \qquad \square$$

Izkaže se, da je pri eksplicitnih Runge-Kutta metodah stopnje 5 ali več red vedno manjši od stopnje. Tako je npr. maksimalni red 5-stopenjske eksplicitne Runge-Kutta metoda enak 4, 6-stopenjske pa 5.

7.3.4 Adaptivna ocena koraka

Če imamo na voljo oceno lokalne napake, lahko h adaptivno prilagajamo. Denimo, da po 4-stopenjski Runge-Kutta metodi reda 4 iz $y(x)$ izračunamo $y(x + 2h)$ enkrat s korakom $2h$, drugič pa v dveh korakih s korakom h. Podobno kot pri Richardsonovi ekstrapolaciji dobimo

$$\begin{aligned} y(x + 2h) &= y^{(1)} + (2h)^5 \cdot C_1 + \mathcal{O}(h^6) \\ y(x + 2h) &= y^{(2)} + 2(h)^5 \cdot C_2 + \mathcal{O}(h^6) \end{aligned}$$

in

$$\Delta = \frac{y^{(2)} - y^{(1)}}{15}$$

je ocena za lokalno napako približka $y^{(1)}$. Ko je ocena Δ velika h zmanjšamo, če pa je ocena dovolj majhna lahko h povečamo. Za izračun Δ in $y^{(1)}$ potrebujemo 11 izračunov vrednosti funkcije f (4 izračune za vsako Runge-Kutta metodo, pri čemer se izračun $f(x_n, y_n)$ ponovi dvakrat).

Boljša je *Runge-Kutta–Fehlbergova metoda*, kjer vzamemo 6 stopenjsko Runge-Kutta metodo

$$k_i = hf(x_n + \alpha_i h, y_n + \sum_{j=1}^{i-1} \beta_{ij} k_j), \quad i = 1, \ldots, 6,$$

potem pa iz istih $k_1, \ldots, k_6$ sestavimo metodo reda 4

$$y_{n+1} = y_n + \sum_{i=1}^{6} \gamma_i k_i$$

in kontrolno metodo reda 5

$$y^*_{n+1} = y_n + \sum_{i=1}^{6} \gamma^*_i k_i.$$

Ocena za napako y_{n+1} je potem kar $y_{n+1} - y^*_{n+1} = \sum_{i=1}^{6} (\gamma_i - \gamma^*_i) k_i$.

Podrobne formule za Runge-Kutta–Fehlbergovo metodo so

$$\begin{aligned}
k_1 &= hf(x_n, y_n) \\
k_2 &= hf(x_n + \tfrac{1}{4}h, y_n + \tfrac{1}{4}k_1) \\
k_3 &= hf(x_n + \tfrac{3}{8}h, y_n + \tfrac{3}{32}k_1 + \tfrac{9}{32}k_2) \\
k_4 &= hf(x_n + \tfrac{12}{13}h, y_n + \tfrac{1932}{2197}k_1 - \tfrac{7200}{2197}k_2 + \tfrac{7296}{2197}k_3) \\
k_5 &= hf(x_n + h, y_n + \tfrac{439}{216}k_1 - 8k_2 + \tfrac{3680}{513}k_3 - \tfrac{845}{4104}k_4) \\
k_6 &= hf(x_n + \tfrac{1}{2}h, y_n - \tfrac{8}{27}k_1 + 2k_2 - \tfrac{3544}{2565}k_3 + \tfrac{1859}{4104}k_4 - \tfrac{11}{40}k_5) \\
y_{n+1} &= y_n + \tfrac{25}{216}k_1 + \tfrac{1408}{2565}k_3 + \tfrac{2197}{4104}k_4 - \tfrac{1}{5}k_5, \quad \text{lokalna napaka : } \mathcal{O}(h^5) \\
y^*_{n+1} &= y_n + \tfrac{16}{135}k_1 + \tfrac{6656}{12825}k_3 + \tfrac{28561}{56430}k_4 - \tfrac{9}{50} + \tfrac{2}{55}k_6, \quad \text{lokalna napaka : } \mathcal{O}(h^6).
\end{aligned}$$

Opazimo lahko, da za izračun y_{n+1} potrebujemo le koeficiente $k_1, \ldots, k_5$, torej je vrednost y_{n+1} v bistvu dobljena s 5-stopenjsko metodo. Koeficient k_6 potrebujemo le za izračun kontrolne vrednosti y^*_{n+1}.

V naslednjem koraku vzamemo razmik qh, kjer je ϵ željena natančnost in

$$q = \left(\frac{\epsilon h}{2|y^*_{n+1} - y_{n+1}|} \right)^{1/4}.$$

Potrebno je poudariti, da pri Runge-Kutta–Fehlbergovi metodi za naslednji približek vzamemo y_{n+1}, kljub temu, da imamo na voljo po vsej verjetnosti natančnejši približek y^*_{n+1}. To naredimo zaradi tega, ker iz razlike $y_{n+1} - y^*_{n+1} = \mathcal{O}(h^5)$ lahko ocenimo vodilni člen lokalne napake y_{n+1}, tega pa ne moremo narediti za y^*_{n+1}. Tako ima Runge-Kutta–Fehlbergova metoda red 4.

7.4 Stabilnost in konvergenca enokoračnih metod

Numerične metode so primerne za uporabo le, če so stabilne in konvergentne. V tem razdelku bomo pokazali, kdaj lahko ti lastnosti predpostavimo za enokoračne metode. Navedli bomo le definicije in glavne izreke. Podrobnosti skupaj z dokazi izrekov lahko najdete v [18].

Vsako enokoračno metodo lahko zapišemo v obliki

$$y_{n+1} = y_n + h\phi(x_n, y_n, h),$$

kjer je ϕ *funkcija prirastka*.

Definicija 7.3 *Enokoračna metoda za numerično reševanje začetnega problema je* konsistentna, *če velja*

$$\lim_{h\to 0} \phi(x, y, h) = f(x, y).$$

Zgled 7.9 *Funkcija prirastka za modificirano Eulerjevo metodo*

$$y_{n+1} = y_n + hf(x_n + \frac{h}{2}, y_n + \frac{1}{2}hf(x_n, y_n))$$

je $\phi(x_n, y_n, h) = f(x_n + \frac{h}{2}, y_n + \frac{1}{2}hf(x_n, y_n))$. *Metoda je očitno konsistentna.* □

Lokalno napako pri izračunu y_{n+1} lahko zapišemo kot

$$T(x_{n+1}) = y(x_{n+1}) - y(x_n) - h\phi(x_n, y_n, h).$$

Če velja $T(x_{n+1}) = \mathcal{O}(h^{p+1})$, potem je metoda reda p.

Numerična metoda za reševanje začetnega problema je stabilna, kadar majhne motnje ne povzročijo divergence numerične rešitve.

Definicija 7.4 *Denimo, da iščemo rešitev začetnega problema na intervalu* $[a, b]$, *ki ga razdelimo z ekvidistantnimi točkami* $x_i = a + ih$, *kjer je* $h = (b - a)/n$ *in* $i = 0, \ldots, n$. *Pravimo, da je enokoračna metoda* stabilna, *če za vsako diferencialno enačbo, ki zadošča Lipschitzovemu pogoju, obstajata taki konstanti* $h_0 > 0$ *in* $K > 0$, *da za poljubni dve rešitvi* y_r, $\widetilde{y}_r$, *ki ju dobimo z* $0 < h \leq h_0$, *velja* $\|y_r - \widetilde{y}_r\| \leq K\|y_0 - \widetilde{y}_0\|$ *za* $x_r \in [a, b]$.

Stabilnost numerične metode ni povezana s stabilnostjo rešitve diferencialne enačbe, ki smo jo definirali v podrazdelku 7.2.2.

Izrek 7.5 *Če funkcija prirastka* ϕ *zadošča Lipschitzovemu pogoju na* y, *potem je metoda stabilna.*

Definicija 7.6 *Enokoračna metoda je* konvergentna, *če za vsako diferencialno enačbo, ki zadošča Lipschitzovemu pogoju, za vsak r velja*

$$\lim_{h\to 0} \|y_r - y(x_r)\| = 0.$$

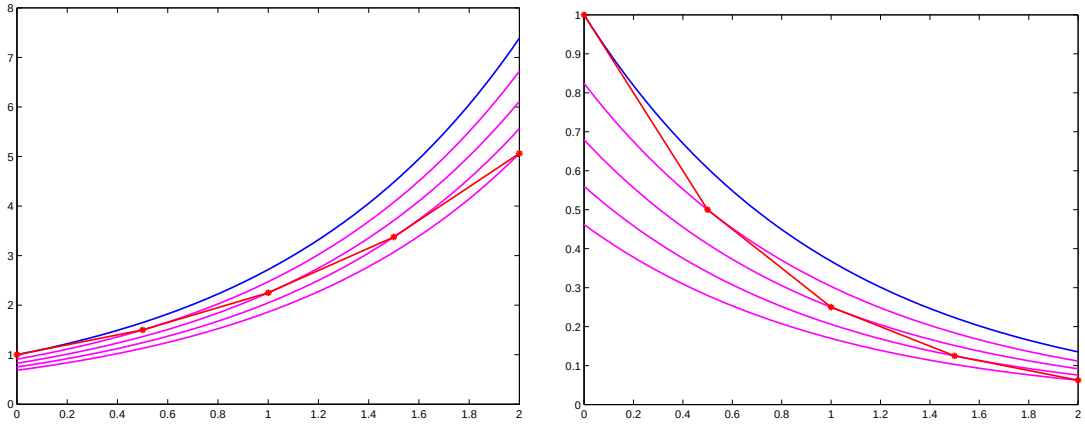
Pri tem predpostavimo, da v limiti upoštevamo le tako majhne h, da je $rh \leq |b - a|$.

Izrek 7.7 *Če je funkcija prirastka ϕ zvezna v x, y, h in zadošča Lipschitzovemu pogoju na y, potem je metoda konvergentna natanko tedaj, ko je metoda konsistentna.*

Izrek 7.8 (O globalni napaki) *Če funkcija prirastka ϕ zadošča pogojem za konvergenco in za lokalno napako velja $\|T(x)\| \leq Dh^{p+1}$, potem za globalno napako velja*

$$\|y_n - y(x_n)\| \leq Dh^p \frac{e^{L(x_n - x_0)} - 1}{L} + e^{L(x_n - x_0)} \|y_0 - y(x_0)\|.$$

Globalna napaka ni preprosto kar vsota lokalnih napak, v grobem pa velja, da ima metoda z lokalno napako reda k globalno napako reda $k-1$. Primera na sliki 7.5 prikazujeta dva primera povezave globalne in lokalne napake. V prvem primeru (levo) je globalna napaka večja od vsote absolutnih vrednosti lokalnih napak. V desnem primeru pa je globalna napaka manjša od posameznih lokalnih napak.



Slika 7.5: Povezava med globalno in lokalno napako.

Zgled 7.10 *Stabilnost diferencialne enačbe lahko razberemo iz obnašanja pri reševanju* modelne enačbe *$y' = \lambda y$, $y(0) = y_0$. Ta diferencialna enačba ima rešitev $y(x) = y_0 e^{\lambda x}$. V primeru, ko je $\mathrm{Re}(\lambda) < 0$, je rešitev asimptotično stabilna.*

Če vzamemo eksplicitno Eulerjevo metodo, potem dobimo $y_1 = (1 + \lambda h) y_0$ in

$$y_k = (1 + \lambda h)^k y_0.$$

Če naj bo pri $\mathrm{Re}(\lambda) < 0$ Eulerjeva metoda stabilna, mora veljati $|1 + \lambda h| < 1$, torej mora biti h omejen ($h\lambda$ mora ležati v krogu z radijem 1 okrog -1).

Če vzamemo implicitno Eulerjevo metodo, potem dobimo $(1 - \lambda h) y_1 = y_0$ in

$$y_k = \left(\frac{1}{1 - \lambda h} \right)^k y_0.$$

Sedaj mora pri $\mathrm{Re}(\lambda) < 0$ veljati $\left| \frac{1}{1-\lambda h} \right| \leq 1$, torej h ni omejen *in za ta začetni problem je implicitna Eulerjeva metoda vedno stabilna.* □

7.5 Večkoračne metode

Pri večkoračnih metodah poleg zadnjega uporabimo tudi še nekaj prejšnjih približkov. Splošni nastavek je

$$\sum_{i=0}^{k} \alpha_i y_{n+1-i} + h \sum_{i=0}^{k} \beta_i f_{n+1-i} = 0,$$

kjer je $f_i = f(x_i, y_i)$, privzamemo pa še $\alpha_0 = 1$. Če je $\beta_0 = 0$, je metoda eksplicitna, sicer pa implicitna. Metodo določata *rodovna polinoma*

$$\begin{aligned} \rho(z) &= \sum_{i=0}^{k} \alpha_{k-i} z^i, \\ \sigma(z) &= \sum_{i=0}^{k} \beta_{k-i} z^i. \end{aligned}$$

Ker na začetku potrebujemo več kot eno začetno vrednost, moramo te približke pridobiti s kakšno drugo metodo. Lahko uporabimo npr. Runge-Kutta metodo ali razvijanje v Taylorjevo vrsto, paziti pa moramo na to, da ne uporabimo metode, ki ima manjši red lokalne napake kot večkoračna metoda. Pri reševanju začetnega problema je namreč tako, da ko so napake enkrat prisotne, se tudi z natančnejšim računanjem v nadaljevanju ne moremo več približati točni rešitvi.

Adamsove metode dobimo tako, da enačbo $y' = f(x, y)$ integriramo na $[x_n, x_{n+1}]$

$$y_{n+1} - y_n = \int_{x_n}^{x_{n+1}} y' dx = \int_{x_n}^{x_{n+1}} f(x, y) dx,$$

f pa nadomestimo z interpolacijskim polinomom na točkah:

a) $x_n, x_{n-1}, \ldots, x_{n-k+1}$: eksplicitne *Adams–Bashworthove formule*

$$\begin{aligned} y_{n+1} &= y_n + h f_n, \quad \text{lokalna napaka } \mathcal{O}(h^2), \\ y_{n+1} &= y_n + \frac{h}{2}(3f_n - f_{n-1}), \quad \text{lokalna napaka } \mathcal{O}(h^3), \\ y_{n+1} &= y_n + \frac{h}{12}(23f_n - 16f_{n-1} + 5f_{n-2}), \quad \text{lokalna napaka } \mathcal{O}(h^4). \end{aligned}$$

b) $x_{n+1}, x_n, \ldots, x_{n-k+2}$: implicitne *Adams–Moultonove formule*

$$\begin{aligned} y_{n+1} &= y_n + h f_{n+1}, \quad \text{lokalna napaka } \mathcal{O}(h^2), \\ y_{n+1} &= y_n + \frac{h}{2}(f_{n+1} + f_n), \quad \text{lokalna napaka } \mathcal{O}(h^3), \\ y_{n+1} &= y_n + \frac{h}{12}(5f_{n+1} + 8f_n - f_{n-1}), \quad \text{lokalna napaka } \mathcal{O}(h^4). \end{aligned}$$

Pri predpostavki, da je $k = n$ in $x_0 = 0$, definiramo linearni funkcional

$$L(y) = \sum_{i=0}^{k} \Big(\alpha_i y((k-i)h) + h\beta_i y'((k-i)h) \Big) = \sum_{j=0}^{k} \Big(\alpha_{k-j} y(jh) + h\beta_{k-j} y'(jh) \Big).$$

Če y in y' razvijemo v Taylorjevo vrsto okrog 0, dobimo

$$L(y) = d_0 y(0) + d_1 h y'(0) + d_2 h^2 y''(0) + \cdots .$$

Metoda je reda p, če je $L(y) = \mathcal{O}(h^{p+1})$ za vsako $(p+1)$-krat zvezno odvedljivo funkcijo y.

Izrek 7.9 *Za večkoračno metodo je ekvivalentno:*

a) $d_0 = d_1 = \cdots = d_m = 0$,

b) $L(q) = 0$ *za poljuben polinom* q *stopnje kvečjemu* m,

c) $L(y) = \mathcal{O}(h^{m+1})$ *za vsako* $(m+1)$*-krat zvezno odvedljivo funkcijo* y.

Red metode je p, če velja $d_0 = d_1 = \cdots = d_p = 0 \neq d_{p+1}$.

Če v Taylorjevo vrsto razvijemo y in y', dobimo

$$\begin{aligned} y(jh) &= \sum_{r=0}^{\infty} \frac{(jh)^r}{r!} y^{(r)}(0), \\ y'(jh) &= \sum_{r=0}^{\infty} \frac{(jh)^r}{r!} y^{(r+1)}(0). \end{aligned}$$

Sedaj dobimo naslednje formule za $d_0, d_1, d_2, \ldots$:

$$\begin{aligned} d_0 &= \sum_{j=0}^{k} \alpha_{k-j}, \\ d_1 &= \sum_{j=0}^{k} (j\alpha_{k-j} + \beta_{k-j}), \\ d_2 &= \sum_{j=0}^{k} \left(\frac{j^2}{2}\alpha_{k-j} + j\beta_{k-j} \right), \\ &\vdots \\ d_q &= \sum_{j=0}^{k} \left(\frac{j^q}{q!}\alpha_{k-j} + \frac{j^{q-1}}{(q-1)!}\beta_{k-j} \right). \end{aligned}$$

Zgled 7.11 *Določi red večkoračne metode* $y_n - y_{n-2} = \frac{h}{3}(f_n + 4f_{n-1} + f_{n-2})$.

Koeficienti so $\alpha_0 = -1$, $\alpha_1 = 0$, $\alpha_2 = 1$, $\beta_0 = \frac{1}{3}$, $\beta_1 = \frac{4}{3}$, $\beta_2 = \frac{1}{3}$. *Po zgornjih formulah lahko izračunamo* $d_0 = d_1 = \cdots = d_4 = 0$ *in* $d_5 = -\frac{1}{90}$, *kar pomeni, da je red metode enak* 4. □

Ponavadi pri večkoračnih metodah uporabljamo *prediktor-korektor metode*, kjer za izračun prediktorja y_{n+1}^P vzamemo eksplicitno metodo, za korektor pa vzamemo implicitno metodo, katere enačbo rešujemo iterativno tako, da na desno stran vstavimo približek y_{n+1}^P, na levi pa dobimo y_{n+1}^K. Tako se izognemo reševanju nelinearne enačbe oziroma je to kar en korak navadne iteracije za reševanje nelinearne enačbe. Postopek lahko ponovimo tako da dobljeni približek vstavimo na desno stran in izračunamo nov približek. Teorija nam zagotavlja, da bo ta postopek skonvergiral, če je h dovolj majhen.

Primer so *Milneove metode*, kjer $y' = f(x,y)$ integriramo na $[x_{n-k}, x_{n+1}]$:

$$y_{n+1} - y_{n-k} = \int_{x_{n-k}}^{x_{n+1}} f(x,y)dx,$$

integral pa računamo preko Newton–Cotesovih formul. Pri odprti dobimo eksplicitno, pri zaprti pa implicitno metodo. Primer je Milneov par

$$\begin{aligned} y_{n+1} &= y_{n-3} + \frac{4h}{3}(2f_n - f_{n-1} + 2f_{n-2}), \quad \text{napaka } \mathcal{O}(h^5) \quad : \text{ prediktor} \\ y_{n+1} &= y_{n-1} + \frac{h}{3}(f_{n+1} + 4f_n + f_{n-1}), \quad \text{napaka } \mathcal{O}(h^5) \quad : \text{ korektor} \end{aligned}$$

Zgled 7.12 *Diferencialno enačbo* $y' = -y - 5e^{-x}\sin(5x)$ *z začetnim pogojem* $y(0) = 1$ *rešujemo z Milneovim parom prediktor-korektor*

$$\begin{aligned} y_{n+1}^{(P)} &= y_{n-3} + \frac{4h}{3}(2f_n - f_{n-1} + 2f_{n-2}), \\ y_{n+1}^{(K)} &= y_{n-1} + \frac{h}{3}(f(x_{n+1}, y_{n+1}^{(P)}) + 4f_n + f_{n-1}). \end{aligned}$$

Denimo, da smo pri $h = 0.1$ *z drugimi metodami že prišli do približkov* $y_1 = 0.79407$, $y_2 = 0.44235$, $y_3 = 0.05239$, *sedaj pa bi z Milneovim parom radi izračunali* y_4.

$y_4^{(P)} = 1 + \frac{0.4}{3}(2 \cdot (-3.7472) + 3.8870 + 2 \cdot (-2.9631)) = -0.27115$

$y_4^{(K)} = 0.44235 + \frac{0.1}{3}(-2.7765 + 4 \cdot (-3.7472) - 3.8870) = -0.27939$

Postopek lahko nadaljujemo, $y_4^{(K)}$ *vnesemo na desno stran in dobimo nov približek.*

$y_4^{(K2)} = 0.44235 + \frac{0.1}{3}(-2.7682 + 4 \cdot (-3.7472) - 3.8870) = -0.27912$

$y_4^{(K3)} = 0.44235 + \frac{0.1}{3}(-2.7685 + 4 \cdot (-3.7472) - 3.8870) = -0.27913$

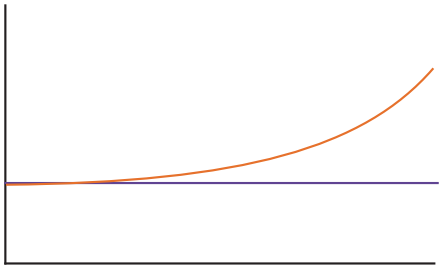
Sedaj nadaljujemo s točko $y_4 = -0.27913$ *in gremo na računanje* y_5. □

7.6 Inherentna in inducirana nestabilnost

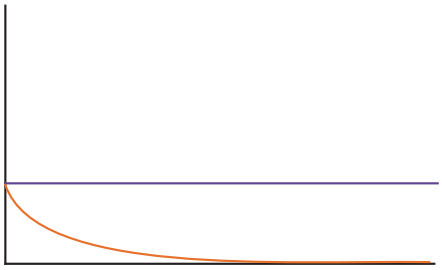
Inherentna nestabilnost je odvisna od problema in neodvisna od numerične metode.

Zgledi so:

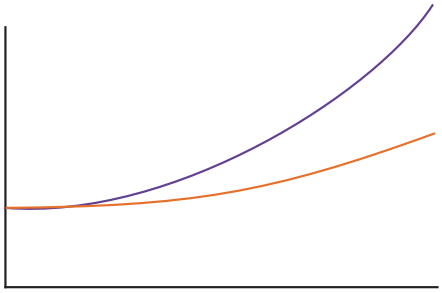
a) $y' = y - 1$, $y(0) = 1$, splošna rešitev je $y(x) = Ce^x + 1$ in $C = 0$, numerično pa dobimo $C \neq 0$. Problem je inherentno absolutno in relativno nestabilen.



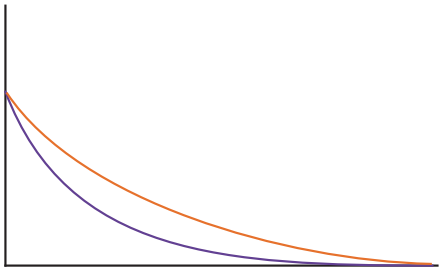
b) $y' = -y + 1, y(0) = 1$, splošna rešitev je $y(x) = Ce^{-x} + 1$ in $C = 0$, numerično pa dobimo $C \neq 0$. Problem je inherentno absolutno in relativno stabilen.



c) $y' = y + e^{2x}, y(0) = 1$, splošna rešitev je $y(x) = Ce^{x} + e^{2x}$ in $C = 0$, numerično pa dobimo $C \neq 0$. Problem je inherentno absolutno nestabilen in relativno stabilen.



d) $y' = -y - e^{2x}, y(0) = 1$, splošna rešitev je $y(x) = Ce^{-x} + e^{-2x}$ in $C = 0$, numerično pa dobimo $C \neq 0$. Problem je inherentno absolutno stabilen in relativno nestabilen.



Pri relativni nestabilnosti včasih pomaga, če obrnemo interval:

a) $y' = y - 1, y(x_0) = y_0$. Vzamemo $y(x_n) = \gamma$ in računamo nazaj,

b) $y' = -y - e^{2x}$, $y(x_0) = y_0$. Vzamemo $y(x_n) = \gamma$ in računamo nazaj.

Parameter γ moramo izbrati tako, da bo nazaj izračunana rešitev imela v točki x_0 pravo začetno vrednost y_0. Za pravo izbiro γ lahko uporabimo katerokoli metodo za reševanje nelinearnih enačb, npr. bisekcijo ali sekantno metodo. Če z $y(x;\gamma)$ označimo numerično rešitev, ki jo dobimo pri začetnem pogoju $y(x_n) = \gamma$ in računanju nazaj, potem moramo rešiti enačbo $y(x_0;\gamma) = y_0$. Podoben postopek uporabimo pri strelski metodi za reševanje robnega problema.

Inducirana nestabilnost je odvisna od izbire numerične metode. Pojavi se lahko le pri večkoračnih metodah. Za večkoračno metodo oblike

$$\sum_{i=0}^{k} \alpha_i y_{n-i} + h \sum_{i=0}^{k} \beta_i f_{n-i} = 0$$

pričakujemo, da je stabilna za enačbo $y' = 0$ (temu pravimo *ničelna stabilnost*), katere rešitev je konstanta. Ničelna stabilnost pomeni, da tudi če začnemo z nekaj začetnimi vrednostmi, kjer je prisotna napaka, mora potem rešitev, dobljena z večkoračno metodo, ostati omejena.

V primeru $y' = 0$ se večkoračna metoda spremeni v diferenčno enačbo

$$\sum_{i=0}^{k} \alpha_i y_{n-i} = 0.$$

Njen karakteristični polinom $\rho(\xi)$ ima k ničel $\xi_1, \ldots, \xi_k$. Splošna rešitev ima pri enostavnih ničlah obliko

$$y_n = \sum_{j=1}^{k} A_j \xi_j^n.$$

Za ničelno stabilnost mora tako veljati (Dahlquistov izrek)

a) $|\xi_i| \leq 1$ za $i = 1, \ldots, k$,

b) če je $|\xi_j| = 1$, mora biti ξ_j enostavna ničla.

Večkoračna metoda je konsistentna natanko tedaj, ko velja $\rho(1) = 0$ in $\rho'(1) + \sigma(1) = 0$, ta dva pogoja pa sta ekvivalentna temu, da je red vsaj 1. O pogojih za konvergentno večkoračno metodo govori naslednji izrek. Tako kot pri enokoračnih metodah lahko dokaz in ostale podrobnosti najdete v [18].

Izrek 7.10 *Večkoračna metoda je konvergentna natanko tedaj, ko je ničelno stabilna in konsistentna.* ■

7.7 Začetni problemi drugega reda

Rešujemo diferencialno enačbo drugega reda

$$\begin{aligned} y'' &= f(x, y, y') \\ y(x_0) &= y_0 \end{aligned}$$

$$y'(x_0) \;=\; y'_0.$$

To lahko prevedemo na sistem enačb prvega reda

$$\begin{array}{rclrcl} y' &=& p, & y(x_0) &=& y_0, \\ p' &=& f(x,y,p), & p(x_0) &=& y'_0. \end{array}$$

Zgled 7.13 *Vzemimo začetni problem drugega reda*

$$y'' = x + y^2$$

in $y(0) = 1$, $y'(0) = 0$. *To prevedemo na sistem dveh enačb prvega reda*

$$\begin{array}{rclrcl} y' &=& p, & y(0) &=& 1, \\ p' &=& x + y^2, & p(0) &=& 0. \end{array}$$

□

Zgled 7.14 *Problem dveh teles opisuje gibanje telesa zaradi sile težnosti telesa z veliko večjo maso. Če je težje telo v izhodišču, dobimo v kartezičnem koordinatnem sistemu enačbi*

$$\begin{aligned} x''(t) &= -x(t)/r(t)^3 \\ y''(t) &= -y(t)/r(t)^3, \end{aligned}$$

kjer je $r(t) = \sqrt{x(t)^2 + y(t)^2}$.

To prevedemo na sistem

$$z(t) = \begin{bmatrix} x(t) \\ y(t) \\ x'(t) \\ y'(t) \end{bmatrix}, \qquad z'(t) = \begin{bmatrix} z_3(t) \\ z_4(t) \\ -z_1(t)/r(t)^3 \\ -z_2(t)/r(t)^3 \end{bmatrix},$$

kjer je $r(t) = \sqrt{z_1(t)^2 + z_2(t)^2}$. □

V posebnem primeru, ko je $y'' = f(x,y)$, obstajajo posebne Runge-Kutta ali večkoračne metode, kot je npr. *metoda Numerova*

$$y_{n+1} - 2y_n + y_{n-1} = \frac{h^2}{12}(f_{n+1} + 10f_n + f_{n-1}) + \mathcal{O}(h^6).$$

7.8 Reševanje začetnih diferencialnih enačb v Matlabu

Na voljo imamo več metod. Izmed njjih je verjetno najbolj pogosto uporabljena `ode45`. Kličemo jo v obliki

```
[x,y]=ode45(fun,span,y0),
```

kjer je `fun` ime funkcije `yodv=fun(x,y)`, ki iz argumentov x in y, kjer je zadnji v primeru sistema vektor, izračuna vrednosti odvoda iz diferencialne enačbe. Seveda je v primeru sistema diferencialnih enačb tudi odvod, ki ga vrne `fun`, v obliki vektorja. Argument `span` poda interval $[a,b]$, na katerem iščemo rešitev, `y0` pa je začetni pogoj.

Zgled uporabe:

```
funcion yprime=fun(t,x)
yprime=-y-5*exp(-t)*sin(5*t);

tspan=[0 3]; yzero=1;
[x,y]=ode45('fun',tspan,y0)
plot(x,y,'*--');
```

Vse metode, ki jih uporablja Matlab, so adaptivne. Na voljo so naslednje metode:

- `ode45`: Temelji na Dormand–Princeovi metodi, ki je eksplicitna Runge-Kutta metoda reda 4 in 5 (podobno kot Fehlbergova metod). To je metoda, ki najpogosteje daje dobre rezultate, zato je priporočljivo problem najprej poskusiti rešiti s to metodo.

- `ode23`: Temelji na eksplicitnem Runge-Kutta pravilu reda 2 in 3. Kadar ne zahtevamo velike natančnosti in v primeru majhne togosti je lahko metoda bolj učinkovita kot pa `ode45`.
- `ode113`: Gre za Adams–Bashworth–Moultonovo večkoračno prediktor-korektor metodo, kjer vedno naredimo en korak korektorja. Lahko je bolj učinkovita od `ode45` v primerih, ko zahtevamo visoko natančnost in pa kadar je funkcija f iz diferencialne enačbe še posebej zapletena in želimo imeti čim manj izračunov vrednosti f.

Prejšnje tri metode so namenjene za netoge sisteme. V primeru, ko ne dajejo dobrih rezultatov imamo lahko opravka s togim sistemom in takrat lahko uporabimo naslednje metode:

- `ode15s`: Večkoračna metoda, ki temelji na metodah za numerično odvajanje. Je sicer manj učinkovita kot `ode45`, a deluje na zmerno togih primerih, ko `ode45` odpove.
- `ode23s`: Uporablja modificirano Rosenbrockovo metodo reda 2. Kadar ne zahtevamo velike natančnosti je lahko bolj učinkovita kot `ode15s` in deluje na nekaterih zelo togih primerih, kjer `ode15s` odpove.
- `ode23t`: Varianta trapezne metode. Uporabna za zmerno toge primere kadar ne zahtevamo velike natančnosti.
- `ode23tb`: Varianta implicitne dvostopenjske Runge-Kutta metode, primerna za zelo toge sisteme.

Vse metode uporabljamo na enak način, zato je dovolj pogledati le uporabo `ode45`.

- Osnovna uporaba je `[x,y] = ode45('f',xab,y0)`, kjer metoda `f` določa diferencialno enačbo $y' = f(x,y)$, $xab = [a\ b]$ je interval, na katerem rešujemo enačbo in y_0 je začetni pogoj $y(a) = y_0$. Kot rezultat dobimo vektorja izračunanih x in y.
- Dodatne opcije, s katerimi nastavimo npr. željeno natančnost, podamo v obliki `[x,y] = ode45('f',xab,y0,options)`. Pri tem moramo opcije `options` podati ali prej definirati s pomočjo funkcije `odeset`.

  Tako npr. z `options=odeset('RelTol',1e-4,'AbsTol',1e-8)` povečamo relativno in absolutno natančnost (privzeti vrednosti sta `1e-3` in `1e-6`).

- Če ima diferencialna enačba f poleg x in y še kakšne dodatne parametre, jih podamo v obliki `[x,y] = ode45('f',xab,y0,options,p1,p2,...)`. Pri tem lahko kot `options` podamo prazem seznam `[]`, kadar ne želimo spreminjati nastavitev.
- Z `[x,y,s] = ode45('f',xab,y0,options)` dobimo še vektor s s šestimi komponentami, ki nam povedo, kaj se je dogajalo med računanjem. Tako je s_1 število uspešnih korakov, s_2 število neuspešnih poskusov, s_3 število izračunov f, s_4 število izračunov Jacobijeve matrike parcialnih odvodov f po y, s_5 število LU razcepov in s_6 število reševanj linearnih sistemov.

Diferencialne enačbe, kjer poleg x in y nastopajo še dodatni parametri, uporabljamo v obliki `[x,y] = ode45('f',xab,y0,options,p1,p2,...)`. Pri tem moramo paziti na to, da ima funkcija f po vrsti parametre x, y, $flag$, p_1, p_2, ... Parameter $flag$ se uporablja za to, da diferencialna enačba po potrebi avtomatično poda začetne pogoje in interval, če tega ne uporabljamo pa moramo le pustiti prostor za nek parameter.

Tako npr. s kombinacijo

```
f=inline('x^2+y+p','x','y','flag','p')
[x1,y1]=ode45(f,[0 1],0,[],1)
[x2,y2]=ode45(f,[0 1],0,[],2)
```

rešimo diferencialni enačbi $y' = x^2 + x + 1$ in $y' = x^2 + y + 2$.

Velikokrat rešujemo diferencialno enačbo $y' = f(x,y)$, $y(x_0) = y_0$, kjer točen interval $[x_0, x_n]$ ni znan, saj je izračun točke x_n, kjer pride do kakšnega dogodka, del same naloge. Tako lahko npr. rešujemo nalogo telesa, ki prosto pada z določene višine, zanima pa nas, v katerem trenutku zadane tla.

Matematično formulirano imamo dano funkcijo $g(x,y)$ in iščemo rešitev začetnega problema $y' = f(x,y)$, $y(x_0) = y_0$ in točko x^*, kjer je $g(x^*, y(x^*)) = 0$. V Matlabu to lahko rešimo tako, da v nastavitvah določimo kontrolno funkcijo g, ki pove, kdaj se je dogodek zgodil.

```
function ydot = f(t,y)
ydot = [y(2); -1+y(2)^2];

function [gstop,isterminal,direction] = g(t,y)
gstop = y(1);       % Dogodek je, ko funkcija g vrne 0
isterminal = 1;     % Metoda se konča, ko pride do dogodka
direction = [];     % Do nič lahko pridemo z obeh strani

opts = odeset('events',@g);
y0 = [1; 0];
[t,y,tfinal] = ode45(@f,[0 Inf],y0,opts);
plot(t,y(:,1),'-',[0 tfinal],[1 0],'o')
```

7.9 Implicitno podane diferencialne enačbe

Diferencialna enačba je lahko namesto v eksplicitni obliki $y' = f(x,y)$ podana v implicitni obliki

$$f(x,y,y') = 0. \tag{7.1}$$

Če se iz enačbe (7.1) ne da direktno izraziti y' kot funkcija x in y, potem lahko sistem numerično rešujemo tako, da v vsakem koraku pri danih vrednostih za y in x dobimo y' iz (7.1) z numeričnim reševanjem nelinearne enačbe.

Reševanje začetnega problema si lahko predstavljamo kot sledenje krivulji, ki se začne pri $y(x_0) = y_0$. V primeru implicitno podane diferencialne enačbe moramo poznati tudi vrednost $y'(x_0)$, saj ima enačba (7.1) lahko več kot le eno rešitev. Ko poznamo x_0, y_0 in y'_0 lahko naredimo npr. en korak Eulerjeve metode in pridemo do $x_1 = x_0 + h$, $y_1 = y_0 + hy'_0$. V novi točki moramo spet rešiti nelinearno enačbo za y'_1, pri čemer pa lahko, če h ni prevelik, za začetni približek vzamemo kar y'_0.

Podobno velja za nelinearne implicitne sisteme nelinearnih enačb. Princip je enak, le da moramo sedaj namesto nelinearne enačbe reševati nelinearne sisteme, da pridemo do vektorja odvodov.

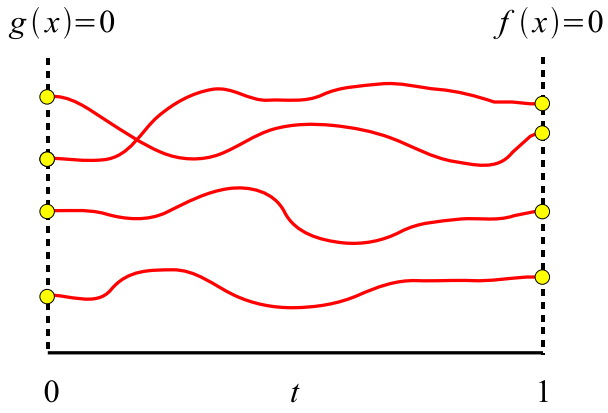
7.10 Metoda zveznega nadaljevanja

To je metoda za reševanje nelinearne enačbe $f(x) = 0$. Če je težko poiskati začetni približek (posebno, kadar je $f : \mathbb{R}^n \to \mathbb{R}^n$), si lahko pomagamo z uvedbo dodatnega parametra. Ideja je, da izberemo sorodni enostavnejši problem $g(x) = 0$, ki ga znamo rešiti, potem pa z dodatnim parametrom t, ki teče od 0 do 1, enostavnejši problem zvezno prevedemo na $f(x) = 0$. Pri tem pričakujemo, da se bodo z zveznim spreminjanjem problema zvezno premikale tudi rešitve in bomo tako s sledenjem iz znanih rešitev enačbe $g(x) = 0$ prišli do iskanih rešitev $f(x) = 0$.

Opazujemo npr. *konveksno homotopijo*

$$F(t,x) = t \cdot f(x) + (1-t) \cdot g(x),$$

kjer je $0 \le t \le 1$ in poznamo rešitve $g(x) = 0$. S sledenjem krivulji od $t = 0$ do $t = 1$ dobimo rešitve $f(x) = 0$.



Sledenje krivulje je povezano z reševanjem diferencialne enačbe. Z odvajanjem po t dobimo

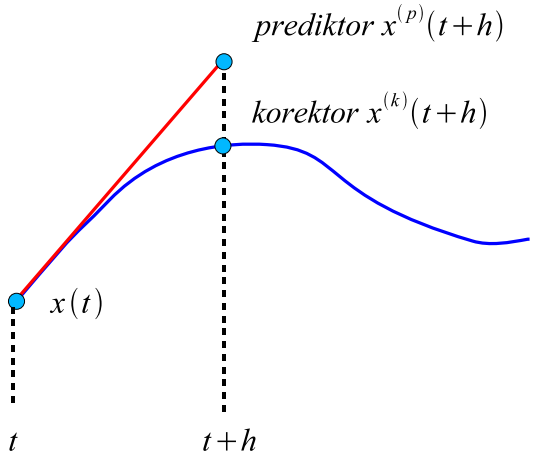
$$F_t(t,x) + F_x(t,x) \cdot \dot{x} = 0,$$

kar nam da navadno diferencialno enačbo

$$\dot{x} = -F_x(t,x)^{-1} \cdot F_t(t,x), \quad x(0) = x_0,$$

kjer je $g(x_0) = 0$.

Z metodami za reševanje navadnih diferencialnih enačb lahko sledimo rešitvi, poleg tega pa pri vsaki vrednosti t lahko upoštevamo še, da mora za $x(t)$ veljati $F(t, x(t)) = 0$. Ponavadi uporabljamo kombinacijo prediktor–korektor, ko najprej uporabimo metodo za reševanje začetnega problema, potem pa iz dobljenega približka izračunamo novo točko na krivulji z kakšno metodo za reševanje nelinearnega sistema.



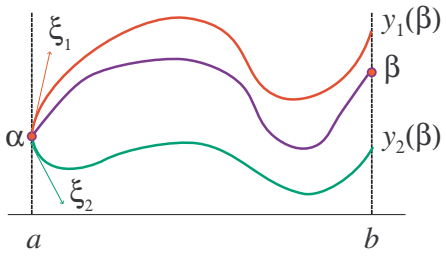
a) prediktor: Iz točke $(t, x(t))$ z eno izmed metod za reševanje začetnega problema (npr. z Eulerjevo metodo) izračunamo prediktor $x^{(P)}(t+h)$, ki je približek za rešitev v točki $t+h$.

b) korektor: Z eno izmed metod za reševanje nelinearnega sistema (npr. z Newtonovo metodo) rešimo sistem $F(t+h, x(t+h)) = 0$, za začetni približek pa vzamemo $(t+h, x^{(P)}(t+h))$.

7.11 Robni problemi drugega reda

Pri robnih problemih imamo opravka z diferencialno enačbo višjega reda, kjer dodatni pogoji, ki določajo pravo vejo, niso podani v isti točki. Tako pri robnem problemu drugega reda rešujemo diferencialno enačbo drugega reda na intervalu $[a, b]$ z robnima pogojema:

$$\begin{aligned} y'' &= f(x, y, y') \\ y(a) &= \alpha \\ y(b) &= \beta. \end{aligned}$$



Slika 7.6: Rešitev linearnega robnega problema lahko sestavimo kot linearno kombinacijo dveh začetnih problemov.

7.11.1 Linearni robni problem

Najpreprostejši je *linearni robni problem drugega reda*, ki ima obliko

$$-y''(x) + p(x)y'(x) + q(x)y(x) = r(x), \tag{7.2}$$
$$y(a) = \alpha, \quad y(b) = \beta.$$

Numerični načini reševanja linearnega robnega problema so:

a) *Kombinacija dveh začetnih problemov*. Izberemo različna ξ_1 in ξ_2 in rešimo začetna problema, ki ustrezata enačbi (7.2) in

$$\begin{aligned} y_1(a) &= \alpha, \quad y_1'(a) = \xi_1, \\ y_2(a) &= \alpha, \quad y_2'(a) = \xi_2. \end{aligned}$$

Za poljuben λ je $y = \lambda y_1 + (1-\lambda)y_2$ tudi rešitev enačbe (7.2) in ustreza pogoju $y(a) = \alpha$, Sedaj λ določimo tako, da bo $y(b) = \beta$, situacija je predstavljena na sliki 7.6. Veljati mora

$$\lambda y_1(b) + (1-\lambda)y_2(b) = \beta,$$

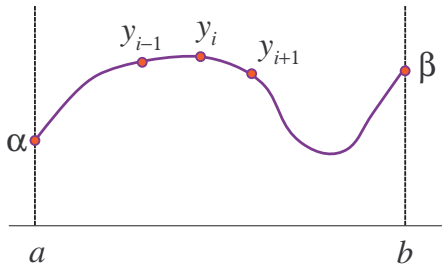
torej

$$\lambda = \frac{\beta - y_2(b)}{y_1(b) - y_2(b)}.$$

Dva začetna problema moramo reševati kot sistem, pa čeprav bi lahko enačbi rešili ločeno drugo od druge. Če jih rešujemo ločeno, se pri adaptivnih metodah lahko namreč zgodi, da rešitvi ne bosta tabelirani v istih točkah, potem pa ne moremo izračunati linearne kombinacije rešitev.

b) *Diferenčna metoda*. Interval $[a, b]$ ekvidistantno razdelimo na $n+1$ delov s točkami $x_0 = a, x_1, \ldots, x_n, x_{n+1} = b$, kjer je $x_i = x_0 + ih$ in $h = (b-a)/(n+1)$. Odvode aproksimiramo s simetričnimi diferencami

$$\begin{aligned} y_i' &= \frac{y_{i+1} - y_{i-1}}{2h} + \mathcal{O}(h^2), \\ y_i'' &= \frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} + \mathcal{O}(h^2). \end{aligned}$$



Slika 7.7: Približki pri diferenčni metodi.

Dobimo sistem enačb

$$\frac{-y_{i+1}+2y_i-y_{i-1}}{h^2}+p_i\frac{y_{i+1}-y_{i-1}}{2h}+q_iy_i=r_i,\quad i=1,\ldots,n,$$

pri čemer je $y_0=\alpha$ in $y_{n+1}=\beta$. Sistem je linearen in tridiagonalen, zato ga lahko rešimo na preprost način. To nam tudi omogoča, da vzamemo velik n, kar izboljša natančnost rešitve.

Če v diferencialni enačbi ne nastopa y', lahko uporabimo metodo Numerova, saj je potem napaka reda $\mathcal{O}(h^6)$ namesto $\mathcal{O}(h^2)$. V primeru diferencialne enačbe

$$-y''(x)+q(x)y(x)=r(x)$$

tako za $i=1,\ldots,n$ dobimo enačbo

$$y_{i+1}-2y_i+y_{i-1}=\frac{h^2}{12}(q_{n+1}y_{n+1}-r_{n+1}+10q_ny_n-r_n-q_{n-1}y_{n-1}+r_{n-1}).$$

Zgled 7.15 *Z diferenčno metodo in $h=1/2$ rešimo robni problem*

$$\begin{aligned}(1+x^2)y''&+2xy'-x^2y=1\\ y(-1)&=0\\ y(1)&=0.\end{aligned}$$

Dobimo sistem enačb za y_1,y_2,y_3:

$$(1+(-0.5)^2)\frac{y_2-2y_1+0}{(0.5)^2}+2(-0.5)\frac{y_2-0}{2(0.5)}-(-0.5)^2y_1=1$$

$$(1+(0.0)^2)\frac{y_3-2y_2+y_1}{(0.5)^2}+2(0.0)\frac{y_3-y_1}{2(0.5)}-(0.0)^2y_2=1$$

$$(1+(0.5)^2)\frac{0-2y_3+y_2}{(0.5)^2}+2(0.5)\frac{0-y_3}{2(0.5)}-(0.5)^2y_3=1,$$

ki ima rešitev

$$y_1=-0.24,\ y_2=-0.365,\ y_3=-0.24.$$ □

Diferenčno metodo lahko uporabljamo tudi pri robnih pogojih, v katerih nastopajo odvodi. Če imamo splošne robne pogoje oblike

$$\alpha_1 f(a) + \beta_1 f'(a) = 0, \quad \text{in} \quad \alpha_2 f(b) + \beta_2 f'(b) = 0,$$

si pomagamo s t.i. navideznimi točkami.

Npr., če imamo v robnem pogoju odvod v točki a, potem ta odvod aproksimiramo s simetrično diferenco

$$y_0' = \frac{y_1 - y_{-1}}{2h}.$$

Nova neznanka y_{-1} je le navidezna. Ker mora tudi v točki a rešitev zadoščati diferencialni enačbi, dobimo še enačbo

$$\frac{-y_1 + 2y_0 - y_{-1}}{h^2} - p_0 \frac{y_1 - y_{-1}}{2h} + q_0 y_0 = r_0,$$

iz teh dveh enačb pa lahko izločimo y_{-1}.

Če podvojimo število točk, pričakujemo, da bomo dobili boljše približke. Ker pa za napako velja, da je reda h^2, lahko z uporabo ekstrapolacije, podobne kot pri Rombergovi metodi, dobimo še boljše približke v tistih točkah, ki nastopajo tako v grobi kot v fini delitvi.

7.11.2 Nelinearni robni problem

Pri *nelinearnem robnem problemu drugega reda* je f nelinearna funkcija y in y'. Zaradi nelinearnosti ne moremo uporabiti kombinacije dveh začetnih problemov. Uporabimo lahko diferenčno metodo, a dobimo nelinearni sistem z veliko neznankami.

Nelinearne enačbe imajo obliko

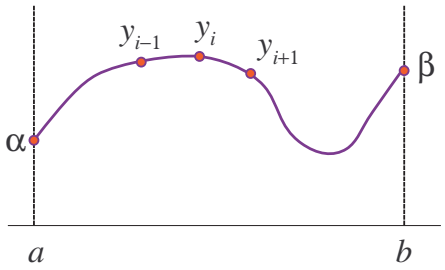
$$\frac{y_{i-1} - 2y_i + y_{i+1}}{h^2} = f\left(x, y_i, \frac{y_{i+1} - y_{i-1}}{2h}\right), \quad i = 1, \ldots, n.$$

Če nelinearni sistem rešujemo z Newtonovo metodo, je Jacobijeva matrika tridiagonalna, tako da se da sistem reševati dokaj ekonomično. Za začetni približek lahko npr. vzamemo kar točke na daljici med (a, α) in (b, β), torej $y_i = \alpha + i(\beta - \alpha)/n$.

Na voljo imamo tudi *strelsko metodo*. Pri strelski metodi rešujemo začetni problem

$$\begin{aligned} y'' &= f(x, y, y'), \\ y(a) &= \alpha, \\ y'(a) &= \xi. \end{aligned} \tag{7.3}$$

Rešitev je odvisna od parametra ξ, ki ga je potrebno izbrati tako, da bo $y(b; \xi) = \beta$.



Če definiramo $E(\xi) := y(b;\xi) - \beta$, potem iščemo tak ξ, da bo $E(\xi) = 0$. Uporabimo lahko katerokoli metodo za reševanje nelinearne enačbe. Če npr. želimo uporabiti tangentno metodo, potem potrebujemo tudi $E'(\xi) = \frac{\partial y(b;\xi)}{\partial \xi}$. Definiramo $z := \frac{\partial y}{\partial \xi}$ in z odvajanjem

$$y'' = f(x, y, y'), \quad y(a) = \alpha,\ y'(a) = \xi$$

dobimo

$$z'' = f_y(x, y, y')z + f_{y'}(x, y, y')z', \quad z(a) = 0,\ z'(a) = 1. \tag{7.4}$$

Če rešujemo sistem (7.3) in (7.4), potem dobimo tudi vrednost odvoda funkcije E.

7.11.3 Prevedba na variacijski problem

Reševanje robnega problema

$$-\frac{d}{dx}\left(p(x)\frac{dy}{dx}\right) + q(x)y = f(x), \quad 0 \le x \le b,$$

z robnima pogojema $y(0) = 0$ in $y'(b) + \sigma y(b) = 0$, kjer je $p(x) \ge \delta > 0$, $q(x) \ge 0$ in $\sigma > 0$, je ekvivalentno iskanju funkcije $y \in C^2[0, b]$, ki zadošča robnima pogojema in minimizira funkcijo

$$F(y) = \int_0^b \left(p(x)y'^2(x) + q(x)y^2(x) - 2f(x)y(x)\right) dx + p(b)\sigma y^2(b).$$

Ideja *Rayleigh–Ritzove metode* je, da namesto $y \in C^2[0, b]$ iščemo približek oblike

$$y(x) = \sum_{i=1}^{n} c_i \phi_i(x),$$

kjer so $\phi_1, \ldots, \phi_n$ t.i. *bazne funkcije*.

V točki, kjer je dosežen minimum, morajo biti vsi parcialni odvodi F po c_i enaki 0. Tako dobimo linearni sistem $Ac = b$ za koeficiente $c = [\, c_1 \quad \cdots \quad c_n \,]^T$, kjer je

$$a_{ij} = \int_0^b \left(p(x)\phi_i'(x)\phi_j'(x) + q(x)\phi_i(x)\phi_j(x)\right) dx$$

in

$$b_i = \int_0^b \frac{\phi_i(x)}{f(x)} dx$$

za $i, j = 1, \ldots, n$. Od izbire baznih funkcij je odvisno, ali bo sistem rešljiv in kako dober bo dobljen približek.

Metoda končnih elementov je poseben primer Rayleigh-Ritzove metoda, kjer za bazne funkcije izberemo t.i. *piramidne funkcije*

$$\phi_i(x) = \begin{cases} 0, & 0 \le x \le x_{i-1}, \\ (x - x_{i-1})/h_{i-1}, & x_{i-1} < x \le x_i, \\ (x_{i+1} - x)/h_i, & x_i < x \le x_{i+1}, \\ 0, & x_{i+1} < x \le b, \end{cases}$$

kjer je $0 = x_0 < x_1 < x_2 < \cdots < x_n = b$ in $x_i - x_{i-1} = h_{i-1}$.

Funkcija ϕ_i, kjer je $i = 1, \ldots, n-1$, ima nosilec na intervalu (x_{i-1}, x_{i+1}), izjema je le funkcija ϕ_n ki ima neničelni del samo na intervalu $(x_{n-1}, x_n]$. Zaradi tega je matrika A sedaj tridiagonalna. Elementi sistema $Ax = b$ so

$$\begin{aligned}
a_{ii} &= \int_{x_{i-1}}^{x_i} \frac{1}{h_{i-1}^2} p(x)dx + \int_{x_i}^{x_{i+1}} \frac{1}{h_i^2} p(x)dx \\
&\quad + \int_{x_{i-1}}^{x_i} \frac{1}{h_{i-1}^2}(x - x_{i-1})^2 q(x)dx + \int_{x_i}^{x_{i+1}} \frac{1}{h_i^2}(x_{i+1} - x)^2 q(x)dx, \quad i = 1, \ldots, n-1, \\
a_{i,i+1} &= \int_{x_i}^{x_{i+1}} \frac{-1}{h_i^2} p(x)dx + \int_{x_i}^{x_{i+1}} \frac{1}{h_i^2}(x_{i+1} - x)(x - x_i) q(x)dx, \quad i = 1, \ldots, n-1, \\
a_{i-1,i} &= \int_{x_{i-1}}^{x_i} \frac{-1}{h_{i-1}^2} p(x)dx + \int_{x_i}^{x_{i+1}} \frac{1}{h_i^2}(x_{i+1} - x)(x - x_{i-1}) q(x)dx, \quad i = 2, \ldots, n, \\
b_i &= \int_{x_{i-1}}^{x_i} \frac{1}{h_{i-1}}(x - x_{i-1}) f(x)dx + \int_{x_i}^{x_{i+1}} \frac{1}{h_i}(x_{i+1} - x) f(x)dx, \quad i = 1, \ldots, n-1, \\
a_{nn} &= \int_{x_{n-1}}^{x_n} \frac{1}{h_{n-1}^2} p(x)dx + \int_{x_{n-1}}^{x_n} \frac{1}{h_{n-1}^2}(x - x_{n-1})^2 q(x)dx + p(b)\sigma, \\
b_n &= \int_{x_{n-1}}^{x_n} \frac{1}{h_{n-1}}(x - x_{n-1}) f(x)dx.
\end{aligned}$$

7.12 Reševanje robnih problemov v Matlabu

V Matlabu imamo za reševanje robnih problemov na voljo metodo `bvp4c`, ki uporablja kolokacijsko metodo. S to metodo lahko rešujemo robne probleme, ki jih zapišemo v obliki

$$\frac{d}{dx} y(x) = f(x, y(x), p), \quad g(y(a), y(b), p) = 0.$$

Tu je $y(x)$ vektor, f je podana funkcija x in y, ki opisuje diferencialno enačbo. Vektor p je neobvezen in opisuje neznane parametre, ki jih iščemo. Rešitev iščemo na intervalu $[a, b]$, s funkcijo g pa podamo robne pogoje. Metoda potrebuje tudi začetni približek za rešitev $y(x)$.

Oblika za klic `bvp4c` je `sol=bvp4c(@odefun,@bcfun,solinit,options)`, pri čemer:

- funkcija `odefun` poda diferencialno enačbo, oblika je `yodv = odefun(x,y);`
- funkcija `bcfun` poda robne pogoje (vrne ostanek, ki mora biti pri izpolnjenih pogojih enak 0), oblika je `res = bcfun(ya,yb);`
- `solinit` je struktura, s katero podamo začetni približek, strukturo zgradimo s pomožno funkcijo `bvpinit` kot `solinit = bvpinit(linspace(a,b,n),@initf)`, kjer je `initf` funkcija oblike `y = initf(x)`, s katero podamo robne pogoje;
- `options` je neobvezen argument, kjer lahko nastavimo delovanje metode, za nastavitve uporabimo metodo `bvpset`.

Zgled 7.16 *Presek oblike kapljice vode na ravni podlagi je podan z rešitvijo robnega problema*

$$u''(x) + (1 - u(x))(1 + u'(x)^2)^{3/2} = 0, \quad u(-1) = 0, \quad u(1) = 0.$$

Za uporabo `bvp4c` *to spremenimo v sistem enačb prvega reda:*

$$\begin{aligned} y_1'(x) &= y_2(x), \\ y_2'(x) &= (y_1(x)-1)(1+y_2(x)^2)^{3/2}, \end{aligned}$$

Za začetni približek vzamemo $y_1(x) = \sqrt{1-x^2}$ *in* $y_2(x) = -x/(0.1+\sqrt{1-x^2})$.

Ustrezna koda za rešitev problema v Matlabu je:

```
function yodv = kaplja(x,y)
yodv = [ y(2); (y(1)-1)*((1+y(2)^2)^(3/2)) ];

function res = kapljarp(ya,yb)
res = [ ya(1); yb(1) ];

function y = kapljainit(x)
y = [ sqrt(1-x^2); -x/(0.1+sqrt(1+x^2)) ];

resinit = bvpinit(linspace(-1,1,20), @kapljainit);
res = bvp4c(@kaplja, @kapljarp, resinit);
plot(res.x, res.y(1,:))
```

□

Zgled 7.17 *Iščemo lastno vrednost* λ*, pri kateri ima robni problem*

$$y''(x) + (\lambda - 10\cos(2x))y(x) = 0, \quad y'(\pi) = 0, \quad y'(0) = 0$$

neničelno rešitev. Ker lahko vsako tako rešitev pomnožimo s poljubnim skalarjem, potrebujemo še en pogoj, npr. $y(0) = 1$*. Podobno kot v prejšnjem zgledu enačbo spremenimo v sistem enačb prvega reda:*

$$y_1'(x) = y_2(x), \quad y_2'(x) = (10\cos(2x) - \lambda)y_1(x).$$

Za začetni približek vzamemo $y(x) = \cos(4x)$ *in* $\lambda = 15$.

Ustrezna koda za rešitev problema v Matlabu je:

```
function yodv = mth(x,y,lambda)
yodv = [  y(2);  (10*cos(2*x)- lambda)*y(1) ];

function res = mthrp(ya,yb,lambda)
res = [ ya(2);  yb(2); ya(1)-1 ];

function y = mthinit(x)
y = [ cos(4*x);  -4*sin(4*x) ];

resinit = bvpinit(linspace(0,pi,10), @mthinit, 15);
res = bvp4c(@mth, @mthrp, resinit);
fprintf('Lastna vrednost je %7.3f.\n', res.parameters)
plot(res.x, res.y(1,:))
```

□

Zgled 7.18 *Naslednje enačbe opisujejo pretok v dolgem navpičnem kanalu, kjer je tekočina vbrizgana skozi eno izmed stranic:*

$$\begin{aligned} f''' - R\left[(f')^2 - ff''\right] + RA &= 0, \\ h'' + Rfh' + 1 &= 0, \\ \theta'' + Pf\theta' &= 0, \end{aligned}$$

pri čemer je R Raynoldsovo število in $P = 0.7R$. Ker parameter A spada med neznanke, imamo 8 robnih pogojev:

$$\begin{aligned} f(0) &= f'(0) = 0, \quad f(1) = 1, \quad f'(1) = 0, \\ h(0) &= h(1) = 0, \\ \theta(0) &= 0, \quad \theta(1) = 1. \end{aligned}$$

Problem bi radi rešili pri $R = 10000$, vendar nimamo nobenih dobrih začetnih približkov. Zato uporabimo metodo zveznega nadaljevanja. Najprej problem rešimo za $R = 100$, potem to uporabimo kot začetni približek za $R = 1000$, tega pa za $R = 10000$.

Ustrezna koda za rešitev problema v Matlabu je:

```
R = 100; resinit = bvpinit(linspace(0,1,10), ones(7,1), 1);
res1 = bvp4c(@pretok, @pretokbc, resinit);
fprintf('Parameter A pri R=100  : %7.4f\n',res1.parameters)

R = 1000; res2 = bvp4c(@pretok, @pretokbc, res1);
fprintf('Parameter A pri R=1000 : %7.4f\n',res2.parameters)

R = 10000; res3 = bvp4c(@pretok, @pretokbc, res2);
fprintf('Parameter A pri R=10000  : %7.4f\n',res3.parameters)
plot(res1.x, res1.y(2,:), res2.x, res2.y(2,:), res3.x, res3.y(2,:));

function yodv = pretok(x,y,A)
  P = 0.7*R;
  yodv = [ y(2); y(3); R*(y(2)^2 - y(1)*y(3) - A);
           y(5); -R*y(1)*y(5) - 1; y(7); -P*y(1)*y(7) ];
end

function res = pretokbc(ya,yb,A)
  res = [ya(1); ya(2); yb(1) - 1; yb(2); ya(4); yb(4); ya(6); yb(6) - 1];
end
```

□

Dodatna literatura

Več o metodi zveznega nadaljevanja lahko najdete v [1].

Literatura

[1] E. L. Allgower, K. Georg, *Numerical Continuation Methods: An Introduction*, Springer-Verlag, New York, 1990.

[2] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, H. van der Vorst, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, SIAM, Philadelphia, 1994.

[3] Z. Bohte, *Numerične metode, 4. ponatis*, DMFA Slovenije, Ljubljana, 1991.

[4] Z. Bohte, *Numerično reševanje nelinearnih enačb*, DMFA Slovenije, Ljubljana, 1993.

[5] Z. Bohte, *Numerično reševanje sistemov linearnih enačb*, DMFA Slovenije, Ljubljana, 1994.

[6] R. L. Burden, J. D. Faires: *Numerical Analysis*, Brooks/Cole, Pacific Grove, 1997.

[7] B. N. Datta, *Numerical Linear Algebra and Applications*, Brooks/Cole, Pacific Grove, 1995.

[8] J. W. Demmel, *Uporabna numerična linearna algebra*, DMFA Slovenije, Ljubljana, 2000.

[9] G. H. Golub, C. F. van Loan, *Matrix Computations, 3rd edition*, The Johns Hopkins University Press, Baltimore, 1996.

[10] G. H. Golub, F. Uhlig, *The QR algorithm: 50 years later its genesis by John Francis and Vera Kublanovskaya and subsequent developments*, IMA Journal of Numerical Analysis **29**, 2009, str. 467—485.

[11] J. F. Grcar, Mathematicians of Gaussian Elimination, AMS Notices **58**, 2011, str. 782–792.

[12] P. C. Hansen, *Regularization Tools. A Matlab Package for Analysis and Solution of Discrete Ill-Posed Problems*, 2008, `http://www2.imm.dtu.dk/~pch/Regutools/RTv4manual.pdf`

[13] M. T. Heath, *Scientific Computing: An Introductory Survey, 2nd edition*, McGraw-Hill, 2002.

[14] D. J. Higham, N. J. Higham, *Matlab Guide, Second Edition*, SIAM, Philadelphia, 2005.

[15] N. J. Higham, *Accuracy and stability of numerical algorithms*, SIAM, Philadelphia, 1996.

[16] C. T. Kelley, *Iterative Methods for Linear and Nonlinear Equations*, SIAM, Philadelphia, 1995.

[17] D. Kincaid, W. Cheney, *Numerical Analysis*, Brooks/Cole, Pacific Grove, 1996.

[18] J. Kozak, *Numerična analiza*, DMFA Slovenije, Ljubljana, 2008.

[19] C.-Y. Lam, *Applied Numerical Methods For Partial Differential Equations*, Prentice Hall, Singapore, 1994.

[20] B.-S. Liao, Z. Bai, L.-Q. Lee, K. Ko, Solving large scale nonlinear eigenvalue problems in next-generation accelerator design. Technical Report CSE-TR, University of California, Davis, 2006.

[21] C. D. Meyer, *Matrix Analysis and Applied Linear Algebra*, SIAM, Philadelphia, 2000.

[22] J. Miller, *Earliest Know Uses of Some of the Words of Mathematics*, `http://jeff560.tripod.com/mathworld.com`.

[23] K. W. Morton, D. F. Mayers, *Numerical Solution of Partial Differential Equations*, Cambridge University Press, Cambridge, 2002

[24] J. J. O'Connor, E. F. Robertson, *MacTutor History of Mathematics archive*, University of St Andrews, Scotland, `http://www-history.mcs.st-andrews.ac.uk/`.

[25] T. Papler, *Stabilno računanje lastnih vektorjev simetrične tridiagonalne matrike v* $\mathcal{O}(n^2)$, diplomsko delo, Ljubljana, 2005.

[26] Y. Saad, *Iterative Methods for Sparse Linear Systems, Second Edition*, SIAM, Philadelphia, 2003.

[27] G. D. Smith, *Numerical Solution of Partial Differential Equations: Finite Difference Methods, 3rd edition*, Clarendon Press, Oxford, 1985.

[28] G. W. Stewart, J.-G. Sun, *Matrix Perturbation Theory*, Academic Press, San Diego, 1990.

[29] F. Tisseur, K. Meerbergen: *The quadratic eigenvalue problem*, SIAM Review **43** (2001), 235–286.

[30] L. N. Trefethen, L. M. Trefethen, *How many shuffles to randomize a deck of cards?*, Proc. Royal Soc. A **456** (2002), str. 2561-–2568.

[31] S. Van Huffel, J. Vandewalle, *The Total Least Square Problem. Computational Aspecs and Analysis*, SIAM, Philadelphia, 1991.

[32] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Oxford University Press, Oxford, 1965.

[33] T. J. Ypma, *Historical development of the Newton-Raphson method*, SIAM Review **37**, 1995, str. 531–555.

[34] E. Zakrajšek, *Uvod v numerične metode, druga izdaja*, DMFA Slovenije, Ljubljana, 2000.