Bor Plestenjak

Iterativne numerične metode v linearni algebri

skripta

verzija: 30. maj 2021

Kazalo

Predgovor

Skripta je namenjena predmetu *Iterativne numerične metode v linearni algebri*, ki se predava na drugi stopnji smeri Matematika in Finančna matematika na Fakulteti za matematiko in fiziko Univerze v Ljubljani.

Skripta je nastala ob prvem izvajanju predmeta v študijskem letu 2011/12 in se nato posodabljala ob izvajanjih predmeta v naslednjih študijskih letih.

prof. dr. Bor Plestenjak

Poglavje 1

Uvod

Pri predmetu bomo obravnavali iterativne numerične metode za reševanje sistemov linearnih enačb in računanje lastnih vrednosti in vektorjev. Gre za nadgradnjo metod, ki smo jih spoznali na prvostopenjskem študiju. Vse potrebno predznanje s področja numeričnega računanja in še posebno numerične linearne algebre, ki ga npr. študenti prve stopnje študijskega programa Matematika lahko pridobijo pri izbirnem predmetu *Numerična linearna algebra* v 3. letniku, lahko najdete v učbeniku [9].

Glede na to, da se bomo ukvarjali z iterativnimi metodami, za začetek definirajmo, kaj bomo privzeli za direktne metode.

a) Če gre za reševanje sistema linearnih enačb, potem z metodami, kot so npr. LU razcep, QR razcep ali razcep Choleskega, ki smo jih spoznali na prvi stopnji, sistem iste velikosti in oblike vedno rešimo z istim številom in vrstnim redom operacij. Npr., za sistem velikosti $n \times n$ reševanje s pomočjo LU razcepa porabi $2/3n^3 + \mathcal{O}(n^2)$ osnovnih računskih operacij. Zato so to očitno direktni algoritmi. Dokler se direktna metoda ne izvede do konca, nimamo še nobenega približka za rešitev sistema.

   Za razliko od direktnih metod bomo pri iterativnih metodah tvorili zaporedje vektorjev $x_0, x_1, \ldots$, ki konvergira proti točni rešitvi sistema $Ax = b$. Pri iterativnih metodah je natančnost približka odvisna od števila porabljenih korakov. Če naredimo več korakov, porabimo več računskih operacij, a je zato tudi natančnost približka boljša.

b) Pri računanju lastnih vrednosti so v resnici vsi algoritmi iterativni. Vemo namreč, da se lastnih vrednosti matrike velikosti 5×5 ali več ne da izraziti z zaključenimi formulami. Kljub temu npr. za QR iteracijo, s katero izračunamo vse lastne vrednosti, pravimo, da je direktna metoda. Sedaj točno število operacij, ki jih potrebujemo za izračun vseh lastnih vrednosti, ni več odvisno le od velikosti matrike temveč tudi od razporeditve lastnih vrednosti in drugih lastnosti matrike. Vseeno pa se da dobiti oceno za splošen primer. Tako npr. QR iteracija ob predpostavki, da v povprečju potrebujemo dva koraka QR iteracije z dvojnimi premiki, da izločimo eno lastno vrednost, potrebuje $10n^3 + \mathcal{O}(n^2)$ osnovnih operacij za izračun vseh lastnih vektorjev oziroma $25n^3 + \mathcal{O}(n^2)$ če računamo tako lastne vrednosti kot tudi lastne vektorje. Zaradi tega QR iteracijo obravnavamo kot direktno metodo.

   V grobem bi lahko pri računanju lastnih vrednosti dejali, da je direktna metoda vsaka, ki izračuna vse lastne vrednosti, vse lastne pare ali Schurovo formo dane matrike. Za razliko

od tega pri iterativnih metodah računamo samo nekaj lastnih vrednosti, ponavadi tiste, ki so najbližje izbrani točki.

Dve takšni metodi že poznamo. Pri potenčni metodi ([9, razdelek 6.4]) računamo dominantno lastno vrednost, pri ortogonalni iteraciji ([9, razdelek 6.7]) pa $k \ll n$ dominantnih lastnih vrednosti.

Iterativne metode običajno uporabljamo v primeru velikih razpršenih matrik, ko reševanje z direktnimi metodi ni možno, saj bi za to porabili preveč pomnilnika ali računskih operacij.

Poglavje 2

Klasične iterativne metode za linearne sisteme

2.1 Uvod

Imamo sistem linearnih enačb

$$Ax = b,$$

kjer je $A \in \mathbb{R}^{n\times n}$ in $x, b \in \mathbb{R}^n$. Poleg direktnih metod kot so npr. LU razcep, QR razcep ali razcep Choleskega, poznamo tudi iterativne metode za reševanje sistemov linearnih enačb. Pri reševanju z iterativnimi metodami dobimo zaporedje približkov $\{x^{(r)}\}$, katerega limita naj bi bila rešitev sistema $Ax = b$.

Reševanja se lahko npr. lotimo na naslednji način. Naj bo M tak približek za matriko A, da znamo sistem z matriko M rešiti bolj učinkovito kot sistem z matriko A. Rešimo sistem

$$Mx_0 = b$$

in tako dobimo približek x_0. Ker to ni točna rešitev, iščemo popravek z_0, da bo $A(x_0 + z_0) = b$ oziroma $Az_0 = b - Ax_0$. Namesto tega rešimo sistem

$$My_0 = b - Ax_0,$$

kjer smo spet A zamenjali s približkom M. Tako smo dobili približek y_0 za z_0 in za nov približek za rešitev sistema $Ax = b$ izberemo $x_1 = x_0 + y_0$. Postopek iterativno ponavljamo in tako pridemo do metode, ki je opisana v algoritmu 2.1.

Algoritem 2.1 Algoritem za iterativno reševanje sistema linearnih enačb $Ax = b$, kjer matriko A nadomestimo z matriko M.

reši $Mx_0 = b$
$k = 0, 1, \ldots$
    reši $My_k = b - Ax_k$
    $x_{k+1} = x_k + y_k$

Z iterativnim reševanjem sistemov linearnih enačb se je ukvarjal že Gauss. Leta 1823 je predlagal iterativno metodo za reševanje sistema $Ax = b$ štirih enačb s štirimi neznankami, ki je bil strogo diagonalno dominanten.

Definicija 2.1 *Pravimo, da je $n \times n$ matrika A* strogo diagonalno dominantna po vrsticah, *če velja*

$$|a_{ii}| > \sum_{j=1, j\neq i}^{n} |a_{ij}| \quad za \quad i = 1, \ldots, n.$$

Podobno je A^T strogo diagonalno dominantna po stolpcih, *če je A strogo diagonalno dominantna po vrsticah.*

Če velja

$$|a_{ii}| \geq \sum_{j=1, j\neq i}^{n} |a_{ij}| \quad za \quad i = 1, \ldots, n,$$

pri čemer vsaj v eni vrstici velja stroga neenakost, potem pravimo, da je matrika šibko diagonalno dominantna po vrsticah. *Podobno definiramo šibko diagonalno dominantnost po stolpcih.*

Zaradi stroge diagonalne dominantnosti je Gauss za M vzel diagonalno matriko, katere diagonala je enaka diagonali matrike A. Če matriko A razdelimo na

$$A = L + D + U,$$

kjer je L spodnja trikotna matrika brez diagonale, D diagonalna matrika in U zgornja trikotna matrika brez diagonale, potem je $M = D$. Tako dobimo Gauss–Jacobijevo metodo, ki jo je uporabljal tudi nemški matematik in astronom Jacobi za računanje tirov planetov. Algoritem je zapisan v algoritmu 2.2.

Algoritem 2.2 Originalna Gauss–Jacobijeva metoda za iterativno reševanje sistema linearnih enačb $Ax = b$.

reši $Dx_0 = b$
$k = 0, 1, \ldots$
    reši $Dy_k = b - Ax_k$
    $x_{k+1} = x_k + y_k$

Kot eno izmed prednosti iterativnega reševanja je Gauss izpostavil, da za reševanje ne potrebuješ tako močne koncentracije kot pri reševanju z eliminacijami. V pismu prijatelju se je navdušeno pohvalil, da te račune lahko opravljaš tudi, ko že napol spiš ali, ko razmišljaš o drugih problemih. Poleg tega direktne metode omogočajo tudi lažje odkrivanje napak med računanjem. Če se npr. zmotimo pri uporabi LU razcepa, bomo to sicer odkrili na koncu, ko Ax ne bo enako b, a ne bomo imeli nobene druge rešitve, kot da gremo ponovno čez vse račune. Pri iterativnih metodah se da hitro ugotoviti, da je en korak narobe izračunan, zato lahko ponovimo le računanje zadnjega koraka.

Ker sedaj račune za nas opravljajo računalniki, ki se naj ne bi motili, zgornje prednosti niso odločilne. A imajo iterativne metode pred direktnimi naslednje pomembne prednosti:

a) Kadar je matrika A velika in razpršena, kar pomeni, da je veliko elementov enakih 0, neničelni elementi pa nimajo kakšne posebne oblike, potem se pri direktnih metodah razpršenost ponavadi izgubi. Zaradi tega nam lahko npr. pri uporabi LU razcepa za zapis matrik L in U zmanjka pomnilnika. V takih primerih je lahko iterativna metoda edina možnost, da sploh lahko pridemo do rešitve oziroma dovolj dobrega približka.

b) Pri iterativnih metodah lahko vplivamo na natančnost izračunanega rezultata. Več iteracij ko izvedemo, več dela porabimo, a pridemo zato do natančnejšega približka. Kadar nam zadošča, da rešitev izračunamo le na nekaj decimalk natančno, lahko naredimo le toliko korakov iterativne metode, kot jih je potrebno, da dosežemo to natančnost. Pri direktnih metodah te izbire nimamo, saj vedno porabimo isto število operacij, da pridemo do rezultata.

Gauss–Jacobijeva (v nadaljevanju jo bomo imenovali samo Jacobijeva) metoda ne konvergira vedno. Pokazali bomo, da npr. vedno konvergira za strogo diagonalno dominantne matrike, za splošne pa ne vedno.

Gauss je vpeljal tudi različico, kjer je za M izbral $L + D$ (cel spodnji trikotnik matrike A). To je Gauss–Seidelova metoda, ki se je izkazala primerna za simetrične pozitivno definitne matrike. Pokazali bomo, da za takšne matrike vedno konvergira. Simetrične pozitivno definitne matrike srečamo v normalnih sistemih pri reševanju predoločenih sistemov po metodi najmanjših kvadratov, s čimer se je prvi ukvarjal ravno Gauss. Algoritem je zapisan v algoritmu 2.3.

Algoritem 2.3 Originalna Gauss–Seidelova metoda za iterativno reševanje sistema linearnih enačb $Ax = b$.

reši $(L + D)x_0 = b$
$k = 0, 1, \ldots$
    reši $(L + D)y_k = b - Ax_k$
    $x_{k+1} = x_k + y_k$

2.2 Iteracijska matrika

V tem razdelku bomo razvili splošno teorijo, ki bo pokrila klasične iterativne metode za reševanje sistemov linearnih enačb. Tako bomo lahko ugotovili, kdaj kakšna iterativna metoda konvergira in kako hitro.

Iteracija pri klasičnih iteracijskih metodah je podobna navadni iteraciji za reševanje sistemov nelinearnih enačb. Linearni sistem $Ax = b$ zapišemo v ekvivalentni obliki $x = Rx + c$ in ga rešujemo iterativno

$$x^{(r+1)} = Rx^{(r)} + c.$$

Matriko R imenujemo *iteracijska matrika*. Upamo, da bo pri čim blažjih pogojih zaporedje $\{x^{(r)}\}$ konvergiralo proti rešitvi sistema $Ax = b$. Vedeti moramo:

a) kdaj zaključiti z iteracijami,

b) pri kakšnih pogojih zaporedje konvergira,

c) kako iz $Ax = b$ pridemo do ekvivalentne oblike $x = Rx + c$.

Najprej si poglejmo dva kriterija, ki ju uporabljamo za zaključek iteracije:

- $\|x^{(r+1)} - x^{(r)}\| \leq \epsilon \|x^{(r)}\|$,

- $\|Ax^{(r+1)} - b\| \leq \epsilon(\|A\|\|x^{(r+1)}\| + \|b\|)$.

Pri drugem kriteriju res testiramo, kako dober približek je $x^{(r+1)}$, a moramo zato množiti z matriko A. Prvi kriterij je cenejši, a je pri počasni konvergenci lahko izpolnjen tudi, ko smo še daleč od prave rešitve.

Izrek 2.2 *Zaporedje $x^{(r+1)} = Rx^{(r)} + c$, $r = 0, 1, \ldots$, za poljuben začetni vektor $x^{(0)}$ konvergira natanko tedaj, ko za spektralni radij matrike R velja $\rho(R) < 1$ (kar pomeni, da za vse lastne vrednosti λ matrike R velja $|\lambda| < 1$).*

Dokaz. Naj bo $\widehat{x}$ točna rešitev. Potem iz $\widehat{x} = R\widehat{x} + c$ in $x^{(r+1)} = Rx^{(r)} + c$ sledi

$$\widehat{x} - x^{(r+1)} = R(\widehat{x} - x^{(r)})$$

in naprej

$$\widehat{x} - x^{(r+1)} = R^2(\widehat{x} - x^{(r-1)}) = \cdots = R^{r+1}(\widehat{x} - x^{(0)}).$$

Očitno je potreben in zadosten pogoj za konvergenco za poljuben $x^{(0)}$, da velja $\lim_{k\to\infty} R^k = 0$. Matriko R lahko zapišemo v Jordanovi formi v obliki $R = XJX^{-1}$, kjer je $J = \operatorname{diag}(J_1, \ldots, J_k)$. Za vsak Jordanov blok J_i velja, da je

$$J_i^k = \begin{bmatrix} \lambda_i^k & \binom{k}{1}\lambda_i^{k-1} & \binom{k}{2}\lambda_i^{k-2} & \cdots & \\ & \lambda_i^k & \binom{k}{1}\lambda_i^{k-1} & \ddots & \vdots \\ & & \ddots & \ddots & \binom{k}{2}\lambda_i^{k-2} \\ & & & \lambda_i^k & \binom{k}{1}\lambda_i^{k-1} \\ & & & & \lambda_i^k \end{bmatrix},$$

torej je $\lim_{k\to\infty} J_i^k = 0$ natanko tedaj, ko je $|\lambda_i| < 1$. ∎

Posledica 2.3 *Zadosten pogoj za konvergenco zaporedja $x^{(r+1)} = Rx^{(r)} + c$, $r = 0, 1, \ldots$, za poljuben začetni vektor $x^{(0)}$ je, da v neki matrični normi velja $\|R\| < 1$.*

Opazimo lahko, da konvergenca ni odvisna od izbire začetnega približka. To je drugače kot pri splošnem sistemu nelinearnih enačb, kjer ponavadi metoda konvergira le, če je začetni približek dovolj dober. V primeru, ko so izpolnjene predpostavke izreka 2.2, bo iterativna metoda v primeru dobrega začetnega vektorja $x^{(0)}$ skonvergirala v manj korakih, a skonvergirala bo tudi, če za začetni približek vzamemo kar $x^{(0)} = 0$.

Kako pridemo do iteracijske matrike R? En način je, da matriko A zapišemo kot $A = M + N$, potem pa sistem $Ax = b$ zapišemo kot $Mx = -Nx + b$ in dobimo $R := -M^{-1}N$. Iteracije seveda izvajamo tako, da rešujemo sisteme linearnih enačb oblike

$$Mx^{(r+1)} = -Nx^{(r)} + b, \quad r = 0, 1, \ldots,$$

matriko M pa izberemo tako, da znamo sistem z matriko M rešiti hitreje od polnega sistema z matriko A. Temu pogoju npr. zadostimo, če za M izberemo diagonalno ali trikotno matriko.

2.3 Jacobijeva in Gauss–Seidelova metoda

Izpeljimo Jacobijevo in Gauss–Seidelovo metodo še na drug način. Sistem $Ax = b$ lahko, pri pogoju $a_{ii} \neq 0$ za $i = 1, \ldots, n$, zapišemo kot

$$\begin{aligned} x_1 &= \frac{1}{a_{11}}(b_1 - a_{12}x_2 - a_{13}x_3 - \cdots - a_{1n}x_n) \\ x_2 &= \frac{1}{a_{22}}(b_2 - a_{21}x_1 - a_{23}x_3 - \cdots - a_{2n}x_n) \\ &\vdots \\ x_n &= \frac{1}{a_{nn}}(b_n - a_{n1}x_1 - a_{n2}x_n - \cdots - a_{n,n-1}x_{n-1}) \end{aligned}$$

V zgornje enačbe na desno stran vstavimo elemente vektorja $x^{(r)}$ in iz njih izračunamo elemente novega približka $x^{(r+1)}$. Tako pridemo do *Jacobijeve metode*

$$x_k^{(r+1)} = \frac{1}{a_{kk}} \left(b_k - \sum_{i=1,\ i\neq k}^{n} a_{ki} x_i^{(r)} \right), \quad k = 1, \ldots, n.$$

Če zapišemo $A = L + D + U$, kjer je L spodnji trikotnik matrike A brez diagonale, D diagonala, U pa zgornji trikotnik matrike A brez diagonale, potem je $M = D$ in $N = L + U$. Jacobijeva iteracijska matrika je $R_J = -D^{-1}(L + U)$.

Ko po vrsti računamo $x_1^{(r+1)}, \ldots, x_n^{(r+1)}$, bi lahko pri računanju $x_k^{(r+1)}$ uporabili že izračunane vrednosti $x_1^{(r+1)}, \ldots, x_{k-1}^{(r+1)}$. Tako dobimo *Gauss–Seidelovo metodo*

$$x_k^{(r+1)} = \frac{1}{a_{kk}} \left(b_k - \sum_{i=1}^{k-1} a_{ki} x_i^{(r+1)} - \sum_{i=k+1}^{n} a_{ki} x_i^{(r)} \right), \quad k = 1, \ldots, n.$$

Pri Gauss–Seidelovi metodi je $M = L + D$, $N = U$ in $R_{GS} = -(L + D)^{-1}U$.

Pri Jacobijevi metodi lahko vse elemente vektorja $x^{(r+1)}$ računamo hkrati in je zato zelo primerna za paralelizacijo. Pri Gauss–Seidelovi metodi to za splošno matriko ni možno.

Izrek 2.4 *Če je A strogo diagonalno dominantna po vrsticah, kar pomeni*

$$|a_{ii}| > \sum_{j=1, j\neq i}^{n} |a_{ij}| \quad za \quad i = 1, \ldots, n,$$

potem Jacobijeva in Gauss–Seidelova metoda konvergirata za poljuben začetni približek.

Dokaz. Pri Jacobijevi metodi iz diagonalne dominantnosti očitno sledi $\|R_J\|_\infty < 1$.

Za dokaz konvergence Gauss–Seidelove metode naj bo (λ, x) lastni par za R_{GS}, torej

$$R_{GS}x = \lambda x$$

in $x \neq 0$. Od tod sledi

$$-\sum_{j=i+1}^{n} a_{ij}x_j = \lambda \sum_{j=1}^{i} a_{ij}x_j$$

in

$$-\lambda a_{ii}x_i = \lambda \sum_{j=1}^{i-1} a_{ij}x_j + \sum_{j=i+1}^{n} a_{ij}x_j.$$

Vzemimo indeks k, pri katerem je $|x_k| = \|x\|_\infty$. Sedaj lahko ocenimo

$$\begin{aligned} |\lambda||a_{kk}| &\leq |\lambda| \sum_{j=1}^{k-1} |a_{kj}| \frac{|x_j|}{|x_k|} + \sum_{j=k+1}^{n} |a_{kj}| \frac{|x_j|}{|x_k|} \\ &\leq |\lambda| \sum_{j=1}^{k-1} |a_{kj}| + \sum_{j=k+1}^{n} |a_{kj}|. \end{aligned}$$

Sledi

$$|\lambda| \leq \frac{\sum_{j=k+1}^{n} |a_{kj}|}{|a_{kk}| - \sum_{j=1}^{k-1} |a_{kj}|} < 1,$$

saj je zaradi stroge diagonalne dominantnosti

$$|a_{kk}| > \sum_{j=1}^{k-1} |a_{kj}| + \sum_{j=k+1}^{n} |a_{kj}|.$$

■

Izrek 2.5 *Gauss–Seidelova metoda konvergira za hermitsko pozitivno definitno matriko A za poljuben začetni približek.*

Dokaz. $A = A^H$, torej $L = U^H$ in $R_{GS} = -(U^H + D)^{-1}U$. Naj bo (λ, x) lastni par za R_{GS}, torej $x \neq 0$ in

$$-Ux = (U^H + D)\lambda x.$$

Enačbo pomnožimo z x^H in dobimo

$$\lambda = -\frac{x^H U x}{x^H U^H x + x^H D x}.$$

Sedaj definiramo

$$\sigma := x^H D x = \sum_{i=1}^{n} a_{ii}|x_i|^2 > 0$$

in

$$\alpha + \beta i := x^H U x,$$

torej $x^H U^H x = \alpha - \beta i$. Dobimo

$$\lambda = -\frac{\alpha + \beta i}{\sigma + \alpha - \beta i}$$

in

$$|\lambda|^2 = \frac{\alpha^2 + \beta^2}{(\sigma + \alpha)^2 + \beta^2}.$$

Ker je A pozitivno definitna, je $x^H A x = x^H U^H x + x^H D x + x^H U x = 2\alpha + \sigma > 0$ in

$$(\sigma + \alpha)^2 = \sigma(\sigma + 2\alpha) + \alpha^2 > \alpha^2,$$

kar pomeni, da je $|\lambda| < 1$. ■

Zgled 2.1 *Za sistem*

$$\begin{aligned} 12x_1 - 3x_2 + x_3 &= 10 \\ -x_1 + 9x_2 + 2x_3 &= 10 \\ x_1 - x_2 + 10x_3 &= 10 \end{aligned}$$

in začetni približek $x^{(0)} = [1\ 0\ 1]^T$ *izračunajmo dva koraka po Jacobijevi in Gauss–Seidelovi metodi. Pri Jacobijevi metodi dobimo*

$$x^{(1)} = \begin{bmatrix} 0.75 \\ 1 \\ 0.9 \end{bmatrix}, \quad x^{(2)} = \begin{bmatrix} 1.00833... \\ 0.99444... \\ 1.025 \end{bmatrix},$$

pri Gauss–Seidelovi metodi pa

$$x^{(1)} = \begin{bmatrix} 0.75 \\ 0.9722... \\ 1.022... \end{bmatrix}, \quad x^{(2)} = \begin{bmatrix} 0.991203... \\ 0.994084... \\ 1.0002881 \end{bmatrix}.$$

Točen rezultat je $\widehat{x} = [1\ 1\ 1]^T$. □

Izrek 2.6 (Geršgorin) *Vse lastne vrednosti* $n \times n$ *matrike* A *ležijo v uniji krogov*

$$K_i := \left\{ z : |z - a_{ii} \leq \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij} \right\}, \quad i = 1, \ldots, n.$$

Dokaz. Naj bo $Ax = \lambda x$ za $x \neq 0$ in naj bo $|x_i| = \|x\|_\infty$. Potem velja

$$\sum_{j=1}^{n} a_{ij} x_j = \lambda x_i.$$

Od tod sledi

$$\lambda - a_{ii} = \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij} \frac{x_j}{x_i}$$

in dobimo oceno

$$|\lambda - a_{ii}| \leq \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}|$$

iz katere sledi $\lambda \in K_i$. ■

Krogom iz zgornjega izreka pravimo Geršgorinovi krogi. Velja tudi, da v primeru, če unija krogov ni povezana, potem vsaka povezana komponenta k krogov vsebuje natanko k lastnih vrednosti.

Izrek 2.7 *Če je* A *strogo diagonalno dominantna matrika, potem je* A *nesingularna.*

Dokaz. To je očitno, saj v tem primeru 0 ne more biti vsebovana v nobenem Geršgorinovemu krogu. ■

Definicija 2.8 *Matrika A je* nerazcepna, *če ne obstaja taka permutacijska matrika P, da je*

$$PAP^T = \begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix}$$

in sta matriki A_{11} in A_{22} kvadratni.

Nerazcepnost matrike A je ekvivalentna temu, da je graf $G(A)$, ki pripada matriki (točke so $1, \ldots, n$, povezave pa ij, kjer je $a_{ij} \neq 0$), krepko povezan, kar pomeni, da med poljubnima dvema točkama v grafu obstaja usmerjena pot.

Izrek 2.9 *Naj bo A nerazcepna matrika. Če njena lastna vrednost λ leži na robu unije Geršgorinovih krogov, potem λ leži na robu vsakega Geršgorinovega kroga.*

Dokaz. Privzemimo, da je x lastni vektor za λ, ki je normiran tako, da je $\|x\|_\infty = 1$. Če je $|x_i| = 1$, potem lahko podobno kot pri prejšnjem izreku pokažemo, da je

$$|\lambda - a_{ii}| \leq \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}|\,|x_j| \leq \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}|, \tag{2.1}$$

torej λ leži v krogu K_i. Ker λ leži na robu unije vseh Geršgorinovih krogov, ne more ležati v notranjosti K_i. Zato morajo v (2.1) veljati same enakosti, kar pomeni, da iz $a_{ij} \neq 0$ sledi $|x_j| = 1$.

Ker je A nerazcepna, obstaja pot od i do poljubne druge točke $k \neq i$, to naj bo npr. pot $i = i_1, i_2, \ldots, i_m = k$. Zato mora biti $a_{i,i_2} \neq 0$ in zato $|x_{i_2}| = 1$. Če za i_2 ponovimo zgornje argumente, mora potem λ ležati tudi na robu kroga K_{i_2}. To ponavljamo in ker je A nerazcepna, lahko tako s potmi pokrijemo vse točke in pokažemo, da λ leži na robu vsakega kroga. ∎

Izrek 2.10 *Če je A šibko diagonalno dominantna matrika in nerazcepna, potem je nesingularna.*

Dokaz. Denimo, da je 0 lastna vrednost matrike A. Zaradi šibke diagonalne dominantnosti bi morala potem 0 ležati na robu unije Geršgorinovih krogov. Toda, ker je matrika nerazcepna, potem po prejšnjem izreku 0 leži na robu vsakega kroga. Tako smo prišli do protislovja, saj bi potem za vsak i veljalo

$$|a_{ii}| = \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}|$$

in matrika ni šibko diagonalno dominantna. ∎

Izrek 2.11 *Če je A šibko diagonalno dominantna matrika in nerazcepna, potem Jacobijeva in Gauss–Seidelova metoda konvergirata za vsak začetni vektor.*

Dokaz. Če ponovimo dokaz izreka 2.4 za šibko diagonalno dominantno matriko, potem sledi, da velja $\rho(R_J) \leq 1$ in $\rho(R_{GS}) \leq 1$. Pokazati moramo tako le, da ni možno, da bi bil spektralni radij ene ali druge matrike enak 1.

Denimo, da je $|\lambda| = 1$ lastna vrednost iteracijske matrike $R = -M^{-1}N$, kjer je bodisi $M = D, N = L + U$ ali $M = L + D, N = U$. Potem iz $M^{-1}Nx = -\lambda x$ sledi $Nx = -\lambda Mx$ in $N + \lambda M$ je singularna matrika. Toda, ker je zaradi $|\lambda| = 1$ matrika $N + \lambda M$ še vedno šibko diagonalno dominantna in nerazcepna, to po zadnjem izreku ni možno in prišli smo do protislovja. ∎

2.4 SOR metoda

Pri metodi *SOR*[1] računamo

$$x_k^{(r+1)SOR} = x_k^{(r)} + \omega \left(x_k^{(r+1)GS} - x_k^{(r)} \right) \quad \text{za} \quad k = 1, \ldots, n,$$

kjer je ω *relaksacijski parameter*, za katerega se izkaže, da mora veljati $0 < \omega < 2$. Pri $\omega = 1$ je SOR kar Gauss–Seidelova metoda.

Dobimo

$$x_k^{(r+1)} = x_k^{(r)} + \omega \frac{1}{a_{kk}} \left(b_k - \sum_{i=1}^{k-1} a_{ki} x_i^{(r+1)} - \sum_{i=k}^{n} a_{ki} x_i^{(r)} \right) \quad \text{za} \quad k = 1, \ldots, n.$$

Gre za pospešitev Gauss–Seidelove metode, ideja pa je, da bo, če je zaporedje monotono, za primerni parameter $\omega > 1$ približek $x^{(r+1)SOR}$ bližje rešitvi kot $x^{(r+1)GS}$, če pa zaporedje alternira, bo boljši približek pri $0 < \omega < 1$. Optimalni ω je težko oceniti.

V matrični obliki velja

$$x^{(r+1)} = x^{(r)} + \omega D^{-1} \left(b - L x^{(r+1)} - (U + D) x^{(r)} \right)$$

oziroma

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

kar pomeni

$$R_{SOR} = (\omega L + D)^{-1}((1 - \omega)D - \omega U). \tag{2.2}$$

Izrek 2.12 *Metoda SOR ne more konvergirati za poljuben začetni vektor v primeru $\omega \notin (0, 2)$.*

Dokaz. Matrika $(\omega L + D)^{-1}$ je spodnja trikotna in ima na diagonali elemente a_{ii}^{-1} za $i = 1, \ldots, n$, matrika $(1 - \omega)D - \omega U$ pa je zgornja trikotna in ima na diagonali elemente $(1 - \omega)a_{ii}$ za $i = 1, \ldots, n$. Od tod iz enakosti (2.2) sledi

$$\det R_{SOR} = (1 - \omega)^n.$$

Ker je determinanta enaka produktu lastnih vrednosti, velja $\rho(R_{SOR}) \geq |1 - \omega|$ in potreben pogoj za konvergenco je $\omega \in (0, 2)$. ∎

Izrek 2.13 *Če je A hermitska pozitivno definitna matrika, potem metoda SOR v primeru $\omega \in (0, 2)$ konvergira za poljuben začetni vektor.*

Dokaz. Dokaz je posplošitev dokaza izreka 2.5, ki je le poseben primer za $\omega = 1$. ∎

Izrek 2.14 *Če je $n \times n$ matrika A nerazcepna in šibko diagonalno dominantna po vrsticah, kar pomeni $|a_{ii}| \geq \sum_{k=1, k \neq i}^{n} |a_{ik}|$ za $i = 1, \ldots, n$, pri čemer je vsaj v eni vrstici stroga neenakost, potem Jacobijeva in Gauss–Seidelova metoda konvergirata in velja $\rho(R_{GS}) < \rho(R_J) < 1$.* ∎

[1] SOR je angleška kratica za Successive Over-Relaxation.

Definicija 2.15 *Matrika $A = L + D + U$ je* konsistentno urejena, *če so lastne vrednosti matrike*

$$R_J(\alpha) = -D^{-1}\left(\alpha L + \frac{1}{\alpha}U\right)$$

neodvisne od $\alpha \neq 0$.

Trditev 2.16 *Če ima matrika obliko*

$$A = \begin{bmatrix} D_1 & A_{12} \\ A_{21} & D_2 \end{bmatrix},$$

kjer sta D_1 in D_2 nesingularni diagonalni matriki, potem je A konsistentno urejena.

Dokaz. Če A razdelimo, dobimo $A = L + D + U$, kjer je

$$L = \begin{bmatrix} 0 & 0 \\ A_{21} & 0 \end{bmatrix}, \quad D = \begin{bmatrix} D_1 & 0 \\ 0 & D_2 \end{bmatrix}, \quad U = \begin{bmatrix} 0 & A_{12} \\ 0 & 0 \end{bmatrix}.$$

Tako je

$$R_J(\alpha) = -D^{-1}\left(\alpha L + \frac{1}{\alpha}U\right) = \begin{bmatrix} 0 & -\frac{1}{\alpha}D_1^{-1}A_{12} \\ -\alpha D_2^{-1}A_{21} & 0 \end{bmatrix}.$$

Ker je

$$\begin{bmatrix} I & \\ & \frac{1}{\alpha}I \end{bmatrix} R_J(\alpha) \begin{bmatrix} I & \\ & \alpha I \end{bmatrix} = R_J(1),$$

so lastne vrednosti $R_J(\alpha)$ res neodvisne od α. ∎

Definicija 2.17 *Matrika A ima* lastnost A, *če obstaja taka permutacijska matrika P, da je*

$$PAP^T = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix},$$

kjer sta A_{11} in A_{22} diagonalni matriki.

Matrika A ima lastnost A natanko tedaj, ko je graf $G(A)$ dvodelen (bipartiten), pri čemer dopuščamo zanke. Točke v takem grafu lahko razdelimo na taka dva dela V_1 in V_2, da so potem vse povezave oblike ik za $i \neq k$ take, da je bodisi $i \in V_1, k \in V_2$ ali $i \in V_2, k \in V_1$.

Vsako matriko z lastnostjo A lahko torej s permutacijsko matriko P spremenimo v konsistentno matriko. Če npr. uporabljamo Jacobijevo ali Gauss–Seidelovo metoda, lahko torej vrstni red spremenljivk izberemo tako, da je matrika konsistentno urejena.

Izrek 2.18 *Naj bo matrika $A = L + D + U$ konsistentno urejena matrika z nesingularno diagonalno matriko. Potem za $\omega \neq 0$ velja:*

a) Če je μ lastna vrednost R_J, je tudi $-\mu$ lastna vrednost R_J.

b) Če je $\lambda \neq 0$ lastna vrednost $R_{\mathrm{SOR}}(\omega)$, je vsak μ ki zadošča

$$(\lambda + \omega - 1)^2 = \lambda\mu^2\omega^2 \tag{2.3}$$

lastna vrednost R_J.

c) Če je μ lastna vrednost R_J in λ zadošča (2.3), je λ lastna vrednost $R_{\mathrm{SOR}}(\omega)$.

Dokaz. Točka a) sledi iz dejstva, da ima zaradi konsistentne urejenosti A matrika $R_J = R_J(1)$ iste lastne vrednosti kot matrika $R_J(-1) = -R_J$.

Naj bo $\lambda \neq 0$ lastna vrednost $R_{\mathrm{SOR}}(\omega)$. To pomeni, da je

$$0 = \det\big((\omega L + D)^{-1}((1-\omega)D - \omega U) - \lambda I\big).$$

Ker je $\omega L + D$ nesingularna, je to ekvivalentno

$$0 = \det\big((1-\omega)D - \omega U - \lambda(\omega L + D)\big)$$

in dalje ekvivalentno

$$\begin{aligned} 0 &= \det\big((1-\omega)I - \omega D^{-1}U - \lambda(\omega D^{-1}L + I)\big) \\ &= \det\big((1-\omega-\lambda)I - \omega(D^{-1}U + \lambda D^{-1}L)\big). \end{aligned}$$

To pomeni, da je $(\lambda+\omega-1)/\omega$ lastna vrednost $\sqrt{\lambda}R_J(\sqrt{\lambda})$, oziroma lastna vrednost $\sqrt{\lambda}R_J$, saj se lastne vrednosti $R_J(\lambda)$ in R_J ujemajo. To je ekvivalentno zvezi

$$\frac{\lambda+\omega-1}{\omega} = \sqrt{\lambda}\mu,$$

od koder sledi (2.3).

Če je $\lambda \neq 0$, lahko gremo tudi v drugo smer in tako pokažemo, da je v primeru, ko je μ lastna vrednost R_J in $\lambda \neq 0$ reši (2.3), potem λ lastna vrednost $R_{\mathrm{SOR}}(\omega)$.

Če je μ lastna vrednost R_J, potem enakost (2.3) lahko velja tudi za $\lambda = 0$. V tem primeru mora potem veljati $\omega = 1$ in ker je $R_{\mathrm{SOR}}(1) = R_{GS}$, je $\lambda = 0$ res lastna vrednost $R_{\mathrm{SOR}}(\omega)$. Hitro lahko namreč preverimo, da je matrika R_{GS} vedno singularna, saj je njen prvi stolpec enak 0. ∎

Izrek 2.19 *Če je A konsistentno urejena in $\mu = \rho(R_J)$, potem velja:*

1) $\rho(R_{GS}) = \mu^2$,

2) $\omega_{\mathrm{opt}} = \dfrac{2}{1+\sqrt{1-\mu^2}}$, $\quad \rho(R_{SOR}(\omega_{\mathrm{opt}})) = \omega_{\mathrm{opt}} - 1$,

3)

$$\rho(R_{SOR}(\omega)) = \begin{cases} \omega - 1, & \text{za } \omega_{\mathrm{opt}} \leq \omega < 2 \\ 1 - \omega + \frac{1}{2}\omega^2\mu^2 + \omega\mu\sqrt{1-\omega+\frac{1}{4}\omega^2\mu^2}, & \text{za } 0 < \omega \leq \omega_{\mathrm{opt}}. \end{cases}$$

Dokaz. *Če iz (2.3) izrazimo λ, dobimo*

$$\lambda_{1,2} = 1 - \omega + \frac{1}{2}\mu^2\omega^2 \pm \omega\mu\sqrt{1-\omega+\frac{1}{4}\mu^2\omega^2}. \tag{2.4}$$

Sedaj ločimo dve možnosti glede na predznak izraza pod kvadratnim korenom. Vemo, da je $0 < \omega < 2$ in $0 < \mu < 1$. Od tod lahko ugotovimo, da je mejni $\omega \in (0,2)$, pri katerem $1 - \omega + \frac{1}{4}\mu^2\omega^2$ spremeni predznak, enak

$$w_0 = \frac{2}{1+\sqrt{1-\mu^2}}.$$

Če je $\omega_0 \leq \omega < 2$, sta rešitvi (2.4) konjugirani kompleksni števili. Za absolutno vrednost velja

$$|\lambda|^2 = \Big(1 - \omega + \frac{1}{2}\omega^2\mu^2\Big)^2 + \omega^2\mu^2\Big(\omega - 1 - \frac{1}{4}\mu^2\omega^2\Big) = (1-\omega)^2.$$

Ker je $1 \leq \omega_0$, sledi $\rho(R_{\mathrm{SOR}}(\omega)) = \omega - 1$. Opazimo še, da je na intervalu $\omega_0 < \omega < 2$ spektralni radij metode SOR naraščajoča funkcija parametra ω.

Če je $0 < \omega \leq \omega_0$, sta obe rešitvi (2.4) realni. Po absolutni vrednosti večjo dobimo, če v (2.4) izberemo $+$, kar da

$$\rho(R_{\mathrm{SOR}}(\omega)) = 1 - \omega + \frac{1}{2}\omega^2\mu^2 + \omega\mu\sqrt{1 - \omega + \frac{1}{4}\omega^2\mu^2}.$$

Z odvajanjem zgornjega izraza bi ugotovili, da je na intervalu $0 < \omega < \omega_0$ spektralni radij padajoča funkcija parametra ω. Zaradi tega je minimum dosežen ravno pri ω_0 in $\omega_{\mathrm{opt}} = \omega_0$. ∎

Zgled 2.2 *Matrika bločne oblike*

$$A = \begin{bmatrix} T & -I & & \\ -I & T & \ddots & \\ & \ddots & \ddots & -I \\ & & -I & T \end{bmatrix}, \tag{2.5}$$

kjer je T tridiagonalna matrika

$$T = \begin{bmatrix} 4 & -1 & & \\ -1 & 4 & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 4 \end{bmatrix},$$

ima lastnost A. Takšne matrike srečamo pri reševanju Poissonove parcialne diferencialne enačbe. Denimo, da rešujemo parcialno diferencialno enačbo

$$u_{xx} + u_{yy} = 0 \tag{2.6}$$

na območju $\Omega = [0,1] \times [0,1]$ z robnim pogojem $u = f$ na $\partial\Omega$. Pri diferenčni metodi območje Ω prekrijemo z mrežo, kjer intervala $[0,1]$ ekvidistantno razdelimo s točkami $x_0 = 0, x_1, \ldots, x_n, x_{n+1} = 1$ in $y_0 = 0, y_1, \ldots, y_n, y_{n+1} = 1$. Če označimo $h = 1/(n+1)$, potem je $x_i = ih$ in $y_j = jh$ za $i, j = 0, \ldots, n+1$.

Če v točki (x_i, y_j) enačbo (2.6) aproksimiramo tako, da druga parcialna odvoda funkcije u po x in y aproksimiramo s simetričnima diferencama, dobimo pettočkovno shemo, ki je predstavljena z enačbo

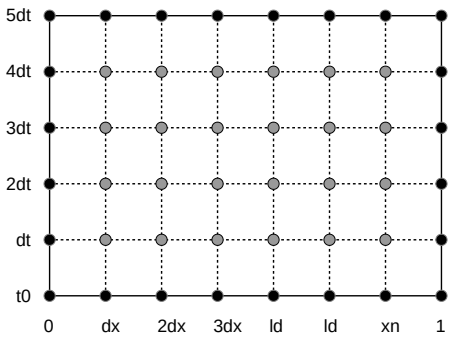
$$\frac{u_{i-1,j} - 2u_{ij} + u_{i+1,j}}{h^2} + \frac{u_{i,j-1} - 2u_{ij} + u_{i,j+1}}{h^2} = 0,$$

od koder sledi

$$u_{ij} = \frac{1}{4}(u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1})$$

za $i, j = 1, \ldots, n$. Vrednost v vsaki točki mreže mora biti torej enaka povprečju vrednosti njenih štirih sosed in tako vrednosti na robu mreže določajo vrednosti v notranjosti.

Če enačbe za vse vrednosti u_{ij} znotraj mreže v pravem vrstnem redu sestavimo v sistem linearnih enačb $Au = b$ velikosti $n^2 \times n^2$, potem ima matrika obliko (2.5). V nadaljevanju bomo sistem s takšno matriko



Slika 2.1: Mreža diferenčne metode za Poissonov problem na $[0,1] \times [0,1]$ z Dirichletovim robnim pogojem. Vrednosti v črnih točkah so znane iz robnih pogojev, druge dobimo iz sistema linearnih enačb.

imenovali modelni problem, saj bomo na njem testirali konvergenco klasičnih iterativnih metod. V praksi sploh ni potrebno, da je matrika A eksplicitno zapisana v pomnilniku, saj lahko en korak Jacobijeve ali Gauss–Seidelove metode izvedemo tako, da gremo v zanki skozi vse točke mreže in posodabljamo njihove vrednosti. Pri Jacobijevi metodi potrebujemo dve mreži, saj moramo imeti shranjene nove in stare vrednosti, pri Gauss–Seidelovi metodi in SOR pa eno samo mrežo.

	Jacobi		Gauss–Seidel		SOR	
n	$\rho(R_j)$	korakov	$\rho(R_{GS})$	korakov	$\rho(R_{SOR})$	korakov
5	0.866054	160	0.750000	80	0.333333	21
10	0.959493	557	0.920628	278	0.560388	40
100	0.999516	47590	0.999033	23795	0.939676	370
1000	0.999995	$4.68 \cdot 10^6$	0.999990	$2.34 \cdot 10^6$	0.993743	3668

Tabela 2.1: Primerjava Jacobijeve, Gauss–Seidelove in SOR metode (pri optimalnem ω) na matrikah $n^2 \times n^2$ oblike (2.5). Stolpci prikazujejo spektralne radije iteracijskih matrik $\rho(R)$ in potrebno število korakov k, da velja $\rho(R^k) < 10^{-10}$.

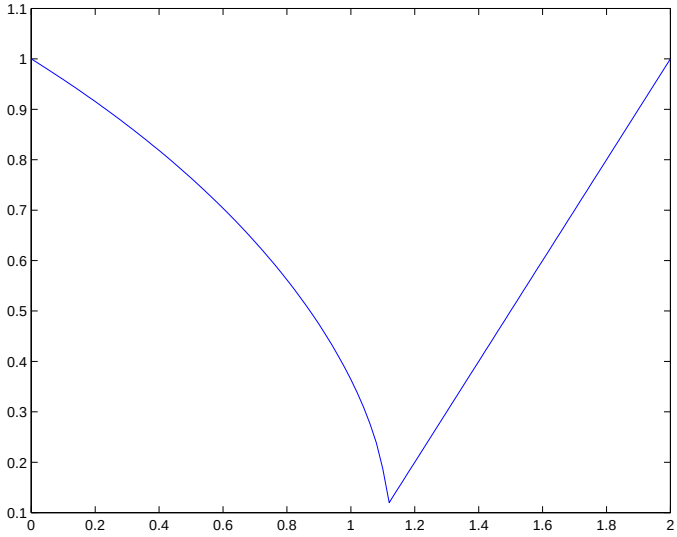
Če je T velikosti $n \times n$ in A velikosti $n^2 \times n^2$, potem tabela 2.1 prikazuje spektralne radije iteracijskih matrik in število potrebnih korakov, da velja $\rho(R^k) < 10^{-10}$, če za iterativno metodo izberemo Jacobijevo metodo, Gauss–Seidelovo metodo in metodo SOR (pri optimalnem ω). Ko n narašča, gredo vsi spektralni radiji proti 1*. Zaradi tega so večji problemi podobne oblike zelo zahtevni za reševanje. Pri večjem n porabimo več operacij v vsakem koraku, poveča pa se tudi število iteracij, ki jih potrebuje iteracijska metoda. Glede na to, da je časovna zahtevnost enega koraka vseh treh metod enaka, vidimo, da metoda SOR daje neprimerno boljše rezultate.* □

Zgled 2.3 *Za matriko*

$$A = \begin{bmatrix} 4 & -1 & & -1 & & \\ -1 & 4 & -1 & & -1 & \\ & -1 & 4 & & & -1 \\ -1 & & & 4 & -1 & \\ & -1 & & -1 & 4 & -1 \\ & & -1 & & -1 & 4 \end{bmatrix} \tag{2.7}$$

velja $\rho(R_J) = 0.6036$, $\rho(R_{GS}) = 0.3634$, $\omega_{\text{opt}} = 1.1128$, graf spektralnega radija $\rho(R_{SOR}(\omega))$ za $\omega \in [0,2]$ pa je prikazan na sliki 2.2. Opazimo lahko, da že v bližini optimalnega ω spektralni radij na

obeh straneh hitro narašča. Pri večjih matrikah je to še bolj očitno in, če ne izberemo dobrega približka za optimalno vrednost ω, se število iteracij lahko močno poveča. □



Slika 2.2: Graf spektralnega radija matrike SOR za matriko (2.7) in $\omega \in [0, 2]$.

Če točke v mreži uredimo kot pri šahovnici na črne in bele, vidimo, da je graf $G(A)$ dvodelen, saj so bele točke povezane le s črnimi in obratno. Če izberemo tako permutacijsko matriko P, da bodo v prvem delu samo bele, v drugem pa same črne točke, vidimo, da ima matrika lastnost A. Ta tako imenovana črno-bela ureditev nam omogoča, da tudi pri Gauss–Seidelovi metodi lahko uporabljamo paralelno računanje, saj lahko vrednosti vseh belih točk oziroma vseh črnih točk izračunamo naenkrat.

2.5 Pospešitev Čebiševa in simetrični SOR

Pri iterativnem reševanju linearnega sistema $Ax = b$ sistem prevedemo v ekvivalentno obliko $x = Rx + c$, izberemo x_0 in tvorimo zaporedje $x_{m+1} = Rx_m + c$. Denimo, da je $\rho(R) < 1$, kar pomeni da, ko gre m proti neskončnosti, približki x_m konvergirajo proti točni rešitvi $\widehat{x}$.

Vektor x_m je približek za $\widehat{x}$, še boljši približek pa bi lahko dobili z linearno kombinacijo vseh do sedaj izračunanih približkov $x_0, \ldots, x_m$ v obliki

$$y_m = \sum_{i=0}^{m} \gamma_{mi} x_i,$$

kjer so $\gamma_{m0}, \ldots, \gamma_{mm}$ primerno izbrani koeficienti. V primeru $x_0 = x_1 = \cdots = x_m = \widehat{x}$ pričakujemo, da bo tudi $y_m = \widehat{x}$, kar pomeni, da mora za vsoto koeficientov veljati $\sum_{i=0}^{m} \gamma_{mi} = 1$. Sedaj lahko zapišemo

$$y_m - \widehat{x} = \sum_{i=0}^{m} \gamma_{mi} x_i - \widehat{x} = \sum_{i=0}^{m} \gamma_{mi}(x_i - \widehat{x}) = \sum_{i=0}^{m} \gamma_{mi} R^i (x_0 - \widehat{x}) = p_m(R)(x_0 - \widehat{x}),$$

kjer je $p_m(t) = \sum_{i=0}^{m} \gamma_{mi} t^i$ polinom stopnje m, za katerega velja $p_m(1) = 1$.

Ker želimo, da bo y_m čim boljši približek za točno rešitev $\widehat{x}$, iščemo tak polinom p_m, da bo spektralni radij matrike $p_m(R)$ čim manjši, obenem pa bo $p_m(1) = 1$. Denimo, da za iteracijsko matriko R vemo:

a) ima same realne lastne vrednosti,

b) $\rho(R) \leq \rho < 1$.

Potem iščemo polinom p_m, ki bo minimalen na $[-\rho, \rho]$ in $p_m(1) = 1$. Do primernih polinomov pridemo s pomočjo polinomov Čebiševa, ki so zelo pomembni na področju enakomerne aproksimacije. Definirani so z rekurzivno zvezo

$$\begin{aligned} T_0(x) &= 1, \\ T_1(x) &= x, \\ T_{m+1}(x) &= 2xT_m(x) - T_{m-1}(x) \quad \text{za} \quad m > 1. \end{aligned}$$

Izrek 2.20 *Za polinome Čebiševa velja:*

a) $T_m(1) = 1$.

b) $T_m(x) = 2^{m-1}x^m + \mathcal{O}(x^{m-1})$.

c) $T_m(x) = \begin{cases} \cos(m \arccos x), & |x| \leq 1, \\ \cosh(m \operatorname{arccosh} x), & |x| \geq 1. \end{cases}$

č) $|T_m(x)| \leq 1$ *za* $|x| \leq 1$.

d) Izmed vseh polinomov stopnje m z vodilnim koeficientom 2^{m-1} *ima* T_m *najmanjšo* ∞*-normo na* $[-1, 1]$.

e) $T_m(1 + \epsilon) \geq 1 + m^2\epsilon$ *za* $\epsilon > 0$

Dokaz. Pokazali bomo le točko d), ostale pustimo za vajo.

Uporabili bomo dokaz s protislovjem. Denimo, da obstaja polinom p stopnje m z vodilnim koeficientom 2^{m-1}, ki ima na intervalu $[-1, 1]$ manjšo $\infty-$normo od T_m. Vemo, da T_m na intervalu $[-1, 1]$ zavzame absolutno vrednost 1 v $m + 1$ točkah $\xi_1 < \xi_2 < \cdots < \xi_{m+1}$ in da v teh točkah alternira predznak, torej $T_m(\xi_k)T_m(\xi_{k+1}) = -1$ za $k = 1, \ldots, m$. Za polinom p mora potem veljati $\|p\|_\infty < \|T_m\| = 1$, torej v vseh točkah ξ_k velja $|p(\xi_k)| < |T_m(\xi_k)| = 1$. Od tod sledi, da se predznak polinoma $T_m - p$ v točkah x_k ujema s predznakom T_m. To pa pomeni, da ima polinom $T_m - p$, ki je stopnje $m - 1$, najmanj m ničel, kar je možno le, če je T_m identično enak p in prišli smo do protislovja. ∎

Izven intervala $[-1, 1]$ vrednosti $|T_n(x)|$ zelo hitro naraščajo, tako da za $x = 1 + \epsilon$ dobimo

$m \backslash \epsilon$	10^{-4}	10^{-3}	10^{-2}
10	1.0	1.1	2.2
100	2.2	44	$6.9 \cdot 10^5$
1000	$6.9 \cdot 10^5$	$1.3 \cdot 10^{19}$	$1.2 \cdot 10^{61}$

Polinom p_m, ki ustreza našim zahtevam, je

$$p_m(t) = \frac{T_m(t/\rho)}{T_m(1/\rho)}.$$

Velja $p_m(1) = 1$, za $x \in [-\rho, \rho]$ pa je

$$p_m(x) \leq \frac{1}{T_m(1/\rho)}.$$

Če je $\rho = 1/(1+\epsilon)$, to pomeni $p_m(x) \leq \frac{1}{T_m(1+\epsilon)}$.

Če definiramo $\mu_m = T_m(1/\rho)$, potem je $p_m(t) = \frac{1}{\mu_m} T_m(t/\rho)$. Za μ_m velja zveza

$$\mu_{m+1} = \frac{2}{\rho}\mu_m - \mu_{m-1}. \tag{2.8}$$

Zaradi tričlenske rekurzivne formule za izračun y_{m+1} ne potrebujemo vseh prejšnjih približkov, temveč zadoščata le y_m in y_{m-1}. Velja namreč

$$\begin{aligned} y_{m+1} &= x + p_{m+1}(R)(x_0 - x) \\ &= x + \frac{2\mu_m}{\rho\mu_{m+1}} R p_m(R)(x_0 - x) - \frac{\mu_{m-1}}{\mu_{m+1}} p_{m-1}(R)(x_0 - x) \\ &= x + \frac{2\mu_m}{\rho\mu_{m+1}} R(y_m - x) - \frac{\mu_{m-1}}{\mu_{m+1}}(y_{m-1} - x) \\ &= \frac{2\mu_m}{\rho\mu_{m+1}} R y_m - \frac{\mu_{m-1}}{\mu_{m+1}} y_{m-1} + \frac{1}{\mu_{m+1}}\Big(\mu_{m+1} - \frac{2}{\rho}\mu_m R + \mu_{m-1}\Big)x. \end{aligned} \tag{2.9}$$

Iz (2.9) lahko izločimo x, saj se zaradi $Rx = x - c$ in (2.8) zadnji člen poenostavi v

$$\begin{aligned} \frac{1}{\mu_{m+1}}\Big(\mu_{m+1} - \frac{2}{\rho}\mu_m R + \mu_{m-1}\Big)x &= \frac{1}{\mu_{m+1}}\Big(\mu_{m+1} - \frac{2}{\rho}\mu_m + \mu_{m-1}\Big)x + \frac{2\mu_m}{\rho\mu_{m+1}} c \\ &= \frac{2\mu_m}{\rho\mu_{m+1}} c. \end{aligned}$$

Tako pridemo do končne formule za naslednji vektor:

$$y_{m+1} = \frac{2\mu_m}{\rho\mu_{m+1}}(R y_m + c) - \frac{\mu_{m-1}}{\mu_{m+1}} y_{m-1}.$$

Pri tem opazimo, da je $Ry_m + c$ v bistvu približek, ki bi ga dobili z izbrano iterativno metodo iz začetnega približka y_m. Na tem mestu torej uporabimo iterativno metodo. Celoten algoritem je zapisan v algoritmu 2.4.

Pospešitev Čebiševa lahko uporabimo za vsako iterativno metodo, katere iteracijska matrika ima realne lastne vrednosti, saj kompleksnih lastnih vrednosti polinomi Čebiševa ne uničijo. Iteracijska matrika za SOR ima lahko kompleksne lastne vrednosti, zato SOR ni primerna metoda. Pomagamo pa si lahko s simetričnim SOR (SSOR), kjer v bistvu naredimo dva koraka metode SOR enega za drugim, pri čemer pri drugem koraku obrnemo vrstni red računanja:

$$\begin{aligned} x_{i+1/2} &= (\omega L + D)^{-1}((1-\omega)D - \omega U)x_i + c_{1/2} \\ x_{i+1} &= (\omega U + D)^{-1}((1-\omega)D - \omega L)x_{i+1/2} + c_1. \end{aligned}$$

Algoritem 2.4 Iterativna metoda s pospešitvijo Čebiševa. Dani so iterativna metoda, podana z iteracijsko matriko R in vektorjem c, tak $\rho < 1$, da vse lastne vrednosti matrike R ležijo na intervalu $[-\rho, \rho]$, in začetni približek x_0.

$\mu_0 = 1$, $\mu_1 = 1/\rho$
$y_0 = x_0$, $y_1 = Rx_0 + c$
$m = 1, 2, \ldots$
  $\mu_{m+1} = \frac{2}{\rho}\mu_m - \mu_{m-1}$
  $z_{m+1} = Ry_m + c$
  $y_{m+1} = \frac{2\mu_m}{\rho\mu_{m+1}} z_{m+1} - \frac{\mu_{m-1}}{\mu_{m+1}} y_{m-1}$

Iteracijska matrika je sedaj

$$R_{SSOR} = (\omega U + D)^{-1}\Big((1-\omega)D - \omega L\Big)(\omega L + D)^{-1}\Big((1-\omega)D - \omega U\Big). \tag{2.10}$$

Zanjo lahko pokažemo, da je v primeru, ko je matrika A simetrična, podobna simetrični matriki in ima zato same realne lastne vrednosti.

Lema 2.21 *Če je matrika A simetrična, potem ima iteracijska matrika za metodo SSOR same realne lastne vrednosti.*

Dokaz. Če upoštevamo simetrijo matrike A in označimo $\widetilde{L} = LD^{-1}$, potem lahko matriko R_{SSOR} iz (2.10) zapišemo kot

$$R_{SSOR} = (I + \omega\widetilde{L})^{-T}\left((2-\omega)I - (I+\omega\widetilde{L})\right)(I+\omega\widetilde{L})^{-1}\left((2-\omega)I - (I+\omega\widetilde{L})^T\right).$$

Če označimo $M = I + \omega\widetilde{L}$, dobimo

$$\begin{aligned} R_{SSOR} &= M^{-T}\Big((2-\omega)I - M\Big)M^{-1}\Big((2-\omega)I - M^T\Big) \\ &= I + (2-\omega)M^{-T}M^{-1}M^T - (2-\omega)M^{-T} + (2-\omega)^2 M^{-T}M^{-1}, \end{aligned}$$

od koder sledi

$$M^T R_{SSOR} M^{-T} = I + (2-\omega)(M^{-1} + M^{-T}) + (2-\omega)^2 M^{-1}M^{-T}.$$

Ker je zgornja matrika simetrična, je R_{SSOR} podobna simetrični matriki in ima zato le realne lastne vrednosti. ∎

Zgled 2.4 *Na matrikah iste oblike, kot v zgledu 2.2, bomo primerjali metodi SOR in SSOR. Medtem, ko optimalni ω pri metodi SOR lahko izračunamo po izreku 2.19, podobna formula za metodo SSOR ne obstaja. Uporabili bomo vrednosti, ki jih je za tovrstne probleme predlagal Young [16]:*

$$\omega = \frac{2}{1+\sqrt{2-2\mu}}, \quad \rho = \frac{\sqrt{2-2\mu}-1+\mu}{\sqrt{2-2\mu}+1-\mu}, \tag{2.11}$$

kjer je $\mu = \rho(R_J)$. Ker se da pokazati, da za matriko A iz zgleda 2.2 velja $\rho(R_J) = \cos(\pi/(n+1))$, lahko tako izračunamo ω in ρ tudi za velik n.

	SOR		SSOR s pospešitvijo Čebiševa		
n	ω_{opt}	korakov	ω	ρ	korakov
50	1.88401814	169	1.88396630	0.94024989	53
100	1.93967633	324	1.93966926	0.96937269	73
200	1.96922267	622	1.96922174	0.98449154	104
400	1.98445317	1184	1.98445305	0.99219619	145
800	1.99218649	2294	1.99218647	0.99608559	205

Tabela 2.2: Primerjava SOR (pri optimalnem ω) in SSOR s pospešitvijo Čebiševa (pri Youngovih približkih (2.11) za optimalni ω in ρ) na $n^2 \times n^2$ matrikah oblike (2.5)

Optimalne vrednosti za ω lahko dobimo s poskušanjem, potem pa iz izračunanega spektralnega radija matrike R_{SSOR} določimo ρ. V praksi si tega seveda ne moremo privoščiti in moramo poiskati druge načine, kako določiti čim boljši vrednosti parametrov ω in ρ.

Primerjava je v tabeli 2.2. Tokrat smo za razliko od prej izračunali dejansko število korakov, ki jih je metoda potrebovala za naključno izbrani vektor b, da je norma ostanka padla pod 10^{-10}. Upoštevati moramo, da je metoda SSOR računsko dvakrat zahtevnejša od metode SOR, saj v bistvu v enem koraku naredimo dva koraka metode SOR (pri čemer je en v obratnem vrstnem redu). Kljub temu število korakov pri metodi SSOR narašča s počasnejšim redom kot pri SOR in pri večjih n se SSOR (pri predpostavki, da znamo oceniti optimalna ω in ρ) obnese bolje kot SOR. □

Pokažimo še teoretično, da je na modelnem primeru, ko gre n proti neskončno, SSOR s pospešitvijo Čebiševa res občutno hitrejši od metode SOR, ta pa je hitrejša od Jacobijeve in Gauss–Seidelove metode.

Potrebovali bomo naslednjo lemo, katere dokaz pustimo za vaje.

Lema 2.22 *Tridiagonalna $n \times n$ matrika A oblike*

$$A = \begin{bmatrix} a & b & & \\ c & a & \ddots & \\ & \ddots & \ddots & b \\ & & c & a \end{bmatrix}$$

ima lastne vrednosti

$$\lambda_k = a + 2\sqrt{bc}\cos\left(\frac{k\pi}{n+1}\right), \quad k = 1, \ldots, n.$$

Prav tako bomo v nadaljevanju potrebovali naslednji definiciji. *Kroneckerjev produkt* matrik $W \in \mathbb{R}^{p\times q}$ in $Z \in \mathbb{R}^{s\times t}$ je matrika velikosti $ps \times qt$ bločne oblike

$$W \otimes Z = \begin{bmatrix} w_{11}Z & \cdots & w_{1p}Z \\ \vdots & & \vdots \\ w_{p1}Z & \cdots & w_{pq}Z \end{bmatrix}.$$

Vektorizacija matrike $W = \begin{bmatrix} w_1 & \cdots & w_q \end{bmatrix} \in \mathbb{R}^{p\times q}$ je vektor velikosti pq bločne oblike

$$\mathrm{vec}(W) = \begin{bmatrix} w_1 \\ \vdots \\ w_q \end{bmatrix}.$$

Hitro lahko preverimo, da veljajo naslednje enakosti

$$\begin{aligned} (A\otimes B)^H &= A^H \otimes B^H, \\ (A\otimes B)(C\otimes D) &= (AC)\otimes(BD), \\ \mathrm{vec}(AX) &= (I_n \otimes A)\mathrm{vec}(X), \\ \mathrm{vec}(XB) &= (B^T \otimes I_m)\mathrm{vec}(X), \\ \mathrm{vec}(AXB) &= (B^T \otimes A)\mathrm{vec}(X), \\ \|\mathrm{vec}(X)\|_2 &= \|X\|_F. \end{aligned}$$

Lema 2.23 *Za matriki* $A \in \mathbb{R}^{m\times m}$ *in* $B \in \mathbb{R}^{n\times n}$ *velja*

a) lastne vrednosti $A \otimes B$ *so* $\lambda_i\mu_j$,

b) lastne vrednosti $I_n \otimes A + B \otimes I_m$ *so* $\lambda_i + \mu_j$,

kjer so λ_i *za* $i = 1,\ldots,m$ *lastne vrednosti matrike* A *in* μ_j *za* $j = 1,\ldots,n$ *lastne vrednosti matrike* B.

Dokaz. Za matriki A in B obstajata unitarni matriki U, V in zgornji trikotni matriki R, S, da je $R = U^HAU$ in $S = V^HBV$. Potem velja

$$A\otimes B = (U\otimes V)(R\otimes S)(U\otimes V)^H,$$

$R \otimes S$ pa je zgornja trikotna matrika, ki ima na diagonali vse možne produkte $r_{ii}s_{jj}$. To pa so ravno vsi možni produkti parov lastnih vrednosti matrik A in B. Podobno je

$$I_n\otimes A + B\otimes I_m = (U\otimes V)(I_n\otimes R + S\otimes I_m)(U\otimes V)^H,$$

kjer je $I_n \otimes R + S \otimes I_m$ zgornja trikotna matrika, katere diagonalni elementi so vse možne vsote $r_{ii} + s_{jj}$. ∎

Lema 2.24 *Če je* A *matrika oblike (2.5) modelnega problema velikosti* $n^2 \times n^2$, *potem je*

$$\rho(R_J) = \cos\left(\frac{\pi}{n+1}\right).$$

Dokaz. Matriko A lahko zapišemo kot $A = M \otimes I + I \otimes M$, kjer je M tridiagonalna matrika

$$M = \begin{bmatrix} 2 & -1 & & \\ -1 & 2 & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{bmatrix}.$$

Po lemi 2.22 in točki b) leme 2.23 so potem lastne vrednosti matrike A enake

$$\lambda_{jk} = 4 + 2\cos\left(\frac{j\pi}{n+1}\right) + 2\cos\left(\frac{k\pi}{n+1}\right)$$

za $j, k = 1, \ldots, n$.

Ker ima matrika A diagonalo $D = 4I$, ima v tem konkretnem primeru Jacobijeva iteracijska matrika R_J obliko

$$R_J = -D^{-1}(A - D) = -\frac{1}{4}(A - D) = I - \frac{1}{4}A,$$

zato so lastne vrednosti R_j enake

$$-\left(\frac{1}{2}\cos\left(\frac{j\pi}{n+1}\right) + \frac{1}{2}\cos\left(\frac{k\pi}{n+1}\right)\right)$$

za $j, k = 1, \ldots, n$. Največjo absolutno vrednost lastne vrednosti dobimo pri $j = k = 1$, zato je $\rho(R_j) = \cos\left(\frac{\pi}{n+1}\right)$. ■

Ker nas zanima, kaj se dogaja, ko n narašča proti neskončno, lahko uporabimo razvoj v vrsto in aproksimiramo

$$\rho(R_j) = \cos\left(\frac{\pi}{n+1}\right) \approx 1 - \frac{\pi^2}{2(n+1)^2}.$$

Vidimo, da je $\rho(R_J) = 1 - \mathcal{O}(n^{-2})$, zato hitro narašča proti 1. V primeru večje mreže lahko tako pri Jacobijevi metodi poleg več dela za en korak pričakujemo tudi večje število korakov. Napaka se bo zanesljivo zmanjšala za faktor e v m korakih, če velja $\rho(R_j)^m \leq e^{-1}$. Če označimo $\delta = \pi^2/(2(n+1)^2)$, nas tako zanima, kdaj je $(1-\delta)^m = e^{-1}$. Z logaritmiranjem dobimo $m\ln(1-\delta) = -1$, od koder dobimo (uporabimo razvoj logaritma v vrsto)

$$m = \frac{-1}{\ln(1-\delta)} = \frac{-1}{-\delta - \delta^2/2 - \delta^3/3 - \cdots} = \frac{1}{\delta} - \frac{1}{2} - \frac{\delta}{12} - \frac{\delta^2}{24} - \cdots,$$

torej $m = 2(n+1)^2/\pi^2$. Vidimo, da Jacobijeva metoda potrebuje $\mathcal{O}(n^2)$ korakov, da napaka pade za izbrani faktor (namesto e bi lahko izbrali poljubno konstanto, izbrali smo jo le zaradi preprostejšega logaritmiranja).

En korak Jacobijeve metode ima zahtevnost $\mathcal{O}(n^2)$, saj za vsako točko v mreži opravimo konstantno mnogo osnovnih operacij. Če označimo $N = n^2$ za število neznank, potem Jacobijeva metoda za modelni sistem potrebuje $\mathcal{O}(N^2)$ operacij, da napaka pade za izbrani faktor. To je bolje kot $\mathcal{O}(N^3)$ operacij, kar potrebujemo za splošni sistem linearnih enačb velikosti $N \times N$, a bomo v nadaljevanju videli, da lahko modelni sistem rešimo še občutno hitreje.

Za Gauss–Seidelovo metodo velja enako kot za Jacobijevo metodo. Je sicer dvakrat hitrejša, a še vedno potrebuje $\mathcal{O}(N^2)$ operacij, da napaka pade za izbrani faktor. Vemo, da za Gauss-Seidelovo metodo velja

$$\rho(R_{GS}) = \rho(R_J)^2 = \cos^2\left(\frac{\pi}{n+1}\right).$$

Če označimo $\mu = \rho(R_J)$, je pri metodi SOR optimalni ω enak

$$w_{\text{opt}} = \frac{2}{1 + \sqrt{1-\mu^2}} = \frac{2}{1 + \sin(\frac{\pi}{n+1})}.$$

Za spektralni radij pri w_{opt} dobimo

$$\rho(R_{SOR}(\omega_{\text{opt}})) = w_{\text{opt}} - 1 = \frac{1 - \sin(\frac{\pi}{n+1})}{1 + \sin(\frac{\pi}{n+1})} \approx 1 - \frac{2\pi}{n+1}.$$

Torej velja $\rho(R_{SOR}(\omega_{\text{opt}})) = 1 - \mathcal{O}(n^{-1})$, kar je za red bolje kot pri Jacobijevi in Gauss–Seidelovi metodi. Podobno kot prej lahko sklepamo, da metoda SOR potrebuje $\mathcal{O}(n)$ korakov, da napaka pade za izbrani faktor. Ker je zahtevnost enega koraka tako kot pri Jacobijevi metodi $\mathcal{O}(n^2)$, tako skupno potrebujemo $\mathcal{O}(N^{3/2})$ operacij, da napaka pade za izbrani faktor.

Za metodo SSOR s pospešitvijo Čebiševa so pokazali, da pri Youngovi izbiri

$$\omega = \frac{2}{1 + \sqrt{2 - 2\mu}}$$

dobimo

$$\rho(R_{SSOR}) \approx 1 - \frac{\pi}{2n^2}.$$

S pospeševanjem Čebiševa se tako napaka na koraku m množi s faktorjem

$$\frac{1}{\mu_m} = \frac{1}{T_m(1 + \pi/(2n^2))} \leq \frac{1}{m^2\pi/(2n^2)}.$$

Ocenimo lahko, da potrebujemo $\mathcal{O}(n^{1/2})$ iteracij, da napaka pade za faktor e. Ker je zahtevnost enega koraka še vedno $\mathcal{O}(n^2)$, tako skupno potrebujemo $\mathcal{O}(N^{5/4})$ operacij, da napaka pade za izbrani faktor.

Modelni primer z matriko oblike (2.5) lahko rešimo še hitreje. V [3] lahko najdete opis metode, ki s pomočjo hitre Fourierove transformacije (FFT) sistem reši z zahtevnostjo $\mathcal{O}(N \log N)$, in opis večmrežne metode, ki sistem reši z zahtevnostjo $\mathcal{O}(N)$.

Poglavje 3

Metode podprostorov Krilova

Gre za razred metod za reševanje sistema $Ax = b$ ali za računanje lastnih vrednosti A, kjer namesto direktnega dostopa do matrike potrebujemo le podprogram, ki zna za poljuben vektor x izračunati Ax (včasih pa tudi $A^T x$). Tako lahko npr. v primeru, ko je matrika A razpršena, produkta z A in A^T običajno izračunamo ekonomično.

3.1 Podprostor Krilova

Definicija 3.1 *Za dano matriko A in vektor b je* podprostor Krilova

$$\mathcal{K}_k(A, b) := \mathrm{Lin}(b, Ab, \ldots, A^{k-1}b).$$

Hitro lahko ugotovimo, da podprostor Krilova $\mathcal{K}_k(A, b)$ sestavljajo vsi vektorji oblike $p_{k-1}(A)b$, kjer gre p_{k-1} po vseh polinomih stopnje manjše ali enake $k - 1$, torej

$$\mathcal{K}_k(A, b) = \{p_{k-1}(A)b : p_{k-1} \in \mathbb{P}_{k-1}\}.$$

Naj bo x_0 začetni približek. Potem kot približek za rešitev sistema $Ax = b$ iščemo rešitev oblike $x_k \in x_0 + \mathcal{K}_k(A, r_0)$, kjer je $r_0 = b - Ax_0$ začetni ostanek. Poznamo štiri pristope, kako lahko v podprostoru Krilova pridemo do ustreznega približka.

1. *Ritz-Galerkin:* Izberemo x_k, pri katerem je ostanek pravokoten na podprostor Krilova, oziroma $b - Ax_k \perp \mathcal{K}_k(A, r_0)$. Za splošno matriko pridemo tako do metode FOM, za simetrično matriko do Lanczoseve metode, če pa je matrika simetrična pozitivno definitna, je to metoda konjugiranih gradientov oz. CG.

2. *Minimalni ostanek:* Izberemo x_k, ki minimizira normo ostanka $\|b - Ax_k\|_2$. V primeru splošne matrike je to metoda GMRES, če je matrika simetrična pa metoda MINRES.

3. *Petrov-Galerkin:* Izberemo x_k, pri katerem je $b - Ax_k \perp \mathcal{W}_k$ za nek testni podprostor $\mathcal{W}_k$, za katerega ponavadi izberemo podprostor Krilova za matriko A^T. Tako pridemo do bikonjugiranih gradientov oz. Bi-CG.

4. *Minimalna napaka:* Izberemo $x_k \in A^T\mathcal{K}_k(A^T, r_0)$, ki minimizira $\|x - x_k\|_2$. Za simetrično matriko je to metoda SYMMLQ.

3.2 Arnoldijev algoritem

Vektorji $r_0, Ar_0, \ldots, A^{k-1}r_0$ so nestabilna baza za $\mathcal{K}_k(A, r_0)$, saj iz potenčne metode vemo, da se z naraščajočim k vektorji vedno bolj približujejo lastnemu vektorju za dominantno lastno vrednost. Namesto tega za stabilnost potrebujemo ortonormirano bazo za $\mathcal{K}_k(A, r_0)$.

Denimo, da stolpci $V_j = [v_1 \ \cdots \ v_j]$ sestavljajo ortonormirano bazo za $\mathcal{K}_j(A, r_0)$ za $j = 1, \ldots, k$. Potem je $\mathcal{K}_{k+1}(A, r_0) = \text{Lin}(v_1, \ldots, v_k, Av_k)$ in naslednji bazni vektor dobimo tako, da Av_k ortogonaliziramo glede na vektorje $v_1, \ldots, v_k$.

Arnoldijev algoritem (1951) za ortogonalizacijo uporablja modificirano Gram-Schmidtovo metodo oz. MGS. Predstavljen je v algoritmu 3.1.

Algoritem 3.1 Arnoldijev algoritem. Vhodni podatki so matrika A, vektor r_0 in dimenzija k. Algoritem vrne ortonormirane stolpce $v_1, \ldots, v_k$, ki tvorijo ortonormirano bazo za podprostor Krilova $\mathcal{K}_k(A, r_0)$.

$v_1 = r_0/\|r_0\|$
$j = 1, \ldots, k$
    $z = Av_j$
    $i = 1, \ldots, j$
        $h_{ij} = v_i^T z$
        $z = z - h_{ij} v_i$
    $h_{j+1,j} = \|z\|$
    če je $h_{j+1,j} = 0$, potem prekini računanje
    $v_{j+1} = z/h_{j+1,j}$

Algoritem se konča pri izbranem k ali pa, ko je $h_{j+1,j} = 0$.

Trditev 3.2 *Arnoldijev algoritem se lahko izvaja do $j = k$, kjer je $k = \dim(\mathcal{K}_n(A, r_0))$. Za $j = 1, \ldots, k$ velja*

$$AV_j = V_j H_j + h_{j+1,j}[0 \ \cdots \ 0 \ v_{j+1}] = V_j H_j + h_{j+1,j} v_{j+1} e_j^T,$$

kjer je H_j zgornja Hessenbergova matrika velikosti $j \times j$, stolpci $V_j = [v_1 \ \cdots \ v_j]$ pa so ortonormirana baza za podprostor Krilova $K_j(A, r_0)$.

Dokaz. Uporabimo indukcijo po j. Denimo, da je $\text{Lin}(v_1, \ldots, v_j) = K_j(A, b)$ in $V_j^T V_j = I_j$.

V primeru $h_{j+1,j} \neq 0$ je $v_{j+1} \perp v_1, \ldots, v_j$ in $v_{j+1} \in \text{Lin}(v_1, \ldots, v_j, Av_j) \subset K_{j+1}(A, r_0)$. Ker pa je $\dim(\text{Lin}(v_1, \ldots, v_j, v_{j+1})) = j + 1$, imamo enakost.

Če je $h_{j+1,j} = 0$, potem je $AV_j = V_j H_j$ in V_j je baza za invariantni podprostor. V tem primeru iz $\text{Lin}(v_1, \ldots, v_j) = K_j(A, r_0)$ sledi $K_n(A, r_0) = K_j(A, r_0)$. ∎

Zahtevnost Arnoldijevega algoritma je k množenj z matriko in $\mathcal{O}(k^2 n)$ za ortogonalizacijo. Množenje z matriko je za polno matriko zahtevnosti $\mathcal{O}(n^2)$, za razpršeno matriko pa je lahko tudi samo $\mathcal{O}(n)$.

Arnoldijev algoritem vrne delno ortogonalno podobnostno transformacijo matrike A v zgornjo

Hessenbergovo matriko. Po k korakih Arnoldijevega algoritma iz

$$H = V^T AV = \begin{bmatrix} V_k^T AV_k & V_k^T AV_u \\ V_u^T AV_k & V_u^T AV_u \end{bmatrix} = \begin{bmatrix} H_k & H_{ku} \\ H_{uk} & H_u \end{bmatrix},$$

poznamo H_k in H_{uk}, ki ima same ničle in v zgornjem desnem kotu element $h_{k+1,k}$.

Pišemo lahko $AV_j = V_{j+1}\widetilde{H}_j$, kjer $\widetilde{H}_j$ dobimo tako, da H_j dodamo še vrstico $[0 \ \cdots \ 0 \ h_{j+1,j}]$.

Če se Arnoldijev algoritem izvede do konca (torej do $k = n$), potem je $V^T AV = H$ ortogonalna redukcije matrike A na zgornjo Hessenbergovo matriko. To lahko uporabimo za alternativno izpeljavo Arnoldijevega algoritma.

Če je prvi stolpec matrike V določen, potem lahko naslednje stolpce izračunamo rekurzivno. Iz zveze $AV = VH$ dobimo za j-ti stolpec enakost

$$Av_j = \sum_{i=1}^{j+1} h_{ij} v_i.$$

S skalarnim množenjem s v_i, kjer je $i = 1, \ldots, j$, dobimo $h_{ij} = v_i^T Av_j$. Vektor v_{j+1} dobimo iz

$$h_{j+1,j} v_{j+1} = Av_j - \sum_{i=1}^{j} h_{ij} v_i.$$

S tem smo pokazali, da so vsi stolpci matrike V do predznaka točno določeni s prvim stolpcem oz. dokazali naslednji izrek.

Izrek 3.3 (Implicitni Q) *Če je $Q = [q_1 \ \cdots \ q_n]$ taka ortogonalna matrika, da je $Q^T AQ = H$ nerazcepna Hessenbergova matrika, potem so stolpci $q_2, \ldots, q_n$ do predznaka natančno določeni s q_1.*

Čeprav bi vektorji $v_1, \ldots, v_k$, ki jih vrne Arnoldijev algoritem, morali biti ortogonalni, se v praksi zaradi zaokrožitvenih napak lahko zgodi, da ortogonalnost preveč pade, ko k narašča. Če želimo ohraniti numerično stabilnost in ortogonalnost baze, se izkaže, da moramo po potrebi Gram-Schmidtov postopek ponoviti in uporabiti RGS (repeated Gram-Schmidt). Ortogonalizacijo ponovimo na vektorjih, ki se jim s tem, ko odštejemo komponente v smeri že ortonormiranih vektorjev iz baze, norma preveč zmanjša. Majhna norma vektorja z je namreč lahko znak potencialnih velikih relativnih napak. Postopek je predstavljen v algoritmu 3.2.

Časovna zahtevnost algoritma 3.2 je k množenj vektorja z matriko in v najslabšem primeru $4k^2 n$ operacij za ortogonalizacijo. Prostorska zahtevnost je $(k+1)n$.

3.2.1 Arnoldijev algoritem s Householderjevimi zrcaljenji

Walker je leta 1980 predstavil Arnoldijev algoritem, ki namesto MGS uporablja Householderjeva zrcaljenja. Za lažjo izpeljavo poglejmo, kakšna je povezava med Arnoldijevim algoritmom in računanjem QR razcepa z uporabo MGS.

Če bi vzeli matriko

$$M = [\, r_0 \quad Av_1 \quad Av_2 \quad \cdots \quad Av_k \,],$$

Algoritem 3.2 Arnoldijev algoritem s ponovljeno ortogonalizacijo. Vhodni podatki so matrika A, vektor r_0 in dimenzija k. Algoritem vrne ortonormirane stolpce $v_1, \ldots, v_k$, ki tvorijo ortonormirano bazo za podprostor Krilova $\mathcal{K}_k(A, r_0)$.

$v_1 = r_0/\|r_0\|$
$j = 1, 2, \ldots, k$
  $z = Av_j,\ \xi = \|z\|$
  $i = 1, \ldots, j$
    $h_{ij} = v_i^T z$
    $z = z - h_{ij} v_i$
  če je $\|z\| < 0.7\xi$, potem
    $i = 1, \ldots, j$
      $\widetilde{h}_{ij} = v_i^T z$
      $h_{ij} = h_{ij} + \widetilde{h}_{ij}$
      $z = z - \widetilde{h}_{ij} v_i$
  $h_{j+1,j} = \|z\|$
  če je $h_{j+1,j} = 0$, potem prekini računanje
  $v_{j+1} = z/h_{j+1,j}$

kjer so vektorji Av_i za $i = 1, \ldots, k$ pobrani iz Arnoldijevega algoritma 3.1 (predpostavimo, da računamo točno), potem lahko hitro vidimo, da ima QR razcep matrike M, ki ga izračunamo z metodo MGS, obliko

$$M = V_{k+1} R_{k+1},$$

kjer je R_{k+1} zgornja trikotna matrika velikosti $(k+1) \times (k+1)$. Iz zveze $AV_k = V_{k+1}\widetilde{H}_k$ sledi, da se zadnjih k stolpcev R_{k+1} ujema z matriko $\widetilde{H}_k$, medtem ko za prvi stolpec R_{k+1} velja, da ima obliko $\|r_0\|e_1$.

Na podlagi zgornje ugotovitve lahko za računanje ortonormirane baze za $\mathcal{K}_k(A, r_0)$ uporabimo tudi metode za računanje QR razcepa, ki temeljijo na Givensovih rotacijah ali Householderjevih zrcaljenjih. Pri tem moramo paziti na to, da na začetku poznamo le prvi stolpec matrike M za katero računamo QR razcep. Ko izračunamo j-ti stolpec v_j, dobimo $(j+1)$-vi stolpec M, ki ima obliko Av_j.

Na kratko ponovimo osnovne lastnosti Householderjevih zrcaljenj. Naj bo

$$P = I - 2ww^T,$$

kjer je w normiran vektor. Potem P imenujemo Householderjevo zrcaljenje in predstavlja zrcaljenje preko hiperravnine, ki je pravokotna na w. Poljuben vektor z prezrcalimo v

$$Pz = z - 2(w^T z)w,$$

iz česar vidimo, da matrike P ne potrebujemo eksplicitno zapisane, potrebujemo le vektor w, ki določa P.

Izbran neničelni vektor x lahko prezrcalimo v $\pm\|x\|e_1$. Ustrezna izbira je $w = \widetilde{w}/\|\widetilde{w}\|$, kjer je

$$\widetilde{w} = x + \operatorname{sign}(x_1)\|x\|_2 e_1.$$

Pri tem mora za funkcijo sign veljati, da je $\operatorname{sign}(t) = 1$ za $t \geq 0$ in $\operatorname{sign}(t) = -1$ za $t < 0$.

Poglejmo, kako bi uporabili Householderjeva zrcaljenja v primeru $k = 2$, če ima matrika A velikost 5×5. Začnemo z r_0 in poiščemo w_1, da $\widetilde{P}_1 = P_1 = I_5 - 2w_1w_1^T$ preslika r_0 v $\pm\|r_0\|e_1$, torej

$$\widetilde{P}_1 r_0 = \begin{bmatrix} \times \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

Vektor v_1 je potem enak prvemu stolpcu $\widetilde{P}_1$ oziroma $v_1 = \widetilde{P}_1 e_1$. V nadaljevanju izračunamo produkt $z_2 = Av_1$ in poiščemo $P_2 = I_4 - w_2w_2^T$, da za

$$\widetilde{P}_2 = \begin{bmatrix} 1 & \\ & P_2 \end{bmatrix}$$

velja

$$\widetilde{P}_2\widetilde{P}_1 \begin{bmatrix} r_0 & z_2 \end{bmatrix} = \begin{bmatrix} \times & \times \\ 0 & \times \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}.$$

Nadaljujemo z $v_3 = \widetilde{P}_1\widetilde{P}_2 e_2$ in za $z_3 = Av_3$ poiščemo $P_3 = I_3 - w_3w_3^T$, da za

$$\widetilde{P}_3 = \begin{bmatrix} I_2 & \\ & P_3 \end{bmatrix}$$

velja

$$\widetilde{P}_3\widetilde{P}_2\widetilde{P}_1 \begin{bmatrix} r_0 & z_2 & z_3 \end{bmatrix} = \begin{bmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & 0 & \times \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} = \widetilde{R}_3.$$

Na koncu dobimo $\widetilde{H}_2$ tako, da iz $\widetilde{R}_3$ odrežemo spodnji dve vrstici in prvi stolpec.

Algoritem za poljuben k je zapisan v algoritmu 3.3. Časovna zahtevnost algoritma je približno dvakrat tolikšna kot pri algoritmu 3.1 in primerljiva z algorotmom 3.2 s ponovno ortogonalizacijo, saj to ponavadi izvedemo v večini korakov. Natančnost je primerljiva z algoritmom 3.2 s ponovno ortogonalizacijo.

3.3 Metoda FOM

Poglejmo, kako lahko s pomočjo Arnodijevega algoritma pridemo do približka za rešitev sistema $Ax = b$. Če je x_0 začetni približek, potem pri Ritz–Galerkinovemu pogoju iščemo $x_k \in x_0 + \mathcal{K}_k(A, r_0)$, da je $r_k \perp \mathcal{K}_k(A, r_0)$. To je ekvivalentno pogoju

$$V_k^T(b - Ax_k) = 0.$$

Vektor x_k lahko zapišemo v obliki $x_k = x_0 + V_k y_k$, kjer je $y_k \in \mathbb{R}^k$. Ker je $r_0 = \|r_0\|v_1$, sledi $V_k^T(b - Ax_0) = \|r_0\|e_1$ in rešiti moramo sistem

$$V_k^T AV_k y_k = \|r_0\|e_1.$$

Algoritem 3.3 Arnoldijev algoritem s Householderjevimi zrcaljenji. Vhodni podatki so matrika A, vektor r_0 in dimenzija k. Algoritem vrne ortonormirane stolpce $v_1, \ldots, v_k$, ki tvorijo ortonormirano bazo za podprostor Krilova $\mathcal{K}_k(A, r_0)$.

$z_1 = r_0$
$j = 1, \ldots, k+1$
določi tak normiran $w_j \in \mathbb{R}^{n-j+1}$, da za $\widetilde{P}_j = \begin{bmatrix} I_{j-1} & \\ & P_j \end{bmatrix}$, kjer je $P_j = I - 2w_j w_j^T$, velja, da ima $\widetilde{P}_j z_j$ zadnjih $n-j$ elementov enakih 0.
$h_{j-1} = \widetilde{P}_j z_j$
$v_j = \widetilde{P}_1 \widetilde{P}_2 \cdots \widetilde{P}_j e_j$
$z_{j+1} = \widetilde{P}_j \cdots \widetilde{P}_1 A v_j \quad (\text{za } j \leq k)$
Iz matrike H vzemi le prvih $k+1$ vrstic.

Matriko $V_k^T A V_k = H_k$ smo že izračunali med konstrukcijo ortonormirane baze podprostora Krilova, kar pomeni, da moramo rešiti sistem z zgornjo Hessenbergovo matriko

$$H_k y_k = \|r_0\| e_1,$$

potem pa vzamemo $x_k = x_0 + V_k y_k$. To je metoda *FOM* (full orthogonal method).

V grobem lahko metodo FOM zapišemo v algoritmu 3.4.

Algoritem 3.4 Metoda FOM za reševanje sistema $Ax = b$. Vhodni podatki so matrika A, desna stran b, začetni približek x_0 in dimenzija k. Algoritem vrne približek x_k po metodi FOM.

$r_0 = b - Ax_0$
Arnoldi$(A, r_0, k) \Longrightarrow AV_k = V_k H_k + h_{k+1,k} v_{k+1} e_k^T$
$y_k = H_k^{-1} \|r_0\| e_1$
$x_k = x_0 + V_k y_k$

V praksi dimenzije k ne določimo vnaprej, temveč sproti povečujemo podprostor Krilova in naredimo toliko korakov, kolikor je potrebnih, da pridemo do dovolj natančnega približka. Izkaže se, da pri tem ni potrebno za vsak k računati približka x_k in norme napake $r_k = b - Ax_k$, saj velja naslednji izrek.

Izrek 3.4 *Če se je Arnoldijev algoritem izvedel do koraka k (kar pomeni $h_{j+1,j} \neq 0$ za $j = 1, \ldots, k-1$), potem za ostanek r_k, ki ga dobimo pri metodi FOM, velja*

$$r_k = b - Ax_k = -h_{k+1,k} e_k^T y_k v_{k+1}$$

in zato

$$\|r_k\| = h_{k+1,k} |e_k^T y_k|.$$

Dokaz. Vemo, da je $x_k = x_0 + V_k y_k = x_0 + V_k H_k^{-1} \|r_0\| e_1$. Če poleg tega upoštevamo še zvezo $AV_k = V_k H_k + h_{k+1,k} v_{k+1} e_k^T$, dobimo

$$\begin{aligned} b - Ax_k &= b - Ax_0 - AV_k H_K^{-1} \|r_0\| e_1 \\ &= r_0 - V_k \|r_0\| e_1 - h_{k+1,k} v_{k+1} e_k^T y_k \\ &= r_0 - r_0 - h_{k+1,k} e_k^T y_k v_{k+1}. \end{aligned}$$

■

Če je matrika A simetrična, se postopek poenostavi, saj je matrika H_k tridiagonalna in dobimo *Lanczosevo metodo*. Še bolj se poenostavi v primeru, ko je A simetrična in pozitivno definitna, kjer dobimo *metodo konjugiranih gradientov*.

Če se Arnoldijev algoritem konča predčasno, je $h_{j+1,j} = 0$. Izkaže se, da je to srečni dogodek, saj je potem $r_j = 0$ in x_j je točna rešitev.

Pri velikem k postane metoda neučinkovita, saj potrebuje veliko spomina za V_k, podraži pa se tudi ortogonalizacija novih vektorjev. Ena od možnih rešitev je, da naredimo samo m korakov metode FOM, potem pa končni približek vzamemo za nov začetni približek. Če naredimo naenkrat m korakov, to imenujemo FOM(m). Konvergenca je sicer počasnejša kot pri metodi FOM, a skupni čas računanja je lahko manjši.

Algoritem 3.5 Metoda FOM(m) za reševanje sistema $Ax = b$. Vhodni podatki so matrika A, desna stran b, začetni približek x_0 in dimenzija m.

$r_0 = b - Ax_0$
Arnoldi$(A, r_0, m) \Longrightarrow AV_m = V_m H_m + h_{m+1,m} v_{m+1} e_m^T$
$y_m = H_m^{-1} \|r_0\| e_1$
$x_m = x_0 + V_m y_m$
če ni konvergence, vzemi $x_0 = x_m$ in se vrni na začetek

3.4 Lanczosev algoritem

Če je matrika A simetrična, je potem tudi matrika H simetrična, torej tridiagonalna. V tem primeru se Arnoldijev algoritem občutno poenostavi. Pišemo $H = T$ in označimo

$$T = \begin{bmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \alpha_2 & \beta_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{n-2} & \alpha_{n-1} & \beta_{n-1} \\ & & & \beta_{n-1} & \alpha_n \end{bmatrix}.$$

Sedaj iz $AV = VT$ dobimo

$$Av_j = \beta_{j-1} v_{j-1} + \alpha_j v_j + \beta_j v_{j-1},$$

kjer je $\alpha_j = v_j^T A v_j$.

Vse skupaj lahko zapišemo v obliki Lanczosevega algoritma, ki je predstavljen v algoritmu 3.6. Če se algoritem ne konča pred korakom k, potem velja

$$AV_k = V_k T_k + \beta_k v_{k+1} e_k^T.$$

Sicer pa, tako kot pri Arnoldijevemu algoritmu, tudi tu v primeru, ko je $\dim \mathcal{K}_n(A, b) = k$, dobimo $\beta_k = 0$.

Po k korakih Lanczosevega algoritma iz

$$T = V^T A V = \begin{bmatrix} T_k & T_{uk}^T \\ T_{uk} & T_u \end{bmatrix}$$

Algoritem 3.6 Lanczosev algoritem. Vhodni podatki so simetrična matrika A, vektor b in dimenzija k. Algoritem vrne ortonormirane stolpce $v_1, \ldots, v_k$, ki tvorijo ortonormirano bazo za podprostor Krilova $\mathcal{K}_k(A, b)$.

$v_1 = b/\|b\|_2$, $\beta_0 = 0$, $v_0 = 0$
$j = 1, 2, \ldots, k$
  $z = Av_j$
  $\alpha_j = v_j^T z$
  $z = z - \alpha_j v_j - \beta_{j-1} v_{j-1}$
  $\beta_j = \|z\|_2$
  Prekini, če je $\beta_j = 0$.
  $v_{j+1} = z/\beta_j$

poznamo matriki T_k in T_{uk}, pri čemer ima T_{uk} same ničle in zgoraj desno β_k.

V primeru simetrične matrike bi morala Arnoldijev in Lanczosev algoritem vrniti iste bazne vektorje in moralo bi veljati $H_k = T_k$. Ker pa pri Lanczosu novi vektor ortogonaliziramo le na zadnja dva vektorja, se v praksi izkaže, da imamo zaradi tega lahko težave z ortogonalnostjo.

Podobno kot pri metodi FOM bi lahko približek za simetrični sistem dobili z Lanczosevo metodo, zapisano v algoritmu 3.7.

Algoritem 3.7 Lanczoseva metoda za reševanje sistema $Ax = b$, kjer je $A = A^T$. Vhodni podatki so matrika A, desna stran b, začetni približek x_0 in dimenzija k.

$r_0 = b - Ax_0$
Lanczos$(A, r_0, k) \Longrightarrow AV_k = V_k T_k + \beta_k v_{k+1} e_k^T$
$y_k = T_k^{-1} \|r_0\| e_1$
$x_k = x_0 + V_k y_k$

Med samim računanjem vektorjev matrike V_k in elementov matrike T_k res potrebujemo le zadnja dva vektorja. Vendar, ko želimo v algoritmu 3.7 v zadnjem koraku izračunati vektor x_k, potrebujemo za izračun vse stolpce matrike V_k. Temu se lahko izognemo in algoritem preuredimo tako, da za izračun končnega x_k ne potrebujemo več vseh stolpcev matrike V_k.

Predpostavimo, da za tridiagonalno matriko T_k obstaja LU razcep brez pivotiranja. Ta ima potem obliko $T_k = L_k U_k$, kjer je

$$T_k = \begin{bmatrix} \alpha_1 & \beta_1 & & \\ \beta_1 & \alpha_2 & \ddots & \\ & \ddots & \ddots & \beta_{k-1} \\ & & \beta_{k-1} & \alpha_k \end{bmatrix} = \begin{bmatrix} 1 & & & \\ \lambda_2 & 1 & & \\ & \ddots & \ddots & \\ & & \lambda_k & 1 \end{bmatrix} \begin{bmatrix} \eta_1 & \beta_1 & & \\ & \eta_2 & \ddots & \\ & & \ddots & \beta_{k-1} \\ & & & \eta_k \end{bmatrix}$$

za ustrezne vrednosti $\lambda_2, \ldots, \lambda_k$ in $\eta_1, \ldots, \eta_k$.

Za rešitev x_k velja

$$x_k = x_0 + V_k T_k^{-1} \|r_0\| e_1 = x_0 + V_k U_k^{-1} L_k^{-1} \|r_0\| e_1.$$

Če definiramo $P_k = V_k U_k^{-1}$ in $z_k = L_k^{-1} \|r_0\| e_1$, dobimo $x_k = x_0 + P_k z_k$. Če zapišemo matriko P_k po stolpcih kot $P_k = [p_1 \ \cdots \ p_k]$ in vektor $z_k = [\zeta_1 \ \cdots \ \zeta_k]^T$ po elementih, potem pridemo do

zveze

$$x_k = x_{k-1} + \zeta_k p_k,$$

kjer je $x_{k-1} = x_0 + P_{k-1} z_{k-1}$.

Če znamo torej ekonomično izračunati zadnji vektor matrike P_k iz prejšnjih in zadnji element vektorja z_k iz prejšnjih, potem lahko po zgornji formuli enostavno iz zadnjega približka x_{k-1} dobimo nov približek.

Iz $P_k U_k = V_k$ sledi $\beta_{k-1} p_{k-1} + \eta_k p_k = v_k$ oziroma

$$p_k = \frac{1}{\eta_k} \left(v_k - \beta_{k-1} p_{k-1} \right).$$

Podobno iz $L_k z_k = \|r_0\| e_1$ sledi

$$\zeta_k = -\lambda_k \zeta_{k-1}.$$

Iz tega lahko sestavimo tako imenovano direktno Lanczosevo metodo, ki je zapisana v algoritmu 3.8. Za razliko od navadne Lanczoseve metode imamo lahko sedaj težave, kadar izračunamo $\eta_j = 0$. V tem primeru pravimo, da je prišlo do *kritične ustavitve*. To pomeni, da naša predpostavka, da za T_k obstaja LU razcep brez pivotiranja, ni bila dobra. Temu se lahko izognemo, če uporabljamo LU razcep z delnim pivotiranjem ali pa QR razcep. V tem primeru potem za izračun novega približka potrebujemo dva zadnja približka namesto le enega.

Algoritem 3.8 Direktna Lanczoseva metoda za reševanje sistema $Ax = b$, kjer je $A = A^T$. Vhodni podatki so matrika A, desna stran b, začetni približek x_0 in dimenzija k.

$v_1 = r_0 / \|r_0\|,\ \beta_0 = 0,\ v_0 = 0,\ \zeta_1 = \|r_0\|$
$j = 1, 2, \ldots, k$
  $z = A v_j$
  $\alpha_j = v_j^T z$
  $z = z - \alpha_j v_j - \beta_{j-1} v_{j-1}$
  $\beta_j = \|z\|$
  če je $j > 1$: $\lambda_j = \beta_{j-1} / \eta_{j-1}$
  $\eta_j = \alpha_j - \beta_{j-1} \lambda_j$, če je $\eta_j = 0$, končaj
  če je $j > 1$: $\zeta_j = -\lambda_j \zeta_{j-1}$
  $p_j = (1/\eta_j) \left(v_j - \beta_{j-1} p_{j-1} \right)$
  $x_j = x_{j-1} + \zeta_j p_j$.
  če je $\beta_j = 0$, prekini računanje
  $v_{j+1} = z / \beta_j$

En korak direktne Lanczoseve metode ima zahtevnost 1 MV (kjer MV označuje produkt vektorja z matriko) in $14n + \mathcal{O}(1)$ za operacije z vektorji.

Pri direktni Lanczosevi metodi nov približek x_k dobimo tako, da prejšnji približek x_{k-1} popravimo v smeri p_k. Za ostanke in smeri $p_1, \ldots, p_k$ velja naslednji izrek.

Izrek 3.5 *Za vektorje, dobljene pri direktni Lanczosevi metodi, velja:*

1) *Ostanki $r_0, r_1, \ldots, r_k$ so paroma pravokotni.*

2) Smeri $p_1, \ldots, p_k$ so A-konjugirane, kar pomeni $p_i^T A p_j = 0$ za $i \neq j$.

Dokaz.

1) Iz zveze
$$r_k = -\beta_k e_k^T y_k v_{k+1} \tag{3.1}$$
sledi da sta vektorja r_k in v_{k+1} kolinearna. Zato so vektorji $r_0, r_1, \ldots, r_k$ paroma pravokotni (saj to velja za vektorje $v_1, \ldots, v_{k+1}$).

2) Iz definicije $P_k = V_k U_k^{-1}$ sledi
$$P_k^T A P_k = U_k^{-T} V_k^T A V_k U_k^{-1} = U_k^{-1} T_k U_k^{-1} = U_k^{-T} L_k.$$
Matrika $U_k^{-T} L_k$ je spodnja trikotna, obenem pa simetrična, saj je $P_k^T A P_k$ simetrična. To pa je možno le, če je $P_k^T A P_k$ diagonalna matrika, kar pomeni $p_i^T A p_j = 0$ za $i \neq j$. ∎

Omenimo še, da točka a) zgornjega izreka velja tudi za metodo FOM in tako ortogonalnost ostankov ni odvisna od simetričnosti matrike A.

Če želimo pri direktni Lanczosevi metodi izračunati velikost ostanka iz enačbe (3.1), naletimo na težavo, saj vektorja y_k ne poznamo. Vemo pa, da je
$$y_k = T_k^{-1} \|r_0\|_2 e_1 = U_k^{-1} L_k^{-1} \|r_0\|_2 e_1 = U_k^{-1} z_k.$$
Velja torej $U_k y_k = z_k$, pri čemer za izračun velikosti ostanka potrebujemo le $e_k^T y_k$, torej zadnji element vektorja y_k. Ker je matrika U_k zgornja trikotna, je $e_k^T y_k = \zeta_k / \eta_k$. Od tod sledi
$$r_k = -\beta_k \frac{\zeta_k}{\eta_k} v_{k+1}$$
in
$$\|r_k\|_2 = \beta_k \frac{|\zeta_k|}{|\eta_k|}.$$

3.5 Konjugirani gradienti

Predpostavimo, da je matrika A simetrična in pozitivno definitna. V takem primeru vemo, da je matrika $T_k = V_k^T A V_k$ iz Lanczosevega algoritma tudi simetrična in pozitivno definitna, torej zanjo vedno obstaja LU razcep brez pivotiranja.

To dejstvo lahko izkoristimo, da v primeru simetrične pozitivno definitne matrike pridemo do še enostavnejšega algoritma.

Iz direktnega Lanczosa lahko razberemo:

1) Vektorja r_{j-1} in v_j sta kolinearna.

2) Vektor p_j dobimo tako, da r_{j-1} prištejemo vektor v smeri p_{j-1}.

3) Vektor x_j dobimo tako, da x_{j-1} prištejemo vektor v smeri p_j.

Ker dolžina nove smeri p_j ni pomembna, velja

$$\begin{aligned} p_j &= r_{j-1} + \beta_{j-1} p_{j-1} \\ x_j &= x_{j-1} + \alpha_j p_j \\ r_j &= r_{j-1} - \alpha_j A p_j \end{aligned}$$

za primerne skalarje α_j, β_j (niso enaki kot pri Lanczosu). Skalarje določimo tako, da velja

a) Vektorji $r_0, r_1, \ldots, r_k$ so paroma pravokotni.

b) Smeri $p_1, \ldots, p_k$ so A-konjugirane, torej $p_i^T A p_j = 0$ za $i \neq j$.

Koeficient α_j dobimo iz enakosti $r_{j-1}^T r_j = r_{j-1}^T (r_{j-1} - \alpha_j A p_j) = 0$. Tako dobimo

$$\begin{aligned} r_{j-1}^T r_{j-1} &= \alpha_j r_{j-1}^T A p_j \\ &= \alpha_j (p_j - \beta_{j-1} p_{j-1})^T A p_j \\ &= \alpha_j p_j^T A p_j, \end{aligned}$$

kjer smo upoštevali, da je sta smeri p_{j-1} in p_j A-konjugirani in je zato $p_{j-1}^T A p_j = 0$. Od tod sledi

$$\alpha_j = \frac{r_{j-1}^T r_{j-1}}{p_j^T A p_j}.$$

Koeficient β_{j-1} dobimo iz A-konjugiranosti p_{j-1} in p_j. Iz

$$p_{j-1}^T A p_j = p_{j-1}^T (A r_{j-1} + \beta_{j-1} p_{j-1}^T A p_{j-1}) = 0$$

sledi

$$\begin{aligned} 0 &= (A p_{j-1})^T r_{j-1} + \beta_{j-1} (r_{j-2} + \beta_{j-2} p_{j-2})^T A p_{j-1} \\ &= \frac{1}{\alpha_{j-1}} (r_{j-2} - r_{j-1})^T r_{j-1} + \beta_{j-1} r_{j-2}^T \frac{1}{\alpha_{j-1}} (r_{j-2} - r_{j-1}) \\ &= -\frac{1}{\alpha_{j-1}} r_{j-1}^T r_{j-1} + \frac{\beta_{j-1}}{\alpha_{j-1}} r_{j-2}^T r_{j-2}, \end{aligned}$$

od koder dobimo

$$\beta_{j-1} = \frac{r_{j-1}^T r_{j-1}}{r_{j-2}^T r_{j-2}}.$$

Tako pridemo do metode konjugiranih gradientov (CG), ki je zapisana v algoritmu 3.9.

Zahtevnost enega koraka metode konjugiranih gradientov je 1 MV in $10n + \mathcal{O}(1)$ za operacije z vektorji. Ker direktno računamo konjugirane smeri in nove vektorje, smo v primerjavi z direktno Lanczosevo metodo prihranili $4n + \mathcal{O}(1)$ operacij na korak.

Reševanje linearnega sistema $Ax = b$, kjer je matrika A simetrična in pozitivno definitna, lahko prevedemo na iskanje minimuma funkcije n spremenljivk, definirane kot

$$f(x) := \frac{1}{2} x^T A x - x^T b. \tag{3.2}$$

Algoritem 3.9 Metoda konjugiranih gradientov za reševanje sistema $Ax = b$, kjer je A simetrična pozitivno definitna. Vhodni podatki so matrika A, desna stran b, začetni približek x_0 in dimenzija k.

$r_0 = b - Ax_0, \quad p_1 = r_0$
$j = 1, 2, \ldots, k$
$\quad \alpha_j = \dfrac{\|r_{j-1}\|^2}{p_j^T A p_j}$
$\quad x_j = x_{j-1} + \alpha_j p_j$
$\quad r_j = r_{j-1} - \alpha_j A p_j$
$\quad \beta_j = \dfrac{\|r_j\|^2}{\|r_{j-1}\|^2}$
$\quad p_{j+1} = r_j + \beta_j p_j$

Lema 3.6 *Če je A simetrična pozitivno definitna matrika, potem funkcija f, definirana s predpisom (3.2), doseže svoj minimum pri $x = A^{-1}b$.*

Dokaz. Velja

$$(Ax - b)^T A^{-1}(Ax - b) = x^T Ax - 2x^T b + b^T A^{-1} b = 2f(x) + b^T A^{-1} b.$$

Ker je A s.p.d, to velja tudi za A^{-1} in očitno je minimum dosežen pri $Ax = b$. ■

Označimo napako približka z $d_k = \widehat{x} - x_k$. Opazimo, da je

$$Ad_k = A\widehat{x} - Ax_k = b - Ax_k = r_k.$$

Velja

$$\begin{aligned} \|d_k\|_A^2 &= d_k^T A d_k = (\widehat{x} - x_k)^T A(\widehat{x} - x_k) \\ &= x_k^T A x_k - 2x_k^T A\widehat{x} + \widehat{x}^T A\widehat{x} \\ &= 2f(x_k) + \widehat{x}^T A\widehat{x}. \end{aligned}$$

Ker je $\widehat{x}^T A\widehat{x}$ konstanta, smo tako pokazali naslednjo lemo.

Lema 3.7 *Naj bo A simetrična pozitivno definitna matrika in $A\widehat{x} = b$. Potem je ekvivalentno*

1) *minimizirati $f(x_k) = \frac{1}{2}x_k^T A x_k - x_k^T b$,*
2) *minimizirati $\|b - Ax_k\|_{A^{-1}}$,*
3) *minimizirati $\|\widehat{x} - x_k\|_A$.*

Pokažimo, da po k korakih metode konjugiranih gradientov res dobimo optimalni približek x_k.

Izrek 3.8 *Naj bo A simetrična pozitivno definitna matrika. Če po k korakih metode konjugiranih gradientov za sistem $Ax = b$ velja $r_j \neq 0$ za $j < k$, potem je x_k tisti vektor iz $\mathcal{K}_k(A, r_0)$, ki minimizira $\|\widehat{x} - x_k\|_A$. Konvergenca je monotona, kar pomeni $\|d_k\|_A \leq \|d_{k-1}\|_A$, in velja $d_m = 0$ za nek $m \leq n$.*

Dokaz. Vzemimo poljuben $x \in \mathcal{K}_k(A, r_0)$, $x = x_k - \Delta x$ in $d = \widehat{x} - x = d_k + \Delta x$. Dobimo

$$\begin{aligned} \|d\|_A^2 &= (d_k + \Delta x)^T A(d_k + \Delta x) \\ &= d_k^T A d_k + \Delta x^T A \Delta x + 2\Delta x^T A d_k \\ &= d_k^T A d_k + \Delta x^T A \Delta x + 2\Delta x^T r_k \\ &= d_k^T A d_k + \Delta x^T A \Delta x, \end{aligned}$$

saj je $r_k \perp \mathcal{K}_k(A, r_0)$. Ker je A s.p.d., je izraz očitno minimalen pri $\Delta x = 0$, to pa pomeni pri $x = x_k$.

Konvergenca je očitno monotona zaradi $\mathcal{K}_{k-1}(A, r_0) \subset \mathcal{K}_k(A, r_0)$. Ker je ostanek r_k vedno pravokoten na $\mathcal{K}_k(A, r_0)$, se najkasneje po n korakih zgodi, da ostanek postane 0 in pridemo do točne rešitve. ∎

3.6 Konvergenca konjugiranih gradientov

Na začetku so metodo CG obravnavali kot direktno metodo, saj pri eksaktnem računanju vedno skonvergira v n korakih. Zaradi numeričnih napak pa se v praksi pogosto zgodi, da metoda CG tudi po n korakih ne skonvergira, čeprav bi po teoriji v eksaktni aritmetiki morala. Zaradi tega so na metodo za nekaj časa pozabili. Čez čas pa so ugotovili, da za mnogo matrik metoda vrne dovolj majhen ostanek v $k \ll n$ korakih in jo lahko uporabljamo kot iterativno metodo.

Za lažji študij konvergence si pomagamo s polinomi. Če je $d_0 = \widehat{x} - x_0$, potem iščemo tak polinom stopnje k, ki minimizira noramo $\|d_k\|_A = \|p_k(A)d_0\|_A$, pri čemer velja $p_k(0) = 1$.

Velja namreč $\widehat{x} - x_k = \widehat{x} - x_0 - q_{k-1}(A)r_0$ za nek polinom q_{k-1} stopnje $k-1$. Ker je $r_0 = A(\widehat{x} - x_0)$, dobimo

$$d_k = \widehat{x} - x_k = (I - Aq_{k-1}(A))d_0 = p_k(A)d_0,$$

kjer je p_k polinom stopnje k, za katerega velja $p_k(0) = 1$.

Izrek 3.9 *Če metoda konjugiranih gradientov ne skonvergira do k-tega koraka ($r_j \neq 0$ za $j < k$), potem velja $d_k = p_k(A)d_0$, kjer p_k minimizira $\|p_k(A)d_0\|_A$. Velja*

$$\frac{\|d_k\|_A}{\|d_0\|_A} = \inf_{\substack{p_k \text{ stopnje } k \\ p_k(0)=1}} \frac{\|p_k(A)d_0\|_A}{\|d_0\|_A} \leq \inf_{\substack{p_k \text{ stopnje } k \\ p_k(0)=1}} \max_{\lambda \in \sigma(A)} |p_k(\lambda)|.$$

Dokaz. Naj bodo $u_1, \ldots, u_n$ ortonormirani lastni vektorji matrike A, pripadajoče lastne vrednosti pa $\lambda_1, \ldots, \lambda_n$. Če je $d_0 = \sum_{i=1}^n a_i u_i$ razvoj začetne napake po lastnih vektorjih, je $p(A)d_0 = \sum_{i=1}^n a_i p(\lambda_i)u_i$. Od tod sledi

$$\|p(A)d_0\|_A^2 = \sum_{i=1}^n a_i^2 \lambda_i p(\lambda_i)^2.$$

Ker je

$$\|d_0\|_A^2 = \sum_{i=1}^n a_i^2 \lambda_i,$$

sledi

$$\frac{\|p(A)d_0\|_A^2}{\|d_0\|_A^2} \leq \max_{\lambda\in\sigma(A)} |p(\lambda)|^2.$$

■

Izkaže se, da je za konvergenco dovolj obravnavati le diagonalne matrike A. Namreč, če je Q ortogonalna matrika in $\widetilde{A} = Q^TAQ$, $\widetilde{x} = Q^Tx$, $\widetilde{b} = Q^Tb$ in vzamemo $\widetilde{x}_0 = Q^Tx_0$, potem CG na sistemu $Ax = b$ z začetnim približkom x_0 vrne iste konstante α_j in β_j kot CG na sistemu $\widetilde{A}\widetilde{x} = \widetilde{b}$ z začetnim približkom $\widetilde{x}_0$. Za ustrezna ostanka velja $\|\widetilde{r}_j\| = \|r_j\|$.

Hitrost konvergence metode konjugiranih gradientov je očitno odvisna od razporeditve lastnih vrednosti matrike A. Če ima matrika le k različnih lastnih vrednosti, potem metoda očitno skonvergira v k korakih, saj obstaja polinom p_k stopnje k, ki ima ničle v vseh lastnih vrednostih in je $p_k(0) = 1$.

Posledica 3.10 *Če ima simetrična pozitivno definitna matrika A le k različnih lastnih vrednosti, potem metoda konjugiranih gradientov skonvergira v največ k korakih.*

Dokaz. Polinom

$$p(x) = \left(1 - \frac{x}{\lambda_1}\right)\cdots\left(1 - \frac{x}{\lambda_k}\right)$$

je stopnje k, $p(0) = 1$ in $p(\lambda_i) = 0$ za $i = 1, \ldots, k$. ■

Ker za oceno konvergence potrebujemo polinom stopnje k, ki bo imel v točki 0 vrednost 1, sicer pa bo imel v lastnih vrednostih matrike A čim manjše absolutne vrednosti, si v ocenah lahko spet pomagamo s polinomi Čebiševa.

Izpeljemo lahko naslednji oceni.

Izrek 3.11 *Naj bodo $0 < \lambda_n \leq \cdots \leq \lambda_1$ lastne vrednosti s.p.d. matrike A. Za približek $x_\ell \in x_0 + \mathcal{K}_\ell(A, r_0)$, dobljen z metodo CG, velja ocena*

$$\|\widehat{x} - x_\ell\|_A \leq \frac{1}{\left|T_\ell\left(\frac{\lambda_1+\lambda_n}{\lambda_1-\lambda_n}\right)\right|}\|\widehat{x} - x_0\|_A.$$

Dokaz. Naj bodo $0 < \lambda_n \leq \cdots \leq \lambda_1$ lastne vrednosti matrike A. Za polinom

$$Q_\ell(\lambda) = \frac{T_\ell\left(\frac{2\lambda-(\lambda_1+\lambda_n)}{\lambda_1-\lambda_n}\right)}{T_\ell\left(-\frac{\lambda_1+\lambda_n}{\lambda_1-\lambda_n}\right)}$$

lahko hitro preverimo, da je $Q_\ell(0) = 1$ in

$$|Q_\ell(\lambda)| \leq \frac{1}{\left|T_\ell\left(\frac{\lambda_1+\lambda_n}{\lambda_1-\lambda_n}\right)\right|}$$

za $\lambda_n \leq \lambda \leq \lambda_1$. ■

Posledica 3.12 *Za približek $x_\ell \in x_0 + \mathcal{K}_\ell(A, r_0)$, dobljen z metodo CG, velja ocena*

$$\|\widehat{x} - x_\ell\|_A \leq 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^\ell \|\widehat{x} - x_0\|_A,$$

kjer je

$$\kappa = \frac{\lambda_1}{\lambda_n}$$

spektralna občutljivost matrike A.

Dokaz. Pomagamo si z naslednjo oceno za polinome Čebiševa:

$$\left|\frac{1}{T_\ell(\xi)}\right| \leq 2\left(\frac{1}{\xi + \sqrt{\xi^2-1}}\right)^\ell = 2\left(\xi - \sqrt{\xi^2-1}\right)^\ell.$$

Če vzamemo $\xi = \frac{\lambda_1+\lambda_n}{\lambda_1-\lambda_n}$, tako dobimo

$$|Q_\ell(\lambda)| \leq 2\left(\frac{\lambda_1+\lambda_2-2\sqrt{\lambda_1\lambda_n}}{\lambda_1-\lambda_n}\right)^\ell = 2\left(\frac{\sqrt{\lambda_1}-\sqrt{\lambda_n}}{\sqrt{\lambda_1}+\sqrt{\lambda_n}}\right)^\ell = 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^\ell. \qquad \blacksquare$$

Iz zgornjega dokaza sledi, da je konvergenca metode konjugiranih gradientov vsaj linearna. V praksi pa je konvergenca (pri predpostavki, da računamo v eksaktni aritmetiki) superlinearna, saj se hitrost konvergence z iteracijami povečuje. To lahko razložimo s povezavo med konvergenco in s tem, kako se lastne vrednosti matrike A ujemajo s približki za lastne vrednosti, ki pripadajo ustreznemu podprostoru Krilova.

Denimo, da stolpci matrike V_k tvorijo ortonomirano bazo za podprostor Krilova $\mathcal{K}_k(A, r_0)$. Približke za lastne vrednosti in vektorje iz podprostora $\mathcal{K}_k$ dobimo iz *Galerkinovega pogoja*

$$Av - \mu v \perp \mathcal{K}_k, \quad v \in \mathcal{K}_k.$$

V tem primeru pravimo, da je μ *Ritzeva vrednost*, v pa pripadajoči *Ritzev vektor*. Ko se povečuje razsežnost podprostora Krilova, Ritzeve vrednosti in vektorji konvergirajo proti lastnim parom matrike A, več o Ritzevih vrednostih in konvergenci pa lahko najdete v razdelku 4.2.

Ker stolpci matrike V_k tvorijo ortonormirano bazo za $\mathcal{K}_k$, to pomeni, da je μ lastna vrednost $k \times k$ matrike $T_k = V_k^T A V_k$, vektor v pa ima obliko $v = V_k w$, kjer je w, $\|w\|_2 = 1$, lastni vektor matrike T_k, ki pripada μ.

Matrika T_k pri metodi konjugiranih gradientov se ujema z matriko T_k, ki jo vrne Lanczoseva metoda. Zaradi tega je nerazcepna, kar pomeni, da so pripadajoče Ritzeve vrednosti vse enostavne. Matrika T_k je tudi simetrična in pozitivno definitna, torej so v tem primeru vse Ritzeve vrednosti strogo pozitivne. Vemo celo več, Ritzeve vrednosti v koraku k se prepletajo z Ritzevimi vrednostmi iz koraka $k+1$, saj se lastne vrednosti T_k prepletajo z lastnimi vrednostmi T_{k+1}. Če so $\theta_k^{(k)} < \cdots < \theta_1^{(k)}$ Ritzeve vrednosti, ki jih dobimo v k-tem koraku Lanczosevega algoritma, potem velja strogo prepletanje

$$\theta_{k+1}^{(k+1)} < \theta_k^{(k)} < \theta_k^{(k+1)} < \cdots < \theta_2^{(k+1)} < \theta_1^{(k)} < \theta_1^{(k+1)}.$$

Očitno velja tudi $\lambda_n \leq \theta_k^{(k)}$ in $\theta_1^{(k)} \leq \lambda_1$. Zaradi tega Ritzeve vrednosti monotono konvergirajo proti lastnim vrednostim matrike A, pri čemer najprej skonvergirajo k zunanjim (največjim in najmanjšim) lastnim vrednostim.

Lema 3.13 *Za polinom p_k stopnje k, ki nastopa v metodi konjugiranih gradientov, velja*

$$p_k(t) = \frac{(\theta_1 - t)(\theta_2 - t)\cdots(\theta_k - t)}{\theta_1\theta_2\cdots\theta_k},$$

kjer so $\theta_1, \ldots, \theta_k$ Ritzeve vrednosti za $V_k^T A V_k$, pri čemer stolpci V_k tvorijo ortonormirano bazo za podprostor Krilova $\mathcal{K}_k(A, r_0)$.

Dokaz. Vemo, da velja

$$AV_k = V_k T_k + \gamma_k v_{k+1} e_k^T,$$

kjer smo z γ_k označili ustrezni element iz Lanczoseve metode. Če to enačbo pomnožimo s prvim enotskim vektorjem e_1, dobimo

$$Av_1 = V_k T_k e_1.$$

Pokažimo, da za $i < k$ velja $A^i v_1 = V_k T_k^i e_1$. Res, indukcijski korak pravi

$$A^{i+1} v_1 = AV_k T_k^i e_1 = (V_k T_k + \gamma_k v_{k+1} e_k^T) T_k^i e_1 = V_k T_k^{i+1} e_1,$$

saj je $e_k^T T_k^i e_1 = 0$ za $i < k$. Podobno pokažemo, da pri $i = k$ dobimo $A^k v_1 = V_k T_k^k e_1 + \gamma v_{k+1}$ za neko konstanto γ. Iz vsega skupaj lahko zaključimo, da velja

$$V_k^T p_k(A) v_1 = p_k(T_k) e_1 = 0, \tag{3.3}$$

saj vemo, da je ostanek $r_k = p_k(A) r_0$ po eni strani kolinearen z vektorjem $p_k(A)v_1$, po drugi strani pa pravokoten na $\mathcal{K}_k(A, r_0)$.

Vidimo, da je matrika $p_k(T_k)$ singularna, pokazali pa bomo, da je kar enaka 0. Naj bodo $u_1, \ldots, u_k$ ortonormirani lastni vektorji matrike T_k, ki po vrsti pripadajo Ritzevim vrednostim $\theta_1, \ldots, \theta_k$, ki so lastne vrednosti matrike T_k. Ker je T_k nerazcepna tridiagonalna matrika, velja $e_1^T u_j \neq 0$ za $j = 1, \ldots, k$, saj lastni vektorji tridiagonalnih nerazcepnih matrik ne morejo imeti prvega ali zadnjega elementa enakega 0. Če vektor e_1 razvijemo po $u_1, \ldots, u_n$, tako velja

$$e_1 = \sum_{i=1}^{k} \tau_i u_i,$$

kjer je $\tau_i = e_1^T u_i \neq 0$ za $i = 1, \ldots, k$. Iz (3.3) lahko sedaj izpeljemo

$$p_k(T_k) e_1 = \sum_{i=1}^{k} \tau_i p_k(\sigma_i) u_i = 0,$$

to pa je zaradi linearne neodvisnosti vektorjev $u_1, \ldots, u_k$ in neničelnosti koeficientov $\tau_1, \ldots, \tau_k$ možno le, če je $p_k(\sigma_i) = 0$ za $i = 1, \ldots, k$. To pomeni, da so Ritzeve vrednosti $\sigma_1, \ldots, \sigma_k$ natanko vse ničle polinoma p_k, ki je stopnje k. Ker mora veljati še $p_k(0) = 1$, polinom ne more imeti drugačne oblike od tiste, ki je navedena v lemi. ∎

Če npr. v nekem koraku Ritzeva vrednost θ_1 dobro aproksimira lastno vrednost λ_1, potem od tega koraka dalje metoda konjugiranih gradientov konvergira tako, kot da lastne vrednosti λ_1 ne bi bilo. Namesto od $\kappa = \lambda_1/\lambda_n$ je sedaj v nadaljevanju konvergenca odvisna od razmerja λ_2/λ_n oziroma od razmerja med največjo in najmanjšo lastno vrednostjo matrike A, za kateri Ritzeve vrednosti še niso skonvergirale.

3.7 Predpogojevanje

Ko rešujemo sistem $Ax = b$ z iterativno metodo, je konvergenca v veliki meri odvisna od lastnosti matrike A. Tako je npr. pri metodi konjugiranih gradientov konvergenca odvisna od razporeda lastnih vrednosti matrike A. Če lastne vrednosti nastopajo v gručah, potem lahko obstaja polinom nizke stopnje, ki bo imel v vseh lastnih vrednostih majhno absolutno vrednost in konvergenca je hitra. V najslabšem primeru, ko so lastne vrednosti dobro separirane, lahko pri eksaktnem računanju potrebujemo tudi n korakov, pri numeričnem računanju pa zaradi zaokrožitvenih napak in izgube ortogonalnosti metoda lahko potrebuje tudi dosti več kot n korakov.

Kadar je konvergenca počasna, si lahko pomagamo s *predpogojevanjem*. Pri t.i. *levem predpogojevanju* namesto sistema $Ax = b$ rešujemo sistem

$$K^{-1}Ax = K^{-1}b,$$

kjer za nesingularno matriko K, ki ji pravimo *(leva) predpogojenka*, velja:

a) matrika K je dobra aproksimacija matrike A,

b) konstrukcija matrike K ni preveč zahtevna,

c) sistem z matriko K lahko rešimo dosti enostavneje kot sistem z matriko A.

Pri uporabi izbrane metode na predpogojenem sistemu moramo v vsakem koraku izračunati produkt vektorja z matriko $K^{-1}A$. Te matrike razen izjemoma ne izračunamo eksplicitno. Tako kot matrika A je ponavadi tudi matrika K razpršena, matrika $K^{-1}A$ pa ima lahko dosti več neničelnih elementov oz. je lahko celo polna. Namesto tega v vsakem koraku najprej množimo vektor z matriko A, potem pa rešimo sistem z matriko K. Pri levem predpogojevanju moramo biti pozorni tudi na to, da sedaj delamo z ostankom $K^{-1}b - K^{-1}Ax$. Če je norma tega ostanka majhna, to še ne pomeni nujno, da je tudi norma ostanka $b - Ax$ dovolj majhna.

Pri *desnem predpogojevanju* namesto $Ax = b$ rešujemo sistem $AK^{-1}z = b$, kjer je $x = K^{-1}z$. Matrika K je sedaj *desna predpogojenka*. Prednost desnega predpogojevanja je, da ne vpliva na desno stran b. Seveda lahko predpogojujemo tudi z obeh strani in rešujemo sistem $K_1^{-1}AK_2^{-1}z = K_1^{-1}b$, kjer je $x = K_2^{-1}z$. Obojestransko predpogojevanje je uporabno v primeru, ko imamo levo predpogojenko podano v obliki razcepa $K = K_1K_2$.

Za izbiro primerne predpogojenke K obstaja mnogo načinov. V praksi je dostikrat prav izbira ustrezne predpogojenke odločilna za to, da je velik razpršen sistem sploh možno numerično rešiti. Za različne vrste linearnih sistemov, ki izvirajo iz problemov, kot so npr. parcialne diferencialne enačbe, obstaja več različnih konstrukcij predpogojenk, raziskave na tem področju pa so zelo obsežne.

Prva možnost je uporaba katere izmed t.i. klasičnih iterativnih metod, ki smo jih obdelali v drugem poglavju. Pri teh pri reševanju sistema $Ax = b$ matriko A razdelimo na $A = M + N$, potem pa sistem $Mx = -Nx + b$ rešujemo iterativno kot

$$Mx_{k+1} = -Nx_k + b.$$

Tako dobimo iteracijo $x_{k+1} = Rx_k + f$, kjer je $f = M^{-1}b$ in

$$R = -M^{-1}N = -M^{-1}(A - M) = I - M^{-1}A.$$

To je v bistvu reševanje predpogojenega sistema $M^{-1}Ax = M^{-1}b$. Vsako matriko M iz klasične iteracijske metode lahko tako uporabimo za predpogojevanje iterativne metode podprostorov. Če matriko A razdelimo kot $A = L + D + U$, kjer je L spodnji trikotnik brez diagonale, D diagonala in U zgornji trikotnik brez diagonale, potem poznamo npr. naslednje variante:

- Jacobijevo predpogojevanje: $M_J = D$,
- Gauss-Seidelovo predpogojevanje: $M_{GS} = L + D$,
- SOR predpogojevanje: $M_{SOR} = \frac{1}{\omega}(D + \omega L)$,
- SSOR predpogojevanje: $M_{SSOR} = \frac{1}{\omega(2-\omega)}(D + \omega L)D^{-1}(D + \omega U)$.

Druga možnost za konstrukcijo predpogojenke so nepopolni razcepi. V primeru splošne matrike uporabimo nepopolni LU razcep, v primeru simetrične pozitivno definitne matrike pa nepopolni razcep Choleskega.

Predpostavimo lahko, da je LU razcep matrike A tak, da imata matriki L in U tako zelo veliko neničelnih elementov, da nam bodisi zmanjka pomnilnika ali pa je računanje časovno zelo potratno. Če bi namreč za reševanje sistema lahko uporabili kar LU razcep, bi najbrž to že naredili in ne bi potrebovali iterativnih metod. Resnici na ljubo je potrebno priznati, da se v praksi številne sisteme z razpršenimi matrikami da učinkovito rešiti kar z uporabo direktnih razpršenih LU razcepov oziroma razcepov Choleskega.

Pri nepopolnem LU razcepu dopustimo neničelne elemente v matrikah L in U le na izbranih mestih. Zato ne velja $A = LU$ temveč $A = LU + E$. Če lahko najdemo taki matriki L in U, da sta še vedno dovolj razpršeni, hkrati pa produkt LU dobro aproksimira A, lahko z njima pospešimo konvergenco.

Naj bo $P \subset \{(i,j) : \ i \neq j, \ i,j = 1,\ldots,n\}$ množica indeksov, kjer naj bodo elementi iz matrik L in U enaki 0. Preprosta različica nepopolnega LU razcepa, ki ima neničelne elemente le na mestih, ki niso v P, je predstavljen v naslednjem algoritmu.

Algoritem 3.10 Nepopolni LU razcep. Začetna podatka sta matrika A in množica indeksov P, kjer naj imata L in U ničelne elemente. Na koncu v spodnjem trikotniku A dobimo matriko L (brez enic na diagonali), v zgornjem trikotniku A pa matriko U, kjer L in U tvorita nepopolni LU razcep matrike A.

---

za vsak $(i,j) \in P$ nastavi $a_{ij} = 0$
$j = 1,\ldots,n-1$
  $i = j+1,\ldots,n$
    če $(i,j) \notin P$:
      $a_{ij} = a_{ij}/a_{jj}$
      $k = j+1,\ldots,n$
        če $(i,k) \notin P$: $a_{ik} = a_{ik} - a_{ij}a_{jk}$

Ponavadi za množico P velja, da je podmnožica indeksov, na katerih ima matrika A ničle. Če za množico P izberemo ravno

$$P = \{(i,j), \ a_{ij} = 0, \ i,j = 1,\ldots,n\},$$

potem je to t.i. razcep ILU(0), pri katerem ne nastanejo novi neničelni elementi. Elemente matrike P lahko določamo tudi dinamično med algoritmom. Ponavadi izberemo določen prag $\epsilon > 0$, potem pa dopuščamo le neničelne elemente v matrikah L in U, ki so dovolj relativno veliki glede na norme ustreznih stolpcev ali vrstic v matriki A. Manjši ko je ϵ, več neničelnih elementov bosta imeli matriki L in U in boljše bosta aproksimirali matriko A. Konvergenca bo glede na število korakov hitrejša, po drugi strani pa bo izračun matrik L in U terjal več časa, prav tako bo počasnejše tudi reševanje sistemov z matrikama L in U. Pri $\epsilon = 0$ bi dobili kar poln LU razcep, ki nam vrne rešitev že v začetnem koraku. Zato je optimalna izbira ponavadi ne premajhen in ne prevelik ϵ.

Pri računanju L in U se lahko uporabljajo še dodatni triki, ki pri določenih vrstah problemov izboljšajo konvergenco. Tako si lahko npr. zapomnimo elemente, ki so bili v določenem stolpcu ali vrstici matrike L oziroma U premajhni, da bi obdržali neničelne vrednosti, potem pa te vrednosti upoštevamo pri računanju naslednjih elementov matrik L in U. Več podrobnosti o predpogojevanju lahko najdete npr. v [7, 11].

3.7.1 Predpogojevanje konjugiranih gradientov

Pri metodi konjugiranih gradientov je matrika A simetrična in pozitivno definitna. Smiselno je, da ima to lastnost tudi predpogojenka K, ki naj bi dobro aproksimirala matriko A. Pri uporabi levega predpogojevanja potem nastopi težava, saj matrika $K^{-1}A$ ni več simetrična.

Pomagamo si lahko na dva načina. Prva varianta je, da si pomagamo z razcepom Choleskega $K = VV^T$ matrike K. Če poznamo matriko V, lahko uporabimo obojestransko predpogojevanje in rešujemo sistem

$$V^{-1}AV^{-T}z = V^{-1}b,$$

kjer je $x = V^{-T}z$. Ker je matrika $V^{-1}AV^{-T}$ simetrična in pozitivno definitna, lahko za ta sistem uporabimo metodo konjugiranih gradientov. Če označimo $\widetilde{A} = V^{-1}AV^{-T}$ in $\widetilde{b} = V^{-1}b$, potem lahko hitro preverimo, da za napako tekočega približka v predpogojenem sistemu velja

$$\|\widetilde{z} - z\|_{\widetilde{A}} = \|\widetilde{x} - x\|_A,$$

kjer je $\widetilde{x}$ rešitev originalnega, $\widetilde{z}$ pa rešitev predpogojenega sistema.

Druga varianta je, da definiramo skalarni produkt $[x, y] := y^T Kx$, ki je dobro definiran zaradi pozitivne definitnosti matrike K. V tem skalarnem produktu je matrika $K^{-1}A$ sebiadjungirana, saj je

$$[K^{-1}Ax, y] = y^T KK^{-1}Ax = y^T Ax = (K^{-1}Ay)^T Kx = [x, K^{-1}Ay].$$

V skalarnem produktu $[.,.]$ je matrika $K^{-1}A$ tudi pozitivno definitna, saj za $x \neq 0$ velja

$$[K^{-1}Ax, x] = x^T KK^{-1}Ax = x^T Ax > 0.$$

Metoda konjugiranih gradientov s skalarnim produktom $[.,.]$ je zapisana v algoritmu 3.11.

Če primerjamo zahtevnosti standardne metode konjugiranih gradientov v algoritmu 3.9 in predpogojene različice v algoritmu 3.11, imamo pri predpogojenem algoritmu dodatno še eno reševanje sistema z matriko K in en pomožni vektor, kamor shranimo $K^{-1}r_j$.

Izkaže se, da je algoritem 3.11 ekvivalenten uporabi navadne metode konjugiranih gradientov za sistem

$$K^{-1/2}AK^{-1/2}y = K^{-1/2}b, \qquad x = K^{-1/2}y.$$

Algoritem 3.11 Predpogojena metoda konjugiranih gradientov za reševanje sistema $Ax = b$, kjer je A simetrična pozitivno definitna. Vhodni podatki so matrika A, predpogojenka K, desna stran b, začetni približek x_0 in dimenzija k.

$$
\begin{aligned}
&r_0 = b - Ax_0, \quad p_1 = K^{-1}r_0\\
&j = 1, 2, \ldots, k\\
&\qquad \alpha_j = \frac{r_{j-1}^T K^{-1} r_{j-1}}{p_j^T A p_j}\\
&\qquad x_j = x_{j-1} + \alpha_j p_j\\
&\qquad r_j = r_{j-1} - \alpha_j A p_j\\
&\qquad \beta_j = \frac{r_j^T K^{-1} r_j}{r_{j-1}^T K^{-1} r_{j-1}}\\
&\qquad p_{j+1} = K^{-1} r_j + \beta_j p_j
\end{aligned}
$$

Če označimo $\widehat{A} = K^{-1/2} A K^{-1/2}$, $\widehat{b} = K^{-1/2} b$ in za sistem $\widehat{A}y = \widehat{b}$ uporabimo metodo konjugiranih gradientov z začetnim približkom $y_0 = K^{1/2} x_0$, potem veljajo enakosti $y_k = K^{1/2} x_k$, $\widehat{r}_k = \widehat{b} - \widehat{A} y_k = K^{-1/2} r_k$, $\widehat{p}_k = K^{1/2} p_k$, koeficienti α_k in β_k pri obeh metodah pa se ujemajo.

Za predpogojevanje lahko pri konjugiranih gradientih od klasičnih metod uporabimo Jacobijevo metodo ali SSOR, saj matrika M pri Gauss-Seidelovi metodi in SOR ni simetrična. Za matriko M_{SSOR} se da pokazati, da je v primeru, ko je A simetrična in pozitivno definitna, taka tudi M_{SSOR} za $0 < \omega < 2$.

Druga možnost za konstrukcijo predpogojenke je nepopoln razcep Choleskega, ki ga izvedemo na podoben način kot nepopoln LU razcep. Tudi tu lahko dopuščamo neničelne elemente le na istih mestih kot v matriki A ali pa vpeljemo prag $\epsilon > 0$ in v matriki V iz razcepa Choleskega pustimo neničelne le tiste elemente, ki so med računanjem razcepa dovolj veliki.

3.8 GMRES

Naj bo x_0 začetni približek za rešitev linearnega sistema $Ax = b$, iščemo pa približek x_k v prostoru $x_0 + \mathcal{K}_k(A, r_0)$, kjer je $r_0 = b - Ax_0$. Denimo, da smo s pomočjo Arnoldijevega algoritma zgradili ortonormirano bazo za $\mathcal{K}_k(A, r_0)$, torej

$$AV_k = V_k H_k + h_{k+1,k} v_{k+1} e_k^T = V_{k+1} \widetilde{H}_k.$$

Pri metodi FOM (in posledično tudi pri D-Lanczosu in konjugiranih gradientih) za približek x_k vzamemo tak vektor, da je ostanek $r_k = b - Ax_k$ pravokoten na podprostor Krilova $\mathcal{K}_k(A, r_0)$. Rešitev je $x_k = x_0 + V_k y_k$, kjer je $y_k \in \mathbb{R}^k$ rešitev sistema $H_k y_k = \|r_0\| e_1$.

Namesto tega pri metodah, ki temeljijo na minimalnem ostanku, vzamemo $x_k \in x_0 + \mathcal{K}_k(A, r_0)$, ki minimizira normo ostanka $\|r_k\| = \|b - Ax_k\|$. Ker je $v_1 = r_0 / \|r_0\|$, velja

$$\|b - A(x_0 + V_k y_k)\| = \|r_0 - AV_k y_k\| = \|r_0 - V_{k+1} \widetilde{H}_k y_k\| = \|\|r_0\| e_1 - \widetilde{H}_k y_k\|.$$

Sedaj za y_k vzamemo rešitev predoločenega sistema velikosti $(k+1) \times k$ z matriko $\widetilde{H}_k$. Tako dobimo metodo posplošenega minimalnega ostanka oz. GMRES.

Algoritem 3.12 Metoda GMRES za reševanje sistema $Ax = b$. Vhodni podatki so matrika A, desna stran b, začetni približek x_0 in dimenzija k. Algoritem vrne približek x_k po metodi GMRES.

$r_0 = b - Ax_0$
Arnoldi$(A, r_0, k) \Longrightarrow AV_k = V_{k+1}\widetilde{H}_k$
$y_k = \widetilde{H}_k^+ \|r_0\| e_1$ (reši predoločen sistem $\widetilde{H}_k y_k = \|r_0\| e_1$)
$x_k = x_0 + V_k y_k$

V praksi se seveda ne odločimo vnaprej za dimenzijo k, temveč sproti za vsak k poiščemo x_k, z iteracijo pa končamo, ko je ostanek r_k dovolj majhen. Ker se, ko gremo iz k v $k+1$, matrika predoločenega sistema le malo spremeni, pri reševanju le tega ni potrebno vse računati od začetka.

Tako kot pri FOM tudi pri GMRES za oceno napake $\| \|r_0\|_2 e_1 - \widetilde{H}_k y_k\|_2$ vektorja y_k v resnici ni potrebno izračunati. Zaradi oblike matrike $\widetilde{H}_k$ je predoločeni sistem z matriko $\widetilde{H}_k$ najenostavneje reševati s pomočjo Givensovih rotacij, saj moramo pri QR razcepu uničiti le elemente na poddiagonali. Če so $R_{12}, \ldots, R_{k,k+1}$ take Givensove rotacije, da je $R_{k,k+1}^T \cdots R_{12}^T \widetilde{H}_k$ zgornja trapezna matrika, potem je minimum $\| \|r_0\|_2 e_1 - \widetilde{H}_k y_k\|_2$ po vseh vektorjih $y_k \in \mathbb{R}^k$ enak absolutni vrednosti $(k+1)$-vega elementa vektorja $\|r_0\|_2 R_{k,k+1}^T \cdots R_{12}^T e_1$. Če je Givensova rotacija $R_{j,j+1}$ določena s c_j in s_j, lahko hitro vidimo, da je napaka v k-tem koraku enaka $\|r_0\|_2 |s_1 \cdots s_k|$. Ko dimenzijo povečamo za 1, moramo dodati le še novo Givensovo rotacijo, saj se matriki $\widetilde{H}_k$ in $\widetilde{H}_{k+1}$ razlikujeta le v zadnjem stolpcu in vrstici.

Ko je k velik, imamo v algoritmu veliko dela na mestu, ko moramo nov vektor ortogonalizirati na vse prejšnje, poleg tega pa moramo imeti v spominu tudi k vektorjev, kar je lahko hitro preveč. Zaradi tega uporabljamo *GMRES s ponovnim zagonom*. Ko k preveč naraste, vzamemo za nov začetni približek x_k in spet zaženemo GMRES od začetka.

Kaj lahko povemo o konvergenci GMRES? Očitno velja $\|r_{k+1}\|_2 \leq \|r_k\|_2$, torej je konvergenca monotona. Približek x_k lahko zapišemo kot $x_k = x_0 + q_{k-1}(A) r_0$, kjer je q_{k-1} nek polinom stopnje $k-1$. Od tod sledi

$$r_k = p_k(A) r_0,$$

kjer je $p_k(A) = I - A q_{k-1}(A)$, torej je p_k polinom stopnje k za katerega velja $p_k(0) = 1$. Pri GMRES iščemo tak polinom stopnje k, da je norma $\|p_k(A) r_0\|_2$ minimalna. Zato velja

$$\frac{\|r_k\|_2}{\|r_0\|_2} \leq \min_{\substack{p_k \text{ stopnje } k \\ p_k(0)=1}} \|p_k(A)\|_2.$$

Vprašanje je torej, kako majhna je lahko norma $\|p_k(A)\|_2$, če je p_k polinom stopnje k in velja $p_k(0) = 1$. Če predpostavimo, da se da matrika A diagonalizirati kot $A = X \Lambda X^{-1}$, potem je $p_k(A) = X p_k(\Lambda) X^{-1}$ in dobimo oceno

$$\frac{\|r_k\|_2}{\|r_0\|_2} \leq \kappa_2(X) \min_{\substack{p_k \text{ stopnje } k \\ p_k(0)=1}} \max_{\lambda \in \sigma(A)} |p_k(\lambda)|.$$

Tako kot pri FOM tudi pri GMRES velja, da v primeru, ko ima matrika A le k različnih lastnih vrednosti, potem metoda GMRES skonvergira v največ k korakih.

Kot kaže naslednji izrek, konvergenca GMRES ni odvisna le od razporeditve lastnih vrednosti.

Izrek 3.14 (Greenbaum, Ptak, Strakoš (1996)) *Za poljubna nenaraščajoča pozitivna števila*

$$f_0 \geq f_1 \geq \cdots \geq f_{n-1}$$

in poljubna neničelna kompleksna števila $\lambda_1, \ldots, \lambda_n$ obstaja taka matrika A, da ima lastne vrednosti $\lambda_1, \ldots, \lambda_n$ in tak vektor b z normo $\|b\| = f_0$, da za ostanke pri metodi GMRES za reševanje sistema $Ax = b$ z $x_0 = 0$ velja $\|r_k\| = f_k$ za $k = 0, \ldots, n-1$.

Na konvergenco tako poleg lastnih vrednosti vplivajo tudi lastni in korenski vektorji. Če se matrika da diagonalizirati in je matrika lastnih vektorjev blizu ortogonalne, kar je ekvivalentno temu, da je matrika A blizu normalne matrike, je pomemben samo razpored lastnih vrednosti. V praksi se tudi tu pojavi superlinearna konvergenca. Če Ritzeve vrednosti matrike H_k dobro aproksimirajo lastno vrednost matrike A, potem ta lastna vrednost in pripadajoči lastni vektor ne vplivata več na konvergenco.

Pri FOM smo iskali tak vektor $x_k \in x_0 + \mathcal{K}_k(A, r_0)$, da je ostanek $r_k = b - Ax_k$ pravokoten na podprostor $\mathcal{K}_k(A, r_0)$. Tudi GMRES si lahko predstavljamo v podobni obliki, le da sedaj iščemo ostanek, ki je pravokoten na $A\mathcal{K}_k(A, r_0)$.

Lema 3.15 *Denimo, da se Arnoldijev postopek za A in $r_0 = b - Ax_0$ ni končal pred k-tim korakom. Potem je vektor $x_k \in x_0 + \mathcal{K}_k(A, r_0)$, pri katerem je ostanek $r_k = b - Ax_k$ pravokoten na podprostor $A\mathcal{K}_k(A, r_0)$, rešitev po metodi GMRES.*

Dokaz. Iz Arnoldijevega postopka smo dobili $AV_k = V_{k+1}\widetilde{H}_k$. Vektor $x_k \in x_0 + \mathcal{K}_k(A, r_0)$ lahko zapišemo v obliki $x_k = x_0 + V_k y_k$ za $y_k \in \mathbb{R}^k$. Za ostanek tako velja

$$r_k = r_0 - AV_k y_k = r_0 - V_{k+1}\widetilde{H}_k y_k.$$

Pogoj, da je ostanek pravokoten na $A\mathcal{K}_k(A, r_0)$, je ekvivalenten temu, da je $(AV_k)^T r_k = 0$. Od tod sledi

$$(AV_k)^T r_k = (V_{k+1}\widetilde{H}_k)^T (r_0 - V_{k+1}\widetilde{H}_k y_k) = 0$$

oziroma

$$\widetilde{H}_k^T \widetilde{H}_k y_k = \widetilde{H}_k^T \|r_0\|_2 e_1,$$

kar je ravno normalni sistem za predoločen sistem $\widetilde{H}_k y_k = \|r_0\|_2 e_1$, ki ga rešimo pri metodi GMRES, torej se rešitvi ujemata. ∎

Tako kot pri metodi konjugiranih gradientov je tudi pri GMRES v praksi zelo pomembno, da izberemo pravo predpogojevanje. Pogosta izbira je npr. nepopolni LU razcep. Pri predpogojevanju moramo paziti na to, da če namesto $Ax = b$ rešujemo $K^{-1}Ax = K^{-1}b$, potem pri GMRES dobimo rešitev x_k, ki minimizira ostanek $\|K^{-1}(b - Ax_k)\|$, ne pa nujno tudi $\|b - Ax_k\|$.

Varianta GMRES za simetrične matrike, kjer se namesto Arnoldijevega uporabi Lanczosev algoritem, je MINRES.

3.8.1 Primerjava GMRES in FOM

Pri obeh metodah najprej z Arnoldijevim algoritmom zgeneriramo ortonormirano bazo podprostora Krilova $\mathcal{K}_k(A, r_0)$ in dobimo $AV_k = V_k H_k + h_{k+1,k} v_{k+1} e_k^T = V_{k+1} \widetilde{H}_k$. Potem pa:

- Pri FOM vzamemo $x_k^F = x_0 + V_k y_k^F$, kjer je $H_k y_k^F = \|r_0\| e_1$.
- Pri GMRES vzamemo $x_k^G = x_0 + V_k y_k^G$, kjer y_k^G minimizira $\|\widetilde{H}_k y_k^G - \|r_0\| e_1\|$.

Za ostanek pri FOM vemo, da velja $\|r_k^F\| = h_{k+1,k} |e_k^T y_k^F|$. Denimo, da smo z uporabo $k-1$ Givensovih rotacij spravili matriko H_k v zgornjo trikotno matriko $R_k = R_{k-1,k}^T \cdots R_{12}^T H_k$. Če je Givensova rotacija $R_{j,j+1}$ določena s c_j in s_j, potem ima vektor $R_{k-1,k}^T \cdots R_{12}^T e_1$ zadnji element enak $(-1)^{k-1} s_1 \cdots s_{k-1}$. Torej je

$$\|r_k^F\| = \frac{h_{k+1,k}}{|r_{kk}|} |s_1 \cdots s_{k-1}| \|r_0\|,$$

kjer je r_{kk} zadnji element matrike R_k.

Pri GMRES naredimo še eno Givensovo rotacijo in dobimo $\|r_k^G\| = |s_1 \cdots s_{k-1} s_k| \|r_0\|$. Rotacija $R_{k,k+1}$ je določena z elementoma r_{kk} in $h_{k+1,k}$ ter velja

$$c_k = \frac{r_{kk}}{\sqrt{r_{kk}^2 + h_{k+1,k}^2}} \quad \text{in} \quad s_k = \frac{h_{k+1,k}}{\sqrt{r_{kk}^2 + h_{k+1,k}^2}}.$$

Če označimo $\rho_k^F = \|r_k^F\|$ in $\rho_k^G = \|r_k^G\|$, potem smo zgoraj izpeljali, da velja $\rho_k^F = \frac{1}{|c_k|} \rho_k^G$.

Iz $\rho_k^G = |s_1 \cdots s_{k-1} s_k| \|r_0\|$ sledi $\rho_k^G = |s_k| \rho_{k-1}^G$. Tako dobimo

$$|c_k| = \sqrt{1 - \left(\frac{\rho_k^G}{\rho_{k-1}^G} \right)^2}$$

in

$$\rho_k^F = \frac{\rho_k^G}{\sqrt{1 - \left(\frac{\rho_k^G}{\rho_{k-1}^G} \right)^2}},$$

od tod pa

$$\frac{1}{(\rho_k^F)^2} + \frac{1}{(\rho_{k-1}^G)^2} = \frac{1}{(\rho_k^G)^2}.$$

Tako pridemo do zveze

$$\sum_{i=0}^{k} \frac{1}{(\rho_i^F)^2} = \frac{1}{(\rho_k^G)^2}.$$

Iz zgornje enakosti sledi, da v primeru, ko pri metodi GMRES v koraku k dobimo majhen ostanek, moramo majhen ostanek dobiti tudi pri metodi FOM v vsaj enem izmed prvih k korakih. Konvergenci metod sta torej povezani in če skonvergira ena, potem skonvergira tudi druga.

3.9 MINRES

V primeru simetrične matrike lahko, podobno kot smo iz metode FOM izpeljali metodo D-Lanczos, izpeljemo metodo MINRES. Tu je treba priznati, da je bila metoda MINRES razvita že leta 1974, torej pred GMRES, ki izvira iz leta 1986, in je v bistvu GMRES posplošitev metode MINRES za nesimetrične matrike, kot sledi tudi iz samega imena metode.

Naj bo x_0 začetni približek za rešitev sistema $Ax = b$, kjer je matrika A simetrična. Zaradi simetrije namesto Arnoldijevega algoritma uporabimo Lanczosev algoritem 3.6, ki vrne vektorje $v_1, \ldots, v_k$ ki tvorijo ortonormirano bazo za podprostor Krilova $\mathcal{K}_k(A, r_0)$, kjer je $r_0 - Ax_0$, in tridiagonalno matriko

$$T_k = \begin{bmatrix} \alpha_1 & \beta_1 & & \\ \beta_1 & \alpha_2 & \ddots & \\ & \ddots & \ddots & \beta_{k-1} \\ & & \beta_{k-1} & \alpha_k \end{bmatrix},$$

da je

$$AV_k = V_k T_k + \beta_k v_{k+1} e_k^T = V_{k+1} \widetilde{T}_k,$$

kjer je $\widetilde{T}_k$ matrika velikosti $(k+1) \times k$, ki jo dobimo tako, da T_k razširimo z vrstico $\beta_k e_k^T$.

Če iščemo vektor x_k oblike $x_k \in x_0 + \mathcal{K}_k(A, r_0)$, ki minimizira $\|r_k\|_2 = \|b - Ax_k\|_2$, potem na podoben način kot pri GMRES ugotovimo, da je $x_k = x_0 + V_k y_k$, kjer je y_k rešitev predoločenega sistema $\widetilde{T}_k y_k = \|r_0\|_2 e_1$.

Tako kot pri GMRES predoločeni sistem z matriko $\widetilde{T}_k$ rešimo z uporabo k Givensovih rotacij, s katerimi uničimo elemente na poddiagonali. Ker je matrika $\widetilde{T}_k$ tridiagonalna, ima ustrezna zgornja trapezna matrika iz QR razcepa neničelne le prve tri diagonale. Pišemo lahko kar

$$R_{k,k+1}^T \cdots R_{12}^T \widetilde{T}_k = \begin{bmatrix} f_1 & g_1 & h_1 & & \\ & f_2 & g_2 & \ddots & \\ & & \ddots & \ddots & h_{k-2} \\ & & & \ddots & g_{k-1} \\ & & & & f_k \\ & & & & 0 \end{bmatrix} = \widetilde{R}_k,$$

kjer je $R_{j,j+1}^T$ ustrezno izbrana Givensova rotacija, ki deluje na vrsticah j in $j+1$ za $j = 1, \ldots, k$.

Prvih k vrstic zgornje trapezne matrike $\widetilde{R}_k$ označimo z R_k. Matrika R_k je zgornja trikotna in nesingularna, če se Lanczosev algoritem ni končal pred k-tim korakom. Če podobno označimo prvih k stolpcev matrike $\widetilde{Q}_k = R_{12} \cdots R_{k,k+1}$ s Q_k, potem imamo QR razcep $T_k = Q_k R_k$.

Za vektor y_k, ki reši predoločen sistem $\widetilde{T}_k y_k = \|r_0\|_2 e_1$, potem velja, da je

$$y_k = R_k^{-1} Q_k^T \|r_0\|_2 e_1.$$

Ko to vstavimo v izraz za x_k, dobimo

$$x_k = x_0 + V_k R_k^{-1} Q_k^T \|r_0\|_2 e_1.$$

Če definiramo matriko $P_k = V_k R_k^{-1}$ in vektor $z_k = Q_k^T \|r_0\|_2 e_1$, potem velja

$$x_k = x_0 + P_k z_k = x_0 + \sum_{i=1}^{k} \zeta_i p_i = x_{k-1} + \zeta_k p_k,$$

kjer je $z_k = [\, \zeta_1 \quad \cdots \quad \zeta_k \,]^T$ in $P_k = [\, p_1 \quad \cdots \quad p_k \,]$. Vidimo, da lahko x_k dobimo s posodobitvijo približka x_{k-1} iz prejšnjega koraka v smeri p_k. Ker lahko vektor p_k zaradi tridiagonalne oblike matrike R_k izračunamo iz p_{k-1} in p_{k-2}, tako dobimo kratko rekurzijo za približke x_k, podobno kot pri metodi D-Lanczos. Za razliko od metode D-Lanczos, ki ne deluje, kadar ne obstaja LU razcep brez pivotiranja matrike T_k, pri metodi MINRES s tem nimamo težav, saj smo namesto LU razcepa uporabili QR razcep, ki vedno obstaja.

Iz $V_k R_k^{-1} = P_k$ sledi $P_k R_k = V_k$, kar pomeni, da lahko vektor p_k za $k > 2$ izrazimo kot

$$p_k = \frac{1}{f_k}\big(v_k - h_{k-2} p_{k-2} - g_{k-1} p_{k-1}\big), \tag{3.4}$$

medtem ko za prva vektorja velja $p_1 = (1/f_1)v_1$ in $p_2 = (1/f_2)(v_2 - g_1 p_1)$.

Denimo, da smo že izvedli $k-1$ korakov Lanczosevega algoritma in izračunali približek x_{k-1} po metodi MINRES. To pomeni, da poznamo matriko R_{k-1}, parametre uporabljenih rotacij $c_1, \ldots, c_{k-1}$ in $s_1, \ldots, s_{k-1}$, vektorje $v_1, \ldots, v_k$, matriko T_{k-1} in β_{k-1}, ter vektorje $p_1, \ldots, p_{k-1}$ in skalarje $\zeta_1, \ldots, \zeta_{k-1}$, kjer je

$$z_{k-1} = [\, \zeta_1 \quad \cdots \quad \zeta_{k-1} \,]^T = Q_{k-1}^T \|r_0\| e_1$$

vektor velikosti $k-1$. Če ima Givensova rotacija $R_{j,j+1}^T$ v ravnini $(j, j+1)$ obliko

$$\begin{bmatrix} c_j & s_j \\ -s_j & c_j \end{bmatrix},$$

potem lahko hitro preverimo, da ima vektor $\widetilde{Q}_{k-1}^T \|r_0\|_2 e_1$ obliko

$$\widetilde{Q}_{k-1}^T \|r_0\|_2 e_1 = \|r_0\|_2 \begin{bmatrix} c_1 \\ -s_1 c_2 \\ s_1 s_2 c_3 \\ \vdots \\ (-1)^{k-3} s_1 \cdots s_{k-2} c_{k-1} \\ (-1)^{k-2} s_1 \cdots s_{k-2} s_{k-1} \end{bmatrix} = \begin{bmatrix} \zeta_1 \\ \zeta_2 \\ \zeta_3 \\ \vdots \\ \zeta_{k-1} \\ \chi_k \end{bmatrix},$$

pri čemer začnemo s $\chi_1 = \|r_0\|_2$.

Iz koraka $k-1$ pridemo v korak k na naslednji način. Najprej izvedemo nov korak Lanczosevega algoritma in tako pridemo do skalarjev α_k, β_k in vektorja v_{k+1}. Matriko $\widetilde{T}_{k-1}$ razširimo z novim stolpcem in vrstico v $\widetilde{T}_k$. QR razcep matrike $\widetilde{T}_{k-1}$ moramo najprej uporabiti na dodanem stolpcu, potem pa lahko dodamo še novo rotacijo, ki uniči element β_k.

Predpostavimo, da je $k > 2$, prva dva koraka pa je potrebno ustrezno popraviti. Začnemo z izračunom

$$\begin{bmatrix} h_{k-2} \\ \widetilde{g}_{k-1} \end{bmatrix} = \begin{bmatrix} c_{k-2} & s_{k-2} \\ -s_{k-2} & c_{k-2} \end{bmatrix} \begin{bmatrix} 0 \\ \beta_{k-1} \end{bmatrix},$$

ki mu sledi

$$\begin{bmatrix} g_{k-1} \\ \widetilde{f}_k \end{bmatrix} = \begin{bmatrix} c_{k-1} & s_{k-1} \\ -s_{k-1} & c_{k-1} \end{bmatrix} \begin{bmatrix} \widetilde{g}_{k-1} \\ \alpha_k \end{bmatrix}.$$

S tem smo na zadnjem stolpcu $\widetilde{T}_k$ uporabili rotaciji $R_{k-2,k-1}^T$ in $R_{k-1,k}^T$. Sedaj lahko določimo Givensovo rotacijo $R_{k,k+1}^T$. Ustrezna elementa sta

$$c_k = \frac{\widetilde{f}_k}{\sqrt{\widetilde{f}_k^2 + \beta_k^2}}, \quad s_k = \frac{\beta_k}{\sqrt{\widetilde{f}_k^2 + \beta_k^2}}.$$

S to rotacijo dobimo

$$\begin{bmatrix} f_k \\ 0 \end{bmatrix} = \begin{bmatrix} c_k & s_k \\ -s_k & c_k \end{bmatrix} \begin{bmatrix} \widetilde{f}_k \\ \beta_k \end{bmatrix}.$$

Izračunati moramo še ζ_k

$$\begin{bmatrix} \zeta_k \\ \chi_{k+1} \end{bmatrix} = \begin{bmatrix} c_k & s_k \\ -s_k & c_k \end{bmatrix} \begin{bmatrix} \chi_k \\ 0 \end{bmatrix}.$$

Sedaj lahko izračunamo p_k po formuli (3.4) in tako dobimo nov približek $x_k = x_{k-1} + \zeta_k p_k$.

Za ostanek velja $\|r_k\|_2 = |\chi_{k+1}|$. Algoritem je seveda potrebno implementirati tako, da imamo hkrati v spominu le končno število vektorjev. Tako potrebujemo le vektor x (tekoči približek, ki ga posodabljamo iz koraka v korak), zadnja dva stolpca P_k in vektor v.

3.10 Bikonjugirani gradienti

Če matrika A ni simetrična, potem pri Arnoldijevem algoritmu dobimo Hessenbergovo matriko. Ker je potrebno vsak nov bazni vektor podprostora Krilova ortogonalizirati na vse prejšnje, ne moremo dobiti algoritmov s kratko rekurzijo, kot jo imamo npr. pri konjugiranih gradientih ali direktni Lanczosevi metodi.

Do kratkih rekurzij lahko pridemo, če delamo z biortogonalnimi bazami. To pomeni, da zgradimo biortogonalni bazi za podprostora Krilova $\mathcal{K}_k(A, v_1)$ in $\mathcal{K}_k(A^T, w_1)$, tako da:

- stolpci $V_k = [v_1 \ \cdots \ v_k]$ tvorijo bazo za $\mathcal{K}_k(A, v_1)$,
- stolpci $W_k = [w_1 \ \cdots \ w_k]$ tvorijo bazo za $\mathcal{K}_k(A^T, w_1)$,
- velja $v_i^T w_j = 0$ za $i \neq j$ (in $v_i^T w_i = 1$).

Poglejmo, kako lahko zgradimo biortogonalni bazi. Denimo, da imamo matriki $V_j = \begin{bmatrix} v_1 & \cdots & v_j \end{bmatrix}$ in $W_j = \begin{bmatrix} w_1 & \cdots & w_j \end{bmatrix}$, za kateri velja, da prvih m stolpcev tvori biortogonalni bazi za $\mathcal{K}_m(A, v_1)$ in $\mathcal{K}_m(A^T, w_1)$ za $m = 1, \ldots, j$. Bazi moramo razširiti z novima vektorjema v_{j+1} in w_{j+1}. Nova vektorja imata pred skaliranjem, s katerim dosežemo, da je $v_{j+1}^T w_{j+1} = 1$, obliko

$$\begin{aligned} \widetilde{v}_{j+1} &= Av_j - \sum_{i=1}^{j} (w_i^T A v_j) v_i \\ \widetilde{w}_{j+1} &= Aw_j - \sum_{i=1}^{j} (v_i^T A w_j) w_i. \end{aligned}$$

Hitro lahko preverimo, da za zgornja vektorja res velja $w_i^T \widetilde{v}_{j+1} = v_i^T \widetilde{w}_{j+1} = 0$ za $i = 1, \ldots, j$. Ker je $A^T w_i \in \mathcal{K}_{i+1}(A^T, w_1)$, to pomeni, da je $w_i^T A v_j = 0$ za $i < j-1$. Podobno lahko pokažemo, da je $v_i^T A^T w_j = 0$ za $i < j-1$. To pomeni, da lahko vektorja $\widetilde{v}_{j+1}$ in $\widetilde{w}_{j+1}$ zapišemo v obliki

$$\begin{aligned} \widetilde{v}_{j+1} &= Av_j - (w_j^T A v_j) v_j - (w_{j-1}^T A v_j) v_{j-1} \\ \widetilde{w}_{j+1} &= Aw_j - (v_j^T A^T w_j) w_j - (v_{j-1}^T A^T w_j) w_{j-1}. \end{aligned}$$

Opazimo, da je prvi skalar v obeh razvojih enak. Če ga označimo z $\alpha_j = w_j^T A v_j = v_j^T A^T w_j$, lahko pišemo

$$\begin{aligned} \widetilde{v}_{j+1} &= Av_j - \alpha_j v_j - (w_{j-1}^T A v_j) v_{j-1} \\ \widetilde{w}_{j+1} &= Aw_j - \alpha_j w_j - (v_{j-1}^T A^T w_j) w_{j-1}. \end{aligned}$$

Sedaj vpeljemo še skalarje β_j in δ_j. To so skalarji, ki jih uporabimo, da skaliramo vektorje $\widetilde{v}_{j+1}$ in $\widetilde{w}_{j+1}$. Naj bo $v_{j+1} = \widetilde{v}_{j+1}/\delta_j$ in $w_{j+1} = \widetilde{w}_{j+1}/\beta_j$. Skalarja nista enolično določena, veljati mora le $\beta_j \delta_j = \widetilde{v}_{j+1}^T \widetilde{w}_{j+1}$. Sedaj dobimo

$$w_{j-1}^T A v_j = (A^T w_{j-1})^T v_j = \widetilde{w}_j^T v_j = \beta_{j-1} w_j^T v_j = \beta_{j-1},$$

saj je $w_{j-1}^T v_j = w_{j-2}^T v_j = 0$. Podobno lahko izpeljemo $v_{j-1}^T A^T w_j = \delta_{j-1}$. Tako pridemo do končne formule

$$\begin{aligned} \widetilde{v}_{j+1} &= Av_j - \alpha_j v_j - \beta_{j-1} v_{j-1} \\ \widetilde{w}_{j+1} &= Aw_j - \alpha_j w_j - \delta_{j-1} w_{j-1}. \end{aligned}$$

Vse skupaj lahko zapišemo v obliki algoritma za izračun biortogonalnih baz. Imenuje se dvostranski Lanczos in je predstavljen v algoritmu 3.13.

Algoritem 3.13 Dvostranski Lanczosev algoritem. Vhodni podatki so matrika A, vektorja v_1 in w_1, da je $v_1^T w_1 = 1$, in dimenzija k. Algoritem vrne vektorje $v_1, \ldots, v_k$ in $w_1, \ldots, w_k$, ki tvorijo biortonormirano bazo za podprostora Krilova $\mathcal{K}_k(A, v_1)$ in $\mathcal{K}_k(A^T, w_1)$.

$\beta_0 = \delta_0 = 0$, $v_0 = w_0 = 0$
$j = 1, 2, \ldots, k$
  $\alpha_j = w_j^T A v_j$
  $\widetilde{v}_{j+1} = Av_j - \alpha_j v_j - \beta_{j-1} v_{j-1}$
  $\widetilde{w}_{j+1} = A^T w_j - \alpha_j w_j - \delta_{j-1} w_{j-1}$
  $\delta_j = |\widetilde{w}_{j+1}^T \widetilde{v}_{j+1}|^{1/2}$
  če je $\delta_j = 0$, potem prekini računanje
  $\beta_j = \widetilde{w}_{j+1}^T \widetilde{v}_{j+1}/\delta_j$
  $v_{j+1} = \widetilde{v}_{j+1}/\delta_j$
  $w_{j+1} = \widetilde{w}_{j+1}/\beta_j$

V vsakem koraku algoritma 3.13 moramo enkrat množiti z matriko A in enkrat z matriko A^T. Če se algoritem izvede do konca (velja $\delta_j \neq 0$ za $j \leq k$), potem na koncu dvostranskega Lanczosevega algoritma dobimo tridiagonalno matriko

$$T_k = \begin{bmatrix} \alpha_1 & \beta_1 & & \\ \delta_1 & \alpha_2 & \ddots & \\ & \ddots & \ddots & \beta_{k-1} \\ & & \delta_{k-1} & \alpha_k \end{bmatrix},$$

za katero velja

$$AV_k = V_k T_k + \delta_k v_{k+1} e_k^T,$$
$$A^T W_k = W_k T_k^T + \beta_k w_{k+1} e_k^T,$$
$$W_k^T A V_k = T_k$$

in

$$W_k^T V_k = I_k,$$

kjer je I_k matrika identitete velikosti k. Dodatno množenje z A^T nam omogoča, da tudi za nesimetrične matrike pridemo do algoritmov s kratkimi rekurzijami.

Metoda dvostranskega Lanczosa se lahko zalomi, če je $\delta_j = 0$ za $j \leq k$. Do tega lahko pride zaradi:

a) $\widetilde{v}_{j+1} = 0$: Če rešujemo sistem $Ax = b$ in je $v_1 = r_0 = b - Ax_0$, je to *srečni zalom*, saj smo naleteli na invarianten podprostor v katerem leži točna rešitev.

b) $\widetilde{w}_{j+1} = 0$: Če rešujemo sistem z matriko A^T je to srečni zalom, sicer pa ne pomaga.

c) $\widetilde{v}_{j+1} \neq 0$, $\widetilde{w}_{j+1} \neq 0$, toda $\widetilde{v}_{j+1}^T \widetilde{w}_{j+1} = 0$: To imenujemo *resni zalom*, rešitev za to situacijo pa je nekaj:

   - Ponovni zagon. Takoj, ko se metoda zalomi (oziroma ko je numerično δ_j preblizu 0), naredimo ponovni zagon. Na ta način sicer lahko nadaljujemo z računanjem, ampak izgubimo podprostor Krilova in možnost superlinearne konvergence.
   - Look-Ahead Lanczos. Pokvarimo tridiagonalno strukturo, tako da uporabimo več vektorjev in pridemo do bločno biortogonalne baze in bločne tridiagonalne matrike T_k. To pomeni, da lahko npr. najdemo par (v_{j+2}, w_{j+2}), čeprav par (v_{j+1}, w_{j+1}) ne obstaja.

Poglejmo, kako si lahko z Lanczosevo biortogonalizacijo pomagamo, da pridemo do približka za rešitev linearnega sistema $Ax = b$. Če je začetni približek x_0, potem vzamemo $v_1 = r_0/\|r_0\|$, kjer je $r_0 = b - Ax_0$. Izberemo še vektor w_1, da je $w_1^T v_1 = 1$, in z algoritmom 3.13 skontruiramo biortogonalni bazi za $\mathcal{K}_k(A, v_1)$ in $\mathcal{K}_k(A^T, w_1)$. Pri tem predpostavimo, da pri Lanczosevi biortogonalizaciji ne pride do resnega zaloma. Po k korakih imamo tako matrike T_k, V_k in W_k, da je $AV_k = V_k T_k + \delta_k v_{k+1} e_k^T$ in $A^T W_k = W_k T_k^T + \beta_k w_{k+1} e_k^T$.

Pri biortogonalnih bazah pri metodah, ki temeljijo na Petrov–Galerkinovemu pogoju, iščemo $x_k = x_0 + V_k y_k$, da je $W_k^T(b - Ax_k) = 0$, torej

$$W_k^T(b - Ax_0 - AV_k y_k) = 0,$$

od koder sledi $T_k y_k = \|r_0\| e_1$.

Za ostanek velja $\|r_k\| = \delta_k |e_k^T y_k| \cdot \|v_{k+1}\|$.

Do metode bikonjugiranih gradientov pridemo na podoben način, kot smo za simetrične pozitivno definitne matrike prišli iz Lanczosa do konjugiranih gradientov. Najprej predpostavimo, da obstaja LU razcep brez pivotiranja za matriko T_k iz algoritma 3.14. To nam omogoča, da pridemo do algoritma s kratko rekurzijo, kjer približek za vektor x_j posodabljamo iz koraka v korak in ne koncu ne potrebujemo več vseh vektorjev matrike V_k. Dobljena metoda je predstavljena v algoritmu 3.15.

Algoritem 3.14 Bi-Lanczoseva metoda za reševanje sistema $Ax = b$. Vhodni podatki so matrika A, desna stran b, začetni približek v_1, vektor w_1, da je $v_1^T w_1 = 1$ in dimenzija k.

$r_0 = b - Ax_0$
$v_1 = r_0/\|r_0\|$, izberi w_1, da je $v_1^T w_1 = 1$
Bi-Lanczos$(A, v_1, w_1, k) \implies AV_k = V_k T_k + \delta_k v_{k+1} e_k^T$ in $A^T W_k = W_k T_k^T + \beta_k w_{k+1} e_k^T$
$y_k = T_k^{-1}\|r_0\| e_1$
$x_k = x_0 + V_k y_k$

Algoritem 3.15 Metoda bikonjugiranih gradientov (BiCG) za reševanje sistema $Ax = b$. Vhodni podatki so matrika A, desna stran b, začetni približek x_0 in dimenzija k.

$r_0 = b - Ax_0, \quad p_1 = r_0$
izberi $\widetilde{r}_0$, da je $\widetilde{r}_0^T r_0 \neq 0, \quad \widetilde{p}_1 = \widetilde{r}_0$
$j = 1, 2, \ldots, k$
$\alpha_j = \dfrac{\widetilde{r}_{j-1}^T r_{j-1}}{\widetilde{p}_j^T A p_j}$
$x_j = x_{j-1} + \alpha_j p_j$
$r_j = r_{j-1} - \alpha_j A p_j$
$\widetilde{r}_j = \widetilde{r}_{j-1} - \alpha_j A^T \widetilde{p}_j$
$\beta_j = \dfrac{\widetilde{r}_j^T r_j}{\widetilde{r}_{j-1}^T r_{j-1}}$
$p_{j+1} = r_j + \beta_j p_j$
$\widetilde{p}_{j+1} = \widetilde{r}_j + \beta_j \widetilde{p}_j$

Za razliko od konjugiranih gradientov, kjer imamo zagotovljen obstoj LU razcepa brez pivotiranja, to ne velja za bikonjugirane gradiente. Lahko pride do zaloma, če LU razcep ne obstaja. Temu pravimo *sekundarni zalom*.

Izrek 3.16 *Če se metoda BiCG ne zalomi, potem so ostanki biortogonalni, oziroma* $\widetilde{r}_j^T r_i = 0$, *smeri pa so bikonjugirane, oziroma* $\widetilde{p}_j^T A p_j = 0$, *za* $i \neq j$.

Obstajajo številni primeri, ko imamo opravka z nesimetričnimi matrikami in množenje z matriko A^T sploh ni na voljo ali pa ni tako ekonomično kot množenje z matriko A. V takih primerih ne moremo uporabiti metode BiCG.

Zgled 3.1 *Denimo, da rešujemo nelinearni sistem* $F(x) = 0$ *z Newtonovo metodo. V vsakem koraku moramo rešiti sistem z Jacobijevo matriko* $JF(x)$. *Če to rešujemo preko metod podprostorov Krilova, lahko za poljuben vektor v produkt* $JF(x)v$ *dobro aproksimiramo z*

$$JF(x)v = \frac{F(x + \epsilon v) - F(x)}{\epsilon}$$

za primerno majhen ϵ. *Podobna ekonomična formula za izračun produkta* $JF(x)^T w$ *ne obstaja.*

3.10.1 QMR

Podobno, kot sta povezani GMRES in FOM, je metoda QMR povezana z BiCG. Dvostranski Lanczos vrne $AV_k = V_k T_k + \delta_k v_{k+1} e_k^T = V_{k+1} T_{k+1,k}$. Za razliko od simetrične matrike sedaj stolpci matrike V_{k+1} niso ortonormirani.

Iščemo vektor oblike $x_k = x_0 + V_k y_k$, za katerega bi bila norma ostanka

$$\|r_k\| = \|b - Ax_k\| = \|r_0 - V_k y_k\| = \|V_{k+1}(\|r_0\|e_1 - T_{k+1,k} y_k)\|$$

minimalna. Kljub temu, da stolpci V_{k+1} niso ortonormirani, kot približek vzamemo y_k, ki minimizira t.i. *kvazi-ostanek*

$$\|\|r_0\|e_1 - T_{k+1,k} y_k\|.$$

Tako dobimo metodo *minimalnega kvazi-ostanka* oziroma QMR. Metodo QMR izvedemo tako, da z uporabo Givensovih rotacij delamo QR razcep matrike T_k. Tako spet lahko delamo kratke rekurzije in posodabljamo x_k.

Če označimo $\|r_k^Q\| = \min(\|\|r_0\|e_1 - T_{k+1,k} y_k\|)$, potem za ostanek $r_k^{QMR} = b - Ax_k$ velja

$$\|r_k^{QMR}\| \leq \|V_{k+1}\|_F \|r_k^Q\|.$$

3.11 Linearni problemi najmanjših kvadratov

3.11.1 CGLS

Denimo, da imamo predoločen sistem $Ax = b$, kjer je $A \in \mathbb{R}^{m \times n}$, $m \geq n$ in $\operatorname{rang}(A) = n$. Rešitev x po metodi najmanjših kvadratov, ki minimizira $\|b - Ax\|_2$ lahko dobimo iz normal-

nega sistema $A^TAx = A^Tb$. Matrika A^TA je simetrična in pozitivno definitna, zato lahko za iterativno reševanje uporabimo metodo konjugiranih gradientov.

Pri tem matrike A^TA ne izračunamo eksplicitno, temveč namesto tega množimo z matrikama A in A^T. Ustrezen algoritem se imenuje CGLS (konjugirani gradienti za problem najmanjših kvadratov). Izpeljemo ga iz algoritma 3.9 in je predstavljen v algoritmu 3.16. V primerjavi z originalno metodo konjugiranih gradientov iz algoritma 3.9 smo sedaj ostanke originalnega sistema označili z $r_i = b - Ax_i$, za ostanke normalnega sistema pa smo uporabili oznako $z_i = A^Tr_i$.

Algoritem 3.16 Metoda konjugiranih gradientov za problem najmanjših kvadratov, kjer rešujemo normalni sistem $A^TAx = A^Tb$. Vhodni podatki so matrika A, desna stran b, začetni približek x_0 in dimenzija k.

$r_0 = b - Ax_0, \quad z_0 = A^Tr_0, \quad p_1 = z_0$
$j = 1, 2, \ldots, k$
$\quad w_j = Ap_j$
$\quad \alpha_j = \dfrac{\|z_{j-1}\|^2}{\|w_j\|^2}$
$\quad x_j = x_{j-1} + \alpha_j p_j$
$\quad r_j = r_{j-1} - \alpha_j w_j$
$\quad z_j = A^Tr_j$
$\quad \beta_j = \dfrac{\|z_j\|^2}{\|z_{j-1}\|^2}$
$\quad p_{j+1} = z_j + \beta_j p_j$

Lema 3.17 *Metoda CGLS vrne $x_k \in x_0 + \mathcal{K}_k(A^TA, A^Tr_0)$, ki minimizira ostanek $\|b - Ax_k\|_2$ v tem afinem podprostoru.*

To lahko uporabimo tudi za reševanje $n \times n$ sistema $Ax = b$, kjer je A nesimetrična matrika. Na ta način se izognemo dolgim rekurzijam, ki jih zaradi nesimetričnosti potrebujeta FOM in GMRES. Ker je občutljivost matrike A^TA kvadrat občutljivosti matrike A, je to primerno le za matrike, ki niso zelo občutljive.

3.11.2 LSQR

Kot pove naslednja lema, lahko namesto iz normalnega sistema rešitev predoločenega sistema dobimo tudi iz razširjenega sistema velikosti $(m+n) \times (m+n)$.

Lema 3.18 *Če je $A \in \mathbb{R}^{m\times n}$, $m \geq n$, $\operatorname{rang}(A) = n$, potem rešitev predoločenega sistema $Ax = b$ po metodi najmanjših kvadratov dobimo iz razširjenega sistema*

$$\begin{bmatrix} I & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} r \\ x \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix}. \tag{3.5}$$

Dokaz. Iz (3.5) sledi $r + Ax = b$ in $A^Tr = 0$. Če iz prve enačbe izrazimo $r = b - Ax$ in vstavimo v drugo, dobimo $A^Tb = A^TAx$, kar je ravno normalni sistem. Iz rešitve sistema (3.5) torej dobimo rešitev x in ostanek r. ∎

Matrika v sistemu (3.5) je simetrična. Za reševanje lahko uporabimo eno izmed metod podprostorov Krilova, za katero ortonormirano bazo zgradimo z Lanczosevim algoritmom. Zaradi posebne oblike matrike v enakosti (3.5) lahko to naredimo še bolj učinkovito.

Denimo, da za začetni približek za rešitev sistema (3.5) vzamemo $\widetilde{x}_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$. Označimo

$$\widetilde{A} = \begin{bmatrix} I & A \\ A^T & 0 \end{bmatrix}.$$

Prvi vektor v bazi za $\mathcal{K}_k(\widetilde{A}, \widetilde{r}_0)$, kjer je $\widetilde{r}_0 = \begin{bmatrix} b \\ 0 \end{bmatrix}$, ima obliko

$$w_1 = \begin{bmatrix} u_1 \\ 0 \end{bmatrix},$$

kjer je $u_1 = b/\|b\|_2$. Ker je

$$\widetilde{A}w_1 = \begin{bmatrix} u_1 \\ A^T u_1 \end{bmatrix},$$

ima naslednji vektor iz ortonormirane baze obliko

$$w_2 = \begin{bmatrix} 0 \\ v_1 \end{bmatrix},$$

kjer je $v_1 = A^T b/\|A^T b\|_2$. Iz

$$\widetilde{A}w_2 = \begin{bmatrix} Au_2 \\ 0 \end{bmatrix}$$

sledi, da se v ortonormirani bazi za $\mathcal{K}_k(\widetilde{A}, \widetilde{x}_0)$ izmenjujejo vektorji oblike $\begin{bmatrix} u \\ 0 \end{bmatrix}$ in $\begin{bmatrix} 0 \\ v \end{bmatrix}$. Postopek za izračun je zapisan v algoritmu 3.17.

Algoritem 3.17 Lanczosev algoritem za bazo podprostora Krilova za sistem (3.5). Vhodni podatki so matrika A, desna stran b in dimenzija k. Za začetni približek vzamemo $x_0 = 0$. Skalarje α_i in β_i v algoritmu je potrebno izbrati tako, da so vektorji u_i in v_i normirani.

$\beta_1 u_1 = b$
$\alpha_1 v_1 = A^T u_1$
$j = 1, \ldots, k$
  $\beta_{j+1} u_{j+1} = A v_j - \alpha_j u_j$
  $\alpha_{j+1} v_{j+1} = A^T u_{j+1} - \beta_{j+1} v_j$

Po k korakih algoritma 3.17 za $U_{k+1} = [\, u_1 \quad u_2 \quad \cdots \quad u_{k+1} \,]$ in $V_k = [\, v_1 \quad v_2 \quad \cdots \quad v_k \,]$ velja

$$\beta_1 U_{k+1} e_1 = b, \tag{3.6}$$
$$A V_k = U_{k+1} B_k, \tag{3.7}$$
$$A^T U_{k+1} = V_k B_k^T + \alpha_{k+1} v_{k+1} e_{k+1}^T, \tag{3.8}$$

kjer je

$$B_k = \begin{bmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \ddots & \ddots & \\ & & \beta_k & \alpha_k \\ & & & \beta_{k+1} \end{bmatrix}$$

matrika velikosti $(k+1) \times k$.

Izrek 3.19 *Če izvedemo k korakov algoritma 3.17, potem velja $T_k = B_k^T B_k$, kjer je T_k tridiagonalna simetrična matrika, ki bi jo dobili, če bi uporabili Lanczosev algoritem za $A^T A$, desno stran $A^T b$ in začetni približek $x_0 = 0$. Velja*

$$A^T A V_k = V_k T_k + \alpha_{k+1}\beta_{k+1} v_{k+1} e_k^T.$$

Dokaz. Zveza sledi iz enakosti (3.6), (3.7) in (3.8). Začnemo tako, da (3.7) z leve pomnožimo z A^T in dobimo

$$\begin{aligned} A^T A V_k &= A^T U_{k+1} B_k \\ &= (V_k B_k + \alpha_{k+1} v_{k+1} e_{k+1}^T) B_k \\ &= V_k B_k^T B_k + \alpha_{k+1} v_{k+1} e_{k+1}^T B_k \\ &= V_k B_k^T B_k + \alpha_{k+1}\beta_{k+1} v_{k+1} e_{k+1}^T, \end{aligned}$$

kjer smo v zadnjem koraku upoštevali, da je $e_{k+1}^T B_k = \beta_{k+1} e_{k+1}^T$. ∎

Ko rešujemo sistem (3.5), iščemo tak približek $\begin{bmatrix} r_k \\ x_k \end{bmatrix} \in \mathcal{K}_{2k+1}(\widetilde{A}, \widetilde{r}_0)$, da bo ostanek

$$\begin{bmatrix} I & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} r_k \\ x_k \end{bmatrix} - \begin{bmatrix} b \\ 0 \end{bmatrix}$$

pravokoten na $\mathcal{K}_{2k+1}(\widetilde{A}, \widetilde{r}_0)$. Pri tem je $x_k = V_k y_k$ in $r_k = U_{k+1} t_{k+1}$ za $y_k \in \mathbb{R}^k$ in $t_{k+1} \in \mathbb{R}^{k+1}$, saj stolpci matrike

$$\begin{bmatrix} U_{k+1} & 0 \\ 0 & V_k \end{bmatrix}$$

tvorijo ortonormirano bazo za $\mathcal{K}_{2k+1}(\widetilde{A}, \widetilde{r}_0)$. Veljati mora

$$\begin{bmatrix} U_{k+1}^T & 0 \\ 0 & V_k^T \end{bmatrix} \left(\begin{bmatrix} I & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} U_{k+1} t_{k+1} \\ V_k y_k \end{bmatrix} - \begin{bmatrix} b \\ 0 \end{bmatrix} \right) = 0,$$

iz česar sledi

$$\begin{bmatrix} I & B_k \\ B_k^T & 0 \end{bmatrix} \begin{bmatrix} t_{k+1} \\ y_k \end{bmatrix} = \begin{bmatrix} \beta_1 e_1 \\ 0 \end{bmatrix},$$

kar je ravno razširjen sistem oblike (3.5) za predoločen sistem $B_k y_k = \beta_1 e_1$. Pri metodi LSQR torej najprej uporabimo algoritem 3.17, nato rešimo predoločen sistem $B_k y_k = \beta_1 e_1$ velikosti $(k+1) \times k$ in za končni približek za rešitev vzamemo $x_k = V_k y_k$.

3.11.3 Regularizacija Tihonova

Denimo, da iščemo rešitev problema najmanjših kvadratov za predoločen sistem $Ax = b$, kjer matrika A ni polnega ranga. V tem primeru je rešitev več, saj je $\|Ax - b\|_2 = \|A(x+z) - b\|_2$ za poljuben tak $z \neq 0$, da je $Az = 0$. V tem primeru potem izmed vseh vektorjev x, ki minimizirajo normo $\|Ax - b\|_2$, za rešitev vzamemo x z najmanjšo normo $\|x\|_2$.

Če matrika A sicer ni singularna, a ima zelo majhne singularne vrednosti, se to odraža v tem, da majhne motnje b povzročijo zelo velike spremembe rešitve. Ker si lahko pri numeričnem

računanju predstavljamo, da namesto s točnimi delamo z malo zmotenimi podatki (če nič drugega že zaradi tega, ker računamo s premično piko in imamo na voljo samo končno število predstavljivih števil), bomo, če numerično rešimo tak sistem, dobili neuporabno rešitev x z zelo veliko normo.

Rešitev je regularizacija. Poznamo več vrst, ena izmed njih je regularizacija Tihonova, kjer izberemo regularizacijski parameter $\lambda \geq 0$ in rešimo sistem

$$(A^T A + \lambda^2 I)x = A^T b. \tag{3.9}$$

Lema 3.20 *Regularizacija Tihonova vrne vektor x, ki reši problem*

$$\min_{x \in \mathbb{R}^n} \left(\|b - Ax\|_2 + \lambda^2 \|x\|^2 \right). \tag{3.10}$$

Dokaz. Pišemo lahko

$$\|b - Ax\|_2 + \lambda^2 \|x\|^2 = \left\| \begin{bmatrix} b \\ 0 \end{bmatrix} - \begin{bmatrix} A \\ \lambda I \end{bmatrix} x \right\|_2^2.$$

Na desni imamo predoločen sistem, katerega normalni sistem je ravno (3.9). ∎

Opazimo, da večji, ko je parameter λ, manj je sistem (3.9) občutljiv. Po drugi strani večja vrednost λ pomeni, da je v (3.10) bolj pomembno, da je norma vektorja x čim manjša, kot pa velikost ostanka $b - Ax$. V limiti, ko λ pošljemo proti neskončno, bomo z regularizacijo Tihonova dobili $x = 0$. Obratno, ko gre λ proti 0, norma vektorja x narašča in norma ostanka pada.

Poglejmo, kako lahko regularizacijo vključimo v metodi CGLS in LSQR. Pri metodi CGLS moramo upoštevati, da sedaj rešujemo sistem $(A^T A + \lambda^2 I)x = A^T b$. Algoritem 3.16 se spremeni v algoritem 3.18, kjer sta spremembi le v izračunu z_j in α_j.

Algoritem 3.18 Metoda konjugiranih gradientov za regularizirani problem najmanjših kvadratov, kjer rešujemo normalni sistem $(A^T A + \lambda^2 I)x = A^T b$. Vhodni podatki so matrika A, desna stran b, začetni približek x_0, regularizacijski parameter λ in dimenzija k.

$r_0 = b - Ax_0, \quad z_0 = A^T r_0 - \lambda^2 x_0, \quad p_1 = z_0$
$j = 1, 2, \ldots, k$
  $w_j = Ap_j$
  $\alpha_j = \dfrac{\|z_{j-1}\|^2}{\|w_j\|^2 + \lambda^2 \|p_j\|^2}$
  $x_j = x_{j-1} + \alpha_j p_j$
  $r_j = r_{j-1} - \alpha_j w_j$
  $z_j = A^T r_j - \lambda^2 x_j$
  $\beta_j = \dfrac{\|z_j\|^2}{\|z_{j-1}\|^2}$
  $p_{j+1} = z_j + \beta_j p_j$

Opazimo, da moramo algoritem za vsak λ zagnati znova. Ker pri regularizaciji običajno ne poznamo optimalnega parametra in zato račun ponovimo za različne vrednosti λ, to ni najboljše. Izkaže se, da pri metodi LSQR zadošča le en izračun.

Lema 3.21 *Če je $A \in \mathbb{R}^{m\times n}$, $m \geq n$, $\operatorname{rang}(A) = n$, potem je uporaba regularizacije Tihonova s parametrom λ za rešitev predoločenega sistema $Ax = b$ ekvivalentna reševanju razširjenega sistema*

$$\begin{bmatrix} I & A \\ A^T & -\lambda^2 I \end{bmatrix} \begin{bmatrix} r \\ x \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix}. \tag{3.11}$$

Dokaz. Iz (3.11) sledi $r + Ax = b$ in $A^T r - \lambda^2 x = 0$. Če iz prve enačbe izrazimo $r = b - Ax$ in vstavimo v drugo, dobimo $A^T b = (A^T A + \lambda^2 I)x$, kar je ravno (3.9). ∎

Z malce računanja lahko ugotovimo, da je podprostor Krilova za matriko iz sistema (3.11) neodvisen od λ in tako za vse vrednosti λ lahko uporabimo bazo, ki jo dobimo z algoritmom 3.17.

Podobno kot prej mora za rešitev veljati

$$\begin{bmatrix} U_{k+1}^T & 0 \\ 0 & V_k^T \end{bmatrix} \left(\begin{bmatrix} I & A \\ A^T & -\lambda^2 I \end{bmatrix} \begin{bmatrix} U_{k+1} t_{k+1} \\ V_k y_k \end{bmatrix} - \begin{bmatrix} b \\ 0 \end{bmatrix} \right) = 0,$$

torej

$$\begin{bmatrix} I & B_k \\ B_k^T & -\lambda^2 I \end{bmatrix} \begin{bmatrix} t_{k+1} \\ y_k \end{bmatrix} = \begin{bmatrix} \beta_1 e_1 \\ 0 \end{bmatrix},$$

kar je ravno razširjen sistem oblike (3.11) za regularizirano rešitev predoločenega sistema $B_k y_k = \beta_1 e_1$. Pri metodi LSQR torej najprej uporabimo algoritem 3.17, nato rešimo sistem

$$(B_k^T B_k + \lambda^2 I) y_k = \beta_1 B_k^T e_1$$

velikosti $(k+1) \times k$ in za končni približek vzamemo $x_k = V_k y_k$. Če potrebujemo rešitev pri drugi vrednosti regularizacijskega parametra, moramo ponoviti le zadnja dva koraka, prvega, ki je računsko najbolj zahteven, pa je dovolj izvesti le enkrat. Za regularizacijo je tako metoda LSQR primernejša od CGLS.

Poglavje 4

Iterativne metode za računanje lastnih vrednosti

4.1 Pomožni rezultati

V tem razdelku je navedenih nekaj rezultatov iz numerične linearne algebre, ki jih bomo potrebovali v nadaljevanju. Za lažje razumevanje so dodani tudi dokazi iz [9].

Kot prvo si poglejmo, kako lahko ocenimo spremembe lastnih vrednosti, če zmotimo matriko. 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 4.1 (Bauer–Fike) *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

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

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

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

Iz

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

sledi

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

■

Opomba 4.1 *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 nadaljevanju razdelka bomo predpostavili, da je matrika A simetrična. V tem primeru lahko pokažemo veliko dodatnih lastnosti. Za začetek 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 $x_1, \dots, x_n$ pa lahko izberemo tako, da tvorijo ortonormirano bazo. Spektralna občutljivost matrike lastnih vektorjev je potem 1 in iz Bauer–Fikeovega izreka dobimo naslednjo posledico.

Posledica 4.2 *Če sta matriki A in E simetrični, potem pri predpostavkah izreka 4.1 za vsako lastno vrednost $\lambda(\epsilon)$ matrike $A + \epsilon E$ velja*

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

Ker imamo pri simetričnih matrikah opravka le z realnimi lastnimi vektorji, v tem primeru *Rayleighov kvocient* računamo le za realne vektorje. Za $x \neq 0$ je

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

in 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 $\sigma = \rho(x, A)$.

Lema 4.3 *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 4.4 (Courant–Fischerjev minimaks izrek) *Če je A simetrična matrika z lastnimi vrednostmi $\lambda_n \le \cdots \le \lambda_1$, potem za $i = 1, \ldots, n$ velja*

$$\lambda_i = \min_{\substack{S \subset \mathbb{R}^n \\ \dim(S) = n-i+1}} \max_{\substack{x \in S \\ x \neq 0}} \rho(x, A) = \max_{\substack{R \subset \mathbb{R}^n \\ \dim(R) = i}} \min_{\substack{x \in R \\ x \neq 0}} \rho(x, A). \tag{4.1}$$

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

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

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 (4.1). Torej

$$\max_{\substack{R \subset \mathbb{R}^n \\ \dim(R) = i}} \min_{\substack{x \in R \\ x \neq 0}} \rho(x, A) = \min_{\substack{x \in \widetilde{R} \\ x \neq 0}} \rho(x, A) \le \rho(x_{\widetilde{R}\widetilde{S}}, A) = \max_{\substack{x \in \widetilde{S} \\ x \neq 0}} \rho(x, A) \le \min_{\substack{S \subset \mathbb{R}^n \\ \dim(S) = n-i+1}} \max_{\substack{x \in S \\ x \neq 0}} \rho(x, A). \tag{4.2}$$

Po drugi strani 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

$$\max_{\substack{R \subset \mathbb{R}^n \\ \dim(R) = i}} \min_{\substack{x \in R \\ x \neq 0}} \rho(x, A) \ge \min_{\substack{x \in \widehat{R} \\ x \neq 0}} \rho(x, A) = \lambda_i = \max_{\substack{x \in \widehat{S} \\ x \neq 0}} \rho(x, A) \ge \min_{\substack{S \subset \mathbb{R}^n \\ \dim(S) = n-i+1}} \max_{\substack{x \in S \\ x \neq 0}} \rho(x, A), \tag{4.3}$$

saj je

$$\max_{\substack{R \subset \mathbb{R}^n \\ \dim(R) = i}} \min_{\substack{x \in R \\ x \neq 0}} \rho(x, A) \ge \min_{\substack{x \in \widehat{R} \\ x \neq 0}} \rho(x, A) qquad \text{in} \qquad \textit{inpodobno} \quad \min_{\substack{S \subset \mathbb{R}^n \\ \dim(S) = n-i+1}} \max_{\substack{x \in S \\ x \neq 0}} \rho(x, A) \le \max_{\substack{x \in \widehat{S} \\ x \neq 0}} \rho(x, A).$$

Iz (4.2) in (4.3) sledi (4.1). ■

Posledica 4.5 *Če sta A in E simetrični matriki in so $\lambda_n \le \cdots \le \lambda_1$ lastne vrednosti matrike A, $\widehat{\lambda}_n \le \cdots \le \widehat{\lambda}_1$ pa lastne vrednosti matrike $A + E$, potem za $i = 1, \ldots, n$ velja*

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

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

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

in uporabimo Courant–Fischerjev minimaks izrek. ■

Od tod sledi t.i. Weylov izrek, ki smo ga kot posledico Bauer–Fikeovega izreka zapisali že v posledici 4.2.

Posledica 4.6 (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 4.7 (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 $p'_1, \ldots, p'_{k-1}$ linearno neodvisni, smo izpustili, 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.

Izrek 4.8 *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

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

∎

Lema 4.9 *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). Potem so vse lastne vrednosti matrike T enostavne, za vsak lastni vektor x pa velja $x_1 \neq 0$ in $x_n \neq 0$.

Dokaz. Zaradi nerazcepnosti je za vsak λ rang matrike $A - \lambda I$ vsaj $n-1$, saj je prvih $n-1$ stolpcev linearno neodvisnih. Zaradi tega nobena lastna vrednost ne more biti večkratna.

Če je x lastni vektor za lastno vrednost λ, potem za elemente x veljajo enačbe

$$\begin{aligned} (a_1 - \lambda)x_1 + b_1 x_2 &= 0 \\ b_{i-1}x_{i-1} + (a_i - \lambda)x_i + b_i x_{i+1} &= 0, \quad i = 2, \ldots, n-1, \\ b_{n-1}x_{n-1} + (a_n - \lambda)x_n &= 0 \end{aligned}$$

Če vstavimo $x_1 = 0$ v prvo enačbo, potem po vrsti sledi $x_i = 0$ za $i = 2 \ldots, n$, torej $x = 0$, kar je protislovje. Podobno, če vstavimo $x_n = 0$ v zadnjo enačbo, potem v obratnem vrstnem redu spet za vse elemente x sledi, da so enaki 0. ∎

4.2 Ritzeve vrednosti

Denimo, da bi radi izračunali lastne vrednosti in lastne vektorje matrike $A \in \mathbb{R}^{n\times n}$. Obstaja več algoritmov, kot je npr. QR iteracija, ki znajo to narediti v pričakovanem času $\mathcal{O}(n^3)$. Čeprav so vsi ti algoritmi v principu iterativni, saj se lastnih vrednosti pač ne da izračunati drugače kot z iterativno metodo, jih bomo obravnavali kot direktne metode. Te metode so primerne za polne matrike in za zmerno velikost n. Delujejo v glavnem tako, da matriko najprej z ortogonalno podobnostno transformacijo pretvorijo v zgornjo Hessenbergovo obliko (za nesimetrične matrike) oz. tridiagonalno obliko (za simetrične matrike). V nadaljevanju uporabimo na zgornji Hessenbergovi (oz. tridiagonalni) matriki enega izmed učinkovitih iterativnih algoritmov, za katerega lahko ocenimo, da v splošnem primeru porabi $\mathcal{O}(n^3)$ operacij.

Če pa imamo dano veliko (razpršeno) matriko, potem si ne moremo privoščiti, da bi izračunali vse njene lastne vrednosti, npr. z uporabo QR iteracije, saj bi na ta način porabili preveč časa in pomnilnika. Namesto tega lahko s pomočjo iterativnih metod podprostorov, ki jih bomo spoznali v tem poglavju, izračunamo nekaj lastnih vrednosti in pripadajočih lastnih vektorjev.

Najpreprostejša iterativna metoda, ki jo lahko uporabimo za izračun manjšega števila lastnih vrednosti, je *potenčna metoda*, s katero lahko izračunamo dominantni lastni par.

Potenčna metoda temelji na tem, da za naključno izbran začetni vektor v_1 vektorji

$$v_j = A^{j-1}v_1 / \|A^{j-1}v_1\|,$$

ko gre j proti neskončnosti, po smeri konvergirajo proti dominantnemu lastnemu vektorju.

Vektorji $v_1, \ldots, v_k$, ki jih dobimo pri potenčni metodi, razpenjajo podprostor Krilova

$$\mathcal{K}_k(A, v_1) = \mathrm{Lin}(v_1, Av_1, \ldots, A^{k-1}v_1) = \{p(A)v_1 : \ p \in \mathbb{P}_{k-1}\},$$

kjer je $\mathbb{P}_{k-1}$ prostor vseh polinomov stopnje manjše ali enake $k-1$. Izkaže se, da v podprostoru $\mathcal{K}_k(A, v_1)$, ki vsebuje še prejšnje vektorje iz potenčne metode, lahko dobimo še boljšo aproksimacijo za dominanten lastni vektor.

Čeprav najpogosteje uporabljamo podprostore Krilova, lahko približke za lastne vrednosti dobimo iz poljubnega podprostora. Denimo, da imamo matriko $A \in \mathbb{C}^{n\times n}$ in podprostor $\mathcal{V}_k$ dimenzije k. Sedaj bi radi poiskali približke za lastne vrednosti in lastne vektorje, povezane s tem podprostorom. Podobno, kot pri reševanju linearnih sistemov, približke za lastne vrednosti in lastne vektorje dobimo iz *Ritz–Galerkinovega pogoja*

$$Az - \mu z \perp \mathcal{V}_k, \quad z \in \mathcal{V}_k,$$

ki pravi, da mora biti ostanek pravokoten na podprostor.

Če stolpci matrike V_k tvorijo ortonormirano bazo za podprostor $\mathcal{V}_k$, potem lahko vektor $z \in \mathcal{V}_k$ zapišemo kot $z = V_k s$ za nek $s \in \mathbb{C}^k$. Ritz–Galerkinov pogoj se tako spremeni v

$$V_k^H A V_k s = \mu s.$$

Če je μ lastna vrednost matrike $H_k = V_k^H A V_k$, potem pravimo, da je μ *Ritzeva vrednost*, pripadajoči vektor $z = V_k s$ pa *Ritzev vektor*. Skupaj tvorita *Ritzev par* (μ, z).

Za nesimetrične matrike lahko uporabimo podprostor Krilova, ki ga zgeneriramo z Arnoldijevim postopkom, ki smo ga zapisali v algoritmu 3.1. Algoritem se konča pri izbranem k ali pa, ko je $h_{j+1,j} = 0$. Če se Arnoldijev postopek izvaja do $j = k$, dobimo

$$AV_k = V_k H_k + h_{k+1,k} v_{k+1} e_k^T = V_{k+1} \widetilde{H}_k,$$

kjer je H_k $k \times k$ zgornja Hessenbergova, stolpci V_k pa tvorijo ortonormirano bazo za $\mathcal{K}_k(A, v_1)$.

Od tod dobimo Arnoldijevo metodo za aproksimacijo lastnih parov. Najprej z Arnoldijevim postopkom zgradimo matriki V_k in H_k, potem pa je (μ, z) Ritzev par, če je $z = V_k s$, kjer je μ lastna vrednost matrike H_k, $s \in \mathbb{C}^k$ pa njen pripadajoči lastni vektor. Za ostanek potem dobimo

$$r = Az - \mu z = h_{k+1,k} v_{k+1} e_k^T s,$$

torej

$$\|r\|_2 = |h_{k+1,k}| \cdot |e_k^T s|.$$

Če naj bo (μ, z) dober približek za lastni par matrike A, mora biti ostanek majhen in to lahko testiramo brez direktnega računanja vektorja z.

V primeru nesimetrične matrike majhen ostanek še ni dovolj dobro zagotovilo, saj za matrike, ki se dajo diagonalizirati kot $A = X\Lambda X^{-1}$, velja Bauer–Fikeov izrek

$$\min_i |\lambda_i - \mu| \leq \|r\| \|X\| \|X^{-1}\|.$$

V primeru, ko je matrika lastnih vektorjev zelo občutljiva, se lahko kljub majhnemu ostanku Ritzeva vrednost dosti razlikuje od lastne vrednosti. Če pa je npr. matrika simetrična, potem majhen ostanek pomeni tudi dobro aproksimacijo, saj je matrika lastnih vektorjev ortogonalna.

Če se Arnoldijev postopek konča s $h_{k+1,k} = 0$, potem je podprostor Krilova $\mathcal{K}_k(A, v_1)$ invarianten za matriko A. To pomeni, da so vsi Ritzevi pari, ki pripadajo $\mathcal{K}_k(A, v_1)$, kar lastni pari matrike A. Kljub temu, da tako izračunamo točne lastne vrednosti, to ni nujno dobrodošel dogodek. Lahko se namreč zgodi, da lastne vrednosti, ki jih dobimo v tem invariantnem podprostoru, niso lastne vrednosti, ki jih iščemo. Ker podprostora ne moremo povečati in tako priti do novih približkov, nam preostane edino to, da ponovno začnemo postopek z novo izbiro začetnega vektorja.

4.2.1 Računanje zunanjih in notranjih lastnih vrednosti

Hitro se lahko prepričamo, da za podprostor Krilova velja

$$\mathcal{K}_k(\alpha A + \beta I, v_1) = \mathcal{K}_k(A, v_1)$$

za poljuben β in $\alpha \neq 0$. Torej dobimo isti podprostor Krilova ne glede na to, če matriko pomnožimo s skalarjem oziroma uporabimo premik. To se odraža tudi na Ritzevih vrednostih. Ker so metode podprostorov Krilova invariantne na skaliranje in premike, je položaj lastnih vrednosti glede na koordinatno izhodišče nepomemben. Namesto tega je pomembno, katere lastne vrednosti so zunanje in katere notranje.

Če vzamemo najmanjši krog oz. elipso, ki vsebuje vse lastne vrednosti, potem lahko pričakujemo, da bo, za naključno izbrane začetne vektorje, Arnoldijeva metoda najprej skonvergirala k tistim lastnim vrednostim, ki so blizu roba tega kroga. To je podobno kot pri potenčni metodi, kjer dobimo dominantno lastno vrednost.

To pomeni, da Arnoldijeva metoda v obliki, kot je, ni primerna za računanje notranjih lastnih vrednosti. Kadar iščemo lastne vrednosti v bližini izbranega cilja $\tau \in \mathbb{C}$, obstaja nekaj pristopov, kako lahko metode podprostorov pripravimo do tega, da konvergirajo k notranjim lastnim vrednostim.

Prva varianta je *premakni-in-obrni Arnoldi* (shift-and-invert Arnoldi), kjer vzamemo podprostor Krilova, ki ga generira matrika $(A - \tau I)^{-1}$, kjer je $\tau \in \mathbb{C}$ dani cilj. To pomeni, da moramo pri tej metodi v vsakem koraku izračunati produkt z matriko $(A - \tau I)^{-1}$. Ta produkt moramo izračunati točno in ne le približno rešiti sistem s kakšno iterativno metodo. Pri tej metodi lahko pričakujemo, da bomo najprej izračunali lastne vrednosti, ki so najbližje τ.

Posplošitev tega pristopa je *racionalna Arnoldijeva metoda*. Denimo, da nas zanimajo lastne vrednosti blizu točk $\tau_1, \ldots, \tau_m$. Sedaj najprej zgradimo podprostor Krilova za matriko $(A - \tau_1 I)^{-1}$ in izračunamo lastne vrednosti v okolici τ_1. Potem gremo na τ_2, a ne zavržemo podprostora. Obstaja transformacija, s katero lahko podprostor spremenimo v podprostor generiran z $(A - \tau_2 I)^{-1}$.

Denimo, da za matriko A iščemo približke za lastne vektorje v prostoru $\mathcal{V}_k$, zahtevamo pa, da bo ostanek pravokoten na prostor $\mathcal{W}_k$. V tem primeru približke za lastne vrednosti in lastne vektorje dobimo iz *Petrov–Galerkinovega pogoja*

$$Az - \mu z \perp \mathcal{W}_k, \quad z \in \mathcal{V}_k.$$

Če sta V_k in W_k matriki z ortonormiranimi stolpci, ki razpenjajo *iskalni podprostor* $\mathcal{V}_k$ in *testni podprostor* $\mathcal{W}_k$, potem je μ lastna vrednost posplošenega problema lastnih vrednosti velikosti $k \times k$

$$W_k^H A V_k s = \mu W_k^H V_k s.$$

Sedaj pravimo, da je μ *vrednost Petrova*, $z = V_k s$ pa *vektor Petrova*.

Ena možnost je, da iskalni in testni podprostor izberemo tako, da sta V_k in W_k biortogonalni bazi, kar pomeni $W_k^H V_k = I$. Takšni bazi lahko skonstruiramo z dvostranskim Lanczosevim algoritmom, ki je zapisan v algoritmu 3.13. Potem so vrednosti Petrova kar lastne vrednosti matrike $W_k^H A V_k$, ki je tridiagonalna.

Pri *harmoničnih Ritzevih vrednostih* za testni podprostor $\mathcal{W}_k$ izberemo $A\mathcal{V}_k$. Iz pogoja

$$Az - \theta z \perp A\mathcal{V}_k, \quad z \in \mathcal{V}_k$$

dobimo posplošeni problem lastnih vrednosti

$$V_k^H A^H A V_k s - \theta V_k^H A^H V_k s. \tag{4.4}$$

Denimo, da stolpci V_k sestavljajo bazo za $\mathcal{V}_k$, ki jo izberemo tako, da stolpci $W_k = AV_k$ sestavljajo ortonormirano bazo za $A\mathcal{V}_k$. Potem se posplošeni problem (4.4) spremeni v

$$V_k^H A^H V_k s = \frac{1}{\theta} s = W_k^H A^{-1} W_k s.$$

To pomeni, da je θ^{-1} Ritzeva vrednost za matriko A^{-1} glede na podprostor $A\mathcal{V}_k$. V tem primeru θ imenujemo *harmonična Ritzeva vrednost* in pričakujemo, da bo θ dober približek za notranje lastne vrednosti. Za razliko od premakni-in-obrni Arnoldijeve metode, tu izračun inverza oz. reševanje sistema z matriko A^{-1} ni nikoli potreben, saj je $W_k^H V_k s = \theta^{-1} s$.

Če iščemo lastne vrednosti v bližini cilja τ, potem dobimo harmonične Ritzeve vrednosti in vektorje iz posplošenega problema lastnih vrednosti

$$V_k^H (A - \tau I)^H (A - \tau I) V_k s = (\theta - \tau) V_k^H (A - \tau I)^H V_k s.$$

4.3 Lanczosev algoritem za lastne vrednosti

V prejšnjem razdelku smo pogledali možnosti, ki jih imamo, da iz podprostora dobimo približke za lastne vrednosti in lastne vektorje. V tem razdelku bomo podrobneje pogledali možnosti, ki jih imamo v primeru simetrične matrike, če za iskalni podprostor uporabimo podprostor Krilova.

Če je A simetrična, se simetričnost pri Arnoldijevem algoritmu prenese na zgornjo Hessenbergovo matriko, ki postane simetrična in tridiagonalna. Arnoldijev algoritem se tako spremeni v Lanczosev algoritem, zapisan v algoritmu 3.6, katerega rezultat je

$$AV_k = V_k T_k + \beta_k v_{k+1} e_k^T,$$

kjer je

$$T_k = \begin{bmatrix} \alpha_1 & \beta_1 & & \\ \beta_1 & \alpha_2 & \ddots & \\ & \ddots & \ddots & \beta_{k-1} \\ & & \beta_{k-1} & \alpha_k \end{bmatrix}.$$

Lastne vrednosti matrike T_k so Ritzeve vrednosti. Ker je matrika T_k simetrična, so seveda vse Ritzeve vrednosti realne, kakor velja tudi za lastne vrednosti matrike A.

Če Lanczosevo metodo primerjamo z Arnoldijevo, potem za Lanczosevo metodo velja:

- časovna zahtevnost in prostorske zahteve so manjše kot pri Arnoldiju,
- več se da povedati o konvergenci,
- do izgube ortogonalnosti pride hitreje kot pri Arnoldiju,
- ne moremo ugotoviti večkratnosti lastne vrednosti,

- lahko imamo težave z navidezno konvergenco,
- pojavijo se prividi lastnih vrednosti.

Nezmožnost detektiranja večkratnih lastnih vrednosti sledi iz posledice **??**. Vse matrike T_k, ki jih vrne Lanczoseva metoda, so nerazcepne, torej so vse Ritzeve vrednosti enostavne.

Ritzeve vrednosti v koraku k se prepletajo z Ritzevimi vrednostmi iz koraka $k+1$, saj se lastne vrednosti T_k prepletajo z lastnimi vrednostmi T_{k+1}. Če so $\theta_k^{(k)} < \cdots < \theta_1^{(k)}$ Ritzeve vrednosti, ki jih dobimo v k-tem koraku Lanczosevega algoritma, potem velja strogo prepletanje

$$\theta_{k+1}^{(k+1)} < \theta_k^{(k)} < \theta_k^{(k+1)} < \cdots < \theta_2^{(k+1)} < \theta_1^{(k)} < \theta_1^{(k+1)}.$$

Zaradi tega Ritzeve vrednosti monotono konvergirajo proti lastnim vrednostim matrike A, pri čemer najprej skonvergirajo zunanje lastne vrednosti.

Izrek 4.10 *Za Ritzeve vrednosti, ki jih dobimo v k-tem koraku Lanczosevega algoritma, velja:*

1. *Obstaja k lastnih vrednosti matrike A, da je* $|\theta_i^{(k)} - \lambda_i| \le \beta_k$ *za* $i = 1, \ldots, k$.
2. *Če je* $z = V_k s$ *Ritzev vektor za Ritzevo vrednost* θ*, potem velja*
$$\min_i |\lambda_i - \theta| \le \|Az - \theta z\|_2 = \beta_k |e_k^T s|.$$
3. $\theta_i^{(k)} = \max\limits_{\dim(S)=i} \min\limits_{0 \neq x \in S \subset \mathcal{K}_k(A, v_1)} \rho(A, x)$.

4.3.1 Ocene o hitrosti konvergence

Naj bodo $\lambda_n \le \cdots \le \lambda_1$ lastne vrednosti matrike A in $x_n, \ldots, x_1$ pripadajoči ortonormirani lastni vektorji. Najprej poglejmo ocene o hitrosti konvergence približkov za lastne vektorje. Merilo za konvergenco je velikost kota med lastnim vektorjem in iskalnim podprostorom Krilova, ki ga označimo s $\varphi(x_i, \mathcal{K}_k(A, v_1))$.

V naslednji lemi bomo uporabili spektralni projektor. Če je Γ zaključena krivulja v $\mathbb{C}$, potem je

$$P_\Gamma(A) := \oint_\Gamma (zI - A)^{-1} dz$$

spektralni projektor, ki pripada lastnim vrednostim matrike A, ki ležijo znotraj krivulje Γ. Če je λ_i enostavna lastna vrednost in Γ_i vsebuje samo lastno vrednost λ_i, potem je $P_i = x_i y_i^H$, kjer je x_i normiran levi in y_i normiran desni lastni vektor za lastno vrednost λ_i. V primeru simetrične matrike je $x_i = y_i$ in $P_i = x_i x_i^T$.

Lema 4.11 *Če je* $P_i = x_i x_i^T$ *spektralni projektor, ki pripada enostavni lastni vrednosti* λ_i *matrike A, potem v primeru* $P_i v_1 \neq 0$ *velja*

$$\tan \varphi(x_i, \mathcal{K}_k(A, v_1)) = \min_{p \in \mathbb{P}_{k-1},\ p(\lambda_i)=1} \|p(A) y_i\| \tan \varphi(x_i, v_1),$$

kjer je

$$y_i = \begin{cases} \frac{(I-P_i)v_1}{\|(I-P_i)v_1\|}, & v\ primeru\ (I-P_i)v_1 \neq 0 \\ \\ 0, & sicer. \end{cases}$$

Dokaz. Za poljuben vektor $z \in \mathcal{K}_k(A, v_1)$ obstaja polinom $q \in \mathbb{P}_{k-1}$, da je $z = q(A)v_1$. Pišemo lahko

$$z = q(A)v_1 = q(A)(P_i v_1 + (I - P_i)v_1) = q(A)P_i v_1 + q(A)(I - P_i)v_1,$$

pri čemer sta vektorja v zadnji vsoti ortogonalna, vektor $q(A)P_i v_1$ pa je kolinearen z lastnim vektorjem x_i. Od tod sledi, da je

$$\tan\varphi(x_i, z) = \frac{\|q(A)(I-P_i)v_1\|}{\|q(A)P_i v_1\|} = \frac{\|q(A)y_i\|\,\|(I-P_i)v_1\|}{|q(\lambda_i)|\,\|P_i v_1\|} = \frac{\|q(A)y_i\|}{|q(\lambda_i)|}\tan\varphi(x_i, v_1).$$

Za polinom $p(\lambda) = q(\lambda)/q(\lambda_i)$ velja, da je $p(\lambda_i) = 1$, njegova stopnja pa je manjša ali enaka $k-1$. Tako dobimo

$$\tan\varphi(x_i, z) = \|p(A)y_i\|\tan\varphi(x_i, v_1), \tag{4.5}$$

oceno iz leme pa dobimo, ko vzamemo minimum po vseh polinomih $p \in \mathbb{P}_{k-1}$, za katere velja $p(\lambda_i) = 1$. ∎

Izrek 4.12 *Pri enakih predpostavkah kot zgoraj velja*

$$\tan\varphi(x_i, \mathcal{K}_k(A, v_1)) \leq \frac{\kappa_i}{T_{k-i}(1+2\gamma_i)}\tan\varphi(x_i, v_1),$$

kjer je

$$\kappa_1 = 1, \quad \kappa_i = \prod_{j=1}^{i-1}\frac{\lambda_j - \lambda_n}{\lambda_j - \lambda_i}\ \ za\ i > 1, \quad in \quad \gamma_i = \frac{\lambda_i - \lambda_{i+1}}{\lambda_{i+1} - \lambda_n}.$$

Dokaz. Izrek bomo najprej dokazali za primer $i = 1$. V tem primeru vektor y_1 iz prejšnje leme razvijemo po bazi lastnih vektorjev kot

$$y_1 = \sum_{j=2}^{n}\alpha_j x_j, \quad \sum_{j=2}^{n}\alpha_j^2 = 1.$$

Od tod sledi

$$\|p(A)y_1\|^2 = \sum_{j=2}^{n}p(\lambda_j)^2\alpha_j^2 \leq \max_{j=2,\ldots,n}p(\lambda_j)^2 \leq \max_{\lambda\in[\lambda_n,\lambda_2]}p(\lambda)^2.$$

Sedaj uporabimo oceno (4.5). Potrebujemo polinom stopnje $k-1$, za katerega velja $p(\lambda_1) = 1$ in ima najmanjšo neskončno normo na intervalu $[\lambda_n, \lambda_2]$. Rešitev dobimo s pomočjo polinomov Čebiševa.

Če imamo tri realna števila $\alpha < \beta \leq \gamma$, potem je

$$\min_{p\in\mathbb{P}_m,\ p(\gamma)=1}\ \max_{t\in[\alpha,\beta]}|p(t)|$$

dosežen pri polinomu

$$\widehat{T}_m(t) = \frac{T_m\left(1+2\frac{t-\beta}{\beta-\alpha}\right)}{T_m\left(1+2\frac{\gamma-\beta}{\beta-\alpha}\right)},$$

kjer je T_m polinom Čebiševa.

V našem primeru vzamemo $\alpha = \lambda_n$, $\beta = \lambda_2$ in $\gamma = \lambda_1$. Tako dobimo

$$\|p(A)y_1\| \leq \frac{1}{T_{k-1}\left(1 + 2\frac{\lambda_1 - \lambda_2}{\lambda_2 - \lambda_n}\right)} = \frac{1}{T_{k-1}(1 + 2\gamma_1)}.$$

Za $i > 1$ vzamemo polinom oblike

$$p(\lambda) = \frac{(\lambda_1 - \lambda)\cdots(\lambda_{i-1} - \lambda)}{(\lambda_1 - \lambda_i)\cdots(\lambda_{i-1} - \lambda_i)} q(\lambda),$$

kjer je $q \in \mathbb{P}_{k-i}$ in $q(\lambda_i) = 1$. Potem lahko ocenimo

$$\|p(A)y_1\| \leq \max_{\lambda \in [\lambda_n, \lambda_{i+1}]} \left| \prod_{j=1}^{i-1} \frac{\lambda_j - \lambda}{\lambda_j - \lambda_i} q(\lambda) \right| \leq \prod_{j=1}^{i-1} \frac{\lambda_j - \lambda_n}{\lambda_j - \lambda_i} \max_{\lambda \in [\lambda_n, \lambda_{i+1}]} |q(\lambda)|$$

in, podobno kot pri $i = 1$, do končne ocene pridemo s pomočjo polinomov Čebiševa. ■

Poglejmo, kaj lahko povemo o konvergenci Ritzevih vrednosti proti lastnim vrednostim. Naj bodo $\theta_k^{(k)} \leq \cdots \leq \theta_1^{(k)}$ Ritzeve vrednosti matrike A glede na podprostor Krilova $\mathcal{K}_k(A, v_1)$. Ker vemo, da so Ritzeve vrednosti Rayleighovi kvocienti Ritzevih vektorjev, lahko pričakujemo, da v oceni za napako Ritzeve vrednosti $\theta_1^{(k)}$ nastopa kvadrat $\tan \varphi(x_1, v_1)^2$, o čemer govori naslednja lema.

Lema 4.13 *Velja*

$$0 \leq \lambda_1 - \theta_1^{(k)} \leq (\lambda_1 - \lambda_n) \min_{\substack{p \in \mathbb{P}_{k-1} \\ p(\lambda_i) = 1}} \max_{\lambda \in [\lambda_n, \lambda_2]} |p(\lambda)|^2 \tan \varphi(x_i, v_1)^2.$$

Dokaz. Prva neenakost $0 \leq \lambda_1 - \theta_1^{(k)}$ je dokaj očitna, saj je

$$\theta_1^{(k)} = \max_{\substack{x \in \mathcal{K}_k \\ x \neq 0}} \rho(A, x) \leq \max_{\substack{x \in \mathbb{R}^n \\ x \neq 0}} \rho(A, x) = \lambda_1,$$

kjer je $\mathcal{K}_k = \mathcal{K}_k(A, v_1)$.

Za drugo neenakost najprej razliko $\lambda_1 - \theta_1^{(k)}$ zapišemo v obliki

$$\lambda_1 - \theta_1^{(k)} = \min_{\substack{x \in \mathcal{K}_k \\ x \neq 0}} \rho(\lambda_1 I - A, x) = \min_{\substack{q \in \mathbb{P}_{k-1} \\ q \not\equiv 0}} \rho(\lambda_1 I - A, q(A)v_1).$$

Če začetni vektor v_1 razvijemo po lastnih vektorjih kot $v_1 = \sum_{j=1}^n \alpha_j x_j$, potem dobimo

$$\lambda_i - \theta_1^{(k)} = \min_{\substack{q \in \mathbb{P}_{k-1} \\ q \not\equiv 0}} \frac{\sum_{j=2}^n (\lambda_1 - \lambda_j)\alpha_j^2 q(\lambda_j)^2}{\sum_{j=1}^n \alpha_j^2 q(\lambda_j)^2}.$$

To lahko ocenimo kot

$$\begin{aligned}
\lambda_1 - \theta_1^{(k)} &\leq (\lambda_1 - \lambda_n) \min_{\substack{q \in \mathbb{P}_{k-1} \\ q \neq 0}} \frac{\sum_{j=2}^n \alpha_j^2 q(\lambda_j)^2}{\sum_{j=1}^n \alpha_j^2 q(\lambda_j)^2} \\
&\leq (\lambda_1 - \lambda_n) \min_{\substack{q \in \mathbb{P}_{k-1} \\ q \neq 0}} \frac{\sum_{j=2}^n \alpha_j^2 q(\lambda_j)^2}{\alpha_1^2 q(\lambda_1)^2} \\
&\leq (\lambda_1 - \lambda_n) \min_{\substack{q \in \mathbb{P}_{k-1} \\ q \neq 0}} \max_{j=2,\ldots,n} \frac{q(\lambda_j)^2}{q(\lambda_1)^2} \frac{\sum_{j=2}^n \alpha_j^2}{\alpha_1^2} \\
&\leq (\lambda_1 - \lambda_n) \min_{\substack{q \in \mathbb{P}_{k-1} \\ p(\lambda_1)=1}} \max_{\lambda \in [\lambda_n, \lambda_2]} p(\lambda)^2 \tan \varphi(x_1, v_1)^2.
\end{aligned}$$

Izrek 4.14 *Velja*

$$0 \leq \lambda_i - \theta_i^{(k)} \leq (\lambda_1 - \lambda_n) \left(\frac{\kappa_i^{(k)} \tan \varphi(x_i, v_1))}{T_{k-i}(1 + 2\gamma_i)} \right)^2,$$

kjer je

$$\kappa_1^{(k)} = 1, \quad \kappa_i^{(k)} = \prod_{j=1}^{i-1} \frac{\theta_j^{(k)} - \lambda_n}{\theta_j^{(k)} - \lambda_i} \ \ za\ i > 1 \quad in \quad \gamma_i = \frac{\lambda_i - \lambda_{i+1}}{\lambda_{i+1} - \lambda_n}.$$

Zgornje ocene nam zagotavljajo konvergenco, a za praktično uporabo niso primerne, saj v njih nastopajo količine, ki jih ne poznamo. Praktične ocene, ki jih lahko uporabimo in so v večini primerov tudi zelo dobre, uporabljajo Ritzeve vrednosti namesto lastnih vrednosti. Naj bodo $\theta_m^{(m)} \leq \cdots \leq \theta_1^{(m)}$ Ritzeve vrednosti matrike A glede na podprostor $\mathcal{K}_m(A, v_1)$ in $u_m, \ldots, u_1$ pripadajoči Ritzevi vektorji, kjer je $u_i = V_m y_i$.

Označimo ostanek $r_j = Au_j - \theta_j^{(m)} u_j$. Vemo že, da velja $\min_i |\lambda_i - \theta_j^{(m)}| \leq \|r_j\|_2 = \beta_m |e_m^T y_j|$. Če je $\theta_i^{(m)}$ približek za λ_i, potem definiramo $\mathrm{gap}(\theta_i^{(m)}) = \min_{j \neq i} |\theta_i^{(m)} - \lambda_j|$. Velja (podrobna izpeljava je v [8])

$$|\lambda_i - \theta_i^{(m)}| \leq \frac{\|r_i\|_2^2}{\mathrm{gap}(\theta_i^{(m)})}$$

in

$$\sin \varphi(u_i, x_i) \leq \frac{\|r_i\|_2}{\mathrm{gap}(\theta_i^{(m)})}.$$

Da lahko oceno uporabimo, potrebujemo še oceno za $\mathrm{gap}(\theta_i^{(m)})$. Tu uporabimo predpostavko, da sta, kadar je $\theta_i^{(m)}$ dober približek za lastno vrednost λ_i, vrednosti $\theta_{i-1}^{(m)}$ in $\theta_{i+1}^{(m)}$ prav tako dobra približka za λ_{i-1} in λ_{i+1}. Tako lahko $\mathrm{gap}(\theta_i^{(m)})$ ocenimo z

$$\mathrm{gap}(\theta_i^{(m)}) \approx \min_{j \neq i} (|\theta_i^{(m)} - \theta_j^{(m)}| - \|r_j\|),$$

kamor smo vključili še oceno, da je $\min_i |\lambda_i - \theta_j^{(m)}| \leq \|r_j\|_2$.

4.3.2 Težave z ortogonalnostjo

Če izvajamo osnovno Lanczosevo metodo brez reortogonalizacije, potem lahko pride do izgube ortogonalnosti in dobimo navidezne večkratne lastne vrednosti. Opazimo, da se ortogonalnost baznih vektorjev za podprostor Krilova poslabša takrat, ko kak izmed Ritzevih vektorjev skonvergira do lastnega vektorja.

Izrek 4.15 (Paige) *Če izvajamo Lanczosev algoritem v aritmetiki z osnovno zaokrožitveno napako u, potem v k-tem koraku za izračunane Ritzeve vektorje $u_1, \ldots, u_k$ velja*

$$u_i^T v_{k+1} = \frac{\mathcal{O}(u\|A\|)}{\|r_i\|_2},$$

kjer je $r_i = Au_i - \theta_i u_i$, oziroma

$$u_i^T v_{k+1} = \frac{\mathcal{O}(u\|A\|)}{\beta_k |e_k^T y_i|},$$

kjer je $u_i = V_k y_i$, za $i = 1, \ldots, k$

Dokaz je v knjigi [3], temelji pa na formuli

$$\widetilde{\beta}_j \widetilde{v}_{j+1} + f_j = (A - \widetilde{\alpha}_j I)\widetilde{v}_j - \widetilde{\beta}_{j-1}\widetilde{v}_{j-1},$$

kjer je $\|f_j\| = \mathcal{O}(\|A\|u)$ posledica zaokrožitvenih napak.

Izkaže se, da imamo pri osnovni različici Lanczoseve metode, ki je zapisana v algoritmu 3.6, velike težave z izgubo ortogonalnosti. Ker stolpci matrike V_k niso tako ortogonalni, kot bi morali biti, se v matriki T_k lahko pojavijo navidezne večkratne lastne vrednosti. V praksi sta lahko tako npr. za dovolj velik k tako $\theta_1^{(k)}$ kot $\theta_2^{(k)}$ približka za λ_1, $\theta_3^{(k)}$ pa je približek za λ_2.

Temu se lahko izognemo s polno reortogonalizacijo, kjer nov vektor ortogonaliziramo na vse stolpce matrike V_k. To moramo po potrebi narediti dvakrat, da bo zadeva stabilna. Postopek je zapisan v algoritmu 4.1.

Algoritem 4.1 Lanczoseva metoda s polno reortogonalizacijo. Vhodni podatki so simetrična matrika A, vektor v_1 in dimenzija k. Algoritem vrne ortonormirane stolpce $v_1, \ldots, v_k$, ki tvorijo ortonormirano bazo za podprostor Krilova $\mathcal{K}_k(A, v_1)$.

---

izberi začetni vektor v_1 z $\|v_1\|_2 = 1$
$j = 1, 2, \ldots, k$
  $z = Av_j$
  $\alpha_j = v_j^T z$
  $\xi = \|z\|_2$
  $z = z - \sum_{\ell=1}^{j} (z^T v_\ell) v_\ell$
  če je $\|z\|_2 < 0.7\xi$, potem
    $z = z - \sum_{\ell=1}^{j} (z^T v_\ell) v_\ell$
  $\beta_j = \|z\|_2$
  če je $\beta_j = 0$, potem prekini računanje
  $v_{j+1} = z/\beta_j$

Izkaže se, da ni potrebno, da nove smeri ortogonaliziramo na vse prejšnje. Teoretično tega seveda (razen za zadnji dve smeri) sploh ni potrebno narediti, saj bi vektorji avtomatično morali biti ortogonalni. Pri selektivni reortogonalizaciji nov vektor ortogonaliziramo le na nekaj vektorjev, saj si časovno ne moremo privoščiti, da bi ortogonalizirali na vse prejšnje vektorje. Iz Paigeovega izreka sledi, da lahko pričakujemo veliko napako pri vektorjih, kjer je vrednost $e_k^T y_i$ blizu nič. To uporabimo kot kriterij pri selektivni ortogonalizaciji, zapisani v algoritmu 4.2. Pri tem vektorji $u_{i,k}$, ki nastopajo v algoritmu 4.2, predstavljajo Ritzeve vektorje za Ritzeve vrednosti $\theta_i^{(k)}$.

Algoritem 4.2 Lanczoseva metoda s selektivno reortogonalizacijo. Vhodni podatki so simetrična matrika A, vektor v_1 in dimenzija k. Algoritem vrne ortonormirane stolpce $v_1, \ldots, v_k$, ki tvorijo ortonormirano bazo za podprostor Krilova $\mathcal{K}_k(A, v_1)$.

izberi začetni vektor v_1 z $\|v_1\|_2 = 1$, $\beta_0 = 0$, $v_0 = 0$
$j = 1, 2, \ldots, k$
  $z = Av_j$
  $\alpha_j = v_j^T z$
  $z = z - \alpha_j v_j - \beta_{j-1} v_{j-1}$
  izračunaj Ritzeve vektorje $u_1, \ldots, u_j$
  za vse $i = 1, \ldots, j$, kjer je $\beta_j |e_j^T y_i| \leq \sqrt{u}\|T_j\|$
    $z = z - (u_i^T z) u_i$
  $\beta_j = \|z\|_2$
  če je $\beta_j = 0$, potem prekini računanje
  $v_{j+1} = z/\beta_j$

4.3.3 Harmonične Ritzeve vrednosti

Za računanje notranjih lastnih vrednosti lahko uporabimo harmonične Ritzeve vrednosti. Vemo, da je θ harmonična Ritzeva vrednost in neničelni vektor $w \in A\mathcal{K}_k(A, v_1)$ pripadajoči harmonični Ritzev vektor, če je

$$A^{-1}w - \theta^{-1}w \perp A\mathcal{K}_k(A, v_1). \tag{4.6}$$

Harmonične Ritzeve vrednosti so namreč inverzi Ritzevih vrednosti matrike A^{-1} glede na podprostor $A\mathcal{K}_k(A, v_1)$. Ekvivalenten pogoj za harmonično Ritzevo vrednost θ je, da obstaja neničelen vektor $u \in \mathcal{K}_k(A, v_1)$, da je

$$Au - \theta u \perp A\mathcal{K}_k(A, v_1).$$

Lema 4.16 *Po k korakih Lanczosa dobimo* $AV_k = V_k T_k + \beta_k v_{k+1} e_k^T = V_{k+1}\widetilde{T}_k$. *Za harmonično Ritzevo vrednost* θ *velja, da je lastna vrednost posplošenega problema lastnih vrednosti*

$$\widetilde{T}_k^T \widetilde{T}_k y = \theta T_k y. \tag{4.7}$$

Dokaz. Če zapišemo (4.6), pri čemer zapišemo $w = AV_k y$, dobimo

$$(AV_k)^T (A^{-1} AV_k y - \theta^{-1} AV_k y) = 0.$$

To se zmnoži v

$$V_k^T AV_k y - \theta^{-1} V_k^T A^2 V_k y = 0$$

in naprej v

$$T_k y - \theta^{-1}(V_{k+1}\widetilde{T}_k)^T(V_{k+1}\widetilde{T}_k)y = 0$$

in

$$T_k y - \theta^{-1}\widetilde{T}_k^T\widetilde{T}_k y = 0,$$

od tod pa že sledi (4.7). ■

Če uporabimo premik σ, je θ harmonična Ritzeva vrednost, če obstaja neničelni vektor $z \in \mathcal{K}_k(A, v_1)$, da je

$$Az - \theta z \perp (A - \sigma I)\mathcal{K}_k(A, v_1).$$

θ je harmonična Ritzeva vrednost za premik σ, če je $(\theta - \sigma)^{-1}$ Ritzeva vrednost $(A - \sigma I)^{-1}$ glede na podprostor $(A - \sigma I)\mathcal{K}_k(A, v_1)$.

To je ekvivalentno $V_k^T(A - \sigma I)^2 V_k y = (\theta - \sigma)V_k^T(A - \sigma I)V_k y$ oziroma

$$(\theta - \sigma)^{-1}(\widetilde{T}_k^T\widetilde{T}_k - 2\sigma T_k + \sigma^2 I)y = (T_k - \sigma I)y.$$

Nekaj nam o konvergenci harmoničnih Ritzevih vrednosti povedo t.i. *Lehmannovi intervali*. Naj bo σ izbrana točka, v okolici katere iščemo lastne vrednosti. Lastne vrednosti A, ki naj bodo vse enostavne, uredimo kot

$$\lambda_{-r} < \cdots < \lambda_{-1} < \sigma < \lambda_1 < \cdots < \lambda_{n-r}.$$

Podobno uredimo in indeksiramo tudi harmonične Ritzeve vrednosti. Če je $\theta_j^{(k)}$ harmonična Ritzeva vrednost glede na podprostor $(A - \sigma I)\mathcal{K}_k(A, v_1)$, potem preko minimaks izreka za matriko $(A - \sigma I)^{-1}$ sledi

$$\theta_{-1}^{(k)} < \lambda_{-1} \qquad \text{in} \qquad \lambda_1 < \theta_1^{(k)}.$$

Izrek 4.17 (Lehmann) *Naj bodo*

$$\theta_{-s}^{(k)} < \cdots < \theta_{-1}^{(k)} < \sigma < \theta_1^{(k)} < \cdots < \theta_{k-s}^{(k)}$$

harmonične Ritzeve vrednosti matrike A glede na podprostor $(A - \sigma I)\mathcal{K}_k(A, v_1)$. Vsak interval $[\sigma, \theta_j^{(k)}]$, kjer je $j = 1, \ldots, k - s$, vsebuje vsaj j lastnih vrednosti matrike A. Enako, vsak interval $[\theta_{-j}^{(k)}, \sigma]$, kjer je $j = 1, \ldots, s$, vsebuje vsaj j lastnih vrednosti matrike A.

Ko k narašča, harmonična Ritzeva vrednost $\theta_{-j}^{(k)}$ monotono naraščajoče konvergira k λ_{-j} in podobno zaporedje $\theta_j^{(k)}$ monotono padajoče konvergira k λ_j

4.4 Arnoldijev algoritem za lastne vrednosti

Za nesimetrične matrike podprostor Krilova zgeneriramo z Arnoldijevim postopkom, ki smo ga že zapisali v algoritmu 3.1. Če se Arnoldijev postopek izvaja do koraka k, dobimo

$$AV_k = V_k H_k + h_{k+1,k} v_{k+1} e_k^T = V_{k+1}\widetilde{H}_k,$$

kjer je H_k zgornja Hessenbergova matrika velikosti $k \times k$, stolpci matrike V_k pa tvorijo ortonormirano bazo za podprostor Krilova $\mathcal{K}_k(A, v_1)$.

Podobno, kot pri Lanczosevi metodi, je tudi pri Arnoldijevi metodi (μ, z) Ritzev par, če velja $z = V_k s$, kjer je μ lastna vrednost matrike H_k, s pa njen pripadajoči lastni vektor. Za ostanek potem dobimo

$$r = Az - \mu z = h_{k+1,k} v_{k+1} e_k^T s,$$

torej

$$\|r\| = |h_{k+1,k}| \cdot |e_k^T s|.$$

Če naj bo (μ, z) dober približek za lastni par matrike A, mora biti ostanek majhen in to lahko testiramo brez računanja približka za lastni vektor z.

V primeru nesimetrične matrike majhen ostanek še ni nujno dovolj, saj za matrike, ki se dajo diagonalizirati kot $A = X\Lambda X^{-1}$, velja Bauer-Fikeov izrek

$$\min_i |\lambda_i - \mu| \leq \|r\| \|X\| \|X^{-1}\|.$$

Če je matrika lastnih vektorjev zelo numerično občutljiva, potem lahko pričakujemo, da Ritzeve vrednosti ne aproksimirajo tako dobro lastnih vrednosti.

Za razliko od Lanczoseve metode pri Arnoldiju nimamo tako lepih rezultatov o konvergenci. Težavo predstavlja med drugim dejstvo, da nimamo na voljo minimaks izreka, prav tako pa nimamo ne prepletanja Ritzevih vrednosti ne monotone konvergence.

Poglejmo nekaj zanimivih zgledov.

Zgled 4.1 *Vzamemo matriko*

$$A = [e_2\ e_3\ \cdots\ e_n\ e_1],$$

ki ima ortonormirane stolpce, lastne vrednosti pa so enakomerno razporejene po enotski krožnici v $\mathbb{C}$.

Če za začetni vektor vzamemo $v_1 = e_1$, *potem dobimo* $v_2 = Av_1 = e_2$, $v_3 = Av_2 = e_3$, $\ldots$, $v_n = e_n$. *Za* $k < n$ *je* $H_k = A(1:k, 1:k)$, *kar pomeni, da so vse Ritzeve vrednosti enake* 0. *Šele v koraku* $k = n$ *dobimo neničelne Ritzeve vrednosti, ki se seveda ujemajo z lastnimi vrednostmi matrike* A.

Konvergenca je torej očitno zelo slaba, a to ni posledica tega, da bi bila matrika lastnih vektorjev X *zelo občutljiva, saj je* $\|X\|\|X^{-1}\| = 1$.

Vemo, da za podprostor Krilova velja

$$\mathcal{K}_k(\alpha A + \beta I, v_1) = \mathcal{K}_k(A, v_1)$$

za poljuben β in $\alpha \neq 0$. Torej dobimo isti podprostor Krilova ne glede na to, če matriko pomnožimo s skalarjem oziroma uporabimo pomik. To pomeni, da lahko podobne težave pričakujemo v primeru, ko so lastne vrednosti enakomerno razporejene na poljubni krožnici v kompleksni ravnini. V tem primeru nobena lastna vrednost ne izstopa, metoda pa konvergira dobro ravno proti zunanjim lastnim vrednostim, ki so dobro separirane od ostalih.

Za konvergenco je pomembno tudi, da je matrika čim bolj blizu normalni matriki. Realna matrika A je normalna, če je $A^T A - A^T A = 0$. Obstaja nekaj ekvivalentnih lastnosti, kot so npr.:

1. Matrika A je normalna.
2. $\|A\|_F^2 = \sum_{i=1}^n |\lambda_i|^2$.
3. $A = X\text{diag}(\lambda_i)X^{-1}$ za neko nesingularno matriko X, da je $\|X\|_2\|X^{-1}\|_2 = 1$.
4. Singularne vrednosti so absolutne lastne vrednosti, oziroma $|\lambda_i| = \sigma_i$ za $i = 1, \ldots, n$.

Na podlagi tega lahko definiramo različne mere za to, koliko je matrika daleč od normalnosti, npr.:

1. $\mu_1(A) = \min\{\|A - F\| \mid F \text{ normalna}\}$,
2. $\mu_2(A) = \|A^TA - AA^T\|$,
3. $\mu_3(A) = \left(\|A\|_F^2 - \sum_{i=1}^n |\lambda_i|^2\right)^{1/2}$,
4. $\mu_4(A) = \max_{i=1,\ldots,n} ||\lambda_i| - \sigma_i|$.

Vse mere so ekvivalentne in, če je katera izmed njih za matriko A velika, lahko pričakujemo težave oz. slabo konvergenco pri uporabi Arnoldijeve metode.

Zgled 4.2 *Vzamemo matriko*

$$A = \text{diag}\left(\text{diag}(1, 2, \ldots, 98), \begin{bmatrix} 100 & 1 \\ -1 & 100 \end{bmatrix}\right),$$

ki ima lastne vrednosti $1, 2, \ldots, 98, 100 + i, 100 - i$. *Potrebnih je kar nekaj korakov, preden se pojavijo kompleksne Ritzeve vrednosti. Tik pred tem, ko se pojavijo kompleksne Ritzeve vrednosti, se za največjo Ritzevo vrednost zdi, da je že skonvergirala.*

Zgled 4.3 *Vzamemo petdiagonalno matriko s konstantnimi vrednostmi na diagonalah*

$$A = \begin{bmatrix} 2 & 1 & -2 & & & \\ 1 & 2 & 1 & -2 & & \\ 2 & 1 & 2 & 1 & -2 & \\ & \ddots & \ddots & \ddots & \ddots & \ddots \end{bmatrix}.$$

Konvergence skoraj ni, razen k ekstremnim štirim lastnim vrednostim.

Zgled 4.4 *Spet vzamemo petdiagonalno matriko s konstantnimi vrednostmi na diagonalah, tokrat*

$$A = \begin{bmatrix} 2 & 1 & -0.4 & & & \\ 0 & 2 & 1 & -0.4 & & \\ 2 & 0 & 2 & 1 & -0.4 & \\ & \ddots & \ddots & \ddots & \ddots & \ddots \end{bmatrix}.$$

Konvergenca je zelo počasna, tako kot v prejšnjem primeru. Se pa izkaže, da lahko že z zelo majhnim podprostorom Krilova zelo dobro aproksimiramo zalogo vrednosti matrike A, ki je definirana kot

$$F(A) = \left\{ \frac{x^HAx}{x^Hx} : x \in \mathbb{C}^n,\ x \neq 0 \right\}.$$

4.4.1 Konvergenca

Pri Arnoldijevi metodi je konvergenca odvisna od izbire začetnega vektorja. Denimo, da nas zanima konvergenca proti lastnemu paru (λ, x).

Naj bo $X = [x\ X_2]$ obrnljiva matrika. Potem je

$$X^{-1}AX = \begin{bmatrix} \lambda & 0 \\ 0 & A_{22} \end{bmatrix}.$$

Če s transformacijo X preslikamo začetni vektor v_1, dobimo $X^{-1}v_1 = \begin{bmatrix} v_\lambda \\ v_r \end{bmatrix}$.

Velja ocena:

$$\tan\theta(x, \mathcal{K}_k) \leq \kappa(X) \min_{\substack{p\in\mathbb{P}_{k-1} \\ p(\lambda)=1}} \frac{\|p(A_{22})v_r\|_2}{|v_\lambda|}.$$

To bo majhno, če bo polinom imel majhne absolutne vrednosti pri vseh ostalih lastnih vrednostih. Iščemo polinom p stopnje $k-1$, za katerega velja $p(\lambda_1) = 1$ in zavzame najmanjšo absolutno vrednost pri ostalih lastnih vrednostih. To je povezano s problemom najboljše enakomerne aproksimacije.

Če je Ω kompaktna množica, lahko definiramo $\|f\|_\infty = \max_{z\in\Omega} |f(z)|$. V našem primeru bomo za Ω vzeli spekter matrike A brez λ_1. Sedaj v bistvu iščemo polinom oblike $(z-\lambda_1)q(z)$, kjer je $q \in \mathbb{P}_{k-1}$, ki je najboljša enakomerna aproksimacija za $f(z) = 1$ na Ω oziroma minimizira

$$\|1 - (z-\lambda_1)q(z)\|_\infty$$

po vseh $q \in \mathbb{P}_{k-1}$. Ker so sedaj lastne vrednosti lahko tudi kompleksne, je take polinome težko poiskati, delno pa tudi tukaj v določenih primerih lahko dobimo ocene s pomočjo polinomov Čebiševa.

4.4.2 Ponovni zagon

Za razliko od Lanczoseve metode, se pri Arnoldijevi metodi količina dela, ki ga porabimo za nov korak, povečuje s tem, ko velikost podprostora Krilova narašča. Vsak nov vektor moramo namreč ortogonalizirati na vse prejšnje bazne vektorje. Ker količina dela in potrebnega prostora lahko hitro preveč naraste, Arnoldijevo metodo ponavadi uporabljamo v kombinaciji s ponovnim zagonom, ko ponovno zaženemo metodo z novim začetnim podprostorom.

Najbolj enostaven ponovni zagon je, da najprej naredimo m korakov Arnoldijeve metode, potem pa metodo ponovno zaženemo z začetnim vektorjem v_+, ki ga izberemo iz prostora $\mathrm{Lin}(v_1, \ldots, v_m)$.

Za tako izbrani vektor obstaja polinom p stopnje kvečjemu $m-1$, da je $v_+ = p(A)v_1$. Če v_1 razvijemo po bazi lastnih vektorjev kot

$$v_1 = a_1x_1 + \cdots + a_nx_n,$$

potem velja

$$v_+ = a_1p(\lambda_1)x_1 + \cdots + a_np(\lambda_n)x_n.$$

Pričakujemo lahko, da bo $\mathcal{K}_m(A, v_+)$ vseboval dobre aproksimacije za tiste lastne vrednosti, kjer je absolutna vrednost $p(\lambda)$ velika in slabe aproksimacije za lastne vrednosti, kjer je ta vrednost majhna.

Izbira v_+ v bistvu pomeni, da izberemo polinom, ki dobro separira lastne vrednosti na željene in neželjene. Ker lastnih vrednosti ne poznamo, določimo polinom p na podlagi izračunanih Ritzevih vrednosti. To je osnova Arnoldijeve metode s polinomskim filtriranjem, ki je predstavljena v algoritmu 4.3.

Algoritem 4.3 Arnoldijeva metoda s polinomskim filtriranjem. Vhodni podatki so matrika A, vektor v_1 in dimenzija m.

---

1. Izvedi m korakov Arnoldijeve metode in izračunaj Ritzeve vrednosti.
2. Ritzeve vrednosti razdeli na željene $\mu_1, \ldots, \mu_k$ in neželjene $\mu_{k+1}, \ldots, \mu_m$.
3. Izberi tak polinom q stopnje $\leq m$, da je $|q(\mu_i)| \gg |q(\mu_j)|$ za $i = 1, \ldots, k$ in $j = k+1, \ldots, m$.
4. Vzemi $v_1 = q(A)v_1 / \|q(A)v_1\|$ in nadaljuj v točki 1.

Tako lahko dobimo en lastni par. Kako jih lahko izračunamo več? Ko imamo izračunanih nekaj (denimo $k-1$) lastnih parov, dobimo delni Schurov razcep $AU_{k-1} = U_{k-1}R_{k-1}$, kjer je R_{k-1} zgornja trikotna matrika in U_{k-1} matrika z ortonormiranimi stolpci ($U_{k-1}^H U_{k-1} = I_{k-1}$). Vektorji $u_1, u_2, \ldots, u_{k-1}$ so Schurovi vektorji.

Lastne vrednosti $\widetilde{A} = A(I - U_{k-1}U_{k-1}^H)$ so $k-1$ ničel in preostale lastne vrednosti matrike A. Postopek za računanje je tako naslednji.

Denimo, da so $v_1, \ldots, v_{k-1}$ že izračunani Schurovi vektorji. Sedaj izberemo vektor v_k, ki je pravokoten na $v_1, \ldots, v_{k-1}$ in normiran. Arnoldijev algoritem poganjamo od v_k dalje, s tem da nove vektorje ortogonaliziramo tudi na $v_1, \ldots, v_{k-1}$. Tako zgradimo ortonormirano bazo za podprostor $\mathrm{Lin}(v_1, \ldots, v_{k-1}, v_k, Av_k, \ldots, A^{m-k}v_k)$. Prvi del, ki ga razpenjajo vektorji $v_1, \ldots, v_{k-1}$, je fiksen ali zaklenjen, drugi del, ki ga razpenjajo vektorji $v_k, Av_k, \ldots, A^{m-k}v_k$ pa je aktiven.

Ko dobimo nov lastni par, ga dodamo v zaklenjeni del (temu postopku pravimo zaklepanje ali locking), če pa je nezaželen, ga lahko tudi izločimo (temu pravimo purging).

Matrika H_m je še vedno zgornja Hessenbergova, a je njena vodilna podmatrika velikosti $(k-1) \times (k-1)$ zgornja trikotna (oziroma kvazi zgornja trikotna, če imamo realno matriko in računamo v realni aritmetiki). Npr. za $k = 3$ in $m = 6$ ima obliko

$$H_m = \begin{bmatrix} \times & \times & \times & \times & \times & \times \\ & \times & \times & \times & \times & \times \\ & & \times & \times & \times & \times \\ & & \times & \times & \times & \times \\ & & & \times & \times & \times \\ & & & & \times & \times \end{bmatrix}.$$

Grob algoritem je zapisan v algoritmu 4.4.

Algoritem 4.4 Arnoldijeva metoda z računanjem več lastnih parov. Vhodni podatki so matrika A, vektor v_1, dimenzija m in želeno število lastnih parov r.

k=1
ponavljaj
  $j = k, \ldots, m$
    $w = Av_j$
    $\ell = 1 \ldots, j$
      $h_{\ell j} = v_\ell^H w$
      $w = w - h_{\ell j} v_\ell$
    $h_{j+1,j} = \|w\|_2$
    $v_{j+1} = w/h_{j+1,j}$
  izračunaj Ritzeve pare iz $H(k:m, k:m)$ in izberi (μ_k, z_k)
  ortogonaliziraj z_k na $v_1, \ldots, v_{k-1}$ in tako pridobi v_k
  če je Ritzev par dober, nastavi $k = k + 1$.
dokler ni $k > r$

4.4.3 Arnoldijeva metoda z implicitnim ponovnim zagonom

Če delamo ponovni zagon samo z enim vektorjem, lahko izgubimo preveč podatkov. Bolje je nadaljevati s primerno izbranim k-razsežnim podprostorom, kjer je $k > 1$. Pri Arnoldijevi metodi z implicitnim ponovnim zagonom (Sorensen (1992)) oz. IRA (Implicitly Restarted Arnoldi) naredimo $M = k + m$ korakov Arnoldijeve metode. Dobimo

$$AV_M = V_M H_M + h_{M+1,M} v_{M+1} e_M^T = V_{M+1} \widetilde{H}_M. \tag{4.8}$$

Na zgornji Hessenbergovi matriki H_M izvedemo m korakov QR iteracije s premiki $\theta_1, \ldots, \theta_m$ in dobimo

$$H_M^{(m)} = Q^H H_M Q.$$

Poglejmo si podrobnosti. Ko izberemo prvi premik θ_1, naredimo en korak QR iteracije z začetno matriko H_M. Torej $H_M - \theta_1 I = Q_1 R_1$ in $H_M^{(1)} = R_1 Q_1 + \theta_1 I = Q_1^H H_M Q_1$. Matrika $H_M^{(1)}$ je zgornja Hessenbergova, saj se ta oblika med QR iteracijo ohranja. Če pogledamo situacijo pri Arnoldijevi metodi, imamo tam pred QR iteracijo enakost

$$AV_M - V_M H_M = re_M^T,$$

kjer je $r = h_{M+1,M} v_{M+1}$. Od tod sledi

$$(A - \theta_1 I)V_M - V_M(H_M - \theta_1 I) = re_M^T.$$

Ko vstavimo $H_M - \theta_1 I = Q_1 R_1$ in enakost z desne pomnožimo s Q_1, dobimo

$$(A - \theta_1 I)V_M Q_1 - V_M Q_1 R_1 Q_1 = re_M^T Q_1.$$

To lahko preoblikujemo v

$$AV_M Q_1 - V_M Q_1 (R_1 Q_1 + \theta_1 I) = re_M^T Q_1$$

in dobimo

$$AV_M Q_1 - V_M Q_1 H_M^{(1)} = re_M^T Q_1.$$

Vidimo, da se z enim korakom QR iteracije zveza (4.8) preoblikuje tako, da se V_M zamenja z $V_M Q_1$, matrika H_M se zamenja s $H_M^{(1)}$, namesto vrstice e_M^T pa imamo $e_M^T Q_1$. Vektor $(e_M^T Q_1)^T$ ima neničelna le zadnja dva elementa, saj je Q_1 zgornja Hessenbergova.

Na začetku je matrika H_M nerazcepna. Če za θ_1 izberemo Ritzevo vrednost (lastno vrednost matrike H_M), bo matrika $H_M^{(1)}$ imela zadnjo vrstico oblike $[\,0 \quad \cdots \quad 0 \quad \theta_1\,]$. Če pa bo θ_1 blizu Ritzeve vrednosti, lahko pričakujemo majhen obdiagonalen element v zadnji vrstici.

Nadaljujemo z drugim korakom QR iteracije. Izberemo premik θ_2, naredimo QR razcep $H_M^{(1)} - \theta_2 I = Q_2 R_2$ in zmnožimo nazaj v $H_M^{(2)} = R_2 Q_2 + \theta_2 I = Q_2^H H_M^{(1)} Q_2$. Zveza (4.8) se po dveh korakih spremeni v

$$AV_M Q_1 Q_2 - V_M Q_1 Q_2 H_M^{(2)} = r e_M^T Q_1 Q_2.$$

Ko naredimo še preostalih $m-2$ QR iteracij, imamo

$$AV_M Q - V_M Q H_M^{(m)} = r e_M^T Q,$$

kjer je $Q = Q_1 \cdots Q_m$. Vektor $(e_M^T Q)^T$ ima prvih $k-1$ elementov enakih 0, saj so vse matrike $Q_1, \ldots, Q_m$ zgornje Hessenbergove. To pomeni, da lahko bločno pišemo

$$h_{M+1,M} e_M^T Q = [\,\beta e_k^T \quad b^T\,]$$

za ustrezna $\beta \in \mathbb{C}$ in $b \in \mathbb{C}^m$. Vodilna podmatrika velikosti $(k+1) \times k$ matrike

$$\begin{bmatrix} H_M^{(m)} \\ h_{M+1,M} e_M^T Q \end{bmatrix}$$

ima razširjeno zgornjo Hessenbergovo obliko, kot smo jo vajeni iz Arnoldijevega algoritma (zgornja Hessenbergova in še neničelni zadnji element v dodatni vrstici).

Če je $\widetilde{V}_M = V_M Q$ in je $\widetilde{V}_k$ prvih k stolpcev te matrike, potem velja zveza

$$A\widetilde{V}_k = \widetilde{V}_k H_k^{(m)} + h_{k+1,k}^{(m)} \widetilde{v}_{k+1} e_k^T + \beta v_{M+1} e_k^T = \widetilde{V}_k H_k^{(m)} + \widetilde{h}_{k+1,k} s e_k^T,$$

kjer je s normiran vektor. Tako pridemo do končne zveze

$$A\widetilde{V}_k = \widetilde{V}_{k+1} \widetilde{H}_{k+1,k},$$

kjer je

$$\widetilde{H}_{k+1,k} = \begin{bmatrix} H_M^{(m)} \\ \widetilde{h}_{M+1,M} e_k^T \end{bmatrix} \quad \text{in} \quad \widetilde{V}_{k+1} = [\,\widetilde{V}_k \quad s\,].$$

Vse QR iteracije seveda izvedemo v implicitni obliki s preganjanjem grbe. Za premike izberemo tiste Ritzeve vrednosti, ki so v območju, ki nas ne zanima. Tako se te Ritzeve vrednosti izločijo v spodnjem delu matrike, v zgornjem delu, s katerim nadaljujemo, pa ostanejo lastne vrednosti iz željenega območja.

Na koncu za nov začetni podprostor $\widetilde{V}_k$ vzamemo prvih k stolpcev matrike $V_M Q$, za $\widetilde{H}_k$ pa vodilno podmatriko matrike $H_M^{(m)}$. Izkaže se, da je

$$\widehat{v}_1 = \gamma (A - \theta_1 I) \cdots (A - \theta_m I) v_1,$$

stolpci $\widetilde{V}_k$ pa so ortonormirana baza za $\mathcal{K}_k(A, \widehat{v}_1)$.

Ustrezna metoda je zapisana v algoritmu 4.5. Arnoldijeva metoda z implicitnim ponovnim zagonom je bila v preteklosti osnova za metodo `eigs` v Matlabu, ki je namenjena računanju lastnih vrednosti velikih razpršenih matrik, ped nekaj leti pa jo je zamenjala metoda iz naslednjega razdelka.

Algoritem 4.5 Arnoldijeva metoda z implicitnim ponovnim zagonom. Vhodni podatki so matrika A, vektor v_1, ter dimenziji m in k.

izvedi k korakov Arnoldija $\Longrightarrow V_{k+1}$, $\widetilde{H}_k$.
$r = h_{k+1,k} v_{k+1}$
dokler $\|r\| > \epsilon$ ponavljaj
  izvedi m korakov Arnoldija $\Longrightarrow V_{M+1}$, $\widetilde{H}_M$.
  izberi m premikov $\theta_1, \ldots, \theta_m$ (npr. neželene Ritzeve vrednosti)
  naredi m korakov implicitne QR iteracije za H_M s premiki $\theta_1, \ldots, \theta_m \Longrightarrow Q$, $\widetilde{V}_{M+1}$, $H_M^{(m)}$
  $V_k = \widetilde{V}_k$
  $H_k = H_k^{(m)}$
  $r = h_{k+1,k}^{(m)} \widetilde{v}_{k+1} + h_{M+1,M} v_{M+1} e_M^T Q e_k$
  $h_{k+1,k} = \|r\|$
  $v_{k+1} = r / h_{k+1,k}$

4.4.4 Krilov–Schurova metoda

Pri Arnoldijevi metodi z implicitnim ponovnim zagonom smo v vsakem koraku naredili m korakov QR iteracije na matriki H_M. Pri Krilov–Schurovem algoritmu, ki ga je predstavil Stewart leta 2001, namesto tega izračunamo Schurovo formo za matriko H_M. Tako dobimo $H_M = QSQ^H$, kjer je S zgornja trikotna in Q unitarna matrika. Pri tem diagonalne elemente S uredimo tako, da so na začetku na mestih $s_{11}, \ldots, s_{kk}$ želene Ritzeve vrednosti. Ker sam algoritem za računanje Schurove forme (QR iteracija) Ritzevih vrednosti ne uredi pravilno, moramo za to poskrbeti z naknadno preureditvijo.

Če je matrika A realna in s_{kk} ter $s_{k+1,k+1}$ tvorita konjugiran par, potem k povečamo na $k+1$, da lahko nadaljujemo z računanjem v realni aritmetiki. V tem primeru je S kvazi zgornja trikotna matrika.

Iz $AV_M = V_M H_M + h_{M+1,M} v_{M+1} e_M^T$ sledi

$$AV_M Q = V_M Q S Q^H Q + h_{M+1,M} v_{M+1} e_M^T Q.$$

Če označimo $\widehat{H}_k = S(1:k, 1:k)$ in $\widehat{V}_k = V_M Q(:, 1:k)$, potem velja

$$A\widehat{V}_k = \widehat{V}_k \widehat{H}_k + h_{M+1,M} v_{M+1} e_M^T Q(:, 1:k).$$

To pomeni, da matriko $\widehat{V}_k$ razširimo v $\widehat{V}_{k+1}$ z vektorjem v_M in vzamemo

$$\widehat{h}_{k+1}^T = h_{M+1,M} e_M Q(:, 1:k)$$

za vrstico, s katero razširimo $\widehat{H}_k$ v matriko $\widetilde{\widehat{H}}_k$. Vrstica $\widehat{h}_{k+1}^T$ je v splošnem primeru polna, zato

ima matrika $\widetilde{\widehat{H}}_k$ obliko (za primer $k = 4$)

$$\widetilde{\widehat{H}}_k = \begin{bmatrix} \widehat{H}_k \\ \widehat{h}_{k+1}^T \end{bmatrix} = \begin{bmatrix} \times & \times & \times & \times \\ & \times & \times & \times \\ & & \times & \times \\ & & & \times \\ \times & \times & \times & \times \end{bmatrix}.$$

Če je norma $\|h_{k+1}\|$ dovolj majhna, imamo invariantni podprostor velikosti k v katerem naj bi bile želene lastne vrednosti.

Matriko $\widetilde{\widehat{H}}_k$ bi v nadaljevanju lahko pretvorili na zgornjo Hessenbergovo obliko, a v bistvu to ni potrebno. Nadaljujemo podobno kot pri IRA od $\widetilde{H}_k$ dalje. Matrika H_M bo tako v nadaljevanju imela v splošnem primeru obliko (npr. za $k = 4$ in $m = 3$)

$$\begin{bmatrix} \times & \times & \times & \times & \times & \times & \times \\ & \times & \times & \times & \times & \times & \times \\ & & \times & \times & \times & \times & \times \\ & & & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times & \times \\ & & & & \times & \times & \times \\ & & & & & \times & \times \end{bmatrix}.$$

Algoritem za Krilov–Schurovo metodo je zapisan v algoritmu 4.6.

Algoritem 4.6 Krilov–Schurova verzija Arnoldijeve metode s ponovnim zagonom. Vhodni podatki so matrika A, vektor v_1, ter dimenziji m in k.

izvedi k korakov Arnoldija $\Longrightarrow V_{k+1}$, $\widetilde{H}_k$.
$r = h_{k+1,k} v_{k+1}$
dokler $\|r\| > \epsilon$ ponavljaj
    izvedi m korakov Arnoldija $\Longrightarrow V_{M+1}$, $\widetilde{H}_M$.
    izračunaj urejeno Schurovo formo $H_M = QSQ^H$,
        kjer so Ritzeve vrednosti urejene tako, da je na začetku k želenih
    $V = [\, V_M Q(:, 1:k) \quad v_{M+1} \,]$
    $H = \begin{bmatrix} S(1:k, 1:k) \\ h_{M+1,M} Q(M, 1:k) \end{bmatrix}$
    $r = H(k+1, 1:k)$

4.4.5 Harmonične Ritzeve vrednosti

Tudi pri Arnoldiju lahko uporabljamo harmonične Ritzeve vrednosti. Če uporabimo premik σ, je θ harmonična Ritzeva vrednost in $z \in \mathcal{K}_k(A, v_1)$ harmonični vektor, če je

$$Az - \theta z \perp (A - \sigma I)\mathcal{K}_k(A, v_1).$$

Lema 4.18 *Po k korakih Arnoldija dobimo* $AV_k = V_k H_k + h_{k+1,k} v_{k+1} e_k^T = V_{k+1} \widetilde{H}_k$. *Če je* ϑ *lastna vrednost posplošenega problema lastnih vrednosti*

$$\vartheta (\widetilde{H}_k - \sigma I_{k+1,k})^H (\widetilde{H}_k - \sigma I_{k+1,k}) y = (H_k - \sigma I_k) y,$$

potem je $\sigma + 1/\vartheta$ harmonična Ritzeva vrednost matrike A.

Izkaže se, da so harmonični Ritzevi vektorji kvalitetnejše aproksimacije za lastne vektorje, kot so harmonične Ritzeve vrednosti za lastne vrednosti. Zato kot približek za lastno vrednost vzamemo Rayleighov kvocient harmoničnega vektorja.

Harmonični vektorji so posebej primerni za ponovni zagon. Če računamo notranje lastne vrednosti, bo tudi v primeru, ko je Ritzeva vrednost blizu tarče, Ritzev vektor lahko daleč od ustreznega lastnega vektorja. Pri harmoničnih vrednostih in vektorjih se to ne more zgoditi, saj ima harmonični vektor vedno močno komponento v smeri iskanega lastnega vektorja.

Namesto harmoničnih Ritzevih vrednosti lahko za izračun notranjih lastnih vrednosti uporabimo metodo premakni-in-obrni Arnoldi (shift-and-invert Arnoldi). Pri tej metodi vzamemo podprostor Krilova, ki ga generira matrika $(A - \tau I)^{-1}$, kjer je $\tau \in \mathbb{C}$ dani cilj. Produkt z matriko $(A - \tau I)^{-1}$ je potrebno izračunati točno, zato lahko to metodo uporabljamo le v primerih, kjer si to lahko privoščimo. Reševanje sistema večinoma poteka preko LU razcepa, to pa si lahko privoščimo le, če sta matriki L in U dovolj razpršeni, da nam zanju ne zmanjka pomnilnika.

4.5 Jacobi–Davidsonova metoda

Pri metodah, ki temeljijo na podprostorih Krilova, kot sta Arnoldijeva in Lanczoseva metoda, je podprostor, v katerem iščemo aproksimacije (Ritzeve, Petrovove, harmonične Ritzeve vrednosti, ...) v bistvu odvisen le od začetnega vektorja v_1.

Pri Jacobi–Davidsonovi metodi podprostor širimo na drug način. Metodo sta razvila van der Vorst in Sleijpen leta 1996, temelji pa na kombinaciji Jacobijeve metode iz leta 1840 in Davidsonove metode iz leta 1975.

4.5.1 Davidsonova metoda

Naj vektorji $v_1, \ldots, v_k$ tvorijo ortonormirano bazo podprostora $\mathcal{V}_k$ dimenzije k in naj bo θ_k dominantna Ritzeva vrednost za A in $\mathcal{V}_k$ z ustreznim Ritzevim vektorjem u_k. Vprašamo se lahko, v kateri smeri naj razširimo podprostor $\mathcal{V}_k$ v $\mathcal{V}_{k+1}$, da bomo dobili še boljši približek za dominantno lastno vrednost?

Če je u_k pripadajoči Ritzev vektor in označimo ostanek kot

$$r = Au_k - \theta_k u_k,$$

potem pri Davidsonovi metodi rešimo sistem

$$(D_A - \theta_k I)t = r,$$

kjer je D_A diagonala matrike A. Z vektorjem t (ki ga ortonormiramo glede na prejšnje vektorje) potem razširimo podprostor $\mathcal{V}_k$. Metoda je predstavljena v algoritmu 4.7.

Metodo je Davidson razvil za velike diagonalno dominantne matrike, ki so nastopale v problemih iz kemije, ki jih je reševal. Za takšne matrike metoda v praksi deluje zelo dobro, če pa poskusimo ozadje razložiti matematično, naletimo na nekaj težav.

Matriko $D_A - \theta_k I$ si lahko predstavljamo kot aproksimacijo za $A - \theta_k I$. A kot kaže naslednji razmislek, ki se lahko potrdi tudi z numeričnimi eksperimenti, ta aproksimacija ne sme biti preveč dobra. Če namreč namesto $D_A - \theta_k I$ vzamemo kar točno matriko $A - \theta_k I$, potem dobimo $t = u_k$, ker pa je $u_k \in \mathcal{V}_k$, s tem vektorjem ne moremo razširiti baze.

Algoritem 4.7 Davidsonova metoda.

izberi vektor $\|v_1\| = 1$
$k = 1, \ldots, m$
  izračunaj k-to vrstico in k-ti stolpec $B_k = V_k^H A V_k$.
  izračunaj dominantno Ritzevo vrednost θ_k in pripadajoči Ritzev vektor $u_k = V_k s_k$
  $r = Au_k - \theta_k u_k$
  $t = (D_A - \theta_k I)^{-1} r$
  vektor t ortogonaliziraj glede na $v_1, \ldots, v_k$ in ostanek vzemi za v_{k+1}

Namesto $(D_A - \theta_k I)^{-1}$ bi lahko uporabili katerokoli drugo aproksimacijo za $(A - \theta_k I)^{-1}$. Tako lahko npr. sistem $(A - \theta_k I)t = -r$ rešimo približno z eno izmed iterativnih metod, kot je npr. GMRES. Paziti moramo, da ga res rešimo le približno, saj v primeru, če ga rešimo preveč natančno, metoda ne bo konvergirala.

Za ortogonalizacijo vektorja t uporabimo MGS oziroma še bolje RGS.

Če za aproksimacijo $(A - \theta_k I)^{-1}$ vzamemo kar $(I - \theta_k I)^{-1}$, dobimo Arnoldijevo metodo. To očitno dobimo tudi v primeru, ko ima A konstantno diagonalo.

Matrika B_k za razliko od Arnoldijeve metode nima nobene posebne oblike in je polna. Razlika od Arnoldijeve metode je tudi ta, da pri Davidsonovi metodi računamo le eno lastno vrednost.

4.5.2 Jacobijeva metoda ortogonalnih popravkov

Poznamo Jacobijevo metodo za računanje lastnih vrednosti simetričnih matrik, kjer z Jacobijevimi rotacijami matriko vedno bolj spreminjamo v diagonalno obliko. Jacobi pa je razvil tudi drugačno metodo, kjer rotacij uporabimo le toliko, da matrika postane močno diagonalno dominantna, za nadaljnje računanje pa uporabimo metodo podobno Davidsonovi.

Naj bo $a_{11} = \alpha$ največji diagonalni element diagonalno dominantne matrike A. Potem je α približek za dominantno lastno vrednost λ, približek za lastni vektor pa je e_1.

Lastni problem

$$A \begin{bmatrix} 1 \\ z \end{bmatrix} = \begin{bmatrix} \alpha & c^T \\ b & F \end{bmatrix} \begin{bmatrix} 1 \\ z \end{bmatrix} = \lambda \begin{bmatrix} 1 \\ z \end{bmatrix},$$

kjer je F kvadratna matrika, α je skalar, b, c, z pa vektorji reda $n-1$, je ekvivalenten sistemu

$$\begin{aligned} \lambda &= \alpha + c^T z, \\ (F - \lambda I)z &= -b. \end{aligned}$$

Jacobi je predlagal, da sistem rešimo iterativno tako, da vzamemo $z_1 = 0$ in nato računamo po vrsti za $k = 1, 2, \ldots$:

$$\theta_k = \alpha + c^T z_k,$$

$$(D - \theta_k I)z_{k+1} \quad = \quad (D - F)z_k - b,$$

kjer je D diagonala matrike F.

V vseh korakih Jacobijevega pristopa iščemo lastni vektor oblike $u = \left[\begin{smallmatrix} 1 \\ z \end{smallmatrix}\right]$, torej iščemo ortogonalne popravke $\left[\begin{smallmatrix} 0 \\ z \end{smallmatrix}\right] = u - (e_1^H u)e_1$ za začetni približek e_1. Ne upoštevamo, da med postopkom dobimo boljše približke $u_k = \left[\begin{smallmatrix} 1 \\ z_k \end{smallmatrix}\right]$ in, da bi potem lahko iskali ortogonalne popravke $u - (u_k^H u)u_k$ za u_k.

Če primerjamo Jacobijevo in Davidsonovo metodo, opazimo nekaj podobnosti. Denimo, da ima matrika obliko

$$A = \begin{bmatrix} \alpha & c^T \\ b & F \end{bmatrix},$$

kot pri Jacobijevi metodi, in vzemimo $v_1 = e_1$. Vektor u_k skaliramo tako, da je $u_k = \left[\begin{smallmatrix} 1 \\ z_k \end{smallmatrix}\right]$. Naj bo θ_k aproksimacija za lastno vrednost. Za ostanek potem velja

$$r_k = (A - \theta_k I)u_k = \begin{bmatrix} \alpha - \theta_k + c^T z_k \\ (F - \theta_k I)z_k + b \end{bmatrix}.$$

Pri Davidsonovi metodi novo smer t_k dobimo iz sistema $(D_A - \theta_k I)t_k = -r_k$. Če označimo $t_k = \left[\begin{smallmatrix} * \\ y_k \end{smallmatrix}\right]$, potem velja

$$(D - \theta_k I)y_k = -(F - \theta_k I)z_k - b = (D - F)z_k - (D - \theta I)z_k - b,$$

oziroma

$$(D - \theta_k I)(z_k + y_k) = (D - F)z_k - b.$$

Vektor $z_k + y_k$ se tako ujema z vektorjem z_{k+1}, ki ga dobimo po enem koraku Jacobijeve metode, če začnemo z vektorjem z_k.

Pri Jacobijevi metodi kot naslednji približek za lastni vektor vzamemo $\widehat{u}_{k+1} = \left[\begin{smallmatrix} 1 \\ z_k + y_k \end{smallmatrix}\right] = u_k + \widehat{y}_k$. Torej vzamemo komponento $\widehat{y}_k = \left[\begin{smallmatrix} 0 \\ y_k \end{smallmatrix}\right]$ vektorja t_k, ki je ortogonalna na u_1 in za naslednji približek vzamemo $u_{k+1} = u_k + \widehat{y}_k$ in $\theta_{k+1} = e_1^H A u_{k+1}$. Pri Davidsonovi metodi pa vzamemo komponento v_{k+1} vektorja t_k, ki je ortogonalna na $u_1, \ldots, u_k$ in razširimo prostor na $\mathcal{V}_{k+1} = \mathcal{L}(v_1, \ldots, v_k, v_{k+1})$, potem pa vzamemo za θ_{k+1}, u_{k+1} Ritzev par matrike A za $\mathcal{V}_{k+1}$. Čeprav velja $\mathcal{L}(u_1, \ldots, u_k, \widehat{u}_{k+1}) = \mathcal{L}(u_1, \ldots, u_k, u_{k+1})$, se metodi ne ujemata, saj približki θ_k niso enaki, to pa pomeni, da se razlikujejo tudi popravki t_j.

V obeh metodah nove smeri iščemo s fiksnimi operatorji (D_A pri Davidsonu oz. D pri Jacobiju), ne pa s takimi, ki bi bili povezani s prostorom v katerem iščemo popravek oz. novo smer. To popravimo v Jacobi–Davidsonovi metodi, ki črpa navdih iz obeh metod.

4.5.3 Jacobi–Davidsonova metoda

Naj bo $\mathcal{V}_k$ podprostor dimenzije k in naj bo θ_k dominantna Ritzeva vrednost za A in $\mathcal{V}_k$ z ustreznim normiranim Ritzevim vektorjem u_k. Sedaj popravek za u_k iščemo v podprostoru $u_k^\perp$. Ortogonalna projekcija matrike A na ta podprostor je $B = (I - u_k u_k^H)A(I - u_k u_k^H)$. Če upoštevamo $\theta_k = u_k^H A u_k$, lahko zapišemo

$$A = B + A u_k u_k^H + u_k u_k^H A - \theta_k u_k u_k^H.$$

Nastavek je $A(u_k + v) = \lambda(u_k + v)$, od tod pa iz $Bu_k = 0$ sledi

$$(B - \lambda I)v = -r + (\lambda - \theta_k - u_k^H Av)u_k,$$

kjer je $r = Au_k - \theta_k u_k$ ostanek. Ker sta tako leva stran kot tudi ostanek r ortogonalna na vektor u_k, mora biti faktor pred u_k enak 0 in dobimo

$$(B - \lambda I)v = (I - u_k u_k^H)(A - \lambda I)(I - u_k u_k^H)v = -r.$$

Tako kot pri Jacobijevi in Davidsonovi metodi namesto neznane vrednosti λ vzamemo približek θ_k. Tako pridemo do t.i. *korekcijske enačbe*

$$(I - u_k u_k^H)(A - \theta_k I)(I - u_k u_k^H)v = -r, \tag{4.9}$$

ki jo rešimo le približno in z novim vektorjem razširimo podprostor $\mathcal{V}_k$. Za približno reševanje ponavadi uporabimo eno izmed iterativnih metod, npr. GMRES za splošno matriko in MINRES za simetrično matriko. Za razliko od Davidsonove metode tu sicer ni omejitve, da sistema (4.9) ne smemo rešiti točno, a se ponavadi izkaže, da je najbolj optimalno, da ga rešimo zgolj približno. Pokazati se da, da imamo v primeru, ko sistem rešimo točno, kvadratično konvergenco (pri simetričnih matrikah celo kubično).

V korekcijski enačbi (4.9) iščemo vektor v, ki je pravokoten na u_k. Zato je $(I - u_k u_k^H)v = v$ in iz korekcijske enačbe sledi

$$(I - u_k u_k^H)(A - \theta_k I)v = -r.$$

Če korekcijsko enačbo rešimo točno, potem iz zgornje enakosti sledi

$$(A - \theta_k I)v = -r + \alpha u_k$$

in za novo smer velja

$$v = u_k + \alpha(A - \theta_k I)^{-1} u_k.$$

Iskalni podprostor $\mathcal{V}_k$ tako razširimo z vektorjem, ki bi ga dobili z enim korakom Rayleighove iteracije, zanjo pa vemo, da ima za simetrično matriko v bližini rešitve kubično konvergenco, sicer pa kvadratično.

Če naredimo npr. le en korak metode GMRES, je to ekvivalentno temu, da za v vzamemo ostanek r. To vodi do Arnoldijeve metode, saj je $\mathrm{Lin}(u_k, r) = \mathrm{Lin}(u_k, Au_k)$.

Če namesto korekcijske enačbe (4.9) vzamemo sistem brez projekcij $(A - \theta_k I)v = -r$, to ne bo v redu, saj je točna rešitev $v = u_k$ in se iskalni podprostor ne more razširiti z vektorjem v.

V primeru simetrične matrike Ritzeve vrednosti monotono konvergirajo proti dominantni lastni vrednosti. V bližini rešitve imamo kubično konvergenco (če bi korekcijsko enačbo reševali točno). Ker namesto točnega reševanja uporabljamo GMRES (lahko pa tudi simetrično varianto MINRES), moramo poiskati kompromis med številom Jacobi–Davidsonovih iteracij in številom računanj produkta Ax.

V primeru nesimetrične matrike ali pri računanju notranjih lastnih vrednosti simetrične matrike imamo težave, ki pa se dajo odpraviti z uporabo harmoničnih Ritzevih vrednosti. Z njimi lahko poiščemo lastne vrednosti, ki so po absolutni vrednosti najmanjše in tako s premiki dobimo iskane lastne vrednosti.

Ko najdemo lastni par (λ, x), nadaljujemo samo s podprostorom, ki ga razpenjajo preostali vektorji. V primeru ortogonalne deflacije tako nadaljujemo z

$$(I - xx^H)A(I - xx^H).$$

Algoritem 4.8 Jacobi–Davidsonova metoda.

1. Start: Izberi začetni vektor $\|v_1\| = 1$.

- Postavi $V_1 = [v_1]$, $u = v_1$, $\theta = u^H Au$ in izračunaj $r = Au - \theta u$.

2. Zunanja zanka: Za $k = 1, \ldots, m-1$:

- Približno reši $(I - uu^H)(A - \theta I)(I - uu^H)v = -r, \quad v \perp u$.
- Ortogonaliziraj v glede na V_k (RGS) in razširi V_k v V_{k+1}.
- Poišči dominantni lastni par (θ, s) matrike $V_{k+1}^H A V_{k+1}$, kjer je $\|s\| = 1$.
- Izračunaj Ritzev vektor $u = V_{k+1}s$ in ostanek $r = Au - \theta u$.
- Testiraj konvergenco in po potrebi končaj.

3. Ponovni zagon: Postavi $V_1 = [u]$ in ponovi točko 2.

V primeru, ko je skonvergiralo že več vektorjev, nadaljujemo z

$$(I - ZZ^H)A(I - ZZ^H),$$

kjer je $AZ = ZS$ delni Schurov razcep. Stolpci matrike Z so ortonormirani, matrika S pa je zgornja trikotna z lastnimi vrednostmi na diagonali.

Ko je dimenzija iskalnega podprostora prevelika, izločimo manjši podprostor in nadaljujemo z njim. V zadnjem koraku poiščemo Schurovo formo matrike $V_k^H A V_k$ in jo preuredimo tako, da so Ritzeve vrednosti, ki so blizu iskani tarči, v zgornjem delu. Potem nov začetni podprostor dobimo iz začetnih Schurovih vektorjev.

4.5.4 Predpogojevanje

Pri reševanju korekcijske enačbe

$$(I - u_k u_k^H)(A - \theta_k I)(I - u_k u_k^H)v = -r$$

je dobro uporabiti predpogojevanje. Denimo, da imamo na voljo matriko K, da je $K^{-1}(A - \theta_k I) \approx I$. Pri uporabi moramo tudi matriko K zožiti na isti podprostor, torej uporabimo

$$\widetilde{K} = (I - u_k u_k^H)K(I - u_k u_k^H).$$

Če za reševanje korekcijske enačbe uporabimo metodo podprostorov Krilova, moramo v vsakem koraku izračunati produkt

$$z = \widetilde{K}^{-1}\widetilde{A}w,$$

kjer je $\widetilde{A} = (I - u_k u_k^H)(A - \theta_k I)(I - u_k u_k^H)$, vektor w pa je pravokoten na u_k.

To lahko učinkovito izračunamo po naslednjem postopku.

1. Ker je w pravokoten na u_k, je $(I - u_k u_k^H)w = w$ in zato $\widetilde{A}w = (I - u_k u_k^H)g$, kjer je vektor $g = (A - \theta_k I)w$. Izračun vektorja g ni težava, saj znamo učinkovito izračunati produkt z matriko A in poznamo skalar θ_k.

2. V drugem delu moramo rešiti sistem $\widetilde{K}z = \widetilde{A}w = (I - u_k u_k^H)g$. Ker je z pravokoten na u_k, je $Kz = g - \beta u_k$ za nek skalar β. Od tod sledi $z = K^{-1}g - \beta K^{-1}u_k$. Ker mora biti z pravokoten na u_k, lahko od tod izračunamo β in dobimo

$$\beta = \frac{u_k^H K^{-1} g}{u_k^H K^{-1} u_k}.$$

Ko poznamo β, lahko seveda vektor z izračunamo iz $z = K^{-1}g - \beta K^{-1}u_k$.

Ker za približno reševanje korekcijske enačbe ponavadi uporabimo metodo (npr. GMRES), ki temelji na podprostorih Krilova, moramo v vsakem koraku te metode izračunati produkt z matriko $\widetilde{K}^{-1}\widetilde{A}$. To pomeni, da moramo v vsakem koraku rešiti sistem z matriko K, da dobimo vektor $K^{-1}g$, poleg tega pa moramo pri prvem koraku izračunati še vektor $K^{-1}u_k$, ki ga potrebujemo v vseh korakih iteracije.

4.5.5 Harmonične Ritzeve vrednosti

Pri Jacobi–Davidsonovi metodi je za notranje lastne vrednosti najbolje uporabiti harmonične Ritzeve vrednosti. Ponovimo, da je θ harmonična Ritzeva vrednost matrike A glede na podprostor $\mathcal{V}_k$, če je θ^{-1} Ritzeva vrednost matrike A^{-1} glede na podprostor $A\mathcal{V}_k$.

Naj stolpci matrike $V_k = [v_1 \ \cdots \ v_k]$ sestavljajo ortonormirano bazo za podprostor $\mathcal{V}_k$, stolpci matrike $W_k = [w_1 \ \cdots \ w_k]$ pa bazo za podprostor $\mathcal{W}_k := A\mathcal{V}_k$. Potem je θ harmonična Ritzeva vrednost A glede na $A\mathcal{V}_k$ natanko tedaj, ko je

$$W_k^H A V_k s = \theta W_k^H V_k s$$

za neničelni vektor $s \in \mathbb{C}^k$.

Naj bo (θ_k, u_k) harmonični Ritzev par v k-tem koraku Jacobi–Davidsonovega algoritma. Ostanek $r = Au_k - \theta u_k$ je ortogonalen na $z_k = Au_k$. Pri korekcijski enačbi sedaj iščemo popravek, ki bo pravokoten na z_k (pri standardni metodi u_k). Namesto ortogonalne projekcije na $z_k^\perp$ uporabimo poševno projekcijo

$$I - \frac{u_k z_k^H}{z_k^H u_k},$$

saj tako u_k pošljemo v 0.

Nova korekcijska enačba je

$$\left(I - \frac{u_k z_k^H}{z_k^H u_k}\right)(A - \theta I)\left(I - \frac{u_k z_k^H}{z_k^H u_k}\right) t = -r.$$

Algoritem 4.9 Jacobi–Davidsonova metoda s harmoničnimi Ritzevimi vrednostmi.

1. Start: Izberi začetni vektor $\|v_1\| = 1$.

- Postavi $V_1 = [v_1]$, $u = v_1$, $\theta = u^H A u$ in izračunaj $r = Au - \theta u$.

2. Zunanja zanka: Za $k = 1, \ldots, m-1$:

- Približno reši

$$\left(I - \frac{uz^H}{z^H u}\right)(A - \theta I)\left(I - \frac{uz^H}{z^H u}\right) t = -r, \quad t \perp z.$$

- Ortogonaliziraj t glede na V_k (RGS) in razširi V_k v V_{k+1}.
- Ortogonaliziraj At glede na W_k (RGS) in razširi W_k v W_{k+1}.
- Poišči minimalni lastni par (θ, s) matrike $(W_k^H V_k)^{-1} W_k^H A V_k$, kjer je $\|s\| = 1$.
- Izračunaj Ritzev vektor $u = V_{k+1} s$, ostanek $r = Au - \theta u$ in vektor $z = Au$.
- Testiraj konvergenco in po potrebi končaj.

3. Ponovni zagon: Postavi $V_1 = [u]$ in ponovi točko 2.

Literatura

[1] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, H. van der Vorst, *Templates for the Solution of Algebraic Eigenvalue Problems. A Practical Guide*, SIAM, Philadelphia, 2000.

[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] J. W. Demmel, *Uporabna numerična linearna algebra*, DMFA Slovenije, Ljubljana, 2000.

[4] G. H. Golub, C. F. van Loan, *Matrix Computations, 3rd edition*, The Johns Hopkins University Press, Baltimore, 1996.

[5] A. Greenbaum, *Iterative Methods for Solving Linear Systems*, SIAM, Philadelphia, 1997.

[6] C. T. Kelley, *Iterative Methods for Linear and Nonlinear Equations*, SIAM, Philadelphia, 1995.

[7] G. Mele, E. Ringh, D. Ek, F. Izzo, P. Upadhyaya, E. Jarlebring, *Preconditioning for Linear Systems*, SIAM, Philadelphia, 2020.

[8] B. N. Parlett, *The Symmetric Eigenvalue Problem*, SIAM, Philadelphia, 1998.

[9] B. Plestenjak, *Razširjen uvod v numerične metode*, DMFA - založništvo, Ljubljana, 2015.

[10] Y. Saad, *Numerical Methods for Large Eigenvalue Problems, Second edition*, SIAM, Philadelphia, 2011.

[11] Y. Saad, *Iterative Methods for Sparse Linear Systems, Revised edition*, SIAM, Philadelphia, 2011.

[12] H. van der Vorst, *Computational Methods for Large Eigenvalue Problems*, P.G. Ciarlet in J.L. Lions (ured.), Handbook of Numerical Analysis, Volume VIII, North-Holand (Elsevier), Amsterdam, 2002, str. 3–179.

[13] H. van der Vorst, *Iterative Krylov Methods for Large Linear systems*, Cambridge University Press, Cambridge, 2003.

[14] D. S. Watkins, *Fundamentals of Matrix Computations, Second Edition*, John Wiley & Sons, New York, 2002.

[15] D. S. Watkins, *The Matrix Eigenvalue Problems. GR and Krylov Subspace Methods*, SIAM, Philadelphia, 2007.

[16] D. M. Young, T.-Z. Mai, The search for omega, v D. R. Kincaid, L. J. Hayes (ured.), *Iterative Methods for Large Linear Systems*, Academic Press, New York, 1990, 293–311.