5.1 Predpogojevanje

Konvergenca metod podprostorov za reševanje linearnega sistema $Ax = b$ je v veliki meri odvisna od razporeditve lastnih vrednosti (in lastnih vektorjev) matrike A.

Kadar je konvergenca počasna, si lahko pomagamo s *predpogojevanjem*. To pomeni, da namesto sistema $Ax = b$ rešujemo sistem

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

kjer za nesingularno matriko K, ki ji pravimo *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 na izračunamo eksplicitno, temveč v vsakem koraku najprej množimo vektor z matriko A, potem pa rešimo sistem z matriko K.

Levo predpogojevanje

Če namesto $Ax = b$ rešujemo sistem $K^{-1}Ax = K^{-1}b$, potem je to *levo predpogojevanje*.

Če uporabljamo katero izmed metod, ki vrača minimalni ostanek (GMRES, MINRES, QMR), potem sedaj namesto ostanka $r_k = b - Ax_k$ minimiziramo predpogojeni ostanek $K^{-1}(b - Ax_k)$. Zaradi tega moramo paziti pri ustavitvenem kriteriju, saj se lahko zgodi, da ostanka nista dovolj usklajena.

Problem z levim predpogojevanjem je, da npr. v primeru, ko sta matriki A in K simetrični, matrika $K^{-1}A$ ni več simetrična.

Če je K s.p.d., potem lahko definiramo skalarni produkt $[x, y] = \langle x, Ky\rangle$. V tem skalarnem produktu je $K^{-1}A$ sebiadjungirana, saj je

$$[K^{-1}Ax, y] = \langle K^{-1}Ax, Ky\rangle = \langle Ax, y\rangle = \langle x, Ay\rangle = [x, K^{-1}Ay].$$

Če je matrika A pozitivno definitna, potem to v $[.,.]$ velja tudi za $K^{-1}A$, saj je

$$[K^{-1}Ax, x] = \langle K^{-1}Ax, Kx\rangle = \langle Ax, x\rangle > 0.$$

Predpogojeni CG

Če je K s.p.d., potem levo predpogojevanje lakko uporabimo v CG v kombinaciji s skalarnim produktom $[x, y] = \langle x, Ky \rangle$. Tako ohranimo simetrijo.

Običajni CG

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

Predpogojeni CG

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

Če primerjamo zahtevnosti, imamo pri predpogojenem CG dodatno: 1 reševanje sistema z matriko K in 1 pomožni vektor ($K^{-1} r_j$) več.

Levo predpogojeni GMRES

Navadna metoda GMRES v grobem je:

1) izberi x_0, določi dimenzijo k za podprostor Krilova, izračunaj $r_0 = b - Ax_0$.
2) naredi k korakov Arnoldijevega algoritma z matriko A in začetnim vektorjem $v_1 = r_0/\|r_0\|$ da dobiš $H_{k+1,k}$ in V_{k+1}.
3) po metodi najmanjših kvadratov poišči y_k, ki minimizira $\|\|r_0\|e_1 - H_{k+1,k}y_k\|$.

Končni približek je potem $x_k = x_0 + V_k y_k$.

Metoda GMRES z levim predpogojevanjem je:

1) izberi x_0, določi dimenzijo k za podprostor Krilova, izračunaj $r_0 = K^{-1}(b - Ax_0)$.
2) naredi k korakov Arnoldijevega algoritma z matriko $K^{-1}A$ in začetnim vektorjem $v_1 = r_0/\|r_0\|$ da dobiš $H_{k+1,k}$ in V_{k+1}.
3) po metodi najmanjših kvadratov poišči y_k, ki minimizira $\|\|r_0\|e_1 - H_{k+1,k}y_k\|$.

Končni približek je potem $x_k = x_0 + V_k y_k$.

Desno predpogojevanje

Pri *desnem predpogojevanju* namesto $Ax =$ rešujemo $AK^{-1}z = b$, kjer je $x = K^{-1}z$.

Prednost desnega predpogojevanja je, da ne vpliva na desno stran b.

Sedaj imamo lahko težave pri metodah, kjer gledamo napako $x - x_k$, saj je

$$x - x_k = K^{-1}(z - z_k),$$

mi pa gledamo $z - z_k$.

Metoda GMRES z desnim predpogojevanjem:

1) izberi x_0, določi dimenzijo k za podprostor Krilova, izračunaj $r_0 = b - Ax_0$.
2) naredi k korakov Arnoldijevega algoritma z matriko AK^{-1} in začetnim vektorjem $v_1 = r_0/\|r_0\|$ da dobiš $H_{k+1,k}$ in V_{k+1}.
3) po metodi najmanjših kvadratov poišči y_k, ki minimizira $\|\|r_0\|e_1 - H_{k+1,k}y_k\|$.

Končni približek je potem $x_k = x_0 + M^{-1}V_k y_k$.

Na začetku ne potrebujemo z_0, saj je $r_0 = b - Ax_0 = r_0 - AM^{-1}z_0$.

Primerjava desnega in levega predpogojevanja

Pri levem predpogojevanju dobimo na koncu

$$x_k^L = x_0 + p(K^{-1}A)s_0,$$

kjer je $s_0 = K^{-1}r_0$ in je p_L tisti polinom stopnje $k-1$, $p_L(0) = 1$, ki minimizira

$$\|s_0 - K^{-1}Ap_L(K^{-1}A)s_0\|.$$

Če upoštevamo

$$s_0 - K^{-1}Ap_L(K^{-1}A)s_0 = K^{-1}(r_0 - Ap_L(K^{-1}A)K^{-1}r_0)$$

in dejstvo, da z vsak polinom velja

$$p(M^{-1}A)M^{-1}r = M^{-1}p(AM^{-1})r,$$

dobimo

$$s_0 - K^{-1}Ap_L(K^{-1}A)s_0 = K^{-1}(r_0 - AK^{-1}p_L(AK^{-1})r_0),$$

torej p_L minimizira

$$\|K^{-1}(r_0 - AK^{-1}p_L(AK^{-1})r_0)\|.$$

Primerjava desnega in levega predpogojevanja 2

Pri levem predpogojevanju dobimo

$$x_k^L = x_0 + p(K^{-1}A)s_0,$$

kjer je $s_0 = K^{-1}r_0$ in p_L minimizira

$$\|K^{-1}(r_0 - AK^{-1}p_L(AK^{-1})r_0)\|.$$

Pri desnem predpogojevanju dobimo na koncu

$$x_k^R = x_0 + K^{-1}p_R(AK^{-1})r_0,$$

kjer je p_R tisti polinom stopnje $k-1$, $p_R(0) = 1$, ki minimizira

$$\|r_0 - AK^{-1}p_R(AK^{-1})r_0\|.$$

Vektorja x_k^R in x_k^L sta iz istega afinega podprostora, saj je

$$x_0 + \mathcal{K}_k(K^{-1}A, s_0) = x_0 + K^{-1}\mathcal{K}_k(AK^{-1}, r_0),$$

pri čemer pri levem predpogojevanju minimiziramo $\|K^{-1}(r_0 - AK^{-1}p(AK^{-1})r_0)\|$, pri desnem pa $\|r_0 - AK^{-1}p(AK^{-1})r_0\|$ po vseh polinomih p stopnje $k-1$ z $p(0) = 1$.

Obojestransko predpogojevanje

Pri *obojestranskem predpogojevanju* namesto $Ax =$ rešujemo

$$K_1^{-1} A K_2^{-1} z = K_1^{-1} b,$$

kjer je $x = K_2^{-1} z$.

To je uporabno v primeru, ko imamo K podan v obliki razcepa $K = K_1 K_2$.

Npr., pri CG lahko v primeru, ko je znan razcep Choleskega $K = LL^T$, metodo CG uporabimo na s.p.d. sistemu

$$L^{-1} A L^{-T} y = L^{-1} b, \qquad x = L^{-T} y.$$

Prav tako lahko uporabimo CG na sistemu

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

Druga varianta je ekvivalentna uporabi levega predpogojevanja v kombinaciji s skalarnim produktom $[x, y] = \langle x, Ky \rangle$.

5.2 Izbira predpogojenke

Prva možnost je uporaba katere izmed "klasičnih"iterativnih metod. Pri teh pri reševanju $Ax = b$ 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 lahko 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 naslednje variante:

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

Izkaže se, da Gauss-Seidel in SOR ne prideta v poštev, saj neprimerno preslikata lastne vrednosti matrike A. Ostaneta Jacobi in SSOR, pri čemer optimalni ω ni tako pomemben.

Nepopolni razcep

Druga popularna možnost je uporaba *nepopolnega LU razcepa*.

Tu računamo LU razcep kot po osnovnem algoritmu, le da delamo neničelne elemente le na izbranih mestih (ponavadi tam, kjer ima A ničle). To pride prav pri razpršenih matrikah, kjer matriki L in U iz popolnega LU razcepa nista več razpršeni.

Pri nepopolnem LU razcepu na koncu sicer ne velja $A = LU$, je pa $K = LU$ lahko dobra matrika za predpogojevanje. Ker že poznamo njej LU razcep, se da sistem z matriko K rešiti na učinkovit način.

Denimo, da je $S \subset \{(i,j) \mid 1 \leq i,j \leq n\}$ množica indeksov, kjer v L in U dopuščamo neničelne elemente. Vedno mora S vsebovati vse diagonalne elemente.

Osnovna varianta nepopolnega LU razcepa, ki povozi A z matrikama L in U, je

Za vse $(i,j) \neq S$ nastavi $a_{ij} = 0$.
$j = 1, 2, \ldots, n-1$
  $i = j+1, \ldots, n$ (samo za $(i,j) \in S$)
    $a_{ij} = a_{ij}/a_{jj}$
    $k = j+1, \ldots, n$ (samo za $(i,k) \in S$)
      $a_{ik} = a_{ik} - a_{ij}a_{jk}$

Zero fill-in ILU

Ponovadi za S vzamemo indekse neničelnih elementov matrike A, oziroma

$$S = \{(i,j) \mid a_{ij} \neq 0\}.$$

V tem primeru je to razcep ILU(0) oziroma (zero fill-in) nepopolni LU.

Pri ILU(0) dobimo matriki L in U, ki imata neničelne elemente le na mestih S in za $M = LU$ velja $m_{ij} = a_{ij}$ za $(i,j) \in S$. Na ostalih mestih ima M lahko neničelne elemente tam, kjer jih A nima.

Z zgornjim pogojem ILU(0) ni enolično določen, običajno pa vzamemo kar razcep, ki ga vrne modificiran algoritem za LU razcep brez pivotiranja.

Množico indeksov lahko določimo tudi tako, da imata L in U več neničelnih elementov kot matrika A. Pri tem imamo več strategij:

- Pri ILU(p) določimo S rekurzivno tako, da zavzema vsa neničelna mesta matrike LU, ki jo dobimo iz ILU($p-1$) razcepa matrike A. Začnemo z ILU(0).
  Množico S lahko določimo vnaprej in ni potrebno računati p nepopolnih LU razcepov.
- Lahko se odločimo, da postavimo določeno mejo za velikost neničelnega elementa. Potem dopustimo tudi neničelne elemente izven S, če so zadosti veliki (kar pomeni, da so dovolj pomembni). Tu množico S določamo sproti med računanjem nepopolnega razcepa. Če uporabimo prag ∞, dobimo ILU(0), pri pragu 0 pa popolni LU razcep.

Ostale možnosti

Pri implementaciji je zelo pomembno, kako je razpršena matrika predstavljena, saj imamo različne variante LU razcepa glede na to, v kakšnem vrstnem redu uredimo tri zanke.

Tudi pri nepopolnem LU lahko pivotiramo.

Podobno kot nepopolni LU obstaja tudi nepopolni razcep Choleskega, ki ga uporabimo za simetrične pozitivno definitne matrike.

6. Na kaj moramo še paziti pri iterativnem reševanju sistemov

Ustavitveni kriterij: Ponavadi kot ustavitveni kriterij uporabimo $\|r_k\| \leq \epsilon$ za izbrani ϵ, radi pa bi imeli, da je norma $\|x - x_k\|$ dovolj majhna.

Pri numeričnem računanju je težava še ta, da ostanek in približek ponavadi posodabljamo kot

$$x_{k+1} = x_k + w_k, \qquad r_{k+1} = r_k - Aw_k.$$

Pri računanju r_k lahko pride do prištevanja vektorjev z veliko normo, zato se natančnost r_k lahko hitro izgubi. Tu izstopa npr. metoda CGS, kjer vmesni veliki ostanki lahko povzročijo v nadaljevanju velike napake.

To, da bi ostanek računali kot $r_k = b - Ax_k$, ni ne ekonomično ne natančneje. Namesto tega lahko posodobimo r_k samo v primeru, ko dobimo nov najmanjši ostanek. V tem primeru vsakič, ko dobimo nov minimalni ostanek, shranimo trenutni $z = x_k$ in vmes shranjujemo le vsoto skupnih popravkov za x_k v y_k. Ko dobimo nov minimalni ostanek, vzamemo $z = z + y_k$ in izračunamo $r = b - Az$.

Na kaj moramo še paziti pri iterativnem reševanju sistemov

Pri MINRES (in ostalih metodah, ki uporabljajo tričlenske rekurzivne formule), lahko pride do velikih zaokrožitvenih napak in bazni vektorji niso več ortogonalni.

Ker je CG v bistvu varianta D-Lanczosa za simetrične pozitivno definitne matrike, pri D-Lanczosu pa imamo prav tako tričlensko rekurzijo, lahko to pričakujemo tudi pri CG.

Izkaže se, da je situacija pri CG vseeno boljša, saj tu uporabljamo le dvočlenske rekurzije. Pri tročlenskih rekurzijah se pojavi parazitska rešitev, ki lahko pokvari in zamegli pravo rešitev, pri dvočlenski pa se to ne more zgoditi.