Eötvös Loránd Tudományegyetem

Természettudományi Kar

Hosszú Ádám Tamás

Rosszul kondicionált egyenletrendszerekre alkalmazott iterációs módszerek

BSc Szakdolgozat

Témavezető:

Dr. Gáspár Csaba

Numerikus Analízis Tanszék

Budapest, 2013

Köszönetnyilvánítás

Ezúton szeretném megköszönni Dr. Gáspár Csabának hogy elvállalta a témavezetői feladatkört, és a konzultációk során iránymutatásával, tanácsaival nagymértékben segítette a dolgozat elkészítését.

Tartalomjegyzék

1. Bevezetés **4**

2. Klasszikus iterációs módszerek **6**

2.1. Fixpont iteráció 6

2.2. Az egyszerű iteráció 7

2.3. A Jacobi-, és a Gauss-Seidel-iteráció 7

2.4. A konjugált gradiens módszer 8

3. Krylov-altérre épülő módszerek **10**

3.1. Az Arnoldi-iteráció 10

3.2. A GMRES-módszer 12

3.3. Az LSQR-módszer 16

3.4. A BiCG-módszer 21

3.5. BiCGStab 26

4. Összehasonlítás **30**

4.1. Az alapmegoldások módszere 30

5. Összefoglalás **36**

6. A módszerek MATLAB-kódjai **37**

1. fejezet

Bevezetés

A gyakorlatban sűrűn merülnek fel $Ax = b$ alakú lineáris egyenletrendszerek, melyeknek a b jobboldala mért adatokat tartalmaz. A mérési eljárás illetve a mérőműszerek hibája miatt a b vektor nem pontos. Ekkor felmerül a kérdés, hogy érdemes-e megkeresni az egyenletrendszer pontos megoldását, mivel a mért adatok pontatlansága következtében a megoldás nem oldja meg a felmerülő gyakorlati feladatot (pontosan). Továbbá gyakran az egyenletrendszer nagy méretű, ekkor a direkt megoldása nagy tárhelyet igényel, mely sokszor nem áll rendelkezésünkre. Ilyen esetekben szoktak iterációs módszereket alkalmazni. A következő fejezetben néhány, széles körben ismert, iterációs módszert említünk meg, melyekről bővebb leírást az [1] könyvben találhatunk.

A probléma nehezebben kezelhető, ha az egyenletrendszer jobb oldalának kis megváltoztatása esetén a megoldás nagy mértékben változik. Az ilyen egyenletrendszereket (és a hozzá tartozó A mátrixot is) rosszul kondicionáltnak nevezzük (ennek mérőszáma[1]: $cond(A) = \|A\|\|A^{-1}\|$ a mátrix kondíciószáma). Minél nagyobb egy mátrix kondíciószáma, annál nagyobb hibát eredményez a megoldásban a mért adatok hibája. Ilyen probléma sűrűn felmerül a gyakorlatban (például interpolációnál, differenciálegyenletek numerikus megoldásánál).

A 3. fejezetben három Krylov-alterekre épülő iterációs módszert mutatok be (GMRES, LSQR, BiCGStab), melyeket a [2] és [3] könyvek alapján dolgoztam fel. Mind a három módszert elterjedten alkalmazzák. A módszereket a 4. fejezetben alkalmazom, illetve hasonlítom össze egy elliptikus parciális differenciálegyenlet numerikus megoldásán keresztül. Ez a probléma is egy rosszul kondicionált egyenletrendszerre vezet, melynek érdekessége hogy nem szükséges a felmerülő rendszert pontosan megoldani, elég egy megfelelő közelítését megadni a megoldásnak. Ekkor a parciális differenciálegyenlet megoldása

még pontos lesz.

A módszerek összehasonlítására a MATLAB programot használtam, a módszerek MATLAB-beli kódjai a dolgozat végén olvashatóak.

2. fejezet

Klasszikus iterációs módszerek

2.1. Fixpont iteráció

2.1.1. Definíció. *Legyen X Banach-tér, $f_n : X \to X$ függvény. Iterációs sorozatnak nevezünk egy olyan $(x_n) \subset X$ sorozatot, melyet a következő rekurzióval kapunk:*

$$x_{n+1} = f_n(x_n)$$

Legyen $A \in \mathbb{R}^{N \times N}$ reguláris, és $b \in \mathbb{R}^N$. Tekintsük az $Ax = b$ lineáris egyenletrendszert, illetve ennek egy átalakítását: $Ax = b$ akkor és csak akkor teljesül, ha $x = Bx + f$, ahol $B \in \mathbb{R}^{N \times N}$, $f \in \mathbb{R}^N$.

2.1.2. Definíció. *Jelölje $\varrho(B)$ a B mátrix spektrálsugarát, azaz*

$$\varrho(B) := \max_{i=1..N} |\lambda_i|, \text{ ahol } \lambda_i \text{ a } B \text{ mátrix sajátértéke } (i=1..N)$$

2.1.3. Tétel. *Ha $\varrho(B) < 1$, akkor tetszőleges $x_0 \in \mathbb{R}^N$ kezdővektor esetén az $x_{n+1} = Bx_n + f$ iterációs sorozat konvergens, és az $x = Bx + f$ egyenlet egyetlen megoldásához tart.*

2.2. Az egyszerű iteráció

Legyen $A \in \mathbb{R}^{N\times N}$ reguláris, és $b \in \mathbb{R}^N$. $\forall \omega > 0$ esetén

$$\begin{aligned} Ax &= b \\ \omega Ax &= \omega b \\ x + \omega Ax &= x + \omega b \\ x &= (I - \omega A)x + \omega b \end{aligned}$$

2.2.1. Állítás. *Legyen A szimmetrikus, pozitív definit. Ekkor minden elég kis $\omega > 0$-ra az $x_{n+1} = (I - \omega A)x_n + \omega b$ tetszőleges $x_0 \in \mathbb{R}^N$ pontból indított iterációs sorozat konvergens, és határértéke az $Ax = b$ lineáris egyenletrendszer egyetlen megoldása.*

2.2.2. Megjegyzés. Elég kis ω alatt azt értjük, hogy $\varrho(I - \omega A)$ legyen 1-nél kisebb. Kiszámítható az optimális paraméterérték: $\omega := 2/(\lambda_N + \lambda_1)$, ahol λ_N az A mátrix legnagyobb, λ_1 a legkisebb sajátértéke (ekkor lesz $\|I - \omega A\|$ minimális). (Ekkor a 2.1.3 tétel értelmében a sorozat konvergens).

2.3. A Jacobi-, és a Gauss-Seidel-iteráció

Legyen $A \in \mathbb{R}^{N\times N}$ reguláris, és $b \in \mathbb{R}^N$. Tegyük fel továbbá hogy A diagonálelemei nem zérusok, azaz $a_{ii} \neq 0$, $i = 1...N$. Tekintsük az A mátrix következő felbontását: $A = L + D + U$, ahol L, illetve U az A mátrix szigorúan alsó, illetve felső része, D főátlója megegyezik A főátlójával, minden más eleme 0.

Jacobi-iteráció:

$$\begin{aligned} Ax &= b \\ (U + D + L)x &= b \\ Dx &= -(U + L)x + b \\ x &= -D^{-1}(U + L)x + D^{-1}b \end{aligned}$$

Az iterációs sorozat: $x_{n+1} = -D^{-1}(U+L)x_n + D^{-1}b$

Gauss-Seidel-iteráció:

$$\begin{aligned} Ax &= b \\ (U+D+L)x &= b \\ (L+D)x &= -Ux+b \\ x &= -(L+D)^{-1}Ux + (L+D)^{-1}b \end{aligned}$$

Az iterációs sorozat: $x_{n+1} = -(L+D)^{-1}Ux_n + (L+D)^{-1}b$

2.3.1. Megjegyzés. A feltételünk szerint D és $(L+D)$ invertálhatóak.

2.3.2. Definíció. *Egy $A \in \mathbb{R}^{N\times N}$ mátrixot soronként diagonálisan dominánsnak nevezünk, ha minden sorára igaz:*

$$|a_{ii}| > \sum_{j\neq i, j=1}^{N} |a_{ij}|$$

2.3.3. Tétel. *Legyen A diagonálisan domináns. Ekkor a Jacobi- és a Gauss-Seidel iterációk konvergensek, határértékük az egyenletrendszer egyetlen megoldása.*

2.3.4. Tétel. *Legyen A szimmetrikus, pozitív definit mátrix. Ekkor a Guass-Seidel iteráció konvergens.*

2.4. A konjugált gradiens módszer

Legyen $A \in \mathbb{R}^{N\times N}$ szimmetrikus, pozitív definit és $b \in \mathbb{R}^N$.

2.4.1. Állítás. *Jelöljük $< x, y >_A := < Ax, y >$. Az így definiált funkcionál skalárszorzat, ezt az A által generált skalárszorzatnak nevezzük. $\|x\|_A := \sqrt{< Ax, x >}$-et A-normának hívjuk. Amennyiben A szimmetrikus, a most definiált A-norma ekvivalens az eredetivel.*

2.4.2. Definíció. *$F(x) := < Ax, x > -2 < x, b >$. Ennek az F funkcionálnak egyetlen minimumhelye van, ez éppen az $Ax = b$ egyenletrendszer pontos megoldása. F-et energetikai funkcionálnak nevezzük.*

Variációs módszereknek nevezzük azon eljárásokat, amelyek F-et minimalizálják. A konjugált gradiens módszer is ebbe a csoportba tartozik.

A módszer alapgondolata hogy az n-edik lépésben minimalizáljuk az F funkcionált a d_n irányban (így kapjuk x_{n+1}-et), majd válasszunk egy új irányt úgy, hogy az új irány A-ortogonális legyen az előzőre (A-skalárszorzatuk 0).
Legyen $x_0 \in \mathbb{R}^N$, tetszőleges közelítés, $r_0 := Ax_0 - b$, $d_0 := -r_0$. Feltehetjük hogy $r_0 \neq 0$, hiszen ekkor már x_0 megoldása az egyenletrendszernek.

$$x_{n+1} := x_n + \lambda_n d_n, \text{ ahol } \lambda_n := -\frac{< r_n, d_n >}{\|d_n\|_A} \tag{2.1}$$

$$r_n := Ax_n - b \tag{2.2}$$

$$d_{n+1} := -r_{n+1} + \mu_n d_n, \text{ ahol } \mu_n := \frac{< r_{n+1}, d_n >_A}{\|d_n\|_A^2} \tag{2.3}$$

2.4.3. Tétel. *Az iteráció legfeljebb N lépés után véget ér.*

3. fejezet

Krylov-altérre épülő módszerek

Jelölje $\|.\|$ az Euklideszi-normát.

3.1. Az Arnoldi-iteráció

3.1.1. Definíció. *Krylov-altérnek nevezünk egy olyan alteret, amely a következő alakban áll elő:*

$$\kappa_m(A, v) = span\{v, Av, ..., A^{m-1}v\}$$

ahol $A \in \mathbb{R}^{N\times N}$, $v \in \mathbb{R}^N$. *Amennyiben nincs félreértés ezt* κ_m*-mel fogjuk jelölni.*

A definícióból könnyen látható, hogy $x \in \kappa_m$ akkor és csak akkor ha $x = p(A)v$ alakba írható, ahol p egy legfeljebb $m-1$-edfokú polinom. Az Arnoldi-iteráció felépít egy ortonormált bázist κ_m-ben. Erre többféle módszer létezik, az egyik:

3.1.2. Algoritmus. *Arnoldi*

1. *Válasszunk egy* $v_1 \in \kappa_m$ *vektort, amelyre* $\|v_1\| = 1$
2. *For* $j = 1...m$ *Do:*
3. $\quad v_{j+1} := Av_j$
4. $\quad$ *For* $i = 1...j$ *Do:*
5. $\quad\quad h_{i,j} :=< Av_i, v_j >$
6. $\quad\quad v_{j+1} := v_{j+1} - h_{i,j}v_i$
7. $\quad$ *EndDo*
8. $\quad h_{j+1,j} := \|v_{j+1}\|$
9. $\quad$ *Ha* $h_{j+1,j} = 0$ *akkor Stop*

10. $v_{j+1} := v_{j+1}/h_{j+1,j}$

11. EndDo

Az algoritmus minden lépésben kibővíti a bázist egy új vektorral, majd Gram-Schmidt ortogonalizációt használva ortonormalizálja az így kapott vektorrendszert.

3.1.3. Állítás. *Tegyük fel hogy az algoritmus nem állt le az m-edik lépésig. Az 3.1.2 algoritmus által megadott* $v_1, v_2, ..., v_m$ *vektorok egy ortonormált bázisát alkotják* κ_m*-nek.*

Bizonyítás. A konstrukció következtében a $v_1, v_2, ..., v_m$ vektorok ortonormált rendszert alkotnak. Generálják κ_m-et, ha $v_i = p_{i-1}(A)v_1$ alakba írható, ahol p_{i-1} egy legfeljebb $i-1$-edfokú polinom. Ezt teljes indukcióval látjuk be. $i = 1$-re triviális teljesül, a $p \equiv 1$ polinommal. Tegyük fel hogy minden $k \leq j$ indexre már tudjuk. Tekintsük a v_{j+1} vektort:

$$h_{j+1,j}v_{j+1} = Av_j - \sum_{k=1}^{j} h_{k,j}v_k = Ap_{j-1}(A)v_1 - \sum_{k=1}^{j} h_{k,j}p_{k-1}(A)v_1$$

ahol az utolsó egyenlőséget az indukciós feltétel felhasználásával kaptuk. Így $v_{j+1} = p_j(A)v_1$ alakban írtuk fel így az állítást igazoltuk. □

Jelöljük V_m-mel azt az ortogonális $N \times m$-es mátrixot, amelynek oszlopai a $v_1, v_2, ..., v_m$ vektorok, H_m-mel azt az $(m+1) \times m$-es mátrixot melynek nem nulla elemei a 3.1.2 algoritmusban definiált $h_{i,j}$ számok. Ekkor H_m felső-Hessenberg mátrix, alakja:

$$H_m = \begin{pmatrix} h_{1,1} & h_{1,2} & h_{1,3} & \cdots & h_{1,m} \\ h_{2,1} & h_{2,2} & h_{2,3} & \cdots & h_{2,m} \\ 0 & h_{3,2} & h_{3,3} & \cdots & h_{3,m} \\ 0 & 0 & h_{4,3} & \cdots & h_{4,m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & h_{m+1,m} \end{pmatrix}$$

3.1.4. Állítás. *Ekkor fennáll a következő:*

$$AV_m = V_{m+1}H_m$$

Bizonyítás. Tekintsük az AV_m mátrix j. oszlopát:

$$(AV_m)_j = Av_j = \sum_{i=1}^{j+1} h_{i,j}v_i = (V_{m+1}H_m)_j$$

ahol az első egyenlőség egyszerűen leolvasható a 3.1.2 algoritmusból. Ebből következik az állítás. □

3.2. A GMRES-módszer

A GMRES (Generalized Minimal Residual) módszer az Arnoldi-iterációra épül. Legyen

$$\kappa_m(A, r_0) = span\{r_0, Ar_0, ..., A^{m-1}r_0\}$$

ahol $r_0 = Ax_0 - b$, x_0 tetszőleges kezdővektor (általában $x_0 = 0$).
Minden $x \in \kappa_m$ vektor felírható $x = x_0 + V_m y$ alakban, ahol V_m oszlopai az Arnoldi iteráció által megadott ortonormált bázisvektorok (tetszőleges vektor helyett $v_1 := r_0/\|r_0\|$), $y \in \mathbb{R}^m$.
Definiáljuk a következő funkcionált:

$$F(y) = \|b - Ax\| = \|b - A(x_0 + V_m y)\| = \|r_0 - AV_m y\|$$

Felhasználva a 3.1.4 állítást, és hogy V_{m+1} ortogonális mátrix:

$$\|r_0 - AV_m y\| = \|\|r_0\|v_1 - V_{m+1}H_m y\| = \|V_{m+1}(\|r_0\|_2 e_1 - H_m y)\| = \|\|r_0\|_2 e_1 - H_m y\| \quad (3.1)$$

Ahol $e_1 = (1, 0, 0, ..., 0)^T \in \mathbb{R}^{m+1}$. Az x_m vektort tehát úgy kapjuk meg hogy megkeressük azt az y vektort, amely minimalizálja az $F(y)$ funkcionált, és $x_m := x_0 + V_m y$. Ez a következő algoritmust adja:

3.2.1. Algoritmus. *GMRES*

1. Válasszunk egy x_0 tetszőleges vektort, és $r_0 := Ax_0 - b$, $v_1 := r_0/\|r_0\|$
2. For $j = 1...m$ Do:
3. $v_{j+1} := Av_j$
4. *For $i = 1...j$ Do:*
5. $h_{i,j} :=< Av_i, v_j >$
6. $v_{j+1} := v_{j+1} - h_{i,j}v_i$
7. *EndDo*
8. $h_{j+1,j} := \|v_{j+1}\|$
9. *Ha $h_{j+1,j} = 0$ akkor $m := j$ és lépjünk a 12.sorra*
10. $v_{j+1} := v_{j+1}/h_{j+1,j}$
11. EndDo
12. Definiáljuk a H_m mátrixot.
13. Számítsuk ki az y vektort, amely minimalizálja az $F(y)$ funkcionált, majd $x_m := x_0 + V_m y$

Most foglalkozzunk az $F(y)$ funkcionál minimalizálásával:
Egy elterjedt eljárás, hogy Givens-forgatások alkalmazásával a H_m mátrixot felső háromszög alakúra hozzuk, majd az így kapott egyenletrendszert oldjuk meg. Definiáljuk a forgatási mátrixot:

$$P_i = \begin{pmatrix} 1 & & & & & & & \\ & \ddots & & & & & & \\ & & 1 & & & & & \\ & & & c_i & s_i & & & \\ & & & -s_i & c_i & & & \\ & & & & & 1 & & \\ & & & & & & \ddots & \\ & & & & & & & 1 \end{pmatrix}$$

ahol $(0, ...0, c_i, -s_i, 0, ..., 0)^T$ a mátrix i. oszlopa, $P_i \in \mathbb{R}^{(m+1)\times(m+1)}$,

$$s_i = \frac{h_{i+1,i}}{\sqrt{h_{i+1,i}^2 + (h_{i,i}^{(i-1)})^2}} \qquad c_i = \frac{h_{i,i}^{(i-1)}}{\sqrt{h_{i+1,i}^2 + (h_{i,i}^{(i-1)})^2}}$$

ahol $h_{i,i}^{(i-1)}$ a már $i-1$-szer elvégzett forgatás után kapott mátrix i. sorának i. eleme. Legyen $Q_m = P_m P_{m-1} ... P_1$. Így $Q_m H_m = \overline{R}_m$, ahol $\overline{R}_m \in \mathbb{R}^{(m+1)\times m}$, és

$$\overline{R}_m = \begin{pmatrix} R_m \\ 0 \end{pmatrix},$$

$R_m \in \mathbb{R}^{m\times m}$ felső-háromszög mátrix. Így

$$\begin{aligned} F(y) &= \|\|r_0\|e_1 - H_m y\| = \|Q_m\|r_0\|e_1 - Q_m H_m y\| \\ &= \|Q_m\|r_0\|e_1 - \overline{R}_m y\| = \|\overline{f}_m - \overline{R}_m y\| \end{aligned}$$

ahol $Q_m\|r_0\|e_1 = \overline{f}_m = (\beta_1, \beta_2, \cdots, \beta_{m+1})^T$, és legyen $f_m = (\beta_1, \beta_2, \cdots, \beta_m)^T$ (elhagytuk $\overline{f}_m$ utolsó koordinátáját). Jelöljük $\overline{f}_{m-1} = (\gamma_1, \gamma_2, \cdots, \gamma_m)$. Ekkor

$$\overline{f}_m = \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_{m+1} \end{pmatrix} = Q_m\|r_0\|e_1 = P_m Q_{m-1}\|r_0\|e_1 = P_m \begin{pmatrix} \gamma_1 \\ \gamma_2 \\ \vdots \\ \gamma_m \\ 0 \end{pmatrix} = \begin{pmatrix} \gamma_1 \\ \gamma_2 \\ \vdots \\ c_m\gamma_m \\ -s_m\gamma_m \end{pmatrix},$$

így speciálisan

$$\beta_{m+1} = -s_m\gamma_m \tag{3.2}$$

3.2.2. Állítás. *Legyen $m < N$, ekkor az eddigi jelöléseket használva:*
1. Az AV_m mátrix rangja megegyezik R_m rangjával, és ha $r_{m,m} = 0$, akkor A szinguláris.
2. Jelölje y_m az $\|\|r_0\|e_1 - H_m y\|$ minimumhelyét. Ekkor: $y_m = R_m^{-1} f_m$.
3. m lépés után:

$$\|b - Ax_m\| = |\beta_{m+1}|$$

Bizonyítás. 1. Felhasználva a 3.1.4 állítást és Q_m ortogonalitását:

$$AV_m = V_{m+1}H_m = V_{m+1}Q_m^T Q_m H_m = V_{m+1}Q_m^T \overline{R}_m$$

Mivel $V_{m+1}Q_m^T$ ortogonális, ezért AV_m rangja megegyezik $\overline{R}_m$ rangjával, amely egyenlő R_m rangjával, hiszen csak egy 0-sorban különböznek. Ha $r_{m,m} = 0$ akkor R_m rangja legfeljebb $m-1$, így AV_m rangja is legfeljebb $m-1$. Mivel $rang(V_m) = m$, ezért A szinguláris.
2. Az eddigi eredményeket, illetve a Pitagorasz-tételt felhasználva kapjuk:

$$\|\|r_0\|e_1 - H_m y\|^2 = \|f_m - R_m y\|^2 + |\beta_{m+1}|^2$$

Mivel R_m teljes rangú, ezért a jobb oldal minimumhelye: $y_m = R_m^{-1} f_m$.
3.

$$\|b - Ax_m\|^2 = \|b - A(x_0 + V_m y_m)\|^2 = \|f_m - R_m y_m\|^2 + |\beta_{m+1}|^2 = |\beta_{m+1}|^2$$

Felhasználva az eddigi eredményeket, illetve hogy $y_m = R_m^{-1} f_m$. □

3.2.3. Következmény. Ezt az algoritmust használva minden lépésben megkapjuk a hiba normáját, ez egy megfelelő leállási kritériumot szolgáltat.

Az algoritmusból leolvasható, hogy az m-edik lépés előtt a GMRES módszer csak a 9. sorban állhat le, azaz amikor az Arnoldi bázis felépítése során $v_j = 0$ valamely $j < m$ indexre.

3.2.4. Állítás. *Legyen A reguláris mátrix. Ekkor a GMRES algoritmus akkor és csak akkor áll le a j. lépésben (azaz $h_{j+1,j} = 0$), ha az x_j közelítő megoldás pontos.*

Bizonyítás. Tegyük fel először hogy $h_{j+1,j} = 0$. Mivel A reguláris, ezért $0 \neq r_{jj} = h_{jj}^{(j-1)}$, így

$$s_j = \frac{h_{j+1,j}}{\sqrt{h_{j+1,j}^2 + (h_{jj}^{(j-1)})^2}} = 0.$$

A 3.2.2 állítás első részéből illetve (3.2) egyenletből következik:

$$r_j = \|Ax - b\| = |\beta_{j+1}| = |s_j \gamma_j| = 0,$$

így x_j pontos.
Visszafele hasonlóan, ha x_j pontos, akkor $s_j = 0$, így $h_{j+1,j} = 0$. $\square$

A H_m mátrix felépítése $\mathcal{O}(m^2)$ lépésben történik, így m növelésével, a lépések száma kvadratikusan nő, továbbá az x_m vektor kiszámításához szükséges a V_m mátrix eltárolása, melynek mérete $N \times m$. Így nagyméretű egyenletrendszerek esetében a módszer memória-igénye nagy. Ennek feloldása hogy az algoritmust néhány lépés után újraindítjuk a kapott, jobb közelítésből.

3.2.5. Algoritmus. *Újraindított GMRES*

1. $r_0 := Ax_0 - b$, $v_1 := r_0/\|r_0\|$
2. *Készítsük el a Arnoldi bázist és a* H_m *mátrixot.*
3. *Számítsuk ki az* y *vektort, amely minimalizálja* $F(y)$*-t, majd* $x_m := x_0 + V_m y$
4. *Ha* x_m *megfelelő közelítése a pontos megoldásnak megállunk,*
   ha nem, $x_0 := x_m$*, és ugorjunk az 1. sorra*

3.2.6. Megjegyzés. Egy ismert nehézség az újraindított GMRES módszerrel, hogy ha az A mátrix nem pozitív definit, akkor stagnálhat (azaz a hiba normája nem csökken).

3.3. Az LSQR-módszer

Legyen $A \in \mathbb{R}^{N\times M}$, $b \in \mathbb{R}^N$, tekintsük az $Ax = b$ egyenletrendszert. Mivel az A mátrix nem négyzetes, így az egyenletrendszernek ritkán van pontos megoldása. Minimalizálhatjuk azonban $\|Ax-b\|$-t, ezt nevezik a legkisebb négyzetek feladatának. Tekintsük a következő bidiagonalizációs módszert:

3.3.1. Algoritmus. *Bidiagonalizálás*

1. Legyen $\beta_1 u_1 = b$ *és* $\alpha_1 v_1 = A^T u_1$
2. For i*=1,2,*$\cdots$ *Do*
3. $\beta_{i+1} u_{i+1} = Av_i - \alpha_i u_i$
4. Ha $u_{i+1} = 0$ *akkor Stop*
4. $\alpha_{i+1} v_{i+1} = A^T u_{i+1} - \beta_{i+1} v_i$
5. Ha $v_{i+1} = 0$ *akkor Stop*
5. EndDo

Ahol $i = 1, 2, ...$-re az $\beta_i, \alpha_i \geq 0$ számokat úgy választjuk hogy $\|u_i\| = \|v_i\| = 1$ legyen (illetve ha $u_i = 0$, akkor $\beta_i := 0$, hasonlóan ha $v_i = 0$, akkor $\alpha_i := 0$). Jelölje $U_k \in \mathbb{R}^{N\times k}$ mátrixot, melynek oszlopai az u_i, $i = 1, ..., k$ vektorok. Hasonlóan jelölje $V_k \in \mathbb{R}^{M\times k}$ azt a mátrixot, melynek oszlopai a v_i vektorok.

3.3.2. Állítás. *U_k és V_k ortogonális mátrixok.*

Bizonyítás. Azt kell tehát belátni hogy $< u_i, u_j >=< v_i, v_j >= 0$ minden $i < j$ párra, illetve $\|u_i\| = \|v_i\| = 1$ minden $i = 1, 2, ...$-re. $\|u_i\| = \|v_i\| = 1$ triviálisan következik α_i, β_i választásából. Az ortogonalitást teljes indukcióval látjuk be. Először tekintsük:

$$< u_1, u_2 > = < u_1, \frac{Av_1 - \alpha_1 u_1}{\|Av_1 - \alpha_1 u_1\|} >= \frac{1}{\|Av_1 - \alpha_1 u_1\|}(< u_1, Av_1 > -\alpha_1 \|u_1\|^2) =$$
$$= \frac{1}{\|Av_1 - \alpha_1 u_1\|}(< A^T u_1, v_1 > -\alpha_1) = \frac{1}{\|Av_1 - \alpha_1 u_1\|}(< \alpha_1 v_1, v_1 > -\alpha_1) =$$
$$= \frac{1}{\|Av_1 - \alpha_1 u_1\|}(\alpha_1 - \alpha_1) = 0$$

$$
\begin{aligned}
< v_1, v_2 > &= < v_1, \frac{A^T u_2 - \beta_2 v_1}{\|A^T u_2 - \beta_2 v_1\|} >= \frac{1}{\|A^T u_2 - \beta_2 v_1\|}(< v_1, A^T u_2 > -\beta_2 \|v_1\|^2) = \\
&= \frac{1}{\|A^T u_2 - \beta_2 v_1\|}(< Av_1, u_2 > -\beta_2) = \\
&= \frac{1}{\|A^T u_2 - \beta_2 v_1\|}(< \beta_2 u_2 + \alpha_1 u_1, u_2 > -\beta_2) = \\
&= \frac{1}{\|A^T u_2 - \beta_2 v_1\|}(\beta_2 \|u_2\|^2 + \alpha_1 < u_1, u_2 > -\beta_2) = 0
\end{aligned}
$$

Most tegyük fel hogy minden $i, j \leq k,\ i < j$ indexpárra teljesül $< u_i, u_j >=< v_i, v_j >= 0$, valamely $k > 2$ természetes számra. Ekkor

$$
< u_i, u_{k+1} >=< u_i, \frac{Av_k - \alpha_k u_k}{\|Av_k - \alpha_k u_k\|} >= \frac{< A^T u_i, v_k > -\alpha_k < u_i, u_k >}{\|Av_k - \alpha_k u_k\|}
$$

Ha $i = 1$, akkor $A^T u_1 = \alpha_1 v_1$, így

$$
< A^T u_1, v_k > -\alpha_k < u_1, u_k >= \alpha_1 < v_1, v_k > -\alpha_k < u_1, u_k >= 0
$$

az indukciós feltevés miatt, így $< u_1, u_{k+1} >= 0$.
Ha $i \neq 1$:

$$
< u_i, u_{k+1} > = \frac{< A^T u_i, v_k > -\alpha_k < u_i, u_k >}{\|Av_k - \alpha_k u_k\|} = \frac{< \alpha_i v_i + \beta_i v_{i-1}, v_k > -\alpha_k < u_i, u_k >}{\|Av_k - \alpha_k u_k\|}
$$

Ha $i \neq k$, akkor az indukciós feltevés miatt $< v_i, v_k >=< v_{i-1}, v_k >=< u_i, u_k >= 0$, így $< u_i, u_{k+1} >= 0$. Ha $i = k$:

$$
< u_k, u_{k+1} >= \frac{< \alpha_k v_k + \beta_k v_{k-1}, v_k > -\alpha_k < u_k, u_k >}{\|Av_k - \alpha_k u_k\|} = \frac{\alpha_k \|v_k\|^2 - \alpha_k \|u_k\|^2}{\|Av_k - \alpha_k u_k\|} = 0
$$

Hasonlóan:

$$
\begin{aligned}
< v_i, v_{k+1} > &= < v_i, \frac{A^T u_{k+1} - \beta_{k+1} v_k}{\|A^T u_{k+1} - \beta_{k+1} v_k\|} >= \frac{< Av_i, u_{k+1} > -\beta_{k+1} < v_i, v_k >}{\|A^T u_{k+1} - \beta_{k+1} v_k\|} = \\
&= \frac{< \beta_{i+1} u_{i+1} + \alpha_i u_i, u_{k+1} > -\beta_{k+1} < v_i, v_k >}{\|A^T u_{k+1} - \beta_{k+1} v_k\|}
\end{aligned}
$$

Mivel már tudjuk hogy $< u_i, u_{k+1} >= 0$ minden $i \leq k$, így ha $i < k$, akkor $< u_{i+1}, u_{k+1} >= < u_i, u_{k+1} >=< v_i, v_k >= 0$, így $< v_i, v_{k+1} >= 0$.
Ha $i = k$:

$$
< v_k, v_{k+1} >= \frac{< \beta_{k+1} u_{k+1} + \alpha_k u_k, u_{k+1} > -\beta_{k+1}}{\|A^T u_{k+1} - \beta_{k+1} v_k\|} = \frac{\beta_{k+1} - \beta_{k+1}}{\|A^T u_{k+1} - \beta_{k+1} v_k\|} = 0
$$

Így U_k, V_k ortogonális mátrixok. $\square$

3.3.3. Állítás. *A $v_1, v_2, ...v_k$ vektorok egy ortonormált bázisát alkotják $\kappa_k(A^TA, b)$-nek.*

Bizonyítás. Már láttuk hogy a $v_1, v_2, ..., v_k$ vektorok ortonormált rendszert alkotnak, így csak annyit kell belátni hogy mindegyik vektor előáll $v_i = p_{i-1}(A^TA)v_1$ alakban, ahol p_{i-1} egy legfeljebb $i-1$-edfokú polinom. Teljes indukcióval látjuk be, v_1-re triviálisan teljesül az állítás a $p \equiv 1$ polinommal. Tegyük fel hogy valamely j indexig már tudjuk. Ekkor

$$\begin{aligned}
\alpha_{j+1}v_{j+1} &= A^Tu_{j+1} - \beta_{j+1}v_j = \frac{A^TAv_j}{\beta_{j+1}} - \frac{\alpha_j}{\beta_{j+1}}A^Tu_j - \beta_{j+1}v_j = \\
&= \frac{A^TAv_j}{\beta_{j+1}} - \frac{\alpha_j^2}{\beta_{j+1}}v_j - \frac{\alpha_j\beta_j}{\beta_{j+1}}v_{j-1} - \beta_{j+1}v_j = \\
&= A^TAp_{j-1}(A^TA)v_1 - p'_{j-1}(A^TA)v_1 - p_{j-2}(A^TA)v_1
\end{aligned}$$

amely egy kívánt felírása a vektornak. □

A 3.3.1 algoritmus összefüggései felírhatók mátrixos alakban is:

$$U_{k+1}(\beta_1 e_1) = b \tag{3.3}$$

$$AV_k = U_{k+1}B_k \tag{3.4}$$

$$A^TU_{k+1} = V_kB_k^T + \alpha_{k+1}v_{k+1}e_{k+1}^T \tag{3.5}$$

ahol $e_i \in \mathbb{R}^{k+1}$, $B_k \in \mathbb{R}^{(k+1)\times k}$ és

$$e_i = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \leftarrow i.sor \qquad B_k = \begin{pmatrix} \alpha_1 & 0 & 0 & \cdots & 0 \\ \beta_2 & \alpha_2 & 0 & \cdots & 0 \\ 0 & \beta_3 & \alpha_3 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \beta_k & \alpha_k \\ 0 & \cdots & 0 & 0 & \beta_{k+1} \end{pmatrix}$$

Minimalizáljuk $\|b - Ax\|$-t az $x \in \kappa_m$ vektorok körében. Ekkor $x = V_my_m$ alakba írható. Így

$$\|b - Ax\| = \|AV_my_m - b\| = \|U_{m+1}B_my_m - U_{m+1}\beta_1e_1\| = \|B_my_m - \beta_1e_1\|$$

A GMRES módszernél látottakhoz hasonlóan forgatásokkal eliminálják a főátló alatti elemeket (β_i). Példaként bemutatjuk az i. forgatást. Az $(i-1)$. forgatás után kapott

mátrix:

$$Q_{i-1}B'_{i-1} = \begin{pmatrix} \theta_1 & \phi_2 & & & \\ & \theta_2 & \phi_3 & & \\ & & \ddots & \ddots & \\ & & & \theta_{i-1} & \phi_i \\ & & & & \overline{\theta}_i \end{pmatrix}$$

ahol $Q_{i-1} = P_{i-1}\begin{pmatrix} Q_{i-2} & 0 \\ 0 & 1 \end{pmatrix}$, és P_{i-1} forgatási mátrix, $Q_1 = P_1$ (így Q_{i-1} forgatások kompozíciója), és B'_{i-1}-t úgy kapjuk meg B_{i-1}-ből hogy hozzávesszük utolsó oszlopnak $(0, .., 0, \alpha_i)^T$-t. Így az i. forgatás:

$$\begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & & c_i & s_i \\ & & & -si & c_i \end{pmatrix} \begin{pmatrix} \theta_1 & \phi_2 & & & \\ & \ddots & \ddots & & \\ & & \theta_{i-1} & \phi_i & \\ & & & \overline{\theta}_i & \\ & & & \beta_{i+1} & \alpha_{i+1} \end{pmatrix} = \begin{pmatrix} \theta_1 & \phi_2 & & & \\ & \ddots & \ddots & & \\ & & \theta_{i-1} & \phi_i & \\ & & & \theta_i & \phi_{i+1} \\ & & & & \overline{\theta}_{i+1} \end{pmatrix}$$

Itt a felülvonással jelölt értékek a következő iterációban még megváltozhatnak, a felülvonás nélküliek már nem. Ebből könnyen látható hogy R_{k-1} és R_k csak egy új sorban és oszlopban különböznek (R_k bal felső $(k-1)\times(k-1)$-es blokkja R_{k-1}). Ekkor

$$\|\beta_1 e_1 - B_m y_m\| = \|Q_m \beta_1 e_1 - \overline{R}_m y_m\| = \|\overline{g}_m - \overline{R}_m y_m\|$$

ahol $Q_m B_m = \overline{R}_m = \begin{pmatrix} R_m \\ 0 \end{pmatrix}$ és $Q_m \beta_1 e_1 = \overline{g}_m$, és g_m-et úgy kapjuk hogy elhagyjuk $\overline{g}_m$ utolsó elemét. Az előbbihez hasonlóan adódik hogy g_m első k eleme megegyezik g_{m-1}-el. Amennyiben R_m rangja teljes: $y_m = R_m^{-1} g_m$ és $r_k = \|b - Ax\| = \|\overline{g}_m - \overline{R}_m y_m\| = |\gamma_{m+1}|$. Azonban y_{m-1} és y_m elemei általában különböznek, így minden lépésben újra kéne számolni a vektort. Egy egyszerű lépéssel azonban megkerülhető y_m kiszámítása. Tekintsük tehát: $x_m = V_m y_m = V_m R_m^{-1} g_m = D_m g_m$, ahol D_m oszlopait az $R_m^T D_m^T = V_m^T$ rendszerből kaphatjuk meg. Mivel $d_0 = x_0 = 0$, így

$$d_k = \frac{1}{\theta_k}(v_k - \phi_k d_{k-1})$$
$$x_k = x_{k-1} + \gamma_k d_k$$

Mivel a k. lépés után θ_k, ϕ_k és γ_k sem változik így, nem kell megjegyeznünk csak az utolsó iterációban kapott értékeket.

Ismeretes hogy ha egy x vektor minimalizálja az $\|Ax-b\|$-t, akkor megoldása a Gauss-féle normálegyenletnek, azaz $A^TAx = A^Tb$, így $\|A^T(Ax-b)\| = \|A^Tr\| = 0$, ahol $r = Ax-b$. Így a legkisebb négyzetek feladatánál $\|A^Tr_m\|$ megfelelő megállási kritériumot szolgáltathat. Az LSQR-módszernél ez szinte ingyen adódik:

$$\begin{aligned} A^Tr_m &= A^T(Ax_m-b) = A^T(AV_my_m-b) = A^TU_{m+1}(B_my_m-\beta_1e_1) = \\ &= \gamma_{m+1}A^TU_{m+1}Q_m^Te_{m+1} = \gamma_{m+1}(V_mB_m^T+\alpha_{m+1}v_{m+1}e_{m+1}^T)Q_m^Te_{m+1} = \\ &= \gamma_{m+1}V_m\overline{R}_m^Te_{m+1}+\gamma_{m+1}\alpha_{m+1}v_{m+1}e_{m+1}^TQ_m^Te_{m+1} = \gamma_{m+1}\alpha_{m+1}c_mv_{m+1} \end{aligned}$$

hiszen, $\overline{R}_m^Te_{m+1} = (0,...,0)^T$ és $e_{m+1}^TQ_m^Te_{m+1} = c_m$.
Ezek alapján:

$$\|A^Tr_m\| = \alpha_{m+1}|\gamma_{m+1}||c_m| \tag{3.6}$$

3.3.4. Algoritmus. *LSQR*

1. Legyen $\beta_1u_1 = b$, $\alpha_1v_1 = A^Tu_1$, $w_1 = v_1$, $x_0 = 0$, $\overline{\gamma}_1 = \beta_1$, $\overline{\theta}_1 = \alpha_1$
2. For i=1,2,... Do
3. $\beta_{i+1}u_{i+1} = Av_i - \alpha_iu_i$
4. $\alpha_{i+1}v_{i+1} = A^Tu_{i+1} - \beta_{i+1}v_i$
5. $\theta_i = \sqrt{\overline{\theta}_i^2+\beta_{i+1}^2}$
6. $c_i = \overline{\theta}_i/\theta_i$
7. $s_i = \beta_{i+1}/\theta_i$
8. $\gamma_i = c_i\overline{\gamma}_i$
9. $\overline{\gamma}_{i+1} = -s_i\overline{\gamma}_i$
10. $\overline{\theta}_{i+1} = c_i\alpha_{i+1}$
11. $\phi_{i+1} = s_i\alpha_{i+1}$
12. $x_i = x_{i-1}+(\gamma_i/\theta_i)w_i$
13. $w_{i+1} = v_{i+1}-(\phi_{i+1}/\theta_i)w_i$
14. Ha $\|A^Tr_i\|$ *elég kicsi akkor Stop*
15. EndDo

M lépés után az algoritmus biztosan véget ér, hiszen $\kappa_M(A^TA,b) = R^M$, így ebben a vektortérben minimalizáljuk $\|r_M\|$-t, így x_M biztosan megoldása a Gauss-féle normálegyenletnek, ebből következendően $\|A^T(Ax_M-b)\| = 0$, így az algoritmus leáll, x_M a keresett megoldás.

3.3.5. Megjegyzés. A számítógép számábrázolási hibájának következtében, a gyakorlatban az algoritmus nem biztos hogy leáll M lépés után.

3.3.6. Állítás. *Legyen $k < M$ a legkisebb olyan index, amire $\alpha_{k+1} = 0$. Ekkor x_k megoldása a Gauss-féle normálegyenletnek, így x_k minimalizálja $\|r\|$-t.*

Bizonyítás. A 3.6 egyenletet felhasználva:

$$\|A^T r_k\| = \alpha_{k+1}|\gamma_{k+1}||c_k|$$

Itt $\gamma_{k+1} = -s_i\gamma_k = -(\beta_{k+1}/\theta_k)\gamma_k$, amely értelmes, hiszen

$$\theta_k = \sqrt{\overline{\theta}_k^2 + \beta_{k+1}^2} = \sqrt{(c_{k-1}\alpha_k)^2 + \beta_{k+1}^2}$$

ahol $\alpha_k \neq 0$, a feltétel miatt, $c_{k-1} = \overline{\theta}_{k-1}/\theta_{k-1}$, és $\theta_{k-1} = 0$ csak akkor ha $\beta_k = 0$, csak akkor ha $\alpha_k = 0$, ami ellentmond a feltételnek, így $\theta_k \neq 0$.
$c_k = \overline{\theta}_k/\theta_k$, ahol $\theta_k \neq 0$ az előbbiek szerint.
Így $\|A^T r_k\| = \alpha_{k+1}|\gamma_{k+1}||c_k|$ értelmes és $\|A^T r_k\| = 0$, hiszen $\alpha_{k+1} = 0$. $\square$

3.4. A BiCG-módszer

Legyen most $A \in \mathbb{R}^{N\times N}$, $b \in \mathbb{R}^N$. A BiCG (bikonjugált gradiens) módszer nem követeli meg az A mátrix szimmetrikusságát, mint a konjugált gradiens módszer. Tekintsük a következő biortogonalizációs eljárást:

3.4.1. Algoritmus. *Lanczos-biortogonalizáció*

1. Válasszunk v_1, $w_1 \in \mathbb{R}^N$ vektorokat, úgy hogy $< v_1, w_1 >:= 1$
2. Legyen $\beta_1 = \delta_1 = 0$, $w_0 = v_0 = 0$.
3. For j=1,2,..,m Do:
4. $\alpha_j =< Av_j, w_j >$
5. $\overline{v}_{j+1} = Av_j - \alpha_j v_j - \beta_j v_{j-1}$
6. $\overline{w}_{j+1} = A^T w_j - \alpha_j w_j - \delta_j w_{j-1}$
7. $\delta_{j+1} = \sqrt{| < \overline{v}_{j+1}, \overline{w}_{j+1} > |}$ *Ha* $\delta_{j+1} = 0$ *akkor Stop*
8. $\beta_{j+1} =< \overline{v}_{j+1}, \overline{w}_{j+1} > /\delta_{j+1}$
9. $w_{j+1} = \overline{w}_{j+1}/\beta_{j+1}$
10. $v_{j+1} = \overline{v}_{j+1}/\delta_{j+1}$
11. EndDo

3.4.2. Megjegyzés. A β_i, δ_i skalárok választása több módon is történhet, annak kell teljesülnie, hogy $\delta_i\beta_i =< \overline{v}_i, \overline{w}_i >$, ez biztosítja hogy $< v_i, w_i >= 1$ legyen. Az algoritmusban azért így választottuk, mert így $\delta_i = |\beta_i|$.

Az algoritmusból leolvasható, hogy $v_i \in \kappa_m(A, v_1)$, illetve $w_i \in \kappa_m(A^T, w_1)$. Továbbá a $v_1, v_2, ...v_m$ vektorok egy bázisát alkotják $\kappa_m(A, v_1)$-nek, a $w_1, w_2, ..., w_m$ vektorok $\kappa_m(A^T, w_1)$-nek.

3.4.3. Állítás. *Ha az algoritmus nem áll le az m-edik lépés előtt, akkor a $v_1, v_2, ..., v_m$ és a $w_1, w_2, ..., w_m$ vektorok egy biortogonális rendszert alkotnak, azaz*

$$< v_i, w_j >= \delta_{ij} \qquad i, j \leq m$$

ahol δ_{ij} a Kronecker-delta függvényt jelöli, azaz

$$\delta_{ij} = \begin{cases} 0 & ha\ i \neq j \\ 1 & ha\ i = j \end{cases}$$

Bizonyítás. Az állítást teljes indukcióval fogjuk bizonyítani. A β_j, δ_j skalárok megválasztásából következik hogy $< v_j, w_j >= 1$. Egyszerűen kapjuk:

$$< v_2, w_1 >=< Av_1 - \alpha_1 v_1, w_1 >=< Av_1, w_1 > -\alpha_1 < v_1, w_1 >= 0$$

α_1 definíciója következtében. Hasonlóan $< w_2, v_1 >= 0$. Tegyük most fel hogy a $v_1, v_2, ..., v_j$ és a $w_1, w_2, ..., w_j$ biortogonális rendszert alkotnak, belátjuk először hogy $< v_{j+1}, w_i >= 0$ minden $i \leq j$ indexre. Ha $i = j$:

$$\begin{aligned} < v_{j+1}, w_j > &= \frac{1}{\delta_{j+1}} < Av_j - \alpha_j v_j - \beta_j v_{j-1}, w_j >= \\ &= \frac{1}{\delta_{j+1}}(< Av_j, w_j > -\alpha_j < v_j, w_j > -\beta_j < v_{j-1}, w_j >) = 0 \end{aligned}$$

α_j definíciója, $< v_j, w_j >= 1$ miatt az első két skalárszorzat különbsége 0, az indukciós feltevést felhasználva a harmadik tag 0.

$$\begin{aligned} < v_{j+1}, w_i > &= \frac{1}{\delta_{j+1}}(< Av_j, w_i > -\alpha_j < v_j, w_i > -\beta_j < v_{j-1}, w_i >) = \\ &= \frac{1}{\delta_{j+1}}(< v_j, A^T w_i > -\beta_j < v_{j-1}, w_i >) = \\ &= \frac{1}{\delta_{j+1}}(< v_j, \beta_{i+1} w_{i+1} + \alpha_i w_i - \delta_i w_{i-1} > -\beta_j < v_{j-1}, w_i >) \end{aligned}$$

$i < j-1$-re a fenti skalárszorzatok eltűnnek, ha $i = j-1$:

$$\begin{aligned} < v_{j+1}, w_{j-1} > &= \frac{1}{\delta_{j+1}}(< v_j, \beta_j w_j + \alpha_{j-1} w_{j-1} - \delta_{j-1} w_{j-2} > - \beta_j < v_{j-1}, w_{j-1} >) = \\ &= \frac{1}{\delta_{j+1}}(\beta_j < v_j, w_j > - \beta_j < v_{j-1}, w_{j-1} >) = 0. \end{aligned}$$

felhasználva az indukciós feltételt, illetve hogy $< v_i, w_i >= 1$ minden $i \leq j$ indexre. Hasonlóan belátható hogy $< w_{j+1}, v_i >= 0$ minden $i \leq j$ indexre, ezzel az állítást beláttuk. □

Jelölje $T_m \in \mathbb{R}^{m \times m}$ a következő tridiagonális mátrixot:

$$T_m = \begin{pmatrix} \alpha_1 & \beta_2 & & & \\ \delta_2 & \alpha_2 & \beta_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \delta_{m-1} & \alpha_{m-1} & \beta_m \\ & & & \delta_m & \alpha_m \end{pmatrix}$$

A 3.4.1 algoritmus alapján felírhatók a következő összefüggések:

$$AV_m = V_m T_m + \delta_{m+1} v_{m+1} e_m^T \tag{3.7}$$

$$A^T W_m = W_m T_m^T + \beta_{m+1} w_{m+1} e_m^T \tag{3.8}$$

$$W_m^T A V_m = T_m$$

ahol V_m, W_m rendre azokat a mátrixokat jelölik, amelynek oszlopai a $v_1, v_2, ..., v_m$, illetve a $w_1, w_2, ..., w_m$ vektorok, $(0, ...0, 1) = e_m^T \in \mathbb{R}^m$.

Vegyük észre, hogy hasonló műveleteket végzünk az A illetve az A^T mátrixokkal, valójában két lineáris egyenletrendszert oldunk meg egyszerre, egyet az A, egyet az A^T mátrixszal. Így ha létezik egy $A^T x^* = b^*$ egyenletrendszer amit meg kell oldanunk, akkor ez a módszer megfelelő.

Válasszunk egy $x_0 \in \mathbb{R}^N$ kezdő közelítést. Legyen $r_0 = b - Ax_0$, és válasszuk $v_1 = r_0/\|r_0\|$-nak. Hasonlóan az előző módszereknél látottakhoz az x_m közelítő megoldást $x_m = x_0 + V_m y_m$ alakban keressük. Így:

$$\|b - Ax_m\| = \|r_0 - AV_m y_m\| = \|V_{m+1}\beta_1 e_1 - V_{m+1}\overline{T}_m y_m\| = \|\beta_1 e_1 - \overline{T}_m y_m\|$$

ahol $\beta_1 = \|r_0\|$, és

$$\overline{T}_m = \begin{pmatrix} T_m \\ \delta_{m+1} e_m^T \end{pmatrix}$$

Mivel $\overline{T}_m$ teljes rangú (ha $\delta_{j+1} \neq 0$), így $y_m = T_m^{-1}(\beta_1 e_1)$. Tekintsük a T_m mátrix LU-felbontását, amely T_m speciális alakja miatt a következőképpen írható fel:

$$T_m = L_m U_m = \begin{pmatrix} 1 & & & & \\ \lambda_2 & 1 & & & \\ & \lambda_3 & 1 & & \\ & & \ddots & \ddots & \\ & & & \lambda_m & 1 \end{pmatrix} \begin{pmatrix} \eta_1 & \beta_2 & & & \\ & \eta_2 & \beta_3 & & \\ & & \ddots & \ddots & \\ & & & \eta_{m-1} & \beta_m \\ & & & & \eta_m \end{pmatrix}$$

Ekkor

$$x_m = x_0 + V_m y_m = x_0 + V_m T_m^{-1}(\beta_1 e_1) = x_0 + V_m U_m^{-1} L_m^{-1}(\beta_1 e_1) = x_0 + P_m L_m^{-1}(\beta_1 e_1)$$

ahol $P_m = V_m U_m^{-1}$, azaz $P_m U_m = V_m$. Tekintsük a szorzat utolsó oszlopát. U_m speciális alakja miatt:

$$u_{(m-1)m} p_{m-1} + u_{mm} p_m = \beta_m p_{m-1} + \eta_m p_m = v_m$$

így p_m az alábbi képlettel egyszerűen számítható a p_{m-1} illetve v_m vektorokból:

$$p_m = \frac{v_m - \beta_m p_{m-1}}{\eta_m}$$

Itt β_m-et a 3.4.1 algoritmus adja, λ_m-et és η_m-et pedig a Gauss-elimináció m-edik lépéséből kapjuk, azaz

$$\lambda_m = \frac{\beta_m}{\eta_{m-1}} \qquad \eta_m = \alpha_m - \lambda_m \beta_m$$

Jelölje $z_m = L_m^{-1}(\beta_1 e_1)$, így az előbbihez hasonlóan $L_m z_m = \beta_1 e_1$, L_m speciális alakja miatt:

$$z_m = \begin{pmatrix} z_{m-1} \\ \zeta_m \end{pmatrix}$$

és $\zeta_m = -\lambda_m \zeta_{m-1}$. Ennek következményeképpen x_m-et minden iterációs lépésben lehet frissíteni:

$$x_m = x_{m-1} + \zeta_m p_m$$

Jelölje $r_j = b - Ax_j$-t, $r_j^* = b^* - A^T x_j^*$ (a korábban említett duál egyenletrendszer).

3.4.4. Állítás. *Minden r_j és r_j^* vektor felírható $r_j = \sigma_{j+1} v_{j+1}$ és $r_j^* = \theta_{j+1} w_{j+1}$ alakban, így az $r_0, r_1, ..., r_j$ és az $r_0^*, r_1^*, ..., r_j^*$ vektorok biortogonális rendszert alkotnak.*

Bizonyítás. Felírva r_j definícióját, illetve felhasználva a (3.7) egyenletet kapjuk:

$$r_j = b - Ax_j = b - AV_j y_j = b - V_j T_j y_j + \delta_{j+1} v_{j+1} e_j^T y_j = \sigma_{j+1} v_{j+1}$$

Hasonlóan, a (3.8)-as egyenletet felhasználva kapjuk az állítást r_j^*-ra. $\square$
Definiáljuk a $P_m^* = W_m (L_m^{-1})^T$ mátrixot. A p_m vektorokhoz hasonlóan belátható hogy p_m^* lineáris kombinációja w_m-nek és p_{m-1}-nek.

3.4.5. Állítás. *Minden $i \neq j$ indexre $< Ap_j, p_i^* >= 0$.*

Bizonyítás. $(P_m^*)^T A P_m = L_m^{-1} W_m A V_m U_m^{-1} = L_m^{-1} T_m U_m^{-1} = I$ $\square$
Tudjuk hogy $x_{m+1} = x_m + \mu_m p_m$. Ebből következik hogy

$$r_{m+1} = b - Ax_{m+1} = b - Ax_m - \mu_m Ap_m = r_m - \mu_m Ap_m$$

hasonlóan

$$r_{m+1}^* = r_m^* - \mu_m Ap_m^*.$$

A 3.4.4 állítás következtében teljesülnie kell:

$$0 =< r_{m+1}, r_m^* >=< r_m - \mu_m Ap_m, r_m^* >$$

amiből kapjuk:

$$\mu_m = \frac{< r_m, r_m^* >}{< Ap_m, r_m^* >}$$

Tudjuk továbbá hogy p_{m+1}^* lineáris kombinációja r_{m+1}^*-nak és p_m^*-nak (mivel r_{m+1}^* egy irányba esik w_{m+1}-el). A p_m^* vektorokat megfelelő skalárral osztva kapjuk hogy:

$$p_{m+1}^* = r_{m+1}^* + \theta_m p_m^*$$

Ebből, felhasználva a 3.4.5 állítást:

$$< Ap_m, r_m^* >=< Ap_m, p_m^* - \theta_m p_{m-1}^* >=< Ap_m, p_m^* >$$

Szintén a 3.4.5 állításból:

$$0 =< p_{m+1}^*, Ap_m >=< r_{m+1}^* + \theta_m p_m^*, Ap_m >$$

és ebből felhasználva hogy $Ap_m = -\frac{1}{\mu_m}(r_{m+1} - r_m)$:

$$\theta_m = -\frac{< r_{m+1}^*, Ap_m >}{< p_m^*, Ap_m >} = \frac{1}{\mu_m} \frac{< r_{m+1}^*, r_{m+1} - r_m >}{< p_m^*, Ap_m >} = \frac{< r_{m+1}^*, r_{m+1} >}{< r_m^*, r_m >}$$

Ezek alapján felírható az algoritmus:

3.4.6. Algoritmus. *BiCG*

1. Számítsuk ki $r_0 = b - Ax_0$*, és válasszunk egy* r_0^* *vektort, melyre* $< r_0, r_0^* > \neq 0$.
2. $p_0 := r_0$*, illetve* $p_0^* := r_0^*$.
3. For $j = 1, 2, ...$ *Do:*
4. $\mu_j = < r_j, r_j^* > / < Ap_j, p_j^* >$
5. $x_{j+1} = x_j + \mu_j p_j$
6. $r_{j+1} = r_j - \mu_j A p_j$
7. $r_{j+1}^* = r_j^* - \mu_j A^T p_j^*$
8. $\theta_j = < r_{j+1}^*, r_{j+1} > / < r_j, r_j^* >$
9. $p_{j+1} = r_{j+1} + \theta_j p_j$
10. $p_{j+1}^* = r_{j+1}^* + \theta_j p_j^*$
11. EndDo

Ha meg kell oldanunk a duál feladatot is, akkor válasszuk $r_0^* = b^* - A^T x_0^*$, és az algoritmus 5. sora után szúrjuk be: $x_{j+1}^* = x_j + \mu_j p_j^*$. Ugyanakkor ha nem oldunk meg duál feladatot, az iteráció minden lépésében elvégzünk egy szorzást A-val és A^T-tal is,de a p_j^* vektor, melynek kiszámításához az A^T mátrixot használtuk, nem járul hozzá közvetlenül a megoldás kiszámításához, csak a μ_j, θ_j konstansok megadásához. Felmerül a kérdés hogy el tudjuk-e kerülni az A^T mátrix használatát.

3.4.7. Megjegyzés. Léteznek olyan gyakorlati problémák, melyeknél az A mátrix csak egy közelítését ismerjük, nem tudjuk (vagy nem szükséges) explicit megadni. Ekkor általában az A^T mátrix nem áll rendelkezésre. Egy egyszerű példa amikor a Newton-iteráció felhasználásával akarjuk megoldani az $F(u) = 0$ egyenletet. Az iteráció minden lépésében egy lineáris egyenletrendszert kell megoldanunk, ezt meg tudjuk oldani a Jacobi-mátrix explicit kiszámítása nélkül, felhasználva hogy:

$$J(u_k)v = \frac{F(u_k + \epsilon v) - F(u_k)}{\epsilon}$$

Ez lehetővé teszi hogy egy tetszőleges v vektorra kiszámítsuk az értéket. Ugyanakkor nem létezik hasonló formula $J(u_k)^T$-hoz.

3.5. BiCGStab

Vegyük észre hogy a BiCG módszerben a maradékvektor felírható:

$$r_j = \phi_j(A) r_0$$

alakban, ahol $\phi_j(t)$ egy j-edfokú polinom, amelyre $\phi_j(0) = 1$. Hasonlóan felírhatjuk:

$$p_j = \pi_j(A)r_0$$

A 3.4.6 algoritmusban r_j^*-ot és p_j^* hasonló képzési szabállyal állítjuk elő, mint r_j-t illetve p_j-t, az A mátrix helyett A^T-ot használva. Így:

$$r_j^* = \phi_j(A^T)r_0^* \qquad p_j^* = \pi_j(A^T)r_0^*$$

Az előbbi felírás helyett a BiCGStab módszer a maradékvektort a következő alakban állítja elő:

$$r_j^{'} = \psi_j(A)\phi_j(A)r_0$$

ahol $\psi_j(A)$ arra szolgál hogy stabilizálja az eredeti algoritmus konvergenciáját. $\psi_j(t)$-t a következő rekurzív képlettel kapjuk meg:

$$\psi_j(t) = (1 - \omega_j t)\psi_{j-1}(t) \tag{3.9}$$

ahol az ω_j paramétert kell meghatároznunk. A 3.4.6 algoritmusból leolvashatjuk:

$$\phi_{j+1}(t) = \phi_j(t) - \mu_j t\pi_j(t) \tag{3.10}$$

$$\pi_{j+1}(t) = \phi_{j+1}(t) + \theta_j\pi_j(t) \tag{3.11}$$

A fenti egyenletekből egyszerűen kapjuk:

$$\begin{aligned}
\psi_{j+1}(t)\phi_{j+1}(t) &= (1 - \omega_j t)\psi_j(t)(\phi_j(t) - \mu_j t\pi_j(t)) = \\
&= (1 - \omega_j t)(\psi_j(t)\phi_j(t) - \mu_j t\psi_j(t)\pi_j(t)) \\
\psi_j(t)\pi_j(t) &= \psi_j(t)(\phi_j(t) + \theta_{j-1}\pi_{j-1}(t)) = \\
&= \psi_j(t)\phi_j(t) + \theta_{j-1}(1 - \omega_{j-1}t)\psi_{j-1}(t)\pi_{j-1}(t)
\end{aligned}$$

Jelölje mostantól:

$$\begin{aligned}
r_j &= \psi_j(A)\phi_j(A)r_0 \\
p_j &= \psi_j(A)\pi_j(A)r_0
\end{aligned}$$

Felhasználva az előző képleteket kapjuk a rekurziókat:

$$\begin{aligned}
r_{j+1} &= (I - \omega_j A)(r_j - \mu_j Ap_j) \\
p_{j+1} &= r_{j+1} + \theta_j(I - \omega_j A)p_j
\end{aligned}$$

Tekintsük most a θ_j konstanst. A BiCG algoritmust tekintve $\theta_j = \rho_{j+1}/\rho_j$, ahol

$$\rho_j =< \phi_j(A)r_0, \phi_j(A^T)r_0^* >=< \phi_j(A)^2 r_0, r_0^* >$$

Azonban nem ismerjük a $\phi_j(A)r_0$, $\phi_j(A^T)r_0^*$ és $\phi_j(A)^2 r_0$ vektorokat, így ezekből nem tudjuk kiszámítani. Tekintsük

$$\overline{\rho}_j =< \phi_j(A)r_0, \psi_j(A^T)r_0^* >=< \psi_j(A)\phi_j(A)r_0, r_0^* >=< r_j, r_0^* >$$

Másrészt $\psi_j(A^T)$ felírható a következő alakban:

$$\psi_j(A^T) = \eta_1^{(j)}(A^T)^j + \eta_2^{(j)}(A^T)^{j-1} + ... + \eta_{j+1}^{(j)}$$

Felhasználva hogy $\phi_j(A)r_0$ ortogonális $(A^T)^k r_0^*$-ra minden $k < j$ indexre, kapjuk:

$$\begin{aligned}
\overline{\rho}_j &= < \phi_j(A)r_0, \psi_j(A^T)r_0^* >= \\
&= < \phi_j(A)r_0, \eta_1^{(j)}(A^T)^j r_0^* + \eta_2^{(j)}(A^T)^{j-1} r_0^* + ... + \eta_{j+1}^{(j)} r_0^* >= \\
&= < \phi_j(A)r_0, \eta_1^{(j)}(A^T)^j r_0^* >=< \phi_j(A)r_0, \frac{\eta_1^{(j)}}{\gamma_1^{(j)}}\phi_j(A^T)r_0^* >= \\
&= \frac{\eta_1^{(j)}}{\gamma_1^{(j)}}\rho_j
\end{aligned}$$

ahol $\gamma_1^{(j)}$ a $\phi_j(t)$ polinom főegyüttható ja. A 3.10 és 3.11 egyenletekből leolvasható hogy ϕ_j és π_j főegyüttható ja megegyezik, továbbá felhasználva a 3.9 egyenletet kapjuk a következő összefüggést a főegyüttható kra:

$$\eta_1^{(j+1)} = -\omega_j \eta_1^{(j)} \qquad \gamma_1^{(j+1)} = -\mu_j \gamma_1^{(j)}$$

Egyszerűen kapjuk az összefüggést:

$$\frac{\overline{\rho}_{j+1}}{\overline{\rho}_j} = \frac{\omega_j}{\mu_j}\frac{\rho_{j+1}}{\rho_j}$$

amit felhasználva

$$\theta_j = \frac{\mu_j}{\omega_j}\frac{\overline{\rho}_{j+1}}{\overline{\rho}_j}$$

Az előbbihez hasonló eljárással felírjuk a μ_j konstanst, felhasználva hogy ϕ_j és π_j főegyüttható ja megegyezik:

$$\begin{aligned}
\mu_j &= \frac{< \phi_j(A)r_0, \phi_j(A^T)r_0^* >}{< A\pi_j(A)r_0, \pi_j(A^T)r_0^* >} = \frac{< \phi_j(A)r_0, \phi_j(A^T)r_0^* >}{< A\pi_j(A)r_0, \phi_j(A^T)r_0^* >} = \\
&= \frac{< \phi_j(A)r_0, \psi_j(A^T)r_0^* >}{< A\pi_j(A)r_0, \psi_j(A^T)r_0^* >} = \frac{< \psi_j(A)\phi_j(A)r_0, r_0^* >}{< A\psi_j(A)\pi_j(A)r_0, r_0^* >} = \frac{\overline{\rho}_j}{< Ap_j, r_0^* >}
\end{aligned}$$

Következő lépésként meghatározzuk az ω_j paramétert. Válasszuk meg ω_j-t úgy, hogy minimális legyen $\|(I - \omega_j A)\psi_j(A)\phi_j(A)r_0\|$. Jelöljük $s_j = r_j - \mu_j A p_j$, így $r_{j+1} = (I - \omega_j A)s_j$. A paraméter optimális értéke:

$$\omega_j = \frac{< As_j, s_j >}{< As_j, As_j >}.$$

Végül

$$r_{j+1} = s_j - \omega_j As_j = r_j - \mu_j Ap_j - \omega_j As_j$$

amiből kapjuk:

$$x_{j+1} = x_j + \mu_j p_j + \omega_j s_j.$$

Ezek alapján felírhatjuk a következő algoritmust:

3.5.1. Algoritmus. *BiCGStab*

1. Számítsuk ki $r_0 = b - Ax_0$, r_0^* *legyen tetszőleges.*
2. $p_0 := r_0$
3. For $j = 0, 1, ...$ *Do:*
4. $\mu_j =< r_j, r_0^* > / < Ap_j, r_0^* >$
5. $s_j = r_j - \mu_j Ap_j$
6. $\omega_j =< As_j, s_j > / < As_j, As_j >$
7. $x_{j+1} = x_j + \mu_j p_j + \omega_j s_j$
8. $r_{j+1} = s_j - \omega_j As_j$
9. $\theta_j = \mu_j < r_{j+1}, r_0^* > /(\omega_j < r_j, r_0^* >)$
10. $p_{j+1} = r_{j+1} + \theta_j(p_j - \omega_j Ap_j)$
11. EndDo

4. fejezet

Összehasonlítás

4.1. Az alapmegoldások módszere

Tekintsük a következő problémát: Legyen $\Omega \subset \mathbb{R}^2$ tartomány. Keressük azt az $u : \Omega \to \mathbb{R}$ kétszer folytonosan differenciálható függvényt, amelyre

$$\triangle\, u(x,y) = 0 \qquad \text{ha } x \in \Omega$$

és

$$u(x) = g(x) \qquad \text{ha } x \in \partial\Omega$$

ahol $\triangle$ a Laplace-operátort jelöli, $\partial\Omega$ az Ω tartomány határát és $g : \partial\Omega \to \mathbb{R}$ függvény (így előírtuk, hogy az u függvény milyen értékeket vesz fel a tartomány határán, ezt nevezik Dirichlet-peremfeltételnek). Tekintsük a

$$\psi : \mathbb{R}^2 \setminus \{0\} \to \mathbb{R}$$
$$\psi(x) = log\|x\|$$

függvényt. Ekkor $\triangle\, \psi(x) = 0$ minden $x \in \mathbb{R}^2 \setminus \{0\}$ pontra. Vegyünk fel n darab pontot a tartomány határán, ezeket jelölje: $\overline{x}_1, \overline{x}_2, ...\overline{x}_n$ és újabb n darab, úgynevezett forráspontot a tartományon kívül: $x_1, x_2, ..., x_n$. Az alapmegoldások módszerében az u függvényt a következő alakban keressük:

$$u(x) = \sum_{j=1}^{n} \alpha_j \psi(x - x_j)$$

ahol az α_j együtthatók ismeretlenek. Ekkor a Laplace-operátor linearitása következtében $\triangle\, u(x) = 0$ minden $x \in \Omega$ esetén (a fenti függvény minden $x \in \Omega$ pont esetén értelmezett,

hiszen $x - x_j \neq 0$).
Az α_j együtthatók a peremfeltétel felhasználásával számíthatók:

$$u(\overline{x}_k) = \sum_{j=1}^{n} \alpha_j \psi(\overline{x}_k - x_j) \qquad k = 1, ..., n$$

Ekkor:

$$A_{ij} := \psi(\overline{x}_i - x_j) \qquad b_i := u(\overline{x}_k)$$

A megoldandó egyenletrendszer:

$$A\alpha = b.$$

Ahogy távolabb választjuk a tartományon kívüli x_j pontokat, a kapott A mátrix kondíciószáma (gyorsan) nő, így a pontos megoldás kiszámítása egyre nagyobb problémát jelent. Ugyanakkor nincs szükségünk az α pontos értékérére: az u függvény még úgy is pontos lehet hogy az α_j együtthatók hibája nem elhanyagolható.
Legyen $\Omega = [-1, 1] \times [-1, 1]$ négyzet, és tekintsük az

$$u(x, y) = 10x^2 - 10y^2 + 5xy + 4x - 2y$$

függvényt. Erre teljesül hogy $\triangle\, u = 0$ minden $x \in \Omega$ pontra. Vegyünk fel n darab pontot a négyzet határáról egyenletesen (a négyzet középpontjából indítunk félegyeneseket, a félegyenesek $2\pi/n$ szöget zárnak be egymással, a félegyenesek és a négyzet metszéspontjai legyenek a határpontok), majd számítjuk ki az $u(\overline{x}_j)$ értékeket. A forráspontokat vegyük az origó középpontú $r\sqrt{2}$ sugarú körről egyenletesen, ahol $r > 1$. Ezen probléma megoldására fogjuk alkalmazni a bemutatott 3 módszert (GMRES, LSQR, BiCGStab), különböző n illetve r értékek esetén. Mindhárom módszer toleranciaértéke 10^{-8}, maximum iterációs száma n. A módszereket a konvergenciához szükséges iterációs szám, az egyenletrendszer megoldásának hibája, és a keresett u függvény hibája (a tartomány belsejében egyenletesen felvett pontokban a pontos megoldás és az általunk adott u függvény eltérésének maximuma) alapján hasonlítjuk össze.

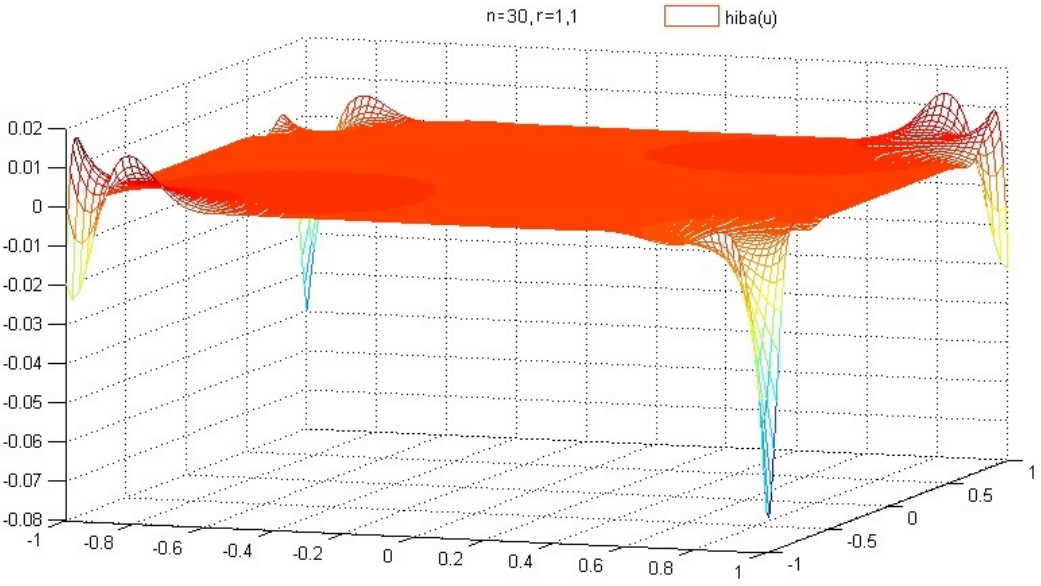
Az adatokat a 4.1 ábra tartalmazza. A mátrix kondíciószáma a forráspontok távolításával, illetve a határpontok számának növelésével nő. Megfigyelhető, hogy a GMRES módszer minden esetben elérte a megfelelő pontosságot, míg a BiCGStab a maximum iterációs szám elérése után állt le. A táblázatból leolvasható hogy több tesztesetnél, annak ellenére hogy a GMRES módszer pontosabb megoldást adott az egyenletrendszerre, az u függvény hibája nagyságrendileg azonos (például $n = 10$, $r = 1,1; 2; 4$ vagy $n = 100$, $r = 1,1$).

A 4.2 ábrán látható hogy a legnagyobb eltérés a tartomány határán, a határpontok között van, a tartomány belsejében a megoldás pontosabb. Ahogy távolítjuk a forráspontokat, a határpontok közötti hiba csökken.

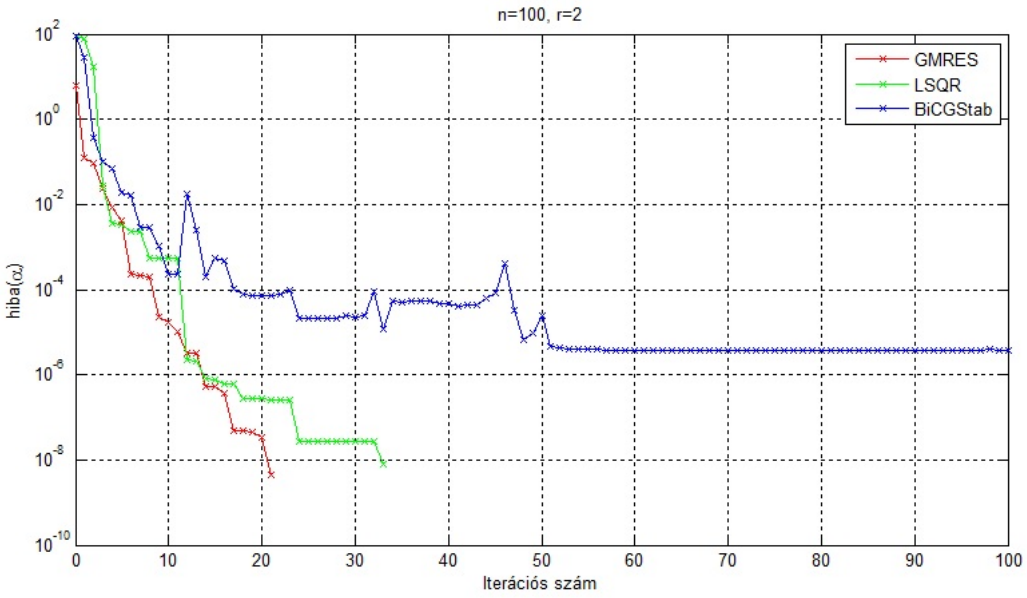
A 4.4 ábráról leolvasható, hogy ha növeljük a határpontok számát, nem kell nagy távolságokra felvenni a forráspontokat, hiszen $r \geq 2$ esetén az u függvény hibájának változása kicsi (már $r = 2$ esetén is pontos).

	Pontok száma	Távolság	Kondíciószám	GMRES	LSQR	BiCGStab
Iterációs szám	n=10	r=1,1	11,18	8	10	10
		r=2	529,32	8	10	10
		r=4	$2{,}82 \cdot 10^{4}$	8	10	10
		r=10	$4{,}21 \cdot 10^{6}$	8	10	10
	n=30	r=1,1	$2{,}06 \cdot 10^{3}$	25	30	30
		r=2	$3{,}18 \cdot 10^{7}$	24	30	30
		r=4	$1{,}73 \cdot 10^{12}$	18	30	30
		r=10	$6{,}88 \cdot 10^{17}$	12	21	30
	n=100	r=1,1	$2{,}82 \cdot 10^{10}$	36	100	100
		r=2	$3{,}79 \cdot 10^{18}$	21	33	100
		r=4	$8{,}7 \cdot 10^{19}$	15	24	100
		r=10	$2{,}38 \cdot 10^{19}$	10	7	100
Hiba(α)	n=10	r=1,1	11,18	$2{,}03 \cdot 10^{-14}$	$2{,}01 \cdot 10^{-6}$	$2{,}01 \cdot 10^{-7}$
		r=2	529,32	$2{,}46 \cdot 10^{-14}$	$4{,}7 \cdot 10^{-3}$	$8{,}95 \cdot 10^{-4}$
		r=4	$2{,}82 \cdot 10^{4}$	$2{,}11 \cdot 10^{-13}$	$2{,}7 \cdot 10^{-3}$	$4{,}14 \cdot 10^{-7}$
		r=10	$4{,}21 \cdot 10^{6}$	$5{,}7 \cdot 10^{-13}$	$1{,}11 \cdot 10^{-4}$	$2 \cdot 10^{-3}$
	n=30	r=1,1	$2{,}06 \cdot 10^{3}$	$2{,}22 \cdot 10^{-9}$	$6{,}68 \cdot 10^{-4}$	$1{,}79 \cdot 10^{-4}$
		r=2	$3{,}18 \cdot 10^{7}$	$4{,}17 \cdot 10^{-9}$	$2{,}82 \cdot 10^{-6}$	$3{,}45 \cdot 10^{-4}$
		r=4	$1{,}73 \cdot 10^{12}$	$3{,}31 \cdot 10^{-9}$	$1{,}05 \cdot 10^{-6}$	$5{,}33 \cdot 10^{-5}$
		r=10	$6{,}88 \cdot 10^{17}$	$9{,}79 \cdot 10^{-9}$	$4{,}72 \cdot 10^{-9}$	$1{,}07 \cdot 10^{-4}$
	n=100	r=1,1	$2{,}82 \cdot 10^{10}$	$9{,}68 \cdot 10^{-9}$	$1{,}36 \cdot 10^{-7}$	$4{,}47 \cdot 10^{-5}$
		r=2	$3{,}79 \cdot 10^{18}$	$4{,}39 \cdot 10^{-9}$	$7{,}98 \cdot 10^{-9}$	$3{,}72 \cdot 10^{-6}$
		r=4	$8{,}7 \cdot 10^{19}$	$3{,}46 \cdot 10^{-10}$	$2{,}33 \cdot 10^{-10}$	$8{,}64 \cdot 10^{-5}$
		r=10	$2{,}38 \cdot 10^{19}$	$4{,}78 \cdot 10^{-9}$	$8{,}8 \cdot 10^{-9}$	$4{,}91 \cdot 10^{-4}$
Hiba(u)	n=10	r=1,1	11,18	1,66	1,66	1,66
		r=2	529,32	$7{,}45 \cdot 10^{-2}$	$7{,}47 \cdot 10^{-2}$	$7{,}44 \cdot 10^{-2}$
		r=4	$2{,}82 \cdot 10^{4}$	$1{,}2 \cdot 10^{-3}$	$4{,}4 \cdot 10^{-3}$	$1{,}5 \cdot 10^{-3}$
		r=10	$4{,}21 \cdot 10^{6}$	$4{,}8 \cdot 10^{-6}$	$1{,}19 \cdot 10^{-4}$	$1{,}7 \cdot 10^{-3}$
	n=30	r=1,1	$2{,}06 \cdot 10^{3}$	$7{,}11 \cdot 10^{-2}$	$7{,}09 \cdot 10^{-2}$	$7{,}09 \cdot 10^{-2}$
		r=2	$3{,}18 \cdot 10^{7}$	$2{,}22 \cdot 10^{-8}$	$2{,}16 \cdot 10^{-6}$	$1{,}79 \cdot 10^{-4}$
		r=4	$1{,}73 \cdot 10^{12}$	$4{,}73 \cdot 10^{-9}$	$4{,}88 \cdot 10^{-7}$	$2{,}99 \cdot 10^{-5}$
		r=10	$6{,}88 \cdot 10^{17}$	$7{,}16 \cdot 10^{-9}$	$1{,}97 \cdot 10^{-9}$	$5{,}3 \cdot 10^{-5}$
	n=100	r=1,1	$2{,}82 \cdot 10^{10}$	$3{,}14 \cdot 10^{-5}$	$3{,}06 \cdot 10^{-5}$	$2{,}46 \cdot 10^{-5}$
		r=2	$3{,}79 \cdot 10^{18}$	$2{,}03 \cdot 10^{-9}$	$2{,}88 \cdot 10^{-9}$	$1{,}59 \cdot 10^{-6}$
		r=4	$8{,}7 \cdot 10^{19}$	$1{,}32 \cdot 10^{-10}$	$6{,}47 \cdot 10^{-11}$	$2{,}94 \cdot 10^{-5}$
		r=10	$2{,}38 \cdot 10^{19}$	$1{,}34 \cdot 10^{-9}$	$1{,}76 \cdot 10^{-9}$	$1{,}35 \cdot 10^{-4}$

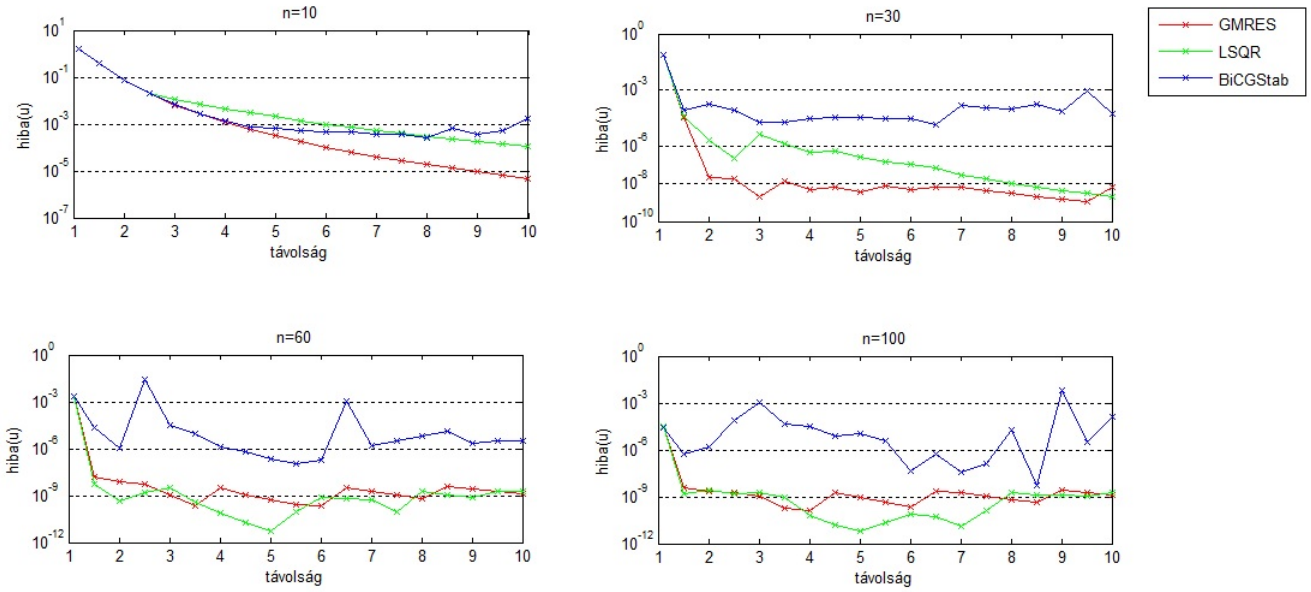
4.1. ábra. A táblázat a problémára alkalmazott GMRES, LSQR és BiCGStab módszerek iterációs számát, az egyenletrendszer megoldásának hibáját illetve a keresett u függvény hibáját tartalmazza. r jelöli a forráspontok távolságát a tartomány szélétől, n a felvett határpontok számát.



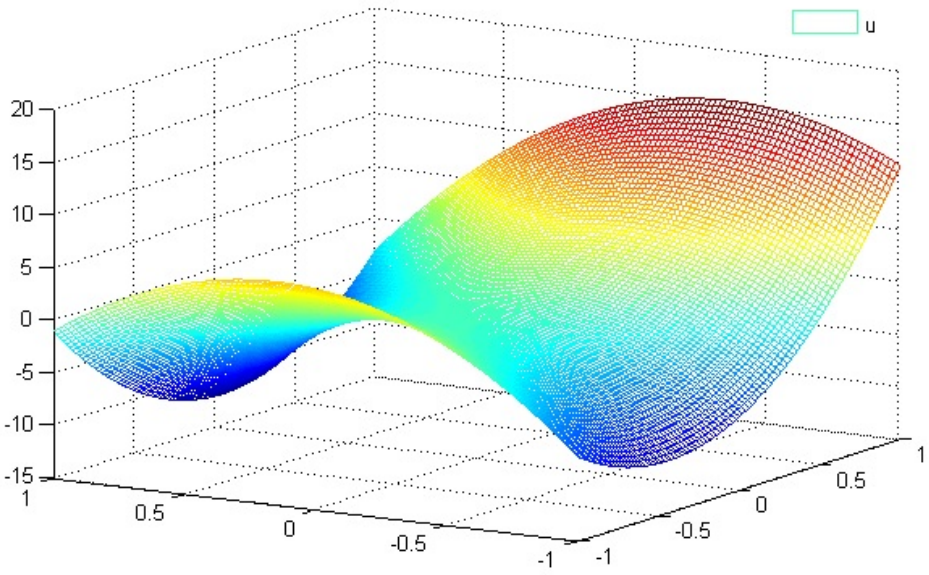
4.2. ábra. A GMRES módszer által adott u függvény hibája $n = 30$ és $r = 1.1$ esetén.



4.3. ábra. $n = 100$ és $r = 2$ esetén a módszerek konvergenciája.



4.4. ábra. A távolság függvényében a kapott u függvény hibája.



4.5. ábra. A kapott u függvény $n = 100$, $r = 4$ esetén.

5. fejezet

Összefoglalás

Az előző fejezetben összehasonlítottuk egy tesztfeladaton a bemutatott, Krylov-altérre épülő módszereket. Megfigyelhető, hogy minden paraméterpár esetén (kivéve $n = 100$, $r = 10$), a GMRES-módszer mutatott a leggyorsabb konvergenciát. Azonban ez általánosságban nem mondható el. Mindegyik módszerhez léteznek olyan mátrixok, melyekre az adott módszer konvergál a legkevesebb iterációs lépés alatt. Ezért a gyakorlatban egy problémára több módszert is szokás alkalmazni, hogy lássuk melyik módszer teljesít a legjobban. A tesztfeladatban használt mátrixok rosszul kondicionáltak, ugyanakkor a konstrukció következtében speciális alakjuk miatt a módszerek gyors konvergenciát mutattak. Egy általános rosszul kondicionált mátrixra a módszerek nem így viselkednek, a konvergencia lassabb, vagy egyáltalán nem következik be. A bemutatott módszereknek számos változata létezik és használt, ezekről bővebben a [2] könyvben olvashatunk (például a már említett Újraindított GMRES-módszer).

6. fejezet

A módszerek MATLAB-kódjai

```
function [x,hiba]=gmres(A,b)

n=size(A,1);
Q(:,1)=b/norm(b,2);
v=A*Q(:,1);
H(1,1)=Q(:,1)'*v;
v=v-H(1,1)*Q(:,1);
H(2,1)=norm(v,2);
Q(:,2)=v/H(2,1);
[T,R]=qr(H);
T=T';
g=T(1,1)*norm(b,2);
R=triu(R(1,1));
y = R\g;
f=1;
i=2;
x=0;
while norm(H*y-norm(b,2)*f)>10^-8 && i<=n;
    v=A*Q(:,i);
    for j=1:i
        H(j,i)=Q(:,j)'*v;
        v=v-H(j,i)*Q(:,j);
    end
```

```
    H(i+1,i)=norm(v,2);
    Q(:,i+1)=v/H(i+1,i);
    if i==2
        [T,R]=qr(H);
        T=T';
    elseif i>2
        c=[T(i,:) 0]*H(:,i);
        s=H(i+1,i);
        c=c/sqrt(c^2+s^2);
        s=sqrt(1-c^2);
        T(i:i+1,1:i+1)=[c,s;-s,c]*[T(i,:) 0;zeros(1,i) 1];
        R(1:i+1,i)=T*H(:,i);
    end
    if(i>1)
        g=T(:,1)*norm(b,2);
        opts.UT = true;
        y = linsolve(triu(R(1:i,1:i)),g(1:i),opts);
        x=Q(:,1:i)*y;
        f=zeros(i+1,1);
        f(1)=1;
        hiba(i-1)=norm(H*y-norm(b,2)*f);
    end
    i=i+1;
end

function [x,hiba1]=lsqr(A,b)

n=size(A,1);
x=zeros(n,1);
q=b;
beta=norm(q);
u=q/beta;
q=A'*u;
alpha=norm(q);
```

```
v=q/norm(q);
a=alpha;
d=beta;
w=v;
hi1=norm(b);
hiba1(1)=norm(A*x-b);
k=1;
while hi1>10^-8 && k<=n
    k=k+1;
    q=A*v-alpha*u;
    beta=norm(q);
    u=q/beta;
    q=A'*u-beta*v;
    alpha=norm(q);
    v=q/alpha;
    e=sqrt(a^2+beta^2);
    c=a/e;
    s=beta/e;
    f=s*alpha;
    a=c*alpha;
    g=c*d;
    d=-s*d;
    x=x+(g/e)*w;
    w=v-(f/e)*w;
    hi1=abs(d);
    hiba1=[hiba1;hi1];
end

function [x,hiba]=bicgstab(A,b)

n=size(A,1);
x=zeros(n,1);
r1=b-A*x;
r0=r1;
```

```
p=r1;
r2=r1;
hiba(1)=norm(r2);
k=1;
while norm(r2)>10^-8 && k<=n
    r1=r2;
    mu=dot(r1,r0)/dot(A*p,r0);
    s=r1-mu*A*p;
    om=dot(A*s,s)/dot(A*s,A*s);
    x=x+mu*p+om*s;
    r2=s-om*A*s;
    hiba=[hiba,norm(r2)];
    th=mu/om*dot(r2,r0)/dot(r1,r0);
    p=r2+th*(p-om*A*p);
    k=k+1;
end
```

Irodalomjegyzék

[1] Stoyan Gisbert, Takó Galina, *Numerikus módszerek 1.*, Typotex, 2002

[2] Yousef Saad, *Iterative Methods for Sparse Linear Systems*, Society for Industrial and Applied Mathematics, 2003

[3] Christopher C. Paige, Michael A. Saunders, *LSQR: An Algorithm for Sparse Linear Equations and Sparse Least Squares*, ACM Transactions on Mathematical Software, Vol.8, No.1, 1982