שיטות איטרטיביות באלגברה ליניארית

איטרציות פשוטות ליניאריות: פתרון מערכת משוואות (בעזרת איטרציות ליניאריות במימד m)

$$x^{(n+1)} = Sx^{(n)} + b$$

$$x^{(1)} = Sx^{(0)} + b$$

$$x^{(2)} = Sx^{(1)} + b = S^2 x^{(0)} + \left(S + I\right)b$$

$$\vdots$$

$$x^{(n)} = S^n x^{(0)} + \left(I + S + S^2 + \ldots + S^{n-1}\right)b$$

פונקציה ממשית $norm(A) \equiv \|A\|$ נקראת נורמה של מטריצה A אם היא ומקיימת:

1) $\|A\| \ge 0$ לכל A ($\|A\| = 0$ אם ורק אם $A \equiv 0$)

2) $\|\alpha A\| = |\alpha|\|A\|$ לכל α ממשי (ליניאריות ביחס למכפלה בסקלר)

3) $\|A + B\| \le \|A\| + \|B\|$ (אי-שוויון המשולש)

4) $\|AB\| \le \|A\|\|B\|$ לכל שני מטריצות רביעיות A ו-B.

ב-MATLAB משתמשים ב *norm(A,1), norm(A,2), norm(A,inf)*

מתי $x^{(n)}$ מתכנס לפתרון? ← כאשר $\|S\| < 1$. נניח ש- $\|S\| = q < 1$

<u>הוכחה:</u> $\|S^n\| \le \|S\|^n = q^n$. $S^n x^{(0)} \to 0$ ולכן זניח.

$$\left\|I + S + S^2 + \ldots + S^{n-1}\right\| \le \|I\| + \|S\| + \left\|S^2\right\| + \ldots + \left\|S^{n-1}\right\| \le \|I\| + \|S\| + \|S\|^2 + \ldots + \|S\|^2 \underset{\neq\Delta}{\le} \frac{1 - \|S\|^n}{1 - \|S\|}$$

הערה: כאשר דרשנו $\|S\| < 1$, כל נורמה בה $\|S\| < 1$ יכולה לגרור התכנסות. יכול לקרות שבנורמה מסוימת $\|S\| < 1$ ואילו באחרת $\|S\| > 1$. במקרה זה מספיק למצוא נורמה אחת.

לדוגמא: $Ax = b$

$$A = D + A'$$

$$D = \begin{pmatrix} a_{11} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & a_{nn} \end{pmatrix}, \quad A' = \begin{pmatrix} 0 & & U \\ & \ddots & \\ L & & 0 \end{pmatrix}$$

(ב-MATLAB *D=diag(diag(A)), U=triu(A-D), L=tril(A-D)*)

כעת, נוכל להציג את מערכת המשוואות באופן הבא: $Dx = -A'x + b$

שיטת יעקובי:

$$x = D^{-1}(b - A'x), \quad S = -D^{-1}(L + U)$$

$$x^{(n+1)} = Sx^{(n)} + b' \xrightarrow{n\to\infty} x = Sx + b' \Leftrightarrow x = D^{-1}A'x + D^{-1}b$$

$$\boxed{JACOBI: \quad x^{(n+1)} = D^{-1} \cdot \left(b - (U + L)x^{(n)}\right)}$$

בפרט, השיטה היא מתכנסת עם מטריצה A מקיימת תנאי "האלכסון שולט": $|a_{ii}| \ge \left|\sum_{i\neq j} a_{ij}\right|$

שיטת גאוס-זיידל:
$$x=(D+L)^{-1}(b-Ux)$$
$$x^{(n+1)}=Sx^{(n)}+b' \xrightarrow{n\to\infty} x=Sx+b', \quad S=-(D+U)^{-1}L$$

מתי כדאי להשתמש בשיטת יעקובי, ומתי בשיטת הדירוג של גאוס?

השגיאה בשיטת הדירוג של גאוס עולה, בעוד בשיטת יעקובי היא יורדת. בגאוס יש $n^3/3$ פעולות, לעומת עיקובי שמס. הפעולות תלוי בגודלה וסוגה של המטריצה.

הגדרה: *sparse matrix* היא מטריצה שרוב האיברים בה הם אפסים. עבור מטריצות כאלו, חישוב בשיטת עיקובי הוא חסכוני ומהיר.

האלגוריתם של *Jacobi* מתכנס כאשר: $\left\|D^{-1}(L+U)\right\|\le q<1$.

ננסה לשפר את האלגוריתם של עיקובי באמצעות כתיבה שונה:

$$A=\{a_{ij}\}_{ij=1}^{m} \qquad Ax=b$$
$$\forall i=1,\dots,n: \quad a_{ii}x_i+\sum_{j<i}a_{ij}x_j+\sum_{j>i}a_{ij}x_j=b_i$$

האלגוריתם של *Gauss-Seidel*:

$$x^{(n+1)}=\frac{-1}{a_{ii}}\left(\underbrace{\sum_{j<i}a_{ij}x_j^{(n+1)}}_{L}+\underbrace{\sum_{j>i}a_{ij}x_j^{(n)}}_{U}\right)+\frac{b_i}{a_{ii}}$$

אלגוריתם זה הוא מדיוק יותר, כי משתמשים בערכי הווקטור שקיבלנו באותה האיטרציה. כלומר, משתמשים ב- $x_j^{(n+1)}$ לחישוב $x_i^{(n+1)}$

$$Ax=b\Rightarrow(D+L+U)x=b$$
$$(D+L)x=-Ux+b\Rightarrow x=(D+L)^{-1}\cdot(b-Ux)$$
$$x^{(n+1)}=(D+L)^{-1}(b-Ux^{(n)})$$
$$\boxed{\text{GAUSS-SEIDEL:}\quad x^{(n+1)}=(D+L)^{-1}\cdot\left(b-Ux^{(n)}\right)}$$

האלגוריתם של *SOR*: (קיצור של *Successive Over Relaxation*) נכפיל ב-w ונקבל:

$$w(D+L)x=w(b-Ux)$$
$$\left[D+(w-1)D+wL\right]x=-wUx+wb$$
$$(D+wL)x=\left[-wU+(1-w)D\right]x+wb$$
$$\Rightarrow x^{(n+1)}=(D+wL)^{-1}\cdot\left[(1-w)D-wU\right]\cdot x^{(n)}+(D+wL)^{-1}wb$$
$$\boxed{\text{SOR:}\quad x^{(n+1)}=(D+wL)^{-1}\cdot\left(\left[(1-w)D-wU\right]\cdot x^{(n)}+wb\right)}$$

קיבלנו $SOR(w)$. ע"י w אפשר להקטין את הנורמה של מטריצה S. $\|S\cdot w\|\to\min$, ואז מהירות ההתכנסות גבוהה הרבה יותר.

SOR הוא האלגוריתם המדויק ביותר, אחריו *Seidel*, ו-*Jacoby* הכי פחות מדויק.

פתרון מערכת משוואות לא ליניארית

שיטה NEWTON-RAPHSON (N~R) למערכת משוואות היא:

$$\begin{cases} f_1(x_1, x_2, \dots, x_m) = 0 \\ f_2(x_1, x_2, \dots, x_m) = 0 \\ \vdots \\ f_m(x_1, x_2, \dots, x_m) = 0 \end{cases}$$

$$N \sim R \quad x^{(n)} = \left(x_1^{(n)}, x_2^{(n)}, \dots, x_m^{(n)}\right)$$

$$x^{(n+1)} = x^{(n)} - \left(f'\left(x^{(n)}\right)\right)^{-1} \cdot f\left(x^{(n)}\right)$$

$$f(x_1, x_2, \dots, x_m) = 0 = \begin{Bmatrix} f_1(x_1, x_2, \dots, x_m) = 0 \\ f_2(x_1, x_2, \dots, x_m) = 0 \\ \vdots \\ f_m(x_1, x_2, \dots, x_m) = 0 \end{Bmatrix}$$

מטריצה יעקובי היא מטריצה שצורתה:

$$f' \overset{def}{=} \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{pmatrix}$$

לדוגמא: ניקח את מערכת המשוואות הבאה: $\begin{cases} f_1(x,y): x^2 - y^2 - 3 = 0 \\ f_2(x,y): 2xy - 2 = 0 \end{cases}$. אז:

$$J = f' = \begin{pmatrix} 2x & -2y \\ 2y & 2x \end{pmatrix}$$

היעקוביאן של מערכת המשוואות הוא $|J| = 4\left(x^2 + y^2\right) \neq 0$

$$\Rightarrow (f')^{-1} = J^{-1} = \frac{1}{4\left(x^2 + y^2\right)} \begin{pmatrix} 2x & 2y \\ -2y & 2x \end{pmatrix}$$

$$\begin{pmatrix} x^{(n+1)} \\ y^{(n+1)} \end{pmatrix} = \begin{pmatrix} x^{(n)} \\ y^{(n)} \end{pmatrix} - \frac{1}{2\left(x_n^2 + y_n^2\right)} \begin{pmatrix} x_n & y_n \\ -y_n & x_n \end{pmatrix} \times \begin{pmatrix} x_n^2 - y_n^2 - 3 \\ 2x_n y_n - 2 \end{pmatrix}$$

אם האלגוריתם מתכנס אז: $x = x - f'(x)^{-1} * \boxed{f(x) = 0}$

הערה: אם האלגוריתם מתכנס אז הוא יתכנס בהכרח לנקודת שבת.

אלגברה ליניארית נומרית

בעיות בהן אנו נתקלים:

1. פתרון מערכת משוואות ליניאריות $Ax=b$. $i=1,\dots,m$ $\quad \sum_{j=1}^{m} a_{ij}x_j = b_i$
2. מציאת ערכים עצמיים $Ax=\lambda x$
3. פולינום אופייני: $p(\lambda)=\det(A-\lambda\cdot I)=a_0\lambda^n+a_1\lambda^{n-1}+\dots$

<u>לדוגמא:</u> $Ax=b$. איך נמדוד את השגיאה?

נניח ש- $\tilde{b}$ היא השגיאה של b, ו- $\tilde{x}$ היא השגיאה של x. כלומר, $A\tilde{x}=\tilde{b}$

הקלט: $b,\tilde{b}$ הפלט: $x,\tilde{x}$

$$\|b\|=\|Ax\|\le\|A\|\cdot\|x\|$$
$$\Rightarrow\left\|b-\tilde{b}\right\|=\|A(x-\tilde{x})\|\le\|A\|\cdot\|x-\tilde{x}\|$$
$$x=A^{-1}\cdot b\Rightarrow\|x\|\le\left\|A^{-1}\right\|\cdot\|b\|$$
$$\delta_x=\frac{\|x-\tilde{x}\|}{\|x\|}\le k\cdot\delta_b=k\cdot\frac{\left\|b-\tilde{b}\right\|}{\|b\|}$$

k משתנה ממערכת משוואות אחת לאחרת. כאשר k גדול (למשל $k\approx 1000$) אז יש למטריצה <u>תנאי חולה</u>, או באנגלית: <u>*ill conditioning*</u>

$$\|x-\tilde{x}\|\le\left\|A^{-1}\right\|\cdot\left\|b-\tilde{b}\right\|$$
$$\|x\|\ge\frac{\|b\|}{\|A\|}\Rightarrow\frac{1}{\|x\|}\le\frac{\|A\|}{\|b\|}$$
$$\frac{\|x-\tilde{x}\|}{\|x\|}\le\frac{\left\|A^{-1}\right\|\cdot\left\|b-\tilde{b}\right\|\cdot\|A\|}{\|b\|}$$

סה״כ נקבל שה-k הוא: $k=\|A\|\cdot\left\|A^{-1}\right\|$. k יהיה מספר התנאי של מטריצה A.

ב-*Matlab* הפקודה תהיה $cond(A,1)$ (ה-1 מציין את מספר הנורמה של A). אם נקבל מספר מאוד גדול, לא כדאי לחשב בשיטה נומרית שכן השגיאה תהיה גדולה מאוד.

כעת, צריך למצוא מהו $\|A\|$:

1. רדיוס ספקטרלי אפשר להגדיר כ- $\rho(A)=\max_{1\le i\le n}|\lambda_i|$
2. "$norm\,2$" (כמו הגדרתה לברירת המחדל ב-*Matlab*): $\|A\|_2\overset{\det}{=}\sqrt{\rho\left(A\cdot A^t\right)}$

<u>הגדרה:</u> נגדיר $L:B_1\to B_2$ (מרחב איקלידי) ע״י: $\|L\|=\sup_{\|x\|_B\ne b}\frac{\|Ax\|_{B_2}}{\|x\|_{B_1}}$

3. נורמת אינסוף: $\|A\|_\infty = \max_i \sum_j |a_{ij}|$

נורמה של אופרטור: $\|Ax\|_\infty = \max_i \left|\sum_{i=1}^{n} a_{ij}x_j\right| \le \max_i \left(\sum_{j=1}^{n}|a_{ij}|\right)\cdot \max_j \left\{|x_j|\right\} \le k\|x\|_\infty$

דוגמא: $A_2 = \begin{pmatrix}1&0&0\\0&1&0\\0&0&1\end{pmatrix} \sim \begin{pmatrix}1&0&0\\0&1&\varepsilon\\0&0&1\end{pmatrix} = A_1$.

לשניהם יש k טוב. $A_1 - A_2 = \begin{pmatrix}0&0&0\\0&0&\varepsilon\\0&0&0\end{pmatrix}$

הערכים העצמיים של A_1 ו A_2 הם: $|A-\lambda I| = (1-\lambda)^3 = 0 \Rightarrow \lambda_{1,2,3} = 1$

ו"ע של A_2: $\begin{pmatrix}0\\0\\1\end{pmatrix}, \begin{pmatrix}0\\1\\0\end{pmatrix}, \begin{pmatrix}1\\0\\0\end{pmatrix}$

אבל יש רק שני הווקטורים העצמיים של A_1, ולכן ל- A_1, A_2 לא יהיו את אותם הוקטורים העצמיים.
קיבלנו שגיאה מעוד גדולה לחישוב ווקטורים עצמים!

ע"מ למצוא דטרמיננטה של מטריצה גדולה ניעזר בחילוץ גאוס (וב- $\sim n^3/3$ פעולות אלמנטאריות).

זהו האלגוריתם הטוב ביותר לפתרון מבחינת כמות הפעולות הדרושות $\sim n^3/3$.

כאשר משתמשים באלגוריתם $x = A^{-1}\cdot b$ נקבל שמספר הפעולות $\sim 2n^3/3$

אלגוריתם גאוס

אלגוריתם גאוס נכשל כאשר הפיבוט (*Pivoting*) בעיזה שהוא שלב ($a_{ii}^{(k)} = 0$)

$$\begin{pmatrix}-1&2&0\\0&0&1\\0&5&-1\end{pmatrix}\begin{pmatrix}x_1\\x_2\\x_3\end{pmatrix} = \begin{pmatrix}1\\2\\3\end{pmatrix} \Rightarrow \begin{pmatrix}-1&2&0\\0&5&-1\\0&0&1\end{pmatrix}\begin{pmatrix}x_1\\x_2\\x_3\end{pmatrix} = \begin{pmatrix}1\\3\\2\end{pmatrix} \Rightarrow x_3 = 2 \Rightarrow x_2 = 1 \Rightarrow x_1 = 1$$

השיטה המתוארת לעיל של החלפת רק שורות נקראת *partial pivoting*, ועם החלפת גם שורות וגם עמודות זוהי השיטה *pivoting*.

נקבל מטריצה אלכסונית עליונה: $E_1\cdot E_2\cdot\ldots\cdot E_n\cdot A = U$ ($E_1,\ldots,E_n$ הן מטריצות אלמנטאריות)

$$E_{ij} = i\begin{pmatrix}1&0&\ldots&0&0\\\ldots&\ldots&\ldots&\ldots&\ldots\\0&\alpha&\ldots&1&0\\\ldots&\ldots&\ldots&\ldots&\ldots\\0&0&\ldots&0&1\end{pmatrix} \qquad row_i + \alpha\, row_j \Rightarrow row_i \quad \Leftrightarrow \quad E_{ij}A \Rightarrow A,$$

$$\begin{array}{cc} j & i \end{array}$$

פירוק LU $\quad P\times A=L\times U$

כאשר U משולשית עליונה, ו-L משולשית תחתונה.

ללא המטריצה P, הפירוק לא היה עובד. לדוגמא: $\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}\neq\begin{pmatrix}a_{11} & 0\\ a_{21} & a_{22}\end{pmatrix}\times\begin{pmatrix}b_{11} & b_{12}\\ 0 & b_{22}\end{pmatrix}$

הפקודה ב-$Matlab$: $[L,U,P]=lu(A)$. אלגוריתם זה דורש $\sim n^3/3$ פעולות (זהה לגאוס).

כיצד נפתור $Ax=b$?

$$PA=LU$$

$$PAx=Pb$$

$$LUx=Pb$$

$$\begin{cases}Ux=y & \sim\dfrac{n^2}{2}\\ Ly=Pb & \sim\dfrac{n^2}{2}\end{cases}$$

ואז מציאת מטריצה הפיכה לוקחת פחות פעולות (צריך בסה"כ למצוא הפיכה למטריצה משולשית תחתונה ועליונה).

פירוק QR $\quad A=QR$ כאשר Q מטריצה אורתוגונאלית, ו-R משולשית עליונה.

תזכורת: מטריצה אורתוגונאלית מקיימת: $Q\times Q^t=I$

הפקודה ב-$Matlab$: $[Q,R]=qr(A)$

כיצד נפתור $Ax=b$?

$$(v_i,v_j)=\delta_{ij}=\begin{cases}1, & i=j\\ 0, & i\neq j\end{cases}$$

$$Ax=b\Rightarrow QRx=b$$

$$\begin{cases}Rx=y\\ Qy=b\Rightarrow y=Q^tQy=Q^tb\end{cases}$$

כדי למצוא ערכים עצמיים נפעל באופן הבא:

$$A=[\ldots]$$

$$for\ i=1:10\ do$$

$$\quad [Q\ R]=qr(A);$$

$$\quad A=R\times Q;$$

$$end$$

$$Q$$

Q תהיה מטריצה אלכסונית, שעל האלכסון יופיעו הע"ע. $Q=\begin{pmatrix}\lambda_1 & & \\ & \ddots & \\ & & \lambda_n\end{pmatrix}$

הליך גרהם-שמידט *(Gram-Schmidt)*

מטרתו: להפוך בסיס לבסיס אורתונורמאלי. ניקח וקטור e_1 בתור וקטור בסיסי ראשון וננרמל אותו.

נגדיר $\overline{e_1} = \frac{e_1}{\|e_1\|}$. בשלב שני ניקח שני וקטורים e_1, e_2 בנבנה וקטור $\overline{e_2}$ האנכי ל $\overline{e_1}$ כפתרון של

משוואות: $\|\overline{e_2}\| = 1$, $(\overline{e_2}, \overline{e_1})$. לדוגמה:

$$\overline{e}_2 = \lambda e_2 + \mu e_1$$
$$\overline{e}_3 = t e_3 + t_2 e_2 + t_3 e_1$$

אז $\lambda(e_2, e_2) + \mu(e_1, e_2) = 1 \quad \lambda(e_2, e_1) + \mu(e_1, e_1) = 0,$ וכו'.

ע"י תהליך זה נקבל את Q מ-A, והמטריצות שנעזרנו בהן מרכיבות את מטריצה R.

פירוק *SVD*

נפרק את A באופן הבא: $A = \underbrace{Q \times D \times S}_{R}$ כאשר D היא אלכסונית, ו-S היא אורתוגונאלית.

שיטת *Krylov* למציאת פולינום אופייני למטריצות ריבועיות

לפי משפט קיילי המילטון, אם נציב את A במקום λ בפולינום האופייני, נקבל 0, $\left(p(A) = 0\right)$.

נבחר כל וקטור b שאינו ו"ע של A:

$$p(A)\cdot b = a_n \cdot \underbrace{A^n \cdot b}_{v_n} + a_{n-1} \cdot \underbrace{A^{n-1} \cdot b}_{v_{n-1}} + \ldots + a_1 \cdot \underbrace{A \cdot b}_{v_1} + a_0 b = 0$$

$$a_n v_n + a_{n-1} v_{n-1} + \ldots + a_1 v_1 = -a_0 b, \quad (a_0 = -1) \Rightarrow$$

$$B = \left(v_1, v_2, \ldots, v_n\right)\begin{pmatrix} a_1 \\ \vdots \\ a_n \end{pmatrix} = b$$

$A^2 \sim n^3$ פעולות, $A^k \sim n^3(k-1)$ פעולות, $v_{k+1} = Av_k \sim n^2$ פעולות.

PRECONDITIONNING

תנאים התחלתיים, הנדרשים לפעמים לפני ביצוע האלגוריתם. לדוגמא: שגיאת תנאי.

לדוגמא:
$$\begin{pmatrix} 1 & \ldots & \ldots & 1 \\ -1 & \ddots & 0 & \vdots \\ \vdots & \ddots & 1 & 1 \\ -1 & \ldots & -1 & 1 \end{pmatrix} \xrightarrow{R_i = R_i + R_1} \begin{pmatrix} 1 & & & 1 \\ 0 & \ddots & & 2 \\ \vdots & -1 & \ddots & \\ \vdots & & \ddots & 1 \\ 0 & & -1 & 2 \end{pmatrix} \sim \begin{pmatrix} 1 & 0 & 1 \\ & 2 & 2 \\ & \ddots & \vdots \\ 0 & & 2^{n-1} \end{pmatrix}$$

אם היינו עובדים עם נקודה צפה אז הייתה שגיאה גבוהה, ולכן טעינו בבחירת האלגוריתם.
(אנו צריכים להשתמש בפעולות אלמנטאריות כך שהמספר הכי גבוהה יהיה באלכסון) התנאי הזה
נקראת תנאי האלכסון השולט, או באנגלית *diagonal dominant*.

פירוק חולסקי (*Cholesky*)

אם A סימטרית וחיובית לחלוטין אז אפשר לפרק את A ל- $A = L \cdot L^t$. L- מטריצה משולשת תחתונה.

הפקודה ב-*Matlab*: $L^t = chol(A)$