**שיטות איטרטביות לפתרון מערכות משוואות נומריות**

לאלימינציה של גאוס כשיטת מציאת פתרון של מערכות לינאריות יש מספר חסרונות: מספר הפעולות הוא גדול O(n3) וזה בעיתי למערכות גדולות (n גדול). חוץ מזה שגיאות העיגול מצטברות בחישובים של האלגוריתם הזה, וגם אם המטריצה המקורית דלילה, המטריצות המיוצרות ע"י האלגוריתם לא חייבות להישאר כאלו.

אחת הדרכים לנסות לעקוף את הבעיות הללו הן להשתמש בשיטות איטרטיביות. שיטה איטרטיבית במקרה הזה פירושו חישוב של וקטורים x(k) כאשר x(k) = (x1(k), x2(k), …, xn(k)} ואילו k = 0, 1, 2, … כמה שצריך עד להתכנסות. זהו מקרה ראשון של אלגוריתם איטרטיבי שבו מחשבים וקטור של ערכים שצריכים להתכנס ולא רק סקלר. השאלה מתי וקטור מתחלי להתכנס היא שאלה יותר מורכבת מאשר מתי ערך סקלרי מתכנס, ונתייחס אליו מאוחר יותר.

החסרונות של השיטות הללו הינן שכמו כל השיטות האיטרטיביות הן לא תמיד חייבות להתכנס, כאשר הן מתכנסות שום דבר לא מבטיח באופן כללי שהן יתכנסו בזמן סביר, ולעיתים מטילות על המשתמש חלק מהאחריות לפתרון (כמו למצוא קירוב ראשוני לפתרון).

**בדיקת התכנסות של אלגוריתם**

במקרה זה אלגוריתמים איטרטיביים שמחשבים תוצאות שהם וקטורים ולא סקלריים הבדיקה של ההתכנסות צריכה להיעשות על וקטור ולא סקלר.

המוסכמה של התכנסות של סידרת וקטורים y(k) = (y1(k), y2(k), …, yn(k)}

אם L(y(k)) < ε כאשר L הוא נורמה של וקטור. הנורמות הנפוצות ביותר הן:

***L1(y) = Σ | yi|***

***L2(y) = Σ yi2|***

***L∞(y) = maxi { | yi| }***

במקרה של פתרון איטרטיבי של מערכות משוואות לינאריות, ישנן שתי אפשרויות לבחור למה ישמש כ- y(k)לבדיקת התכנסות האלגוריתם.

1. אפשרות הראשונה היא לבדוק אם ביצוע איטרציה כבר לא משנה הרבה את הווקטור x כלומר:

***y(k)=x(k) -x(k-1)***

2. אפשרות השנייה הוא לבדוק עד כמה x(k) הוא פתרון של המשוואה:

***y(k)= A x(k) - b***

בדוגמאות יעשה שימוש ב- y(k)=x(k) -x(k-1) ו- ∞L= L.

**שיטות פתרון ג'קובי ו-גאוס-זיידל**

שתי השיטות האיטרטיביות הקלאסיות מבוססות על חילוץ המשתנים xi מתוך מערכת המשוואות. נמחיש את זה בשלושה מימדים ונכליל אחר כך.

נניח שיש לנו מערכת Ax=b בשלושה משתנים, יש לנו למעשה את המערכת

***a11 x1 + a12 x2 + a13 x3 = b1***

***a21 x1 + a22 x2 + a23 x3 = b2***

***a31 x1 + a32 x2 + a33 x3 = b3***

נחלץ את כל אחד מה-xi –ים מתוך המשוואה ה- i -יית:

***x1 = (b1- a12 x2 - a13 x3)/a11***

***x2 = (b2- a21 x1 – a23 x3)/a22***

***x3 = (b2- a31 x1 – a32 x2)/a33***

זה מוביל אותנו לנוסחת ג'קובי לחישוב איטרטיבי של פתרון מערכת לינארית 3x3:

***x1(k) = (b1- a12 x2(k-1) - a13 x3(k-1))/a11***

***x2(k) = (b2- a21 x1(k-1) – a23 x3(k-1))/a22***

***x3(k) = (b2- a31 x1(k-1) – a32 x2(k-1))/a33***

שיטת גאוס-זיידל דומה מאד לשיטת ג'קובי, רק שהיא משתמשת ב- xi(k)במקום ב- xi(k)ברגע שאפשר. במקרה של 3x3 זה נראה כך:

***x1(k) = (b1- a12 x2(k-1) - a13 x3(k-1))/a11***

***x2(k) = (b2- a21 x1(k) – a23 x3(k-1))/a22***

***x3(k) = (b2- a31 x1(k) – a32 x2(k))/a33***

האלגוריתם של ג'קובי וגאוס-זיידל הם פשוטים יחסית לתכנות. בהינתן מטריצה A, וקטור b, גודל n וניחוש התחלתי x הרוטינה הזו מממשת את שיטת ג'קובי:

int test\_convergence(double \*x, double \*oldx, int n)

{

double maxvalue, tempvalue;

int i;

double epsilon = 0.000000001;

maxvalue = 0.0;

for(i=0; i < n; i++)

{

tempvalue =fabs(x[i] - oldx[i]);

if (tempvalue > maxvalue)

maxvalue = tempvalue;

} /\* for \*/

if (maxvalue > epsilon)

return 0;

else

return 1;

} /\* test\_convergence \*/

void jacobi(double \*A[], double b[], int n, double x[])

{

int i, j, k, p;

double \*oldx;

oldx = (double \*)malloc(n\*sizeof(double));

if (oldx == NULL)

return;

do {

for (i=0; i < n; i++)

oldx[i] = x[i];

for (i=0; i < n; i++)

{

x[i] = b[i];

for(j=0; j < n; j++)

if ( i != j)

x[i] = x[i] - A[i][j] \* oldx[j]; /\* \*/

x[i] = x[i]/A[i][i];

} /\* for \*/

} while(test\_convergence(x, oldx, n) == 0);

} /\* jacobi \*/

במקרה הזה נעשה שימוש בנורמת האינסוף בכדי לבדוק התכנסות, אבל באותה מידה אפשר היה לממש כל נורמה אחרת על ידי שינו הקוד של test\_convergence.

מימוש אפשרי של גאוס –זיידל שאינו חסך את שכפול הוקטור הוא כמעט זהה, כאשר הפקודה /\* \*/ מוחלפת מ-

x[i] = x[i] - A[i][j] \* oldx[j]; /\* \*/

ל-

x[i] = x[i] - A[i][j] \* x[j]; /\* \*/

כלומר שהתוכנית של גאוס-זיידל הוא

void gaus\_seidel(double \*A[], double b[], int n, double x[])

{

int i, j, k, p;

double \*oldx;

oldx = (double \*)malloc(n\*sizeof(double));

if (oldx == NULL)

return;

do {

for (i=0; i < n; i++)

oldx[i] = x[i];

for (i=0; i < n; i++)

{

x[i] = b[i];

for(j=0; j < n; j++)

if ( i != j)

x[i] = x[i] - A[i][j] \* x[j];

x[i] = x[i]/A[i][i];

} /\* for \*/

} while(test\_convergence(x, oldx,n) == 0);

} /\* gaus\_seidel \*/

הגרסה הבאה של תוכנית גאוס-זיידל "חוסכת" את המימוש של oldx (כלומר אין צורך יותר במערך נוסף בגודל n). כאן אנחנו מוותרים על test\_convergence ומחשבים את נורמת האינסוף תוך כדי חישוב האיטרציה. כמובן שכאן שינוי הנורמה לפיו נבדקת ההתכנסות תחייב שינוי ברוטינה gaus\_seidel עצמה.

void gaus\_seidel(double \*A[], double b[], int n, double x[])

{

int i, j, k, p;

double temp, diff, maxdiff;

double epsilon = 0.000000001;

do {

maxdiff = 0;

for (i=0; i < n; i++)

{

temp = b[i];

for(j=0; j < n; j++)

if ( i != j)

temp = temp - A[i][j] \* x[j];

temp = x[i]/A[i][i];

diff = fabs(x[i] - temp);

x[i] = temp;

if (diff > maxdiff)

maxdiff = diff;

} /\* for \*/

printf("maxdiff = %lf\n", maxdiff);

} while(maxdiff > epsilon);

} /\* gaus\_seidel \*/

**סיבוכיות**

בשתי השיטות סיבוכיות כל איטרציה היא O(n2). לפיכך סיבוכעות האלגוריתמים במקרה של התכנסות הינו O(kn2) כאשר k מספר האיטרציות. אם k קטן יחסית ל-n, האלגוריתמים יכנסו מהר יותר מאלימינציה של גאוס.

אם המטריצות דלילות, הסיבוכיות של כל איטרציה יכולה להיות יחסית למספר המקדמים שוני האפס של המטריצה. בשיטות הללו המקדמים של המטריצה אינם משתנים מאיטרציה לאיטרציה ושום מקדם לא משנה את ערכו מאפס לערך אחר. זה גם יתרון של השיטות הללו לעומת אלימינציה, בנוסף לדיוק נומרי.

**התכנסות מובטחת**

ישנו לפחות מצב אחד שבו מובטח התכנסות של שתי השיטות בלי קשר לבחירה של הניחוש ההתחלתי של הווקטור x. אם ערכי האלכסון הם דומיננטיים במטריצה, כלומר לכל i מתקיים:

***|aii | > Σj | aij |***

אזי שתי השיטות יתכנסו לכל ערך התחלתי של x.

לא נכנס להוכחה של המשפט הזה כאן.

מדובר בתנאי מספיק, הוא לא הכרחי. האלגוריתמים עשויים להתכנס גם כאשר התנאי לא מתקיים.

באופן כללי קשה לקוות שהתנאי הזה יתקיים לעיתים קרובות, ולפיכך יש לקוות שהשיטה תתכנס גם כאשר התנאי הזה לא מתקיים. נראה בהמשך מהו התנאי הכללי שצריך להתקיים בכדי שהשיטות יתכנסו.

**ייצוג מטריציוני של שיטות גקובי ו-גאוס-זיידל.**

אפשר להציג את השיטות ג'קובי וגאוס-זיידל בייצוג מטריציוני פשוט. זה משמש בעיקר לניתוח תיאורטי מאשר למימוש חישוב. בעיקרו של דבר אם נכתוב

***A = L + D + U***

כאשר L היא מטריצה המכילה את החלק התת-אלכסוני התחתון של A ( כלומר מטריצה עם ערכי אפס בכל הכניסות באלכסון ומעליו) , D הוא מטריצה המכילה את האלכסון של A וכל היתר אפס, U מטריצה המכילה את החלק העל-אלכסוני של A כלומר

***D = {dii = aii, i = 1, …, n ,***

***dij = 0 i≠j, i = 1, .. n, j = 1, …, n}***

***L = { lij = aij, | i < j, i = 1, …, n , , j = 1, …, n ,***

***lij = 0 i>j, i = 1, .. n, j = 1, …, n}***

***U = { uij = aij, | i > j, i = 1, …, n , , j = 1, …, n ,***

***uij = 0 i<j, i = 1, .. n, j = 1, …, n}***

את שיטת ג'קובי ניתן להציג כ-

***Dx(k) = b – (L+U)x(k-1)***

או

***x(k) = D-1(b– (L+U)x(k-1))***

או

***x(k) = D-1b – D-1 (L+U)x(k-1)***

ניתוח דומה על שיטת גאוס-זיידל הוא כלהלן:

***Dx(k) = b – Lx(k) - U x(k-1)***

או

***(D+L) x(k) = b – U x(k-1)***

או

***x(k) = (D+L)-1b – (D+L)-1U x(k-1)***

לפיכך, שתי השיטות הם מהצורה

***x(k) = M x(k-1) + g (1)***

כאשר עבור ג'קובי

***M = – D-1 (L+U)***

***g = D-1b***

ועבור גאוס-זיידל

***M = – (D+L)-1U***

***g = (D+L)-1b***

תהי x\* וקטור הפתרון של המערכת Ax = b כלומר

***Ax\* = b***

אבל (1) הוא רק ארגון מחדש של Ax=b לפיכך x\*מקיים גם את

***x\* = M x\* + g (2)***

אם נפחית את (2) מ-(1)

***(x(k)- x\*) = M (x(k-1) - x\*)***

או, אם נסמן x(k)- x\* = kε אזי

***εk = M εk-1***

או, אם נספור מהתחלה

***εk = Mk ε0***

כלומר, ההתכנסות של השיטות תלויה בשאלה האם Mk מתכנסת למטריצת האפס כלומר האם M מטריצה מתכנסת, המקביל במטריצות למספר שערכו המוחלט קטן מאחד. ישנם אלגוריתמים שנועדו לנסות להתאים את A כך שהשיטה האיטרטיבית תתכנס.