Metodi iterativi per la soluzione di sistemi lineari: Jacobi e Gauss-Seidel

Alvise Sommariva

Università degli Studi di Padova
Dipartimento di Matematica Pura e Applicata

15 aprile 2013

Metodi iterativi e diretti

Supponiamo che siano $A \in \mathbb{R}^{n\times n}$ (matrice non singolare), $b \in \mathbb{R}^n$ (vettore colonna) e di dover risolvere il problema $Ax = b$ avente sol. unica x^*. A tal proposito si può utilizzare la fatt. LU con pivoting. Il costo computazionale è in generale di $O(n^3/3)$ operazioni moltiplicative. Questo diventa proibitivo se n è particolarmente elevato.

L'idea dei metodi iterativi è quello di ottenere una successione di vettori $x^{(k)} \to x^*$ cosicchè per $\bar{k} \ll n$ sia $x^{(\bar{k})} \approx x^*$. In generale, la soluzione non è ottenuta esattamente come nei metodi diretti in un numero finito di operazioni (in aritmetica esatta), ma quale limite, accontentandosi di poche iterazioni ognuna dal costo quadratico. Quindi il costo totale sarà di ordine $O(\bar{k} \cdot n^2)$.

Metodi iterativi stazionari

Supponiamo che la matrice non singolare $A \in \mathbb{R}^{n\times n}$ sia tale che

$$A = M - N, \text{ con M non singolare.}$$

Allora $Mx - Nx = Ax = b$ e quindi $Mx = Nx + b$. Moltiplicando ambo i membri per M^{-1} e posto $\phi(x) = M^{-1}Nx + b$ abbiamo $x = M^{-1}Nx + M^{-1}b = \phi(x)$. Viene quindi naturale utilizzare la succ. del metodo di punto fisso

$$x^{(k+1)} = \phi(x^{(k)}) = M^{-1}Nx^{(k)} + M^{-1}b$$

La matrice $P = M^{-1}N$ si dice di *iterazione* e non dipende, come pure b dall'indice di iterazione k. Per questo motivo tali metodi si chiamano iterativi stazionari.

Quale utile notazione, sia inoltre $A = D - E - F$ con D la matrice diagonale estratta da A, E, F rispettivamente triangolari inferiore e superiore.

Esempio A, E, F

```
>> A=[1 2 3 4; 5 6 7 2; 8 9 1 2; 3 4 5 1]
A =
     1     2     3     4
     5     6     7     2
     8     9     1     2
     3     4     5     1
>> E=-(tril(A)-diag(diag(A)))
E =
     0     0     0     0
    -5     0     0     0
    -8    -9     0     0
    -3    -4    -5     0
>> F=-(triu(A)-diag(diag(A)))
F =
     0    -2    -3    -4
     0     0    -7    -2
     0     0     0    -2
     0     0     0     0
>> % A=diag(diag(A))-E-F.
```

Metodo di Jacobi e di Gauss-Seidel

Nel caso del metodo di Jacobi si ha

$$M = D,\ N = E + F \tag{1}$$

e quindi

$$\begin{aligned} P &= M^{-1}N = D^{-1}(E+F) = D^{-1}(D - D + E + F) \\ &= D^{-1}(D - A) = I - D^{-1}A \end{aligned} \tag{2}$$

Si osservi che se D è non singolare allora il metodo di Jacobi, almeno in questa versione di base, non può essere utilizzato visto che in (4) non ha senso la scrittura D^{-1}.
Il metodo di Gauss-Seidel è definito quale metodo stazionario in cui

$$M = D - E, N = F \tag{3}$$

e quindi

$$P = M^{-1}N = (D - E)^{-1}F \tag{4}$$

Teorema di Hensel

Siano $\{\lambda_k\}_{k=1,\ldots,n}$ gli autovalori di una matrice A. Si definisce raggio spettrale $\rho(A)$ la quantità

$$\max_{k=1,\ldots,n} |\lambda_k|.$$

Teorema. Un metodo iterativo stazionario consistente $x^{(k+1)} = Px^{(k)} + c$ converge per ogni vettore iniziale x_0 se e solo se $\rho(P) < 1$.

Alcuni teoremi di convergenza

Una matrice A si dice tridiagonale se $A_{i,j} = 0$ qualora $|i - j| > 1$.

Esempio:

$$\begin{pmatrix} 1 & 2 & 0 & 0 & 0 \\ 1 & 5 & 7 & 0 & 0 \\ 0 & 2 & 6 & 9 & 0 \\ 0 & 0 & 4 & 4 & 2 \\ 0 & 0 & 0 & 5 & 3 \end{pmatrix}$$

Teorema. Per matrici tridiagonali $A = (a_{i,j})$ con componenti diagonali non nulle, i metodi di Jacobi e Gauss-Seidel sono o entrambi convergenti o divergenti e il tasso di convergenza del metodo di Gauss-Seidel è il doppio di quello del metodo di Jacobi (il che vuol dire che *asintoticamente* sono necessarie metà iterazioni del metodo di Gauss-Seidel per ottenere la stessa precisione del metodo di Jacobi).

Alcune definizioni

Ricordiamo che A è a predominanza diagonale (per righe) se per ogni $i = 1, \ldots, n$ risulta

$$|a_{i,i}| \geq \sum_{j=1, j\neq s}^{n} |a_{i,j}|$$

e per almeno un indice s si abbia

$$|a_{s,s}| > \sum_{j=1, j\neq s}^{n} |a_{s,j}|.$$

Se per tutti gli indici s si ha $|a_{s,s}| > \sum_{j=1, j\neq s}^{n} |a_{s,j}|$ allora la predominanza diagonale è in senso stretto. Ad esempio la matrice

$$A = \begin{pmatrix} 4 & -4 & 0 \\ -1 & 4 & -1 \\ 0 & -4 & 4 \end{pmatrix}$$

è a predominanza diagonale per righe (non stretta).

Teoremi di convergenza

- Sia A una matrice quadrata a predominanza diagonale per righe, in senso stretto. Allora il metodo di Jacobi converge alla soluzione di $Ax = b$, qualsiasi sia il punto $x^{(0)}$ iniziale.
- Sia A è a predominanza diagonale per righe, in senso stretto. Allora il metodo di Gauss-Seidel risulta convergente, qualsiasi sia il punto $x^{(0)}$ iniziale.

Tali teoremi valgono se A^T è a predominanza diagonale per righe in senso stretto (cioè A è a predominanza diagonale per colonne in senso stretto).

Alcune definizioni

La matrice A è simmetrica se $A = A^T$. Una matrice A è definita positiva se ha tutti gli autovalori positivi. Ad esempio la matrice

$$A = \begin{pmatrix} 4 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 4 \end{pmatrix}$$

è simmetrica e definita positiva:

```
>> A=[4 -1 0; -1 4 -1; 0 -1 4];
>> eig(A) % AUTOVALORI DI A.
ans =
    2.5858
    4.0000
    5.4142
>>
```

Alcuni teoremi di convergenza

Ricordiamo che equivalentemente una matrice A è definita positiva se

$$x^T Ax > 0, \text{ per ogni } x \neq 0.$$

Teorema. Sia A una matrice simmetrica, non singolare con elementi principali $a_{i,i} \neq 0$. Allora il metodo di Gauss-Seidel è convergente per qualsiasi scelta del punto iniziale $x^{(0)}$ se e solo se A è definita positiva.

Esercizio in Matlab I

- Eseguire una routine `iterstat` che implementi, date due matrici M ed N, un metodo iterativo stazionario. Quale criterio di arresto si utilizzi

$$\|x^{(k+1)} - x^{(k)}\|_\infty < \text{tol}$$

con `tol` tolleranza richiesta dall'utente (es. $\texttt{tol} = 10^{(-12)}$).
- Usare `iterstat` per definire `jacobi` e `gs` che implementano il metodo di Jacobi e Gauss Seidel, per risolvere il problema $Ax = b$. Se necessario usare i comandi `diag`, `tril` e `triu`.
- Si calcoli la matrice di Poisson P_{20} di ordine 20 aiutandosi con

```
>> help gallery
```

- Sia b il vettore composto di componenti uguali a 1, avente lo stesso numero di righe di P_{20}. Si risolva col metodo di Jacobi e Gauss Seidel il problema $P_{20}x = b$, con tolleranza di $10^{(-12)}$, partendo da $x^0 = [0 \dots 0]$ Convergono *numericamente*? Se sì , quante iterazioni servono?

SOR

La versione di Gauss-Seidel con la scelta del parametro ω è nota in letteratura come *successive over relaxation* (**SOR**)

$$x^{(k+1)} = \left(\frac{D}{\omega} - E\right)^{-1} \left(\left(\frac{1}{\omega} - 1\right) D + F\right) x^{(k)} + \left(\frac{D}{\omega} - E\right)^{-1} b. \tag{5}$$

Deriva dallo descrivere esplicitamente una iter. di Gauss-Seidel

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left[b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right]$$

e sostituirla con la combinazione convessa

$$x_i^{(k+1)} = \frac{\omega}{a_{ii}} \left[b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right] + (1-\omega) x^{(k)}.$$

Gauss-Seidel è SOR per $\omega = 1$. Una versione di SOR scaricabile presso Netlib.

Codice Matlab SOR

```
function [x, error, iter, flag] = sor(A, x, b, w, max_it,
    tol)

%
% Successive Over-Relaxation Method
% (Gauss-Seidel method when omega = 1 ).
%
% input   A        REAL matrix
%         x        REAL initial guess vector
%         b        REAL right hand side vector
%         w        REAL relaxation scalar
%         max_it   INTEGER maximum number of iterations
%         tol      REAL error tolerance
%
% output  x        REAL solution vector
%         error    REAL error norm
%         iter     INTEGER number of iterations performed
%         flag     INTEGER: 0 = solution found to tolerance
%                           1 = no convergence given max_it
```

Codice Matlab SOR

```
flag = 0;  iter = 0; bnrm2 = norm( b );
if  ( bnrm2 == 0.0 ), bnrm2 = 1.0; end
r = b - A*x; error = norm( r ) / bnrm2;
if ( error < tol ) return, end
[ M, N, b ] = split( A, b, w, 2 );
for iter = 1:max_it
   x_1 = x; x   = M \ ( N*x + b );
   error = norm( x - x_1 ) / norm( x );
   if ( error <= tol ), break, end
end
b = b / w;
if ( error > tol ) flag = 1; end;  % no conv.
```