

Risoluzione di sistemi di equazioni lineari

Metodi Iterativi

Ángeles Martínez Calomardo

<http://www.math.unipd.it/~acalomar/DIDATTICA/>
angeles.martinez@unipd.it

Laurea in Informatica
A.A. 2018–2019

Matrici in formato sparso

Si chiamano matrici *sparse* quelle in cui il numero di elementi diversi da zero (nonzeri) è proporzionale ad n invece che a n^2 , essendo n l'ordine della matrice.

Esempio: data la matrice sparsa F

$F =$

1	4	0	-3	0
0	2	0	0	0
0	0	2	0	0
-3	12	0	-7	0
0	0	0	0	2

è possibile memorizzarne solo i nonzeri.

La conversione dalla memorizzazione in formato pieno alla memorizzazione in formato sparso avviene con il comando `sparse`:

```
>> S = sparse(F);
```

<https://sparse.tamu.edu/>

Scaricare dal sito il file 1138_bus.

```
load('1138_bus.mat'); B=pr
```

Matrici in formato sparso

Si noti la differente occupazione di memoria nei due formati di memorizzazione della matrice.

```
>> whos F
```

Variables in the current scope:

Attr	Name	Size	Bytes	Class
	F	5x5	200	double

Total is 25 elements using 200 bytes

```
>> whos S
```

Variables in the current scope:

Attr	Name	Size	Bytes	Class
	S	5x5	132	double

Total is 9 elements using 132 bytes

Per matrici di dimensioni maggiori la differenza di occupazione di memoria diventa ancora più significativa. Oltre ad un risparmio nell'occupazione di memoria, il formato sparso può risultare vantaggioso anche considerando la velocità di esecuzione in quanto tutte le operazioni che coinvolgerebbero gli elementi uguali a zero, non vengono realizzate quando la matrice è memorizzata in formato sparso.

Fattorizzazione LU di matrici sparse

fill-in

Per matrici sparse la fattorizzazione LU non viene utilizzata per il fatto che i fattori L ed U di una matrice sparsa possono essere densi (cioè con pochi elementi nulli) o comunque molto meno sparsi della matrice di partenza.

Per renderci conto di ciò costruiamo una matrice sparsa in MATLAB/OCTAVE nel seguente modo:

```
n = 100;  
A = 4*diag(ones(n,1))-diag(ones(n-1,1),-1)-diag(ones(n-1,1),1);  
A(1,:) = ones(1,n);  
A(:,1) = ones(n,1);
```

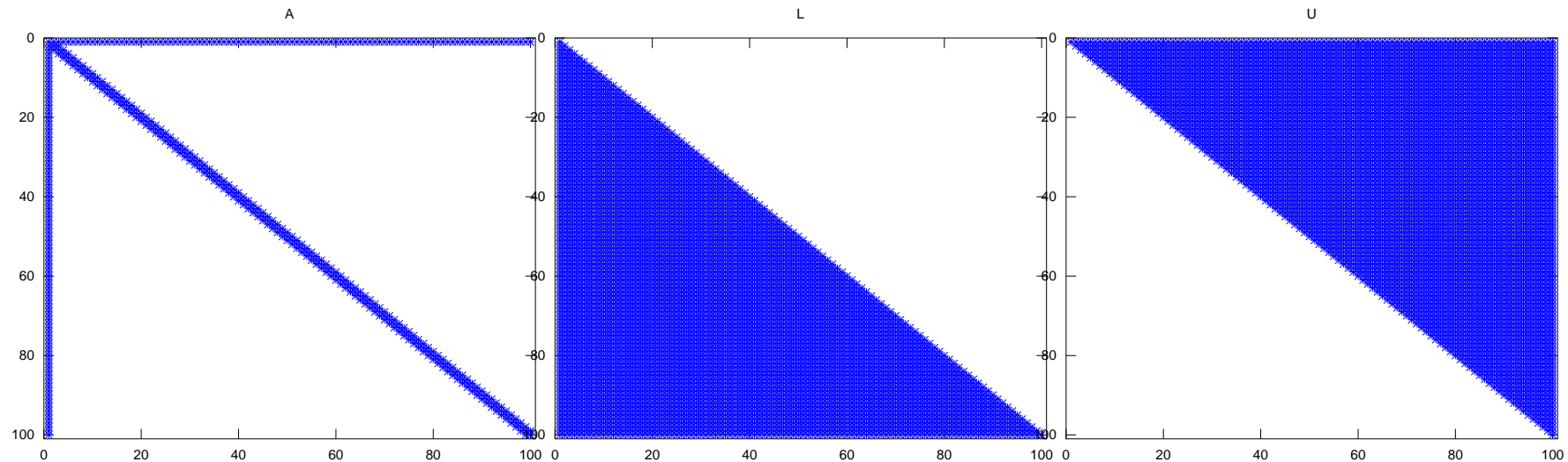
Il *pattern* di sparsità (posizione dei nonzeri) si può visualizzare così:

```
figure; spy(A); title('A');
```

La fattorizzazione LU riempie i fattori triangolari come si vede dal loro *pattern*.

```
[L,U,P] = lu(A);  
figure; spy(L); title('L');  
figure; spy(U); title('U');
```

Fattorizzazione LU di matrici sparse



La soluzione di sistemi lineari sparsi motiva l'uso di metodi alternativi ai metodi diretti: i metodi **iterativi**.

I metodi iterativi cercano la soluzione del sistema generando una successione di vettori che sotto certe condizioni converge alla soluzione “esatta”.

Sotto la condizione che A sia non singolare e che inoltre $a_{ii} \neq 0, i = 1, \dots, n$, scrivendo

Se M non e' invertibile

$$A = M - N,$$

con M invertibile, il sistema lineare $Ax = b$ si può dunque riformulare come segue:

$$\begin{aligned} (M - N)x = b &\implies Mx - Nx = b \implies \\ \boxed{Mx = Nx + b} &\implies x = M^{-1}Nx + M^{-1}b \implies x = Bx + q \end{aligned}$$

M nuova iterata, N qu

La matrice $B = M^{-1}N$ definita nell'ultima equazione si definisce **matrice di iterazione**.

Abbiamo ottenuto il metodo iterativo:

Per fermarsi si possono usare vari criteri, quello con lo scarto o quello sui residui. Og

$$\boxed{Mx^{(k+1)} = Nx^{(k)} + b}$$

$$k = 0, 1, \dots$$

Si predilige ----->rispetto a

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

$$k = 0, 1, \dots \quad (1)$$

$$x^{(k+1)} = Bx^{(k)} + q$$

$$k = 0, 1, \dots$$

I piu' facili sistemi da risolvere sono i diagonali e i triangolari, gli altri hanno un costo di risoluzione cubico. Per questo motivo per M solitamente si sceglie

Metodi iterativi stazionari

Per caratterizzare i singoli metodi iterativi, si parte dal fatto che A si può scrivere come

$$A = -E + D - F$$

dove:

$$-F = \begin{pmatrix} 0 & a_{12} & \dots & \dots & a_{1n} \\ & \ddots & \ddots & & \vdots \\ & & \ddots & \ddots & \vdots \\ & & & \ddots & \vdots \\ & & & & 0 & a_{n-1,n} \\ & & & & & 0 \end{pmatrix} \quad -E = \begin{pmatrix} 0 & & & & & \\ a_{21} & 0 & & & & \\ \vdots & \ddots & \ddots & & & \\ \vdots & & \ddots & \ddots & & \\ \vdots & & & \ddots & \ddots & \\ a_{n1} & \dots & \dots & a_{n,n-1} & 0 \end{pmatrix} \quad D = \begin{pmatrix} a_{11} & 0 & 0 & 0 \\ 0 & a_{22} & 0 & 0 \\ & & \ddots & \\ 0 & 0 & 0 & a_{nn} \end{pmatrix}$$

I metodi iterativi stazionari si ottengono definendo opportunamente N ed M in funzione di $-E, D, -F$:

- 1 **Metodo di Jacobi.** $M = D$; $N = (E + F)$. Il metodo da implementare e' sempre lo stesso, basta cambiare le m
- 2 **Metodo di Gauss-Seidel.** $M = D - E$; $N = F$.
- 3 **Metodo SOR.** $M = D - \omega E$; $N = (1 - \omega)D + \omega F$, con $\omega \in (0, 2)$ parametro reale (per $\omega = 1$ il metodo SOR coincide con quello di Gauss-Seidel).

Nessuno di questi metodi puo' essere utilizzato se anche una sola di queste matrici ha uno zero sulla diagonale principale.

Convergenza dei metodi iterativi

Convergono \Leftrightarrow il grado spettrale della matrice di iterazione e' minore di 1

Esiste una condizione necessaria e sufficiente di convergenza che accomuna tutti i succitati metodi iterativi:

Teorema

Un metodo iterativo stazionario $x^{(k+1)} = Bx^{(k)} + q$ converge per ogni scelta del vettore iniziale $x^{(0)}$ se e solo se tutti gli autovalori della matrice di iterazione B sono in modulo minori di 1, ovvero se $\rho(B) < 1$.

Esercizio

Data la matrice

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

- ① Calcolare il raggio spettrale delle matrici di iterazione di Jacobi e di Gauss-Seidel.
- ② Quale dei due metodi convergerà più velocemente alla soluzione di $Ax = b$?

Convergenza dei metodi iterativi

Risoluzione

```
>>> A = [4 -1 0 0 ; -1 4 -1 0 ; 0 -1 4 -1; 0 0 -1 4];
>>> D = diag(diag(A));
>>> E = -tril(A,-1);
>>> F = -triu(A,1);
>>> B_J = inv(D)*(E+F);
>>> B_S = inv(D-E)*F
>>> eig(B_J)'
ans =
```

```
-0.40451 -0.15451 0.15451 0.40451
```

e' il raggio spettrale

```
>>> eig(B_S)'
ans =
```

```
0.00000 0.16363 0.02387 -0.00000
```

```
>>> rho_J = max(abs(eig(B_J)))
rho_J = 0.40451
>>> rho_S = max(abs(eig(B_S)))
rho_S = 0.16363
```

metodi a conv lineare -> lenti. Sono metodi di punto fisso, quindi devono avere

Sono uno il quadrato dell'altro in questo caso

In questo caso entrambi i metodi sono convergenti come si poteva vedere anche dall'esame della matrice che risulta essere **strettamente diagonale dominante**.

Implementazione del Metodo iterativo di Jacobi

Si scriva una function `jacobi.m` che implementi il metodo iterativo di Jacobi:

$$Mx^{(k+1)} = Nx^{(k)} + b, \quad \text{con} \quad M = D \quad N = -(L + U).$$

La function deve avere come dati di ingresso:

- la matrice A ,
- il termine noto del sistema b ,
- il vettore iniziale $x^{(0)}$,
- il numero massimo di iterazioni `itmax` e
- la tolleranza `tol`.

Si usi il test sulla norma dello scarto come criterio di arresto del metodo iterativo di Jacobi:

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

La function dovrà restituire il vettore soluzione finale, il numero di iterazioni effettuate e il vettore delle norme degli scarti calcolati ad ogni iterazione.

Implementazione del Metodo iterativo di Jacobi

```
function [x,iter , vscarti]=jacobi(A,b,x0,itmax,tol)
% Metodo iterativo di JACOBI
% [x,iter , vscarti]=jacobi(A,b,x0,itmax,tol)
% vettore iniziale x_0
xold = x0;
% partizionamento della matrice
D = diag(diag(A));
E = -tril(A,-1);
F = -triu(A,1);
M = D;
N = (E+F);
% vettore contenente le norme degli scarti
vscarti = [];
scarto = 1;
iter = 0;
% ciclo iterativo  $M x_{k+1} = N x_k + b$ ;
while scarto > tol && iter < itmax
    iter = iter + 1;
    tnoto = N*xold + b;
    x = M\tnoto;
    scarto = norm(x-xold);
    vscarti = [vscarti;scarto];
    xold = x;
end
```

Test della function `jacobi`

Esercizio

Sia data la matrice

$$A = \begin{pmatrix} 4 & 3 & 0 \\ 2 & 3 & 0.5 \\ 0 & 1 & 2.5 \end{pmatrix},$$

*si vuole risolvere il sistema lineare $Ax = b$; con $b = A * \text{ones}(3, 1)$.*

- ① *Si dica perché il metodo di Jacobi converge.*
- ② *Si usi la function `jacobi` con una tolleranza pari a $1e-8$ per il test di arresto, con vettore iniziale il vettore nullo, e `itmax = 200`.*

Il calcolo del raggio spettrale lo facciamo a livello didattico. Nella pratica non ha senso. Ci sono delle famiglie di matrici per cui Jacobi converge se le matrici s

Esercizio proposto

Esercizio

COPERTO PERCHE' NON CORRETTO IN QUESTO CASO, DURANTE LA LEZIONE E' STATA USATA UNA NOTAZIONE DIVERSA DA QUELLA UTILIZZATA

Esercizio

Si confrontino i metodi di Jacobi e di Gauss-Seidel per la risoluzione del sistema lineare dell'esercizio precedente. Si realizzi un unico grafico di convergenza con i profili dei due metodi.

Esercizio proposto 2

- ① Si risolva utilizzando la function `jacobi.m` il sistema lineare $Ax = b$ dove

E' una matrice pentadiagonale

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

Matrice discretizzata con u

`nx = 5; R=numgrid('`

e b è tale che il vettore soluzione ha tutte le componenti pari a 1. Usare `tol` = 10^{-9} , `itmax` = 200 e il vettore nullo come soluzione iniziale.

- ② Si calcoli inoltre la velocità di convergenza del metodo: $R = -\log_{10}(|\lambda_1|)$ essendo $|\lambda_1|$ il raggio spettrale della matrice di iterazione del metodo di Jacobi.
- ③ Si stimi il numero di iterazioni necessarie per ridurre di un fattore 10^{-9} la norma dell'errore iniziale, ovvero: $k = \left\lceil \frac{9}{R} \right\rceil$. Confrontare il numero ottenuto con il numero di iterazioni eseguite dalla function `jacobi.m`.

Metodo SOR

Esercizio

A partire dalla function `jacobi.m`, si scriva una function di nome `sor.m` che implementa il metodo di sovrarilassamento (SOR), definito così:

$$Mx^{(k+1)} = Nx^{(k)} + \omega b,$$

rispetto ai due metodi visti in precedenza, oltre a set

dove

$$M = D - \omega E$$

$$N = (1 - \omega)D + \omega F$$

essendo $\omega \in (0, 2)$ un parametro reale.

La function `sor` dovrà avere come parametri di ingresso la matrice A , il termine noto b , il **vettore** soluzione iniziale $x^{(0)}$, il numero massimo di iterazioni `itmax`, la tolleranza `tol` e il parametro di rilassamento, ω . In output la function deve restituire il **vettore** soluzione x , il numero di iterazioni effettuate e il **vettore** `vscarti` contenente le norme degli scarti calcolati ad ogni iterazione. Si usi il test sulla norma dello scarto come criterio di arresto $\|x^{(k+1)} - x^{(k)}\| < \text{tol}$.

La sintassi della function deve essere la seguente:

```
function [x, iter, vscarti] = sor(A, b, x0, itmax, tol, omega)
```

Esercizio proposto 3

Metodo SOR

Si produca uno script chiamato `scriptsor.m` che risolva il sistema lineare $Ax = b$ dove

$$A = \begin{pmatrix} 2 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 2 \end{pmatrix}$$

e b si ottiene imponendo che la soluzione vera sia $x = \text{ones}(5,1)$ con i metodi di Jacobi, Gauss-Seidel (SOR per $\omega = 1$) e SOR con $\omega = 1.4$. Usare `tol` = 10^{-10} , `itmax` = 200 e il vettore nullo come soluzione iniziale. Si riportino in un unico grafico semilogaritmico i profili di convergenza ottenuti con i tre metodi.

Si spieghi perchè si è certi che tutti e tre i metodi convergono alla soluzione di un sistema lineare con questa matrice dei coefficienti.

Stima della velocità asintotica di convergenza

non detto durante la parte teorica del corso, fatto solo in lab.

- Si può dimostrare che se $|\lambda_1| > |\lambda_2| \geq \dots \geq |\lambda_n|$ allora per $k \rightarrow \infty$

$$\mathbf{e}^{(k+1)} = B\mathbf{e}^{(k)} \approx \lambda_1^{k+1} c_1 \mathbf{v}_1 \quad \Longrightarrow \quad \lim_{k \rightarrow \infty} \frac{\|\mathbf{e}^{(k+1)}\|}{\|\mathbf{e}^{(k)}\|} = |\lambda_1|.$$

- Per $k \rightarrow \infty$ anche il rapporto tra le norme degli scarti tende a $|\lambda_1|$.
- Alla fine delle iterazioni possiamo calcolare una stima del fattore di convergenza del metodo (stima del modulo dell'autovalore dominante della matrice di iterazione B), come rapporto tra la norma euclidea di due scarti consecutivi:

```
asint=vscarti(2:iter)./vscarti(1:iter-1);
```

deve essere ripetuta la condizione in alto. Gli autovalori non possono avere lo stesso modulo di un