Corso di Laurea Specialistica in Ingegneria Meccanica e in Ingegneria dell'Energia

Progetto numerico al calcolatore

Parte I

Implementazione del metodo del Gradiente Coniugato Modificato (GCM) per la soluzione di sistemi lineari sparsi, simmetrici e definiti positivi

Indice

1 Richiami dello schema del GCM

Il metodo del Gradiente Coniugato Modificato (GCM) si basa sulla tecnica del Gradiente Coniugato (GC) sviluppata nei primi anni '50 da Hestenes e Stiefel per la soluzione di sistemi lineari con matrice simmetrica e definita positiva. Secondo il metodo del GC, la soluzione del generico sistema:

$$A\boldsymbol{x} = \boldsymbol{b} \tag{1}$$

si ottiene costruendo una successione di approssimazioni successive $\boldsymbol{x}_k$ a partire da un vettore iniziale arbitrario $\boldsymbol{x}_0$ secondo la formula ricorrente:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k \tag{2}$$

Lo scalare α_k viene scelto in modo da minimizzare una opportuna misura Φ dell'errore commesso assumendo $\boldsymbol{x}_{k+1}$ come soluzione del sistema (1). Adottando come misura dell'errore la seguente forma quadratica:

$$\Phi(\boldsymbol{x}_{k+1}) = \boldsymbol{e}_{k+1}^T A \boldsymbol{e}_{k+1} = (\boldsymbol{x} - \boldsymbol{x}_{k+1})^T A (\boldsymbol{x} - \boldsymbol{x}_{k+1}) \tag{3}$$

si ottiene:

$$\alpha_k = \frac{\boldsymbol{p}_k^T \boldsymbol{r}_k}{\boldsymbol{p}_k^T A \boldsymbol{p}_k} \tag{4}$$

dove $\boldsymbol{r}_k = \boldsymbol{b} - A\boldsymbol{x}_k$ è il residuo associato alla k-esima iterazione. Il nuovo residuo relativo alla soluzione $\boldsymbol{x}_{k+1}$ si ricava facilmente dalla (2):

$$\boldsymbol{r}_{k+1} = \boldsymbol{b} - A(\boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k) = \boldsymbol{r}_k - \alpha_k A \boldsymbol{p}_k \tag{5}$$

La direzione di ricerca $\boldsymbol{p}_{k+1}$ utilizzata per calcolare la nuova approssimazione della soluzione viene determinata A-ortogonalizzando $\boldsymbol{p}_{k+1}$ rispetto a $\boldsymbol{p}_k$ secondo la formula ricorrente:

$$\boldsymbol{p}_{k+1} = \boldsymbol{r}_{k+1} + \beta_k \boldsymbol{p}_k \tag{6}$$

Imponendo la condizione di A-ortogonalità fra i vettori $\boldsymbol{p}_k$ e $\boldsymbol{p}_{k+1}$:

$$\boldsymbol{p}_{k+1}^T A \boldsymbol{p}_k = 0 \tag{7}$$

si ottiene la relazione che definisce lo scalare β_k:

$$\beta_k = -\frac{\boldsymbol{r}_{k+1}^T A \boldsymbol{p}_k}{\boldsymbol{p}_k^T A \boldsymbol{p}_k} \tag{8}$$

Lo schema del GC va inizializzato scegliendo un vettore soluzione arbitrario $\boldsymbol{x}_0$, il corrispondente residuo $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$ e ponendo $\boldsymbol{p}_0 = \boldsymbol{r}_0$. Si applicano quindi in successione le formule ricorrenti (2), (5) e (6) mediante le (4) e (8) finché una opportuna norma del residuo, ad esempio la norma euclidea, risulta inferiore ad una prefissata tolleranza. Si osservi che è generalmente

molto più conveniente fissare la tolleranza in funzione della norma del termine noto. Il residuo, quindi, su cui viene effettivamente eseguito il controllo sarà quello relativo:

$$r_r = \frac{\|\boldsymbol{r}_{k+1}\|}{\|\boldsymbol{b}\|} < \text{toll} \tag{9}$$

rendendo adimensionale la tolleranza prescelta. L'ultima approssimazione $\boldsymbol{x}^*$ del vettore $\boldsymbol{x}$ così ottenuta viene assunta come soluzione del sistema lineare (1). Va osservato che, specialmente in problemi mal-condizionati, gli errori di arrotondamento che si accumulano nel calcolo del residuo mediante la formula ricorrente (5) possono far sì che quest'ultimo sia in realtà anche molto diverso dal residuo vero e che, di conseguenza, $\boldsymbol{x}^*$ sia lontano dalla soluzione esatta del sistema (1). Per evitare tale inconveniente, è opportuno calcolare al termine della procedura iterativa del GC il residuo relativo vero r_r^*:

$$r_r^* = \frac{\|\boldsymbol{b} - A\boldsymbol{x}^*\|}{\|\boldsymbol{b}\|} < \text{toll} \tag{10}$$

e verificare che sia effettivamente inferiore alla tolleranza prefissata.

Si può dimostrare che il vettore $\boldsymbol{r}_{k+1}$ risulta ortogonale ai precedenti $\boldsymbol{r}_0, \ldots, \boldsymbol{r}_k$ e che, pertanto, il residuo n-esimo, avendo indicato con n la dimensione del sistema (1), deve necessariamente essere nullo. A causa degli errori di arrotondamento del calcolatore, tuttavia, ciò non si verifica ed il GC, pur essendo teoricamente un metodo diretto, va catalogato fra le tecniche iterative. In conseguenza a questo aspetto, che incide negativamente sulla performance del solutore, il GC, se non opportunamente accelerato, non offre alcun vantaggio rispetto ai metodi diretti classici.

La convergenza del GC risulta particolarmente accelerata qualora gli autovalori della matrice A siano raggruppati attorno ad unico valore, il che corrisponde ad interpretare geometricamente l'iper-ellissoide associato alla matrice A come quasi-sferico. Si può, pertanto, cercare di risolvere un nuovo sistema:

$$B\boldsymbol{y} = \boldsymbol{c} \tag{11}$$

equivalente al precedente, in cui, tuttavia, la matrice B presenta i propri autovalori raggruppati attorno all'unità. Il nuovo sistema (11), detto sistema precondizionato, è collegato al sistema originale (1) mediante le seguenti relazioni:

$$B = X^{-1}AX^{-1} \qquad \boldsymbol{y} = X\boldsymbol{x} \qquad \boldsymbol{c} = X^{-1}\boldsymbol{b} \tag{12}$$

La matrice $K^{-1} = X^{-1}X^{-1}$ è detta matrice di precondizionamento. Se si sceglie $K^{-1} = A^{-1}$ si osserva facilmente che la matrice B coincide con l'identità e la soluzione di (11) è immediata. È ovvio che tale opportunità è del tutto teorica, in quanto se si conoscesse già l'inversa di A non servirebbe ricorrere ad alcun metodo per risolvere il sistema (1). Tuttavia, questa osservazione ci permette di concludere che la matrice di precondizionamento prescelta sarà tanto più efficace quanto più il prodotto AK^{-1} si avvicinerà in qualche modo all'identità .

Riscrivendo le equazioni del GC per il sistema (11) e successivamente ripristinando le variabili originali, si ottiene lo schema del Gradiente Coniugato Modificato (GCM):

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k \\ \boldsymbol{r}_{k+1} &= \boldsymbol{r}_k - \alpha_k A\boldsymbol{p}_k \\ \boldsymbol{p}_{k+1} &= K^{-1}\boldsymbol{r}_{k+1} + \beta_k \boldsymbol{p}_k \end{aligned} \tag{13}$$

dove:

$$\alpha_k = \frac{\boldsymbol{p}_k^T \boldsymbol{r}_k}{\boldsymbol{p}_k^T A \boldsymbol{p}_k} \qquad \beta_k = -\frac{\boldsymbol{r}_{k+1}^T K^{-1} A \boldsymbol{p}_k}{\boldsymbol{p}_k^T A \boldsymbol{p}_k} \tag{14}$$

Lo schema del GCM va inizializzato partendo da un vettore arbitrario $\boldsymbol{x}_0$, con il relativo residuo $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$, e dalla direzione $\boldsymbol{p}_0 = K^{-1}\boldsymbol{r}_0$. La soluzione iniziale può essere migliorata con alcune (poche) iterazioni effettuate con lo schema delle Correzioni Residue (CR), che coincide con lo schema di Jacobi applicato al sistema (11) ove si assuma la diagonale di B unitaria:

$$\begin{aligned} \boldsymbol{y}_0 &= \boldsymbol{c} \\ \boldsymbol{y}_1 &= (I - B)\,\boldsymbol{y}_0 + \boldsymbol{c} \\ &\vdots \\ \boldsymbol{y}_{k+1} &= (I - B)\,\boldsymbol{y}_k + \boldsymbol{c} \end{aligned} \tag{15}$$

Espresso nelle variabili originali, si verifica facilmente che lo schema (15) risulta:

$$\begin{aligned} \boldsymbol{x}_0 &= K^{-1}\boldsymbol{b} \\ \boldsymbol{x}_1 &= \boldsymbol{x}_0 + K^{-1}\boldsymbol{r}_0 \\ &\vdots \\ \boldsymbol{x}_{k+1} &= \boldsymbol{x}_k + K^{-1}\boldsymbol{r}_k \end{aligned} \tag{16}$$

Lo schema del GCM si colloca, allo stato attuale, fra i metodi più moderni ed efficienti per la soluzione di sistemi lineari sparsi, simmetrici e definiti positivi, quali tipicamente si possono derivare dall'applicazione del metodo degli elementi finiti in svariati problemi di ingegneria civile, ambientale, meccanica, ecc. Uno dei punti vincenti del GCM sta nella semplicità di implementazione e nel ridotto costo computazionale rischiesto ad ogni iterazione, essendo le operazioni più dispendiose limitate a due prodotti matrice-vettore ($A\boldsymbol{p}_k$ e $K^{-1}\boldsymbol{r}_{k+1}$). Tuttavia, per sfruttare appieno da un punto di vista computazionale le potenzialità di tale tecnica è opportuno memorizzare la matrice del sistema non in forma tabellare, bensì in forma compatta. Sarà pertanto necessario ri-programmare l'operazione di prodotto matrice-vettore secondo questa nuova notazione, come vedremo in seguito. A tali costi, va infine aggiunto quello per il calcolo iniziale della matrice di precondizionamento K^{-1} e della sua applicazione nello schema (13)-(14).

2 Memorizzazione di una matrice in forma compatta

Come vedremo meglio in seguito, le matrici ottenute dalla discretizzazione agli elementi finiti di problemi ingegneristici sono tipicamente molto sparse, con una percentuale di elementi nulli (grado di sparsità) anche superiore al 99%. Risulta pertanto molto più conveniente da un punto di vista computazionale memorizzare la matrice A in forma compatta, escludendo, cioè, tutti i coefficienti nulli. Oltre ad un ragguardevole risparmio di memoria, questa tecnica di memorizzazione consente anche di trascurare tutti i prodotti in cui uno dei fattori è a priori nullo ed il cui risultato è ovviamente già noto.

Il modo in cui viene memorizzata la matrice A del sistema è detto Compressed Row Storage (CRS).

2.1 Caso non simmetrico

Data una generica matrice A quadrata di ordine n, la tecnica di memorizzazione CRS prevede la generazione di 3 vettori:

1. `SYSMAT`: vettore di numeri reali contenente gli nt coefficienti non nulli della matrice A memorizzati in successione per righe;

2. `JA`: vettore di numeri interi contenente gli nt indici di colonna dei corrispondenti elementi memorizzati in `SYSMAT`;

3. `IA`: vettore di numeri interi con $n+1$ componenti, contenente le posizioni in cui si trova in `SYSMAT` il primo elemento di ciascuna riga di A.

Il vettore `IA` è chiamato "vettore topologico" e talvolta indicato anche come `TOPOL`. L'uso congiunto di `IA` e `JA` consente di individuare qualsiasi elemento non nullo a_{ij} memorizzato in `SYSMAT`. Infatti, l'elemento a_{ij} si troverà in una posizione k del vettore `SYSMAT` compresa nell'intervallo $\texttt{IA}(i) \leq k \leq \texttt{IA}(i+1) - 1$ e tale per cui $\texttt{JA}(k) = j$. Queste due condizioni permettono di individuare univocamente k per cui $\texttt{SYSMAT}(k) = a_{ij}$. Si noti che l'occupazione di memoria si riduce da n^2 numeri reali (generalmente in doppia precisione) a nt numeri reali (tipicamente $nt < 0.05n^2$) e $nt + n + 1$ numeri interi.

Facciamo un esempio numerico per rendere più chiaro il concetto. Si consideri la seguente matrice A di dimensione 6×6, sparsa e non simmetrica:

$$A = \begin{bmatrix} 4 & 0 & 7 & 0 & 0 & 0 \\ 3 & 4 & 0 & 2 & 0 & 0 \\ 0 & 2 & 4 & 0 & 4 & 0 \\ 0 & 0 & 7 & 4 & 0 & 1 \\ 1 & 0 & 0 & 3 & 4 & 0 \\ 1 & 1 & 0 & 0 & 3 & 4 \end{bmatrix}$$

Gli elementi non nulli sono $nt = 18$, mentre $n^2 = 36$, pertanto il grado di sparsità di A è il 50%. Trascurando i coefficienti nulli, la matrice A diventa:

$$A' = \begin{bmatrix} 4 & & 7 & & & \\ 3 & 4 & & 2 & & \\ & 2 & 4 & & 4 & \\ & & 7 & 4 & & 1 \\ 1 & & & 3 & 4 & \\ 1 & 1 & & & 3 & 4 \end{bmatrix}$$

Il vettore `SYSMAT` ha dimensione 18 e risulta:

$$\overbrace{\begin{array}{cccccccccccccccccc} 4 & 7 & 3 & 4 & 2 & 2 & 4 & 4 & 7 & 4 & 1 & 1 & 3 & 4 & 1 & 1 & 3 & 4 \end{array}}^{nt}$$

Il vettore `JA` ha la stessa dimensione di `SYSMAT` e per ogni coefficiente di quest'ultimo individua l'indice di colonna:

$$\overbrace{\begin{array}{cccccccccccccccccc} 1 & 3 & 1 & 2 & 4 & 2 & 3 & 5 & 3 & 4 & 6 & 1 & 4 & 5 & 1 & 2 & 5 & 6 \end{array}}^{nt}$$

Il vettore `IA`, infine, individua la posizione in `SYSMAT` del primo coefficiente di ciascuna riga:

$$\overbrace{\begin{array}{ccccccc} 1 & 3 & 6 & 9 & 12 & 15 & 19 \end{array}}^{n+1}$$

Si noti che che le componenti di `IA` sono necessariamente ordinate in senso strettamente crescente e che $\texttt{IA}(n+1) = nt+1$. Ciò si deduce immediatamente dal fatto che per individuare un elemento a_{ij} dell'ultima riga di A si deve cercare l'indice k della componente in `SYSMAT` nell'intervallo $\texttt{IA}(n) \leq k \leq \texttt{IA}(n+1) - 1$.

2.2 Caso simmetrico

Nel caso in cui la matrice A sia simmetrica, come accade ad esempio nella soluzione agli elementi finiti dell'equazione differenziale ellittica di Laplace, il risparmio computazionale derivato dal sistema CRS viene incrementato memorizzando la sola parte triangolare alta, inclusa la diagonale

principale, di A. In altri termini, vengono memorizzati solo i coefficienti $a_{ij} \neq 0$ con $j \geq i$ e si sfrutta la proprietà di A secondo cui $a_{ij} = a_{ji}$.

La memorizzazione di A viene effettuata sempre mediante i tre vettori `SYSMAT`, `JA` e `IA` definiti nel paragrafo precedente, in cui tuttavia nt rappresenta il numero di coefficienti non nulli della triangolare alta e `IA` contiene le posizioni in cui si trova in `SYSMAT` l'elemento diagonale di ciascuna riga di A.

Anche in questo caso facciamo un esempio numerico per rendere più chiaro il concetto. Si consideri la seguente matrice A di dimensione 7×7, sparsa e simmetrica:

$$A = \begin{bmatrix} 10 & 0 & 0 & 0 & 1 & 0 & 2 \\ 0 & 1 & 0 & 3 & 0 & -1 & 0 \\ 0 & 0 & 4 & 0 & 0 & 0 & 1 \\ 0 & 3 & 0 & 2 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 5 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 1 & 0 \\ 2 & 0 & 1 & 0 & 0 & 0 & 20 \end{bmatrix}$$

Gli elementi non nulli sono 19, mentre se fosse piena ne avremmo 49. Tuttavia i coefficienti da memorizzare sono solamente quelli relativi alla parte superiore di A, di seguito evidenziati:

$$A' = \begin{bmatrix} \boxed{10} & & & & \boxed{1} & & \boxed{2} \\ & \boxed{1} & & \boxed{3} & & \boxed{-1} & \\ & & \boxed{4} & & & & \boxed{1} \\ & 3 & & \boxed{2} & \boxed{1} & & \\ 1 & & & 1 & \boxed{5} & & \\ & -1 & & & & \boxed{1} & \\ 2 & & 1 & & & & \boxed{20} \end{bmatrix}$$

Il vettore `SYSMAT` avrà perciò dimensione $nt = 13$ (7 elementi diagonali più 6 extra-diagonali) anziché 19, con un risparmio di memoria superiore al 30%, e sarà dato da:

$$\overbrace{\begin{array}{cccccccccccc} 10 & 1 & 2 & 1 & 3 & -1 & 4 & 1 & 2 & 1 & 5 & 1 & 20 \end{array}}^{nt}$$

Il vettore `JA` con gli indici di colonna corrispondenti ha la stessa dimensione di `SYSMAT` e risulta:

$$\overbrace{\begin{array}{cccccccccccc} 1 & 5 & 7 & 2 & 4 & 6 & 3 & 7 & 4 & 5 & 5 & 6 & 7 \end{array}}^{nt}$$

Infine, il vettore `IA` possiede sempre dimensione $n + 1$ ed individua la posizione in `SYSMAT` degli elementi diagonali:

$$\overbrace{\begin{array}{cccccccc} 1 & 4 & 7 & 9 & 11 & 12 & 13 & 14 \end{array}}^{n+1}$$

Ad esempio, IA(3)=7 significa che a_{33} è memorizzato in SYSMAT(7), come confermato anche dal fatto che JA(7)=3.

Come nel caso di matrici non simmetriche, si pone $\texttt{IA}(n+1) = nt + 1$. Si noti anche che dovrà sempre essere IA(1)=1 e $\texttt{IA}(n) = nt$, nonché JA(1)=1 e $\texttt{JA}(nt) = n$.

3 Implementazione del prodotto matrice-vettore

Lo schema del GCM necessita dell'operazione del prodotto matrice-vettore. L'implementazione di tale operazione al calcolatore risulta del tutto banale nel caso di memorizzazione tabellare della matrice, mentre è necessaria qualche attenzione qualora si utilizzi il sistema di rappresentazione CRS. Nel seguito distingueremo fra l'implementazione del prodotto matrice-vettore per matrici non simmetriche, più intuitivo, e per matrici simmetriche in cui si memorizza la sola triangolare alta.

3.1 Caso non simmetrico

Si vuole calcolare il prodotto matrice-vettore:

$$A\boldsymbol{v} = \boldsymbol{w} \tag{17}$$

con A matrice quadrata non simmetrica di ordine n, $\boldsymbol{v}$ e $\boldsymbol{w}$ vettori in $\Re^n$. È ben noto che la componente i-esima di $\boldsymbol{w}$ è pari alla somma dei prodotti degli elementi della riga i di A per i corrispondenti elementi di $\boldsymbol{v}$:

$$w_i = \sum_{j=1}^{n} a_{ij} v_j \tag{18}$$

Se la matrice è memorizzata in modo compatto, gli elementi a_{ij} vanno opportunamente ricercati in SYSMAT mediante l'uso di IA, mentre gli indici j relativi alle colonne si trovano nel vettore intero JA. In particolare, gli elementi di A appartenenti alla riga i sono memorizzati in corrispondenza agli indici k del vettore SYSMAT compresi, per definizione di IA, nell'intervallo $\texttt{IA}(i) \leq k \leq \texttt{IA}(i+1) - 1$. Gli indici di colonna j, di conseguenza, sono memorizzati in $\texttt{JA}(k)$.

Il prodotto matrice-vettore con A non simmetrica e memorizzata in forma compatta può quindi essere calcolato implementando il seguente algoritmo:

001 **Per** $i = 1, n$
002 azzero w_i
003 **Per** $k = \texttt{IA}(i)$, $\texttt{IA}(i+1) - 1$
004 $j :=\texttt{JA}(k)$
005 $w_i := w_i + \texttt{SYSMAT}(k) \cdot v_j$

006 **Fine Per**

007 **Fine Per**

Si noti che è sempre utile azzerare il vettore soluzione prima di procedere all'implementazione del ciclo di sommatoria (riga 2), al fine di evitare l'uso improprio di valori precedentemente contenuti in $\boldsymbol{w}$.

3.2 Caso simmetrico

Si vuole ora calcolare il prodotto (17) in cui A è matrice simmetrica di cui si memorizza la sola triangolare alta. Poiché gli elementi a_{ij} con $j < i$ non sono ora immediatamente disponibili nella sequenza di coefficienti della riga i, conviene scrivere la definizione (18) come:

$$w_i = \sum_{j=1}^{i-1} a_{ji} v_j + \sum_{j=i}^{n} a_{ij} v_j \tag{19}$$

avendo sfruttato la condizione per cui $a_{ij} = a_{ji}$.

Dalla (19) si deduce che il contributo a w_i relativo agli elementi con $j \geq i$ può essere implementato in maniera del tutto analoga a quanto fatto nel paragrafo precedente, mentre il contributo relativo agli elementi con $j < i$ può essere determinato selezionando in `SYSMAT` le componenti k per cui $\texttt{JA}(k) = i$, cioè appartenenti alla colonna i-esima. È del tutto evidente, tuttavia, che questo modo di procedere, seppure intuitivo, risulta inapplicabile da un punto di vista pratico. Sarebbe, infatti, necessario procedere ad una ricerca su nt componenti per n volte, con un dispendio che renderebbe inutile il vantaggio fornito dalla memorizzazione compatta della matrice e, in certi casi, dalla convergenza accelerata del GCM.

Conviene osservare che gli elementi della triangolare alta della riga i, oltre a contribuire al calcolo di w_i, entrano in gioco, in virtù della simmetria di A, anche nel calcolo di w_j, con j pari all'indice di colonna dell'elemento considerato. All'interno del medesimo ciclo sulle righe i si aggiornerà pertanto non solo w_i ma anche tutti i w_j corrispondenti. L'algoritmo descritto nel precedente paragrafo viene quindi modificato nel modo seguente:

001 **Per** $i = 1, n$

002 azzero w_i

003 **Fine Per**

004 **Per** $i = 1, n$

005 $k := \texttt{IA}(i)$

006 $w_i := w_i + \texttt{SYSMAT}(k) \cdot v_i$

007 **Per** $k = \texttt{IA}(i) + 1$, $\texttt{IA}(i+1) - 1$

008 $j := \texttt{JA}(k)$

009 $w_i := w_i + \texttt{SYSMAT}(k) \cdot v_j$

010 $w_j := w_j + \texttt{SYSMAT}(k) \cdot v_i$

011 **Fine Per**

012 **Fine Per**

Si noti che, a differenza dell'algoritmo utilizzato per matrici non simmetriche, in questo caso l'azzeramento del vettore prodotto $\boldsymbol{w}$ va fatto con un ulteriore ciclo (righe 1-3) esterno a quello di calcolo di $\boldsymbol{w}$. Inoltre, il contributo a w_i dato dall'elemento diagonale a_{ii} viene conteggiato a parte (riga 6) per evitare di considerarlo due volte nel ciclo successivo (corrisponde infatti al caso in cui $i = j$).

4 Calcolo del precondizionatore

Una delle chiavi del successo del GCM nella soluzione efficiente di sistemi lineari sparsi, simmetrici e definiti positivi sta nella possibilità di ottenere formidabili accelerazioni della convergenza mediante l'uso di opportune matrici di precondizionamento. La scelta di K^{-1} deve soddisfare alle seguenti caratteristiche:

- deve essere tale che il prodotto AK^{-1} abbia lo spettro, cioè l'insieme degli autovalori, raggruppato attorno all'unità e comunque un numero di condizionamento spettrale inferiore a quello di A;
- il calcolo deve essere semplice e poco costoso per non appesantire lo schema;
- l'occupazione di memoria deve essere inferiore o al più paragonabile a quella della matrice del sistema allo scopo di non vanificare lo sforzo computazionale effettuato per la memorizzazione compatta.

Spesso le suddette caratteristiche confliggono fra loro e necessariamente la scelta di K^{-1} diventa il risultato di un compromesso. Per paradosso, la matrice che soddisfa completamente la prima richiesta è ovviamente A^{-1}, il cui costo ed occupazione di memoria (si ricordi che l'inversa di una matrice sparsa è generalmente una matrice piena) tuttavia rendono del tutto incalcolabile.

Una discreta accelerazione del metodo del GC è ottenuta scegliendo:

$$K^{-1} = D^{-1} \tag{20}$$

dove D è la matrice diagonale contenente gli elementi diagonali di A. In questo caso, il calcolo della matrice di precondizionamento e la sua applicazione nello schema del GCM risultano banali

e vengono lasciati per esercizio al lettore. Il raggruppamento degli autovalori attorno all'unità di AD^{-1}, tuttavia, può essere limitato, specialmente se A presenta coefficienti extra-diagonali grandi rispetto agli elementi diagonali.

Risultati assai più significativi sono invece ottenuti assumendo:

$$K^{-1} = \left(\tilde{L}\tilde{L}^T\right)^{-1} \tag{21}$$

dove $\tilde{L}$, matrice triangolare bassa, è calcolata mediante la fattorizzazione incompleta di A, cioè la decomposta di Cholesky a cui viene assegnato il medesimo schema di sparsità di A. L'uso di questa matrice di precondizionamento, proposto negli anni '70 da Kershaw, si rivela generalmente un ottimo compromesso fra le contrapposte esigenze precedentemente discusse. Il calcolo di K^{-1} secondo la (21) e la sua applicazione nell'algoritmo del GCM verranno esaminati nei due paragrafi seguenti.

4.1 Fattore incompleto di Cholesky secondo Kershaw

Il calcolo del fattore incompleto di Cholesky si basa sulla fattorizzazione triangolare di matrici simmetriche che viene di seguito brevemente richiamata. Poichè con la memorizzazione compatta di A vengono conservati i soli elementi appartenenti alla triangolare alta, conviene riscrivere la (21) come:

$$K^{-1} = \left(\tilde{U}^T\tilde{U}\right)^{-1} \tag{22}$$

dove $\tilde{U} = \tilde{L}^T$.

Supponiamo che A sia una matrice 3×3 piena di cui svolgiamo per esteso la fattorizzazione:

$$\begin{bmatrix} u_{11} & 0 & 0 \\ u_{12} & u_{22} & 0 \\ u_{13} & u_{23} & u_{33} \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{12} & a_{22} & a_{23} \\ a_{13} & a_{23} & a_{33} \end{bmatrix} \tag{23}$$

Si noti che nella (23) si è sfruttata la simmetria di A ed il fatto che i soli elementi a_{ij} memorizzati sono quelli per cui $j \geq i$.

Sviluppiamo il prodotto (23) procedendo secondo la successione delle righe di A. Per la prima riga vale:

$$\begin{aligned} a_{11} = u_{11}^2 \quad &\Rightarrow \quad u_{11} = \sqrt{a_{11}} \\ a_{12} = u_{11}u_{12} \quad &\Rightarrow \quad u_{12} = \frac{a_{12}}{u_{11}} \\ a_{13} = u_{11}u_{13} \quad &\Rightarrow \quad u_{13} = \frac{a_{13}}{u_{11}} \end{aligned} \tag{24}$$

Si può osservare che i coefficienti sottodiagonali di A vengono di volta in volta già utilizzati nel calcolo delle righe superiori a quella corrente. Si procede, pertanto, con le righe 2 e 3 considerando

i soli termini relativi alla triangolare alta di A:

$$\begin{aligned} a_{22} = u_{12}^2 + u_{22}^2 \quad &\Rightarrow \quad u_{22} = \sqrt{a_{22} - u_{12}^2} \\ a_{23} = u_{12}u_{13} + u_{22}u_{23} \quad &\Rightarrow \quad u_{23} = \frac{1}{u_{22}} \left(a_{23} - u_{12}u_{13}\right) \\ a_{33} = u_{13}^2 + u_{23}^2 + u_{33}^2 \quad &\Rightarrow \quad u_{33} = \sqrt{a_{33} - u_{13}^2 - u_{23}^2} \end{aligned} \tag{25}$$

Dalle (24) e (25) si può facilmente generalizzare l'algoritmo di calcolo del fattore completo di Cholesky per una matrice di ordine n:

001 $u_{11} := \sqrt{a_{11}}$
002 **Per** $j = 2, n$
003 $\quad u_{1j} := a_{1j}/u_{11}$
004 **Fine Per**
005 **Per** $i = 2, n$
006 $\quad u_{ii} := \sqrt{a_{ii} - \sum_{l=1}^{i-1} u_{li}^2}$
007 $\quad$ **Per** $j = i + 1, n$
008 $\quad\quad u_{ij} := \left(a_{ij} - \sum_{l=1}^{i-1} u_{li}\, u_{lj}\right) / u_{ii}$
009 $\quad$ **Fine Per**
010 **Fine Per**

Il passaggio concettuale dal fattore completo di Cholesky a quello incompleto secondo Kershaw risulta a questo punto banale, in quanto all'algoritmo precedente è sufficiente aggiungere l'assegnazione:

$$a_{ij} = 0 \Rightarrow u_{ij} := 0 \tag{26}$$

Poiché A è memorizzata in modo compatto, e così anche $\tilde{U}$, si deduce immediatamente che, in virtù dell'assegnazione (26), i vettori `IA` e `JA` descrivono anche la topologia di $\tilde{U}$ per la quale sarà sufficiente adottare un vettore `PREC` contenente i coefficienti non nulli memorizzati sequenzialmente per righe.

Il nuovo algoritmo di calcolo del fattore incompleto di Cholesky secondo Kershaw con la memorizzazione in formato CRS risulta pertanto:

001 `PREC`(1) := $\sqrt{$`SYSMAT`(1)$}$
002 **Per** $k =$ `IA`(1)+1, `IA`(2)-1
003 $\quad$ `PREC`(k) := `SYSMAT`(k)/`PREC`(1)

004 **Fine Per**

005 **Per** $i = 2, n$

006 $k := \texttt{IA}(i)$

007 $l :=$ ogni $\overline{k} < \texttt{IA}(i)$ per cui $\texttt{JA}(\overline{k}) = i$

008 $\texttt{PREC}(k) := \sqrt{\texttt{SYSMAT}(k) - \sum_l \texttt{PREC}(l)^2}$

009 **Per** $k1 = \texttt{IA}(i) + 1, \texttt{IA}(i+1) - 1$

010 $j := \texttt{JA}(k1)$

011 ogni $l1 < \texttt{IA}(i)$ per cui $\texttt{JA}(\overline{k}) = i$ e $\texttt{IA}(l) + 1 < l1 < \texttt{IA}(l+1) - 1$

012 ogni $l2 < \texttt{IA}(i)$ per cui $\texttt{JA}(\overline{k}) = j$ e $\texttt{IA}(l) + 1 < l2 < \texttt{IA}(l+1) - 1$

013 $\texttt{PREC}(k1) := \left(\texttt{SYSMAT}(k1) - \sum_{l1,l2} \texttt{PREC}(l1) \cdot \texttt{PREC}(l2)\right) / \texttt{PREC}(k)$

014 **Fine Per**

015 **Fine Per**

Va osservato, tuttavia, che l'implementazione efficiente delle sommatorie in riga 8 e 13 sugli indici l, $l1$ ed $l2$ è in realtà non banale e va preferibilmente effettuata calcolando separatamente il contributo del coefficiente relativo al k corrente in modo da evitare le dispendiose ricerche previste in riga 7, 11 e 12. Tale implementazione è effettuata nella subroutine `KERSH` a disposizione dello studente.

Si deve infine ricordare che l'uso generalizzato delle (24) e (25) comporta l'estrazione di radici quadrate il cui argomento, nel caso di una fattorizzazione incompleta, non è più garantito essere positivo. In questo caso l'elemento diagonale in questione verrà posto pari ad un numero positivo arbitrario (ad esempio, l'ultimo coefficiente diagonale di $\tilde{U}$ non nullo). Si dimostra comunque teoricamente che se la matrice A è di tipo M, cioè diagonalmente dominante con elementi extra-diagonali di segno opposto rispetto a quelli diagonali, come ad esempio si verifica nella discretizzazione agli elementi finiti dell'equazione ellittica di Laplace, tutte le radici da calcolare nel fattore incompleto esistono nel campo reale.

4.2 Applicazione della decomposta incompleta

Mediante l'algoritmo sviluppato nel precedente paragrafo non si calcola esplicitamente la matrice di precondizionamento K^{-1} ma solamente il fattore incompleto $\tilde{U}$. È quindi necessario sviluppare un algoritmo ad hoc che permetta di calcolare il generico prodotto $K^{-1}\boldsymbol{v}$ senza generare esplicitamente K^{-1}.

Sia $\boldsymbol{w}$ il vettore in $\Re^n$ risultato del prodotto $K^{-1}\boldsymbol{v}$. Per definizione di K^{-1} si ha:

$$\boldsymbol{w} = \left(\tilde{U}^T \tilde{U}\right)^{-1} \boldsymbol{v} \tag{27}$$

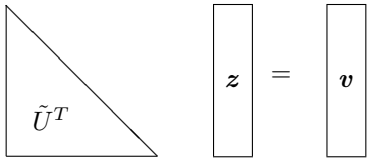

FIGURA 1: *Schema della soluzione del sistema con sostituzioni in avanti.*

cioè, premoltiplicando per K ambo i membri:

$$\left(\tilde{U}^T\tilde{U}\right)\boldsymbol{w} = \boldsymbol{v} \tag{28}$$

Il calcolo di $\boldsymbol{w}$ viene quindi ricondotto alla soluzione di un sistema lineare la cui matrice è K. Poiché K è fattorizzabile nel prodotto di due matrici triangolari, il sistema (28) è risolvibile tramite sostituzioni in avanti e all'indietro. Posto:

$$\tilde{U}\boldsymbol{w} = \boldsymbol{z} \tag{29}$$

la (28) diventa:

$$\tilde{U}^T\boldsymbol{z} = \boldsymbol{v} \tag{30}$$

che si può facilmente risolvere con sostituzioni in avanti (Figura 1). Iniziando dalla prima componente, ricavata in modo immediato come:

$$z_1 = \frac{v_1}{u_{11}} \tag{31}$$

si ottiene con semplici calcoli la formula ricorrente:

$$z_i = \frac{1}{u_{ii}}\left(v_i - \sum_{j=1}^{i-1} u_{ji}z_j\right) \qquad i = 2, \ldots, n \tag{32}$$

Come osservato per il prodotto matrice-vettore e per la determinazione del fattore incompleto di Cholesky, la sommatoria contenuta in (32) non è banale da implementare in modo efficiente memorizzando le matrici secondo il sistema CRS. A tal proposito, risulta conveniente definire un vettore di accumulazione $\boldsymbol{s}$ nelle cui componenti viene aggiornato il prodotto contenuto nella sommatoria dell'equazione (32) procedendo per righe di $\tilde{U}$. L'algoritmo per il calcolo del vettore $\boldsymbol{z}$ può pertanto essere scritto come:

001 **Per** $j = 1, n$
002 azzero s_j
003 **Fine Per**

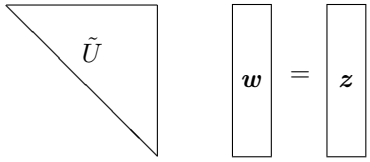

FIGURA 2: *Schema della soluzione del sistema con sostituzioni all'indietro.*

004 $z_1 := v_1/\texttt{PREC}(1)$
005 **Per** $i = 2, n$
006 $k := \texttt{IA}(i)$
006 **Per** $k1 = \texttt{IA}(i-1)+1,\ \texttt{IA}(i)-1$
007 $j := \texttt{JA}(k1)$
008 $s_j := s_j + \texttt{PREC}(k1) \cdot z_{i-1}$
009 **Fine Per**
010 $z_i := (v_i - s_i)\,/\texttt{PREC}(k)$
011 **Fine Per**

Ottenuto il vettore $\boldsymbol{z}$ si può infine calcolare $\boldsymbol{w}$ risolvendo il sistema (29) tramite sostituzioni all'indietro (Figura 2). La formula ricorrente si ricava in modo del tutto analogo a quanto fatto nelle equazioni (31) e (32) partendo in questo caso dalla componente n-esima:

$$w_n = \frac{z_n}{u_{nn}} \tag{33}$$

$$w_i = \frac{1}{u_{ii}} \left(z_i - \sum_{j=i+1}^{n} u_{ij} w_j \right) \qquad i = n-1, \ldots, 1 \tag{34}$$

L'implementazione della (2) risulta stavolta molto semplice anche memorizzando la matrice $\tilde{U}$ in forma compatta. Sempre utilizzando il vettore di accumulo $\boldsymbol{s}$, l'algoritmo corrispondente può essere scritto nel modo seguente:

001 $w_n := z_n/\texttt{PREC}(nt)$
002 **Per** $i = n-1, 1$ con passo -1
003 azzero s_i
004 $k := \texttt{IA}(i)$
005 **Per** $k1 = \texttt{IA}(i)+1,\ \texttt{IA}(i+1)-1$
006 $j := \texttt{JA}(k1)$

007 $s_i := s_i + \texttt{PREC}(k1) \cdot w_j$

008 **Fine Per**

009 $w_i := (z_i - s_i) / \texttt{PREC}(k)$

010 **Fine Per**

Per un confronto, viene messa a disposizione dello studente la subroutine `LSOLVE` che implementa gli algoritmi sopraripportati.