Calcolo del precondizionatore K^{-1}

- $A\mathbf{x} = \mathbf{b}$ sistema lineare sparso, simmetrico, definito positivo.
- Soluzione del sistema utilizzando il GCM con un opportuno precondizionatore K^{-1}.
- K^{-1} deve essere tale che AK^{-1} abbia gli autovalori raggruppati attorno all'unità; il calcolo di K^{-1} deve essere semplice e poco costoso; deve occupare poca memoria (inferiore o paragonabile alla matrice del sistema).

Precondizionatore diagonale

$$K^{-1} = D^{-1}$$

D è la matrice diagonale contenente gli elementi della diagonale principale di A.

➥ Semplice il calcolo di K^{-1} e la sua applicazione nel GCM.

Precondizionatore $K^{-1} = (\tilde{L}\tilde{L}^T)^{-1}$

Indichiamo con

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

il precondizionatore in cui $\tilde{L}$ è una matrice triangolare bassa calcolata mediante la fattorizzazione incompleta di A.

➥ Fattorizzazione incompleta di A: Si applica la decomposta di Cholesky a cui viene assegnato lo stesso schema di sparsità di A, (perciò incompleta).

Questa fattorizzazione fu proposta da Kershaw negli anni '70 ed è un buon compromesso fra i contrapposti requisiti richiesti per un precondizionatore.

Calcolo del fattore incompleto di Cholesky

Teniamo presente che stiamo lavorando con una matrice A simmetrica e che dobbiamo memorizzare le matrici in forma compatta, conservando quindi gli elementi che appartengono alla parte triangolare alta.

Conviene perciò lavorare non su $\tilde{L}$ che è triangolare bassa ma su $\tilde{U} = \tilde{L}^T$ che è triangolare alta.

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

Esempio

A matrice 3×3 simmetrica e piena (facciamo ora la decomposta completa di Cholesky):

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

Prima riga di U

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

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

Seconda e terza riga di U

$$a_{22} = u_{12}^2 + u_{22}^2 \quad \Rightarrow \quad u_{22} = \sqrt{a_{22} - u_{12}^2}$$

$$a_{23} = u_{12}u_{13} + u_{22}u_{23} \quad \Rightarrow \quad u_{23} = \frac{1}{u_{22}} \left(a_{23} - u_{12}u_{13}\right)$$

$$a_{33} = u_{13}^2 + u_{23}^2 + u_{33}^2 \quad \Rightarrow \quad u_{33} = \sqrt{a_{33} - u_{13}^2 - u_{23}^2}$$

Algoritmo: calcolo fattore completo di Cholesky

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**

Passaggio al fattore incompleto di Cholesky

Basta aggiungere all'algoritmo precedente l'assegnazione:

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

➥ A è memorizzata in modo compatto. Anche $\tilde{U}$ deve essere memorizzata in modo compatto. Ma la parte alta di A e $\tilde{U}$ presentano lo stesso schema di sparsità e di A è memorizzata solo la parte alta. Quindi i vettori JA e IA descrivono non solo la topologia di A ma anche quella di $\tilde{U}$.

Ci serve allora costruire solo un vettore, che chiamiamo PREC, in cui andare a memorizzare i coefficienti non nulli di $\tilde{U}$ presi in successione riga per riga.

Algoritmo: fattore incompleto di Cholesky

001 $\mathtt{PREC}(1) := \sqrt{\mathtt{SYSMAT}(1)}$

002 **Per** $k = \mathtt{IA}(1)+1, \mathtt{IA}(2)-1$

003 $\quad \mathtt{PREC}(k) := \mathtt{SYSMAT}(k)/\mathtt{PREC}(1)$

004 **Fine Per**

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

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

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

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

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

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

011 $\quad\quad l1 :=$ ogni $\overline{k} < \mathtt{IA}(i)$ per cui $\mathtt{JA}(\overline{k}) = i$

012 $\quad\quad l2 :=$ ogni $\overline{k} < \mathtt{IA}(i)$ per cui $\mathtt{JA}(\overline{k}) = j$

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

014 $\quad$ **Fine Per**

015 **Fine Per**

Osservazioni:

- le sommatorie sugli indici l, $l1$ e $l2$ non sono banali da implementarsi (la subroutine `kersh.f` è messa a disposizione in rete per facilitare il lavoro della programmazione).
- Annullando in $\tilde{U}$ gli elementi che corrispondono ad elementi nulli di A e andando avanti nell'implementazione dell'algoritmo della decomposta incompleta si possono ottenere elementi diagonali uguali a radici quadrate di elementi negativi! In tal caso, quell'elemento diagonale viene posto uguale ad un numero positivo arbitrario (ad esempio, l'ultimo coefficiente diagonale di $\tilde{U}$ non nullo).

  Teoricamente si prova che se gli elementi extra-diagonali di A sono tutti negativi, allora tutti gli elementi diagonali del fattore incompleto di Cholesky sono reali.

Applicazione della decomposta incompleta

Noi calcoliamo solo $\tilde{U}$ e non tutto il precondizionatore $K^{-1} = \left(\tilde{U}^T \tilde{U}\right)^{-1}$.

Come calcolare $K^{-1}\mathbf{v}$ senza generare esplicitamente K^{-1}?

Sia $K^{-1}\mathbf{v} = \mathbf{w}$, cioè $\left(\tilde{U}^T \tilde{U}\right)^{-1} \mathbf{v} = \mathbf{w}$.

$$\begin{cases} \left(\tilde{U}^T \tilde{U}\right)^{-1} \underset{noto}{\mathbf{v}} = \underset{incognita}{\mathbf{w}} \\ \underset{noto}{\mathbf{v}} = \left(\tilde{U}^T \tilde{U}\right) \underset{incognita}{\mathbf{w}} = \tilde{U}^T \underset{incognita}{\mathbf{z}} \\ \underset{noto}{\mathbf{v}} = \tilde{U}^T \underset{incognita}{\mathbf{z}} \end{cases}$$

$$\begin{cases} \tilde{U}^T \underset{incognita}{\mathbf{z}} = \underset{noto}{\mathbf{v}} & \text{con sostituzione in avanti ricavo } \mathbf{z} \\ \tilde{U} \underset{incognita}{\mathbf{w}} = \underset{noto}{\mathbf{z}} & \text{mediante sostituzione all'indietro ricavo } \mathbf{w} \end{cases}$$

Difatti:

$$\tilde{U}^T \mathbf{z} = \mathbf{v}$$

in forma estesa diventa

$$\begin{cases} u_{11} z_1 & = v_1 \\ u_{12} z_1 + u_{22} z_2 & = v_2 \\ u_{13} z_1 + u_{23} z_2 + u_{33} z_3 & = v_3 \\ \vdots & \vdots \\ u_{1n} z_1 + u_{2n} z_2 + \ldots u_{nn} z_n & = v_n \end{cases}$$

Dalla prima equazione ricaviamo

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

Dalla seconda equazione, esplicitando z_1 appena ricavato, troviamo z_2 e procediamo via via fino all'ultima equazione:

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

Una volta ottenuto $\mathbf{z}$ dobbiamo risolvere il sistema

$$\tilde{U}\mathbf{w} = \mathbf{z}$$

In forma estesa si ha:

$$\begin{aligned} u_{11}w_1 + u_{12}w_2 + \ldots + u_{1(n-1)}w_{n-1} + u_{1n}w_n &= z_1 \\ u_{22}w_2 + u_{23}w_3 + \ldots + u_{2n}w_n &= z_2 \\ \vdots \qquad\qquad & \qquad \vdots \\ u_{(n-1)(n-1)}w_{n-1} + u_{(n-1)n}w_n &= z_{n-1} \\ u_{nn}w_n &= z_n \end{aligned}$$

Partendo dall'ultima equazione si ha

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

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

Algoritmo per il calcolo di z

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

Algoritmo per il calcolo di w

001 $w_n := z_n / \mathsf{PREC}(nt)$

002 **Per** $i = n-1, 1$ con passo -1

003 azzero s_i

004 $k := \mathtt{IA}(i)$

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

006 $j := \mathsf{JA}(k1)$

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

008 **Fine Per**

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

010 **Fine Per**

Nota: per un confronto è messa a disposizione la subroutine `lsolve.f` che implementa i due algoritmi

Esempio (molto semplice)

$$\tilde{U} = \begin{pmatrix} 1 & 0 & 2 \\ 0 & 3 & 0 \\ 0 & 0 & 4 \end{pmatrix}$$

PREC	1	2	3	4
JA	1	3	2	3
IA	1	3	4	5

$\tilde{U}^T \mathbf{z} = \mathbf{v} \Rightarrow z_1 = v_1 \qquad z_2 = v_2/3 \qquad z_3 = (v_3 - 2z_1)/4$

Applico l'algoritmo per ricavare z_i:

- azzero il vettore s di dimensione $n = 3$ (righe 1-3);
- assegno a z_1 il valore v_1/ PREC$(1) = v_1$;
- i= 2
  - $k =$ IA$(2) = 3$;
  - per $k1 =$IA$(1) + 1 = 2$ fino a IA$(2) - 1 = 2$, quindi $k1 = 2$:
    - j= JA$(2) = 3$

- $s_3 = s_3 + \mathsf{PREC}(2) z_1 = 0 + 2z_1$

- $z_2 = (v_2 - s_2) / \mathsf{PREC}(3) = v_2/3$

- i = 3
  - $k = \mathtt{IA}(3) = 4$;
  - per $k1 = \mathtt{IA}(2) + 1 = 4$ fino a $\mathtt{IA}(3) - 1 = 3$, quindi $k1 = \emptyset$: si esce da questo ciclo;
  - $z_3 = (v_3 - s_3) / \mathsf{PREC}(4) = (v_3 - 2z_1)/4$

Calcoliamo $\tilde{U}\mathbf{w} = \mathbf{z} \Rightarrow w_3 = z_3/4 \qquad w_2 = z_2/3 \qquad w_1 = z_1 - 2w_3$

Applichiamo l'algoritmo:

- $w_3 = z_3/\mathsf{PREC}(4)$
- i =2
  - azzero s_2
  - $k = \mathtt{IA}(2) = 3$
  - per $k1 = \mathtt{IA}(2) + 1 = 4$ fino a $\mathtt{IA}(3) - 1 = 3$ quindi $k1 = \emptyset$: si esce da questo ciclo;
  - $w_2 = (z_2 - s_2)/\ \mathsf{PREC}(3) = z_2/3$
- i=1
  - azzero s_1
  - $k = \mathtt{IA}(1) = 1$
  - per $k1 = \mathtt{IA}(1) + 1 = 2$ fino a $\mathtt{IA}(2) - 1 = 2$ quindi $k1 = 2$

- $j = \mathsf{JA}(2) = 3$
- $s_1 = s_1 + \mathsf{PREC}(2) w_3 = 0 + 2w_3$

$\Rightarrow$ $w_1 = (z_1 - s_1)/\ \mathsf{PREC}(1) = z_1 - 2w_3$

Implementazione del CGM

Siamo ora in grado di poter costruire un codice che implementi il metodo del Gradiente Coniugato Modificato per risolvere il sistema lineare $A\mathrm{x} = \mathrm{b}$

- I dati di input sono la matrice A e il vettore b: la matrice A sia data direttamente in forma compatta tramite i vettori `SYSMAT, JA, IA`. Si fissi una tolleranza tol (per esempio $tol = 10^{-9}$ o 10^{-10}) che serve per il controllo della norma del residuo, e un numero massimo di iterazioni.
- Si applica lo schema delle Correzioni Residue (CR) in modo da ricavare il vettore iniziale x_0 per il CGM. Si applichi lo schema CR 0, 1, 2, 5, 10 e 20 volte.
- Una volta ricavato x_0 con le CR, si applica il metodo CGM.

Come precondizionatore si può utilizzare sia $K^{-1} = D^{-1}$ sia $K^{-1} = (\tilde{L}\tilde{L}^T)^{-1}$, confrontando quindi i due precondizionatori.

- Per ciascun caso test si fa quindi il profilo di convergenza mettendo a confronto sia i precondizionatori, sia il diverso numero di iterazioni usato nelle CR. Il profilo di convergenza deve essere fatto in scala semilogaritmica (iterazioni/norma del residuo relativo)
- le matrici test si trovano in rete
- **IL LISTATO DEL CODICE PER IL CGM, LE PROVE EFFETTUATE CON LE MATRICI TEST E I GRAFICI DI CONVERGENZA SONO DA PORTARE ALL'ESAME**

Sui files di dati in FORTRAN

Le matrici test sono messe in un unico file che contiene

- sulla prima riga N e NTERM
- poi il vettore SYSMAT
- il vettore JA
- il vettore IA

In FORTRAN si leggono i dati, per esempio così:

```
open(8, file='matrice_dati.dat')
read(8,*) n,nterm
read(8,*) (sysmat(i), i=1,nterm)
read(8,*) (ja(i), i=1,nterm)
read(8,*)(ia(i), i=1,n+1)
```

E in MATLAB?

Un modo può essere - a parte rispetto al programma sul GCM - spezzare il file di dati in 4 files separati, uno che contiene solo N e NTERM, uno solo per SYSMAT, uno per JA e uno per IA.

Per i files che contengono i tre vettori (SYSMAT, JA e IA) si può operare in questo modo sulla Command Window del MATLAB:

```
s=importdata('sysmat_test_xyz.dat');
s=s';
lunghezza=size(s,1)*size(s,2);
sysmat=reshape(s,lunghezza,1);
sysmat=sysmat(1:NTERM);
save sysmat_test_xyz.mat sysmat -ASCII -DOUBLE
```

In questo modo viene creato il file sysmat_test_xyz.mat che

contiene il vettore sysmat (può essere usato solo in MATLAB)

In questo modo, per ogni matrice test, ci salviamo i vettori che daremo in input al programma sul GCM.

Quando si scriverà il programma sul GCM si farà qualcosa di simile

```
load -ASCII sysmat_test_xyz.mat
sysmat=sysmat_test_xyx;
clear sysmat_test_xyz
```

Oppure

```
s=importdata('sysmat_test_xyz.dat');
s=s';
lunghezza=size(s,1)*size(s,2);
sysmat=reshape(s,lunghezza,1);
sysmat=sysmat(1:NTERM);
save sysmat_test_xyz.dat sysmat -ASCII
```

Sul programma sul GCM si farà qualcosa di simile (forse è meglio)

```
file=input('scrivi il nome del file di sysmat ', 's');
sysmat=load(file)
```