Università degli studi di Napoli “Federico II”

Facoltà di Scienze MM.FF.NN. - Informatica

Corso di Calcolo Scientifico

Elaborato 2

Sviluppo di un software per la risoluzione di un sistema

Ax=b, con A matrice quadrata di ordine n, simmetrica, e x e b

di dimensione n, mediante l'utilizzo della libreria LAPACK

Ferrara Francesco Saverio

566/811

prof.ssa L. D'Amore

A.A. 2004/2005

Scopo

Assegnata la matrice $A \in \Re^{nxn}$ e i vettori $x, b \in \Re^{n}$, con A matrice simmetrica, x vettore dei termini noti, e x vettore delle incognite, si vuole calcolare la risoluzione del sistema $Ax = b$.

Analisi del problema

Abbiamo a che fare con un sistema lineare $Ax = b$, nel quale la matrice A ha delle particolari caratteristiche; essa infatti è una matrice simmetrica di ordine n.
Per fattorizzare una matrice simmetrica possiamo anche evitare di eseguire la costosa fattorizzazione A=LU (dell'ordine di $\theta(n^3)$), ma possiamo usare una sua specializzazione per le metrici simmetriche chiamata fattorizzazione $A = LDL^T$.

<u>Descrizione della strategia utilizzata:</u>
Per risolvere questo problema non è stata utilizzata nessuna particolare strategia in quanto le libreria LAPACK mette a disposizione una vasta gamma di routine per risolvere questo problema.
Tenendo conto della sua struttura, per memorizzare la matrice si è scelto di utilizzare la memorizzazione “packed storage” (già prevista dalle routine di Lapack); essa ci fa risparmiare molto spazio in quanto permette di registrare una matrice triangolare n^2 in uno spazio pari a $\frac{n \cdot (n+1)}{2}$. Per illustrare meglio come funziona la memorizzazione packed storage, supponiamo di voler dare in pasto al programma la seguente matrice:

$$A = \begin{pmatrix} 3 & -2 & 12 & 4 \\ -2 & 9 & 3 & 6 \\ 12 & 3 & 5 & -1 \\ 4 & 6 & -1 & 8 \end{pmatrix}$$

All'interno del computer possiamo scegliere, ad esempio, di memorizzare solo il triangolo inferiore della matrice per colonne, in modo da avere una struttura di questo tipo:

$$AP = \begin{pmatrix} 3 & -2 & 12 & 4 & 9 & 3 & 6 & 5 & -1 & 8 \end{pmatrix}$$

dove:
- 3 -2 12 4 rappresentano la prima colonna del triangolo inferiore di A,
- 9 3 6 rappresentano la seconda colonna del triangolo inferiore di A,
- 5 -1 rappresentano la prima terza del triangolo inferiore di A,
- 8 rappresenta l'ultima colonna del triangolo inferiore di A.

Fatte queste considerazioni, l'utente dovrà intuire subito che il programma non si aspetta l'intera matrice A in input, ma solo il suo triangolo inferiore inserito colonna per colonna.

Descrizione dell'algoritmo:
In una prima fase l'algoritmo esamina gli argomenti passati per riga di comando, in quanto se viene inserita l'opzione '-h` oppure '--help` viene stampata a video una mini guida che aiuta l'utente a inserire un input corretto.
Se invece il programma viene chiamato senza opzioni, allora chiede all'utente di inserire la dimensione della matrice A (che sarà anche la dimensione dei vettori x e b), e in seguito legge dallo standard input il contenuto di queste strutture dati.
Dopo aver fatto questo vengono eseguite le seguenti istruzioni, la restante parte dell'algoritmo non ha nulla di complicato:

```
01 ipiv = (int *) calloc(n, sizeof(int)); // Alloca spazio per il vettore ipiv
02 if (ipiv == NULL) {
03   perror("calloc()");
04   return EXIT_FAILURE;
05 }
06
07 sspsv_(&uplo, &n, &nrhs, ap, ipiv, b, &n, &info); // Chiamo la funzione di lapack
08
09 if (info > 0) { // Non e' possibile calcolare la soluzione
10   printf("L'elemento %d della diagonle e' esattamente zero.\n", info);
11   return EXIT_FAILURE;
12 }
13 else
14   if (info < 0) { // Errore nel passaggio dei parametri alla funzione
15     printf("La funzione sspsv() del pacchetto lapack e' stata chiamata con parametri errati.\n");
16     return EXIT_FAILURE;
17   }
18   else // La funzione e' terminata correttamente
19     printf("Fine della computazione!\n");
```

Alla riga 1 viene allocato lo spazio per il vettore ipiv; quest'ultimo non è inizializzato perchè la routine sspsv_() tratta questo vettore come un parametro di solo output. Per assicurarci che la memoria sia stata allocata correttamente, alle righe 2-5 abbiamo un controllo del valore di ritorno della funzione calloc().
In seguito viene chiamata la subroutine sspsv() di Lapack (riga 7), e subito dopo (righe 9-19) viene testato il suo esito leggendo il valore della variabile “info”. In caso di errore, info diverso da 0, viene stampato un errore sullo standard error e il programma termina immediatamente ritornando un indicatore di errore (righe 11 e 16).

Indicatori di errore

Il programma ritorna un indicatore di errore in caso di passaggio di parametri non riconosciuti oppure di errore nell'allocazione della memoria.

Routine ausiliarie

Per il corretto funzionamento, il programma si avvale di diverse routine classificabili in routine interne, esterne, e appartenenti alla libreria LAPACK.

Routine interne:

```
void help();
```

Questa procedura stampa a video una semplice mini-guida per aiutare l'utente a inserire i dati in input.

Routine esterne:

Sono tutte definite nel file “util.h”.

```
float *spcle(int n);
```

Legge dallo standard input il triangolo inferiore di una matrice simmetrica memorizzando per colonna nella forma "packed storage".

```
float *svele(int n);
```

Legge un vettore di n elementi dallo standard input.

```
void svest(float *x, int n);
```

Stampa sullo standard output il vettore x di n elementi.

Routine esterne (LAPACK):

Sono le routine messe a disposizione dalla libreria LAPACK (Linear Algebra PACKage). In questo programma è bastato usare la driver routine sspsv() per risolvere il problema:

```
sspsv(char *uplo, int *n, int *nrhs, float *ap, int *ipiv, float *b, int *ldb, int *info);
```

- `uplo`: Specifica alla subroutine se il triangolo passato in input è quello superiore o inferiore;
- `n`: E' il numero delle equazioni lineari;
- `nrhs`: Numero di colonne della matrice b dei termini noti (nel nostro caso 1);
- `ap`: Matrice dei coefficienti memorizzata in formato packed storage;
- `ipiv`: Parametro di solo ouput, viene utilizzato dalla subroutine per facilitare il pivoting durante la fattorizzazione;
- `b`: Matrice dei termini noti (in output contiene le soluzioni del sistema);
- `ldb`: E' la leading dimension dell'array B;
- `info`: Valore ritornato dalla subroutine.

Complessità

L'algoritmo è stato pensato per fare calcoli con matrici di grandi dimensioni ottimizzando la complessità di spazio in quanto il pacchetto LAPACK prevede la memorizzazione “Packed Storage” per le matrici simmetriche.

<u>Complessità di tempo:</u>

Eseguendo la fattorizzazione $A = LDL^T$, per matrici simmetriche senza utilizzare la tecnica del pivoting abbiamo una complessità di tempo pari a $T(n) = O\left(\frac{n^3}{6}\right)$.

Purtroppo il solo fatto che la matriche è simmetrica non ci permette di evitare la tecnica del pivoting senza rendere instabile l'algoritmo. Le sole matrici simmetriche che non necessitano di questa tecnica per la fattorizzazione $A = LDL^T$ sono le matrici a diagonale (strettamente) dominante.

Nel nostro caso utilizziamo la tecnica del pivoting, la quale fa perdere la proprietà di simmetria della matrice e quindi la complessità di tempo è uguale a quella della fattorizzazione $A = LU$, cioè:

$$T(n) = O\left(\frac{2}{3}n^3\right)$$

<u>Complessità di spazio:</u>

L'algoritmo alloca spazio per:

- vettore ipiv di n elementi,
- vettore b di n elementi,
- matrice ap di $\frac{n \cdot (n+1)}{2}$ elementi,
- 4 variabili (uplo, info, nrhs, n).

Quindi: $S(n) = 2 \cdot n + \frac{n \cdot (n+1)}{2} + 4 = \frac{n^2}{2} + \frac{5}{2} \cdot n + 4$; asintoticamente si ha:

$$S(n) = O\left(\frac{n^2}{2}\right)$$

Accuratezza fornita

L'algoritmo utilizza numeri float (singola precisione), e restituisce il risultato esatto a meno ad errori di round-off.

Raccomandazioni di utilizzo

Pe far funzionare il programma, non bisogna passargli argomenti dalla linea di comando. Se invece si vuole che venga stampata a video una mini-guida all'uso del programma, basta passargli l'opzione '-h` o '--help`.

Riferimenti bibliografici

“A. Murli”
Lucidi del corso di Calcolo Scientifico
http://www.dma.unina.it/~murli/didattica/mat_didattico_cs0405.html
2004-2005

Esempio d'uso

Facciamo un esempio dando in pasto all'eseguibile la matrice

$$A = \begin{pmatrix} 3 & -2 & 12 & 4 \\ -2 & 9 & 3 & 6 \\ 12 & 3 & 5 & -1 \\ 4 & 6 & -1 & 8 \end{pmatrix} .$$

Appena avviato il programma ci chiede di inserire la dimensione della matrice:

```
l566811@infservlinux ~/calcolo/elaborato2 $ make run
./elaborato
Inserisci la dimensione della matrice A: 4
```

Rispondiamo con 4 e proseguiamo inserendo gli elementi della matrice:

```
Inserisci gli elementi del triangolo inferiore della matrice per colonne,
oppure del triangolo superiore per righe...
 1>> 3
 2>> -2
 3>> 12
 4>> 4
 5>> 9
 6>> 3
 7>> 6
 8>> 5
 9>> -1
10>> 8
```

Dopo aver inserito gli elementi della matrice, ci vengono chiesti quelli del vettore:

```
Inserisci il vettore dei termini noti...
 1>> 3
 2>> -2
 3>> 1
 4>> 4
```

Alla fine dei calcoli avremo il seguente output:

```
Fine della computazione!
Stampa del vettore delle soluzioni:
 [ 0]-> 0.358414  [ 1]-> -0.566735  [ 2]-> -0.175367  [ 3]-> 0.723923
```

Se invece diamo in pasto all'algoritmo una matrice singolare, come ad esempio $A = \begin{pmatrix} 2 & 4 \\ 4 & 8 \end{pmatrix}$, avremo il seguente ouput:

```
l566811@infservlinux ~/calcolo/elaborato2 $ make run
./elaborato
Inserisci la dimensione della matrice A: 2
Inserisci gli elementi del triangolo inferiore della matrice per colonne,
oppure del triangolo superiore per righe...
 1>> 2
 2>> 4
 3>> 8
Inserisci il vettore dei termini noti...
 1>> 3
 2>> 6
La matrice e' singolare in quanto l'elemento 2 della diagonle e' esattamente zero.
make: *** [run] Error 1
```

Codice sorgente

spcle.c :

```
#include "util.h"


float *spcle(int n) {

        float *ap; // Matrice che verra' restituita in output
        int dim = ( (n*(n+1))/2 ); // Dimensione della matrice in formato packed storage
        int i;

        ap = (float *) malloc(dim * sizeof(float)); // Alloco la memoria
        if (ap == NULL) {
                fprintf(stderr, "Errore nell'allocazione della memoria\n");
                return NULL;
        }


        fprintf(stdout, "Inserisci gli elementi del triangolo inferiore della matrice per
colonne,\n");
        fprintf(stdout, "oppure del triangolo superiore per righe...\n");

        for (i=0 ; i<dim ; i++) { // Leggo gli elementi
                fprintf(stdout, "%2d>> ", i+1);
                fscanf(stdin, "%f", &ap[i]);
        }

        return ap; // Ritorno l'indirizzo della matrice al chiamante
}
```

svest.c :

```
#include "util.h"


void svest(float *x, int n) {

        int i;


        fprintf(stdout, "Stampa del vettore delle soluzioni:\n");
        for (i=0 ; i<n ; i++) // Stampo la matrice
                fprintf(stdout, " [%2d]-> %f ", i, x[i]);
        fprintf(stdout, "\n");

}
```

svele.c :

```
#include "util.h"


float *svele(int n) {

        int i;
        float *x; // Vettore che verra' restituito in output


        x = (float *) calloc(n, sizeof(float)); // Alloco la memoria
        if (x==NULL) {
                fprintf(stderr, "Errore nell'allocazione della memoria\n");
                return NULL;
        }


        fprintf(stdout, "Inserisci il vettore dei termini noti...\n");
        for (i=0 ; i<n ; i++) { // Leggo gli elementi
                fprintf(stdout, "%2d>> ", i+1);
                fscanf(stdin, "%f", &x[i]);
        }

        return x; // Ritorno il vettore alla procedura chiamante
}
```

util.h :

```
#ifndef UTIL
#define UTIL



#include <stdio.h>
#include <stdlib.h>



float *spcle(int n);
        /*
        Legge dallo standard input il triangolo inferiore di una matrice simmetrica
        memorizzando per colonna nella forma "packed storage".
        */

float *svele(int n);
        /*
        Legge un vettore di n elementi dallo standard input.
        */

void svest(float *x, int n);
        /*
        Stampa sullo standard output il vettore x di n elementi.
        */


#endif
```

main.c :

```
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include "util.h"



// Dichiarazione della routine di lapack usata
void sspsv(char *uplo, int *n, int *nrhs, float *ap, int *ipiv, float *b, int *ldb, int
*info);



//Stampa di un help
void help() {
        fprintf(stdout,
"################################################################################\n");
        fprintf(stdout, "#                      LABORATORIO DI CALCOLO SCIENTIFICO
#\n");
        fprintf(stdout, "#                               A.A. 2004/2005
#\n");
        fprintf(stdout, "#
#\n");
        fprintf(stdout, "#                      Ferrara Francesco Saverio - 566/811
#\n");
        fprintf(stdout, "#                          fsterrar@studenti.unina.it
#\n");
        fprintf(stdout,
"################################################################################\n\n");
        fprintf(stdout, "Software per la risoluzione di un sistema Ax=b, con A matrice
quadrata di ordine n,\n");
        fprintf(stdout, "simmetrica, e x e b vettori di dimensione n.\n\n");
        fprintf(stdout, "Per la simmetria della matrice A, essa puo' essere memorizzata nella
forma packed\n");
        fprintf(stdout, "storage, quindi nell'input dei dati non e' necessario inserire tutti
gli elementi,\n");
        fprintf(stdout, "ma basta inserire gli elementi del triangolo inferiore della matrice
per colonne,\n");
        fprintf(stdout, "oppure del triangolo superiore per righe.\n\n");
        fprintf(stdout, "Chiamare il programma senza argomenti.\n");
        fprintf(stdout, "Per maggiorni informazioni leggere la documentazione allegata.\n");
}
```

```
int main(int argc, char *argv[]) {

        int n; // Dimensione della matrice A
        int nrhs = 1; // Abbiamo un unico vettore che rappresenta la soluzione
        int *ipiv; // Tiene traccia degli scambi sulla matrice durante le operazioni svolte
da sspsv_()
        int info; // Contiene il valore di terminazione di sspsv_()
        float *ap; // Matrice triangolare di n elementi memorizzata nel formato packed
storage
        float *b; // Vettore contenente i termini noti
        char uplo = 'l'; // Memorizziamo la matrice ap utilizzando solo il suo triangolo
inferiore
```