AA 2015-2016 - Metodi del Calcolo Scientifico - Progetto 1

Algebra lineare numerica - metodi diretti per matrici sparse
4 maggio 2016

Lo scopo di questo progetto è di studiare l'implementazione in ambienti di programmazione standard (Python, Java, C, C++, .NET, C#, ecc.) degli algoritmi di risoluzione diretta di sistemi lineari per matrici sparse.

Le matrici sparse sono matrici con un grandissimo numero di elementi uguali a zero. Spesso il numero di elementi diversi da zero su ogni riga è un numero piccolo (per esempio dell'ordine di 10^1) indipendente dalla dimensione della matrice, che può essere anche dell'ordine di 10^8. Molti problemi del calcolo scientifico si riconducono alla soluzione di uno o più sistemi lineari con matrice dei coefficienti sparsa.

Le matrici sparse si possono memorizzare in modo compatto, tenendo solo conto degli elementi diversi da zero; per esempio, è sufficiente per ogni elemento diverso da zero memorizzare solo la sua posizione (i, j) e il suo valore a_{ij}, e semplicemente ignorare gli elementi uguali a zero.

Il semplice metodo di eliminazione di Gauss (con pivot o senza pivot) applicato alle matrici sparse tende a generare nuovi elementi diversi da zero, causando il riempimento (*fill-in*) della matrice triangolare finale (oppure, equivalentemente, delle matrici L ed U).

Il *fill-in* rende il metodo di elimazione di Gauss inutilizzabile, perché saremmo costretti ad allocare lo spazio sufficiente per rappresentare l'*intera* matrice.

Ci sono tuttavia matrici sparse particolari per le quali l'algoritmo di Gauss non genera fill-in: per esempio le matrici *tridiagonali*, nelle quali gli elementi diversi da zero sono solo sulla diagonale principale e sulle due sottodiagonali.

L'idea per trattare matrici sparse generali è di fare una *permutazione preliminare di righe e colonne* in modo che l'algoritmo di Gauss generi il minor numero possibile di elementi diversi da zero.

Ci sono diverse tecniche per effettuare questa permutazione preliminare; la più efficace al momento sembra essere quella sviluppata da Tim Davis della Texas A&M University (libreria UMFPACK) che è implementata molto efficacemente in MATLAB. Per tutti i dettagli si veda la sua pagina web: `http://faculty.cse.tamu.edu/davis` oppure la pagina Wikipedia associata: https://en.wikipedia.org/wiki/UMFPACK

Tim Davis cura anche il sito `http://www.cise.ufl.edu/research/sparse/matrices/` contenente numerose matrici sparse provenienti da problemi reali che possono essere usate per testare le librerie.

Le matrici che dovrete considerare per la relazione sono quelle del gruppo `FEMLAB` (tranne la matrice `FEMLAB/waveguide3D` (l'ultima della serie) che è a coefficienti complessi).

Gli ambienti di programmazione che dovrete considerare sono:

- MATLAB;
- due a vostra scelta (Python, Java, C, C++, C#, .NET, ... (altro)).

In ciascuno dei tre ambienti di programmazione, utilizzate un solutore diretto per matrici sparse risolvendo il sistema lineare `Ax=b` dove `A` è una delle matrici del gruppo `FEMLAB` e il termine noto `b` è scelto in modo che la soluzione esatta sia il vettore `xe=[1 1 1 1 1 1 ...]` avente tutte le componenti uguali a 1: in altre parole, `b=A*xe`.

Per ognuna delle matrici occorre determinare:

- il tempo necessario per calcolare la soluzione `x`;
- l'errore relativo tra la soluzione calcolata `x` e la soluzione esatta `xe`, definito da:

$$\texttt{errore relativo} = \frac{\|\texttt{x} - \texttt{xe}\|}{\|\texttt{xe}\|}$$

  dove $\|\texttt{v}\|$ è la norma euclidea del vettore `v`;
- la memoria necessaria per risolvere il sistema, ovvero grosso modo l'aumento della dimensione del programma in memoria da subito dopo aver letto la matrice a dopo aver risolto il sistema.

L'implementazione di MATLAB dovrebbe essere la migliore.

Riportate in un grafico queste tre quantità mettendo in ascissa la dimensione della matrice e in ordinata tempo, errore e memoria per ognuno dei tre ambienti. Organizzate il codice in modo che sia poi semplice ripetere l'esperimento numerico con un altro set di matrici (sempre della collezione di Davis).

Riassunto

Riassumendo, la relazione dovrà contenere:

- Per ogni ambiente di programmazione (escludendo MATLAB) una breve descrizione della libreria usata mettendo in evidenza le sue caratteristiche e se è documentata e mantenuta;
- per ogni parametro (velocità, precisione e occupazione di memoria) riportare un grafico che permetta di confrontare direttamente il comportamento dei tre ambienti di programmazione rispetto alle matrici del gruppo `FEMLAB` come descritto sopra; Per quanto riguarda la precisione, probabilmente è meglio usare la scala logaritmica per i tempi altrimenti il grafico risulterebbe troppo schiacciato.
- Il listato del codice.

Al momento della discussione della relazione occorre anche portare il computer sul quale si sono eseguiti i test e, eventualmente, verrà richiesto di ripetere gli esperimenti su altre matrici, provenienti sempre dal set di Davis.

La relazione dovrà essere consegnata almeno 3 giorni lavorativi prima dell'esame all'indirizzo `lorenzo.mascotto@unimi.it`.

E' possibile (e apprezzato) lavorare in gruppo purché i gruppi siano composti al massimo di 3 studenti.

ALTERNATIVAMENTE

Ripetere quanto richiesto sopra anche a metodi iterativi. In particolar modo, confrontare i *vostri* codici relativi ai metodi di Jacobi e di Gauss-Seidel con quelli presenti in altre *due* librerie (al solito in Python, Java, C, C++, C#, .NET, ... (altro)).

Tenete presente che, come visto a lezione, non sempre i metodi iterativi convergono. Nel progetto dovrete evidenziare quali sono le matrici su cui il metodo in considerazione converge.

Oltre che:

- tempo
- errore relativo
- memoria

è richiesto anche, qualora il metodo in considerazione converga, uno studio riguardante:

- numero iterazioni del metodo.

I parametri del metodo sono a vostra scelta e saranno argomento di discussione in sede di esame.

Vi segnalo che non è scontato trovare in tutte le librerie on line i metodi di Jacobi e Gauss-Seidel. Per questa ragione, l'impiego di altri metodi (iterazione di Chebychev, metodo CG, metodo GMRES, ...) è ovviamente ben accetto.