Libreria EJML

Il primo progetto implementato per la risoluzione di sistema lineari associati ad una matrice sparsa, simmetrica e definita positiva è basato sulla libreria open source EJML. Efficient Java Matrix Library (EJML) è una libreria scritta interamente in Java e rilasciata con licenza Apache v2, è stata progettata per essere facile da usare e offre una sintassi chiara e concisa per la manipolazione di vettori e matrici reali, complesse, dense oppure sparse. Più formalmente fornisce un'ampia gamma di funzionalità matematiche, tra cui la risoluzione di sistemi lineari, il calcolo di autovalori e autovettori, la decomposizione QR, la fattorizzazione LU, la decomposizione di Cholesky, la fattorizzazione SVD.

Gli obbiettivi della libreria EJML sono:

* Essere il più efficiente possibile dal punto di vista computazionale e della memoria per matrici di piccole e grandi dimensioni
* Essere accessibile sia a principianti che agli esperti

Questi obbiettivi sono raggiunti grazie alla selezione dinamica dei migliori algoritmi da utilizzare in fase di esecuzione, a un’API pulita e interfacce multiple.

EJML ha tre interfacce distinte con cui poter interagire:

1) Procedural, fornisce pieno accesso a tutte le funzionalità e capacità di EJML e un controllo quasi completo sulla creazione della memoria, sulla velocità e su algoritmi specifici con un'API procedurale

2) SimpleMatrix, è l’interfaccia utilizzata all’interno del progetto, fornisce un sottoinsieme semplificato delle funzionalità principali tramite un'API orientata alla programmazione ad oggetti, ispirata a Jama.

3) Equations, è un'interfaccia simbolica, concepita per la manipolazione di matrici da parte degli utenti in modo simile a Matlab e ad altri CAS, fornendo un modo compatto per scrivere equazioni.

Di seguito sono riportate tutte le funzionalità per l’algebra lineare offerte dalla libreria EJML

Immagine che contiene testo, numero, schermata, Carattere

Descrizione generata automaticamente

Documentazione di EJML

Sul sito della libreria disponibile al link: <http://ejml.org/wiki/index.php?title=Manual> è presente un breve manuale. Esso si divide in 4 sezioni che forniscono una panoramica della libreria:

* The Basics (nozioni base)
* Tutorials
* Example Code (esempi di codice)
* External References (riferimenti esterni).

La sezione del manuale riguardante le nozioni base fornisce un'introduzione a EJML, compresa la lista delle operazioni standard e le funzionalità principali della libreria. Descrive inoltre come utilizzare e sviluppare un'applicazione utilizzando EJML fornendo in aggiunta una lista di domande frequenti con la possibilità di scrivere eventuali domande relative alla libreria sfruttando una bacheca.

La sezione Tutorials del manuale fornisce delle guide per vari problemi di algebra lineare, come ad esempio operare con matrici a valori complessi, risoluzione di sistemi lineari, decomposizioni di matrici ed esecuzione di unit test. Questi tutorial sono progettati per presentare diversi aspetti di EJML e aiutare gli utenti a capire come risolvere diversi problemi di algebra lineare utilizzando la libreria.

La sezione Example Code del manuale fornisce una tabella contenente vari esempi di codice per i più comuni problemi di algebra lineare risolvibili utilizzando la libreria EJML.

Nella sezione External References del manuale viene consigliato materiale aggiuntivo per tutti gli utenti che desiderano approfondire il funzionamento della libreria EJML per poter sviluppare tecniche avanzate di programmazione.

Infine, sempre sul sito internet della libreria, è presente una pagina dedicata in cui viene spiegato come accedere al codice sorgente di EJML sfruttando la pagina Github ufficiale <https://github.com/lessthanoptimal/ejml> e tutti i comandi necessari per importare ed utilizzare tutti i componenti della libreria tramite file JAR, utilizzando Gradle oppure Maven.

Libreria MFL

All’interno del progetto per importare le matrici sparse in formato .mat da analizzare e manipolare con i componenti della libreria EJML si sfrutta la libreria MFL, disponibile al seguente repository Github: https://github.com/HebiRobotics/MFL.git

La MAT File Library (MFL) è una libreria Java per la lettura e la scrittura di file MAT compatibili con il formato MAT-File di MATLAB. Gli obiettivi generali del progetto sono:

* fornire un'API di facile utilizzo che aderisca al comportamento semantico di MATLAB,
* supportare il lavoro con grandi quantità di dati in ambienti con vincoli di heap o allocazione limitata,
* consentire agli utenti di serializzare classi di dati personalizzate senza doverle convertire in oggetti

La libreria è gratuita, scritta al 100% in Java ed è stata rilasciata con licenza Apache v2.0. Funziona con Java 6 e versioni successive, incluse Java 9 e 10.

Inoltre, all’interno della libreria è presente il modulo mfl-ejml che fornisce un supporto preliminare per la conversione tra file MAT e tipi di dati EJML. I wrapper di serializzazione sono molto leggeri e serializzano i dati contenuti nel formato file MAT direttamente senza richiedere memoria aggiuntiva per la memorizzazione di dati intermedi.

Infine, all’interno del file readme del repository Github sono disponibili:

* esempi base, accompagnati dal codice java, utili all’utente per capire come svolgere al meglio le operazioni di creazione, scrittura e lettura di file MAT
* le dipendenze presenti nel Maven central repository, necessarie per importare correttamente la libreria MFL in progetti Maven e Gradle

//todo inserire grafici

//todo listato del codice (poi da sistemare in latex)

Descrizione del programma

Il programma esegue diverse operazioni su una serie di file che contengono matrici sparse e definite positive, con lo scopo di calcolare il tempo di esecuzione, la memoria utilizzata e l'errore relativo, tra la soluzione calcolata e quellla esatta, durante la decomposizione di Cholesky e la risoluzione di un sistema lineare Ax=b. All'interno del programma è possibile trovare diverse funzioni quali:

* Mat5.readFromFile(file.getAbsolutePath()).getStruct(“Problem”).getSparse(“A”): Questo metodo è un'istruzione di lettura del file nel formato Mat5 utilizzando la libreria MFL (MatFile Library). L'obiettivo di questa istruzione è ottenere una matrice sparsa denominata "A" dal campo "Problem" di un file Mat5.
* file.getAbsolutePath(): rappresenta il percorso assoluto del file Mat5 da cui leggere i dati.
* Mat5.readFromFile(file.getAbsolutePath()): questo comando legge il file Mat5 specificato dal percorso assoluto file.getAbsolutePath() utilizzando la libreria MFL. Restituisce un oggetto Mat5 che rappresenta il contenuto del file.
* .getStruct("Problem"): utilizzando l'oggetto Mat5 ottenuto, viene richiamato il metodo getStruct("Problem") per ottenere la struttura denominata "Problem" dal file Mat5. Una struttura in un file Mat5 è un contenitore per dati strutturati.
* .getSparse("A"): utilizzando l'oggetto “Struct” ottenuto, viene richiamato il metodo getSparse("A") per ottenere una matrice sparsa denominata "A" dalla struttura "Problem" del file Mat5.
* MatrixFeatures\_DSCC.isPositiveDefinite(A): è un metodo fornito dalla libreria EJML (Efficient Java Matrix Library) per determinare se una matrice sparsa A è definita positiva.
* MatrixFeatures\_DSCC.isSymmetric(A, tolerance): è un metodo fornito dalla libreria EJML (Efficient Java Matrix Library) per verificare se una matrice sparsa A è simmetrica entro una determinata tolleranza verificando se la differenza assoluta tra gli elementi simmetrici della matrice è inferiore o uguale alla tolleranza specificata.
* CommonOps\_MT\_DSCC.mult(A, B, C): esegue il prodotto di due matrici sparse A e B e memorizza il risultato nel parametro C specificato. Le matrici A e B devono essere compatibili per la moltiplicazione, il numero di colonne di A deve essere uguale al numero di righe di B. Il risultato del prodotto è una nuova matrice C di dimensioni appropriate.
* LinearSolverSparse<DMatrixSparseCSC, DMatrixRMaj> solver = LinearSolverFactory\_DSCC.cholesky(FillReducing.NONE): L'istruzione crea un oggetto solver di tipo LinearSolverSparse utilizzando il metodo statico cholesky della classe LinearSolverFactory\_DSCC. Questo risolutore di sistemi lineari sparsi è basato sulla decomposizione di Cholesky e viene configurato con l'opzione FillReducing.NONE.
* solver.setA(A): L'istruzione imposta la matrice del sistema lineare A per il risolutore solver utilizzando il metodo setA fornito dal risolutore. La matrice A deve essere di tipo DMatrixSparseCSC e rappresenta il sistema di equazioni lineari da risolvere.
* solver.solveSparse(B, x): L'istruzione risolve il sistema lineare sparsamente rappresentato da A \* x = B utilizzando il metodo solveSparse del risolutore solver. Il vettore B rappresenta i termini noti del sistema e il vettore x è il vettore soluzione che verrà calcolato e memorizzato in x.
* CommonOps\_DSCC.add(alpha, A, beta, B, C, structure, workspace): Il metodo “add()” calcola la somma alpha\*A + beta\*B di due matrici sparse A e B e memorizza il risultato nel parametro C specificato. Le matrici A e B devono avere le stesse dimensioni e la stessa struttura sparsa. Il parametro alpha rappresenta un fattore scalare da moltiplicare per la matrice A, mentre il parametro beta rappresenta un fattore scalare da moltiplicare per la matrice B. Questi fattori scalari consentono di regolare l'importanza relativa delle matrici A e B nella somma finale.
* NormOps\_DSCC.normF(A): è un metodo fornito dalla libreria EJML (Efficient Java Matrix Library) per calcolare la norma di Frobenius di una matrice sparsa A di tipo DMatrixSparseCSC (compressed sparse column format).