Marco Lorusso
Matricola: 761302

Relazione per l'esame di Programmazione per la Fisica

L'intento della presente relazione è quello di presentare il programma da me scritto il quale permette di calcolare, a partire da una matrice data dall'utente, la sua inversa utilizzando l'algoritmo di Gauss, come descritto nei testi di Algebra Lineare.

L'algoritmo

In questo paragrafo cercherò di illustrare in maniera sintetica le basi matematiche del metodo utilizzato nel programma.
Consideriamo una matrice denominata A il cui determinante è diverso da 0, altrimenti non sarebbe invertibile, e sfruttiamo la proprietà per la quale il suo prodotto *righe per colonne* con la rispettiva matrice inversa risulta essere la matrice identità I, che ha la caratteristica di avere gli elementi lungo la diagonale uguali a 1 e i restanti uguali a 0.
Per iniziare impostiamo una matrice accostando A con I alla sua destra, ottenendone una rettangolare con un numero di colonne doppio rispetto a quello delle righe.
Bisogna controllare che l'elemento iniziale della prima riga non sia 0, in caso contrario sarà necessario scambiare la riga con un'altra che rispetti questa condizione. Se non è presente alcuna riga avente il primo membro non nullo, il determinante sarà uguale a 0, implicando la non esistenza della matrice inversa.
Nel caso in cui si sia superato il passo precedente, ricorrendo alle seguenti operazioni elementari di riga, si ottiene la matrice identità dove originariamente erano presenti gli elementi di A nella matrice rettangolare:

- scambio di due righe;
- moltiplicazione di una riga per un numero reale diverso da 0;
- sostituzione della riga j-esima con la somma della riga i-esima e della j-esima moltiplicata per un numero reale qualsiasi.

In questo modo l'inversa di A sarà data dai valori presenti nella parte destra della matrice utilizzata per fare queste operazione, dove all'inizio era stata posta I.
Nell'eventualità in cui la soluzione ottenuta presenti una riga composta di soli 0, capiamo che la nostra matrice iniziale possiede un determinante uguale a 0, dunque risulta non invertibile.

Il codice

Il codice qui presentato si basa sulla creazione di una classe, di nome "Matrice", che contiene la ridefinizione degli operatori delle principali operazioni quali somma e differenza tra matrici, moltiplicazione e divisione per uno scalare, prodotto tra matrici *righe per colonne*, assegnamento tra matrici e dell'operatore "<<", rendendone necessaria la dichiarazione all'interno della classe con l'utilizzo della parola di vocabolario *friend*, che permette l'output in terminale di una qualsiasi matrice più immediato.
In questa classe sono presenti 3 costruttori:

- *Matrice (int, int)* è quello più generico utilizzato nella creazione di matrici qualsiasi;
- *Matrice (int, int, double)* permette la creazione di matrici aventi tutti gli elementi con lo stesso valore *double* ed è stato utilizzato come ausilio nella ridefinizione degli operatori.
- *Matrice (int)* è quello utilizzato nel nostro caso poiché costruisce direttamente la matrice rettangolare avente i valori inseriti dall'utente a sinistra e la matrice identità sulla destra.

Ogni oggetto *Matrice* è basato su un doppio puntatore ***m* allocato e inizializzato attraverso cicli *for*, nel secondo caso nidificati in maniera opportuna per scorrere tutti gli indici del doppio array risultante.
Per la visualizzazione della matrice inserita e per la sua inversa a conclusione dell'algoritmo si sono inoltre definiti due metodi, *Anteprima()* e *Risultato()*, che permettono la visualizzazione rispettivamente della prima e della seconda metà della matrice rettangolare creata.
Inoltre, è stato necessario definire 3 funzioni al fine di implementare le 3 operazioni elementari di riga precedentemente elencate, in cui i valori *int* servono a creare dei puntatori ausiliari della stessa dimensione delle righe presenti nella matrice:

- *scambia (double*, double*, int)* che ricevendo i due puntatori a double rappresentanti le due righe per riferimento, assegna al primo il valore del secondo e viceversa;
- *dividi (double*, double, int)* il quale non fa altro che sostituire i valori della riga datagli, per questo il puntatore è sempre ricevuto per riferimento, con il risultato della loro divisione con il *double* ricevuto;
- *sottrai (double*, double*, int, double)* infine permette la sottrazione tra la riga rappresentata dal secondo puntatore e un multiplo *double* della riga contenuta nel primo puntatore.

Grande parte della gestione dell'input è affidata a due funzioni *template*.
All'interno della funzione *template <class X> X InputCheck()* è dichiarata la variabile *Input* del tipo passatole come parametro che, se nell'input è presente solo la pressione del tasto *Invio*, viene inizializzata a zero e poi assegnata alla variabile dalla quale inizializzazione è stata invocata la funzione. Altrimenti, se l'input fallisce, si sposta l'indice di lettura fino alla fine dell'input che ha provocato errore e, richiedendo l'inserimento del tipo adatto attraverso la funzione *TypeSorter()*, anche questa templatizzata e con uno *switch* che ha proprio il compito di smistare i casi relativi ai vari tipi. Per di più è controllata da *InputCheck()* anche la presenza di spazi inopportuni che terminerebbero lo stream al posto del carattere di andata a capo. Tutti questi controlli sono posti in un ciclo *do/while* che perciò permette l'uscita dalla funzione solo in caso di input completamente valido.
Infine, passiamo al nucleo del programma inserito anch'esso in un ciclo *do/*while per consentire un' esecuzione ripetuta più velocemente, a sua volta contenuto nell'ambito della funzione *main()*.
Per prima cosa si costruisce la matrice rettangolare su cui attuare le operazioni inserendo la dimensione della matrice che si vuole invertire e i valori contenuti al suo interno. In questo caso si è voluto utilizzare un controllo più basilare, basato su semplici *if*, per evitare l'inserimento di dimensioni negative o uguali a 0, a causa della loro insensatezza, e troppo grandi in modo tale da rendere più difficile la possibilità che la troppa memoria allocata dal programma dia problemi al calcolatore su cui è utilizzato, dando però comunque la possibilità di superare il limite arbitrario imposto *a proprio rischio e pericolo*.
Dopo la stampa a schermo della matrice inserita si verifica che la prima riga abbia come primo elemento un numero diverso da 0, in caso contrario si effettua uno scambio con la prima riga accettabile e nell'eventualità non ce ne fossero viene riconosciuto che la matrice inserita ha un determinante nullo non permettendone l'esistenza dell'inversa.
Qui finalmente si arriva al vero e proprio algoritmo di Gauss composto da due cicli *for*, uno all'interno dell'altro, che invocano le funzioni *dividi (double*, double, int)* e *sottrai (double*, double*, int, double)* per:

- dividere, ad ogni iterazione del ciclo esterno, tutti gli elementi della riga i-esima per l'elemento nella posizione i-esima della stessa riga: facendo questo l'elemento i-esimo diverrà uguale a 1 in ogni ripetizione;
- sottrarre, nel ciclo interno, a tutte le righe la riga i-esima moltiplicata per il valore in posizione *[k][i]* in maniera tale da renderlo uguale a 0 ad ogni iterazione.

Ripetendo questo procedimento si ottiene: la matrice identità nella parte sinistra della matrice rettangolare costruita e l'inversa di quella inserita inizialmente sulla destra la quale, dopo un ultimo controllo in maniera tale da evitare il caso in cui si sia ottenuta una matrice nulla, viene stampata a schermo, seguita da una richiesta di riesecuzione del programma che, se risposta in maniera negativa, modifica la variabile *booleana* all'interno della clausola del *while* in falsa, permettendo la conclusione dell'esecuzione senza errori.

Per poter scrivere questo programma si sono incluse le librerie <*iostream*> e <*iomanip*>, per l'utilizzo dei relativi oggetti addetti all'input/output e alla sua manipolazione per rendere più gradevole la lettura sul terminale; <*typeinfo*> per riconoscere il tipo delle variabili inserite, necessità presente nella funzione *template* addetta al controllo dell'input.

Bibliografia e sitografia

R.Fioresi, M.Morigi, Introduzione all'Algebra Lineare, Edizioni Ambrosiana, Milano 2015.
Dispense didattiche alla pagina http://www.physycom.unibo.it/labinfo/ curate dal professore G.Servizi.