UNIVERSITÀ DEGLI STUDI DI CAGLIARI

FACOLTÀ DI INGEGNERIA

Fattorizzazione QR e problemi ai minimi quadrati.

Seminario: Numerical Linear Algebra

(Professor Rodriguez)

Giulia Marcello
Francesca Marcello

ANNO ACCADEMICO 2013-2014

1 Introduzione

In questo elaborato è descritta la risoluzione di un sistema lineare:

$$A\mathbf{x}=\mathbf{b}$$

mediante l'uso di alcune fattorizzazioni della matrice dei coefficienti A.

L'obbiettivo del lavoro è stato l'implementazione dell'algoritmo QR per la fattorizzazione, basandoci sulla costruzione delle matrici di Householder, utilizzando poi i minimi quadrati per risolvere il sistema. In seguito sono stati eseguiti dei test di confronto tra gli algoritmi personalizzati e le relative versioni pre-implementate e funzionanti in ambiente di calcolo MATLAB.

2 Fattorizzazione QR

Lo scopo della fattorizzazione è quello di scomporre una matrice in un prodotto di matrici aventi una struttura più semplice (matrici triangolari, diagonali, ortogonali).
È possibile fattorizzare ogni matrice mxn nella forma

$$A= QR$$

Dove Q è una matrice mxm ortogonale ed R è triangolare superiore con le stesse dimensioni di A.
È importante notare che qualunque matrice A può essere fattorizzata in questo modo: in particolare tale scomposizione esiste anche per matrici singolari (nel qual caso, essendo per una matrice ortogonale $\det(Q)\neq 0$, sarà singolare la matrice R).

Esistono diversi modi per ottenere una fattorizzazione A = QR, tra i quali si è scelto di implementare il metodo delle matrici ortogonali di Householder.

3 La fattorizzazione QR di Householder

La fattorizzazione QR di Householder è incentrata sulla determinazione delle matrici elementari di Householder.
Una matrice elementare di Householder è una matrice del tipo:
$H = I - 2\mathbf{w}\mathbf{w}^T$, $\mathbf{w} \in \Re^n$, $||\mathbf{w}|| = 1$

La matrice di Householder ha delle proprietà importanti:

-è simmetrica, quindi sappiamo che $H=H^T$;
-è ortoganale, per cui $H^T=H^{-1}$ e quindi $HH^T=I$
-è una matrice associata ad una riflessione, ogni vettore x moltiplicato per H viene riflesso rispetto al piano.

Si può determinare un valore di **w** tale da fare in modo che la moltiplicazione a sinistra di questa matrice per un qualsiasi vettore di $\Re^n$ lo trasformi in un vettore parallelo al primo versore e_1 della base canonica, ossia:
$H\mathbf{x} = ke_1$

La determinazione analitica del vettore **w** e quindi della matrice di Householder H conduce alla definizione del seguente algoritmo:

1.$\sigma = ||\mathbf{x}||$
2. $k = -\text{sign}(x_1)\sigma$
3. $\lambda = \sqrt{2\sigma(\sigma + |x_1|)}$
4. $\mathbf{w} = (\mathbf{x} - ke_1)/\lambda$
5. $H = I - 2\mathbf{w}\mathbf{w}^T$

Si può esprimere la matrice H in una forma diversa che consente di ridurre gli errori e di scrivere un algoritmo modificato:

1.$\sigma = ||\mathbf{x}||$
2. $k = -\text{sign}(x_1)\sigma$
3. $\beta = \sigma(\sigma + |x_1|)$
4. $\mathbf{v} = (\mathbf{x} - ke_1)$
5. $H = I - \frac{1}{\beta}\mathbf{v}\mathbf{v}^T$

L’ultimo algoritmo appena descritto è stato implemento in MATLAB nella funzione MyHouseholder.m

```
function [H] = MyHouseholder(x)     %il vettore deve essere COLONNA
%Trasformazione di Householder
sigma=norm(x);
if x(1)<0
    k=sigma;
else
    k=-sigma;
end
beta=sigma*(sigma+abs(x(1)));

v=x;
 v(1)=x(1)-k;

 [b,a]=size(x);  %prendo b che è la lunghezza del vettore COLONNA
H=eye(b)-(1/beta)*v*v';

end
```

A partire da un vettore **x** che deve essere trasformato e che viene passato in ingresso alla funzione otteniamo la matrice di Householder H.

4 Fattorizzazione QR di Householder

Le matrici elementari di Householder possono essere utilizzare per determinare la fattorizzazione QR di una matrice A. Consideriamo dapprima il caso in cui A sia quadrata di ordine n. Nel corso dell'algoritmo si genererà una successione di matrici A_i per i = 1,…,n, tali che A_n sia triangolare superiore(e quindi coincida con la matrice R della fattorizzazione).
Abbiamo implementato in MatLab l'algoritmo, presente a pagina 97 del libro "Algoritmi Numerici", nella nostra funzione myQR.m

```
function [Q R]= myQR(A)
[m,n]=size(A); %m è il numero di righe, n è il
                numero di colonne

M=max(m,n);
Qi=eye(M);
if m==n         %matrice quadrata
for i=1:(n-1)
    a= A(i:m,i);
    Hi=MyHouseholder(a); %richiamo funzione per generare  la matrice di  Househ.
```

```
I=eye(M);   %matrice identità della dimensione necessaria
I(i:m,i:m)=Hi;  %orlatura della matrice di Householder
     A=I*A; ;  %aggiornamento della matrice A del sistema

Qi=Qi*I; ; %aggiornamento della matrice Q della fattorizzazione
end
Q=Qi;  %matrice Q della fattorizzazione
  R=A; %matrice R della fattorizzazione

elseif m>n
for i=1:n
        a=A(i:m,i);
        Hi=MyHouseholder(a);
I=eye(M);
        I(i:m,i:m)=Hi;
        A=I*A;

        Qi=Qi*I;
end

Q=Qi;
    R=A;
else
    Qi=eye(m);      %m<n
for i=1:(n-(n-m))
        a=A(i:m,i);
        Hi=MyHouseholder(a);
I=eye(m);
        I(i:m,i:m)=Hi;
        A=I*A;

        Qi=Qi*I;
end

Q=Qi;
    R=A;

end
end
```

Il codice prende in esame 3 casi a seconda delle dimensioni della matrice considerata. Il primo caso è quello di una matrice A quadrata di dimensione mxn.

Al primo passo, a partire dalla matrice in ingresso si costruisce un vettore $\mathbf{a_1}$ considerando i primi termini delle sue righe.

$$A=\begin{pmatrix} \alpha & \alpha & \alpha & \alpha \\ a & a & a & \alpha \\ a & a & a & a \\ a & a & a & a \end{pmatrix} \rightarrow H_1 \rightarrow \begin{pmatrix} \beta & y & y & y \\ 0 & y & y & y \\ 0 & y & y & y \\ 0 & y & y & y \end{pmatrix} = H_1A$$

Questo vettore sarà l'input della funzione che permette di determinare la matrice di Householder H_1. Tale matrice viene applicata a sinistra della matrice A di partenza in modo da generare una seconda matrice A_2 della successione. Al secondo passo sarà da questa matrice che si ricaverà il vettore $\tilde{\mathbf{a}}_2$ di dimensione (n-1) a partire da cui si costruisce la matrice di Householder anche essa di dimensione (n-1). Una volta ottenuta, questa matrice dovrà essere orlata con una riga e una colonna della matrice identità in modo da raggiungere le dimensioni n. Questo nel nostro codice viene fatto sovrascrivendo ad ogni passo la matrice ottenuta dalla funzione MyHouseholder in una matrice identità delle dimensioni volute.

$$A_2=\begin{pmatrix} \beta & y & y & y \\ 0 & y & y & y \\ 0 & y & y & y \\ 0 & y & y & y \end{pmatrix} \rightarrow \tilde{H}_2$$

$$H_2=\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & & & \\ 0 & & \tilde{H}_2 & \\ 0 & & & \end{pmatrix}$$

La nuova matrice così determinata sarà applicata a sinistra della matrice A_2 per ottenere la terza matrice della successione.

$$H_2H_1A = \begin{pmatrix} \beta & y & y & y \\ 0 & \xi & z & z \\ 0 & 0 & z & z \\ 0 & 0 & z & z \end{pmatrix} = A_3$$

L’algoritmo termina al passo n-1 con la matrice triangolare superiore
$A_n=H_{(n-1)}A_{(n-1)} = H_{(n-1)}H_{(n-2)}\ldots H_1A = R$
La matrice Qi è data dalle successive moltiplicazioni delle matrici di Householder che si ottengono passo per passo, ed è una matrice ortogonale in quanto prodotto di matrici ortogonali.
$Q = H_1H_2...H_{(n-1)}$
Quindi la funzione in uscita da le due matrici Q e R della fattorizzazione tali per cui
A=QR.
Gli altri casi considerati riguardano matrici di dimensioni rettangolari.
Nel caso in cui m>n il procedimento è lo stesso del caso precedente ma occorrerà un passo ulteriore per azzerare gli elementi dell’ultima colonna di A. Mentre nel caso in cui m<n il numero di passi è inferiore e pari alla differenza n-(n-m).

5 Minimi quadrati.

In molti casi si incontrano sistemi lineari in cui il numero di equazioni è diverso dal numero delle incognite.
Si possono verificare due casi:

1. Il numero di equazione è maggiore di quello delle incognite (m>n), il sistema risulta sovradeterminato e potrebbero non trovarsi soluzioni;
2. Il numero di incognite è maggiore di quello delle equazioni (m<n), il sistema si dice sottodeterminato e potrebbe avere infinite soluzioni.

Il caso di rango non pieno in cui alcune equazioni sono dipendenti dalle altre si può ricondurre ad una di queste due condizioni.
Il problema in questi casi è mal posto e non si possono trovare soluzioni in senso classico ma si richiede che lo scarto quadratico medio tra il primo e il secondo membro del sistema sia minimo. Si passa quindi ad un problema ai minimi quadrati. La condizione di varianza minima si può esprimere nella forma:

$$\min_{x\in R^n} ||A\mathbf{x}-\mathbf{b}||_2$$

Il nostro obbiettivo è quello di risolvere un sistema con il metodo dei minimi quadrati applicando la fattorizzazione QR.
Dato un certo sistema A**x**=**b**, si può considerare il quadrato della norma del residuo e minimizzare

$$||A\mathbf{x}-\mathbf{b}||^2$$

Per far questo scomponiamo fattorizziamo A=QR:

$$||A\mathbf{x}-\mathbf{b}||^2=||QR\mathbf{x}-\mathbf{b}||^2=||QR\mathbf{x}-QQ^T\mathbf{b}||^2=||Q(R\mathbf{x}-Q^T\mathbf{b})||^2$$

Poiché $||Q\mathbf{x}||_2=||\mathbf{x}||_2$

Allora:

$$||R\mathbf{x}-Q^T\mathbf{b}||^2=\left\| \begin{bmatrix} R_1 \\ 0 \end{bmatrix}\mathbf{x}-\begin{bmatrix} C_1 \\ C_2 \end{bmatrix}\right\|^2=\left\| \begin{matrix} R_1\mathbf{x}-C_1 \\ -C_2 \end{matrix}\right\|^2=\left|\left|R_1\mathbf{x}-C_1\right|\right|^2+\left|\left|C_2\right|\right|^2$$

$R_1\mathbf{x}=C_1 \quad \min||A\mathbf{x}-\mathbf{b}||^2=||C||^2$

Quindi

1) A=QR
2) $C=Q^T\mathbf{b}=\begin{bmatrix} C_1 \\ C_2 \end{bmatrix}_{n-m}^{n}$
3) $R=\begin{bmatrix} R_1 \\ 0 \end{bmatrix}$
4) Risolvo $R_1x=C_1$

Se $C_2=0$ la soluzione coincide con la soluzione classica.
Se $C_2\neq 0$la soluzione coincide con quella dei minimi quadrati e la sua norma fornisce la misura del residuo.
Se il problema è sovradeterminato troveremo soltanto un minimo diverso da 0, invece nel caso di un problema sottodeterminato i minimi sono più di uno e in generale si cerca **x** tale che $||\mathbf{x}||_2^2$ sia minimo.

Abbiamo scritto una funzione in MATLAB che implementasse quest'algoritmo: soluzionesistemaQR.m, in cui sono considerate tutte le possibili situazioni m>n, m=n, m<n.

```
function [x_qr] = soluzionesistemaQR(A,Q,R,b)

[m n]=size(A);
C=Q'*b;
```

```
if m>=n
C1=C(1:n);
C2=C((n+1):m);
R1=R(1:n,:);
x_qr=R1\C1;

else
   C1=C(1:m);
   C2=zeros(n-m,1);
   R1=R(1:m,:);
x_qr=R1\C1;

end

end
```

6 Test e verifica

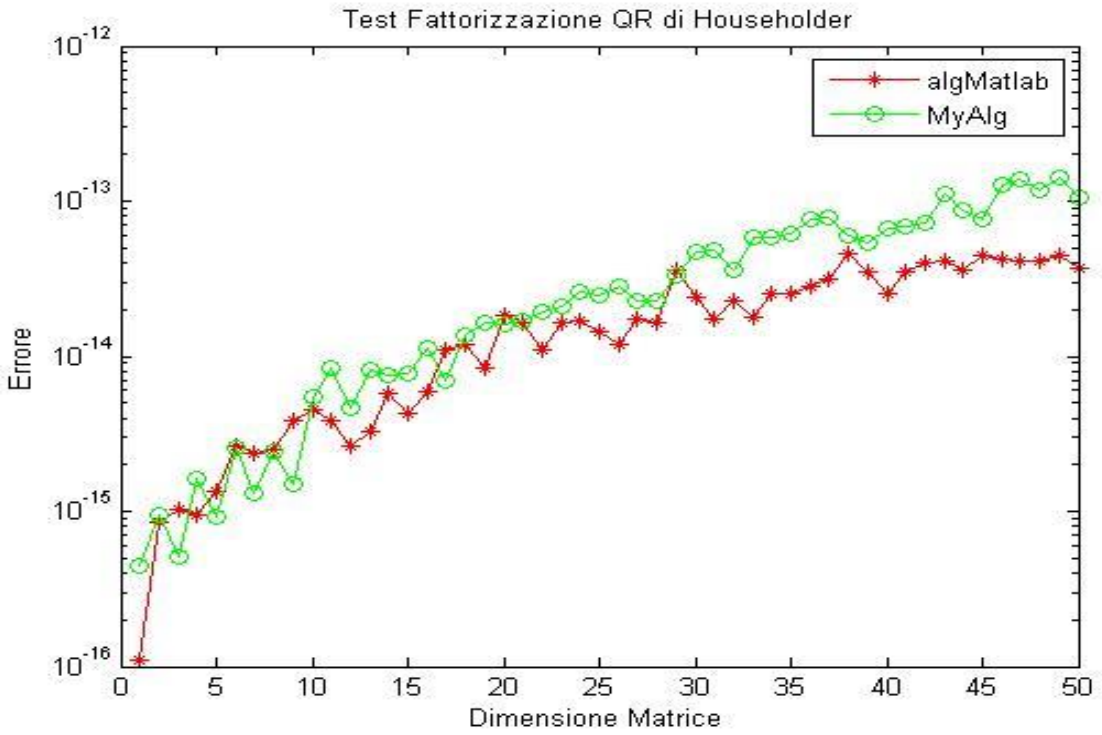
A questo punto ci proponiamo di testare gli algoritmi prodotti confrontandoli con quelli di Matlab.
Procediamo in questo modo:

1. Creiamo un sistema lineare del tipo A**x**=**b** del quale conosciamo la soluzione esatta **x**.
2. Fattorizziamo la matrice A utilizzando rispettivamente l'algoritmo di Matlab e l'algoritmo implementato.
3. Andiamo a valutare l'errore ossia lo scostamento dalla soluzione esatta a noi nota.

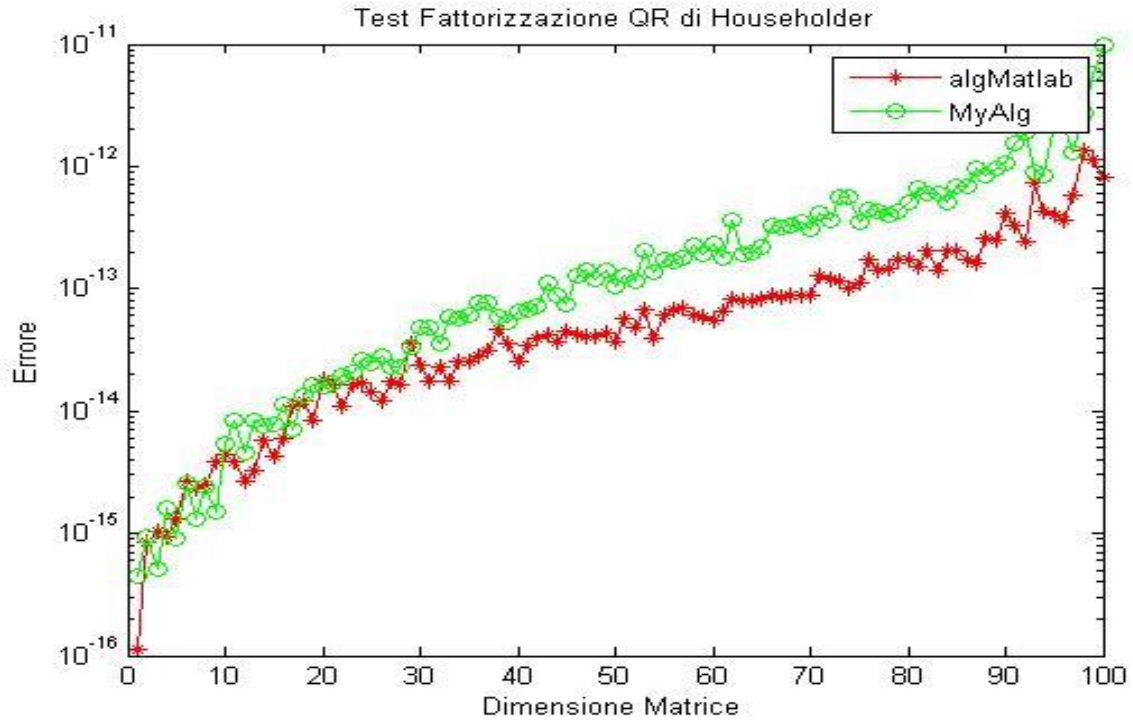
Questo è possibile farlo considerando il caso quadrato, sovradeterminato e sottodeterminato.

```
m =100;
n =50
for i= 1:n
A =rand(m,i);
e = ones(i,1);
b = A*e;
```

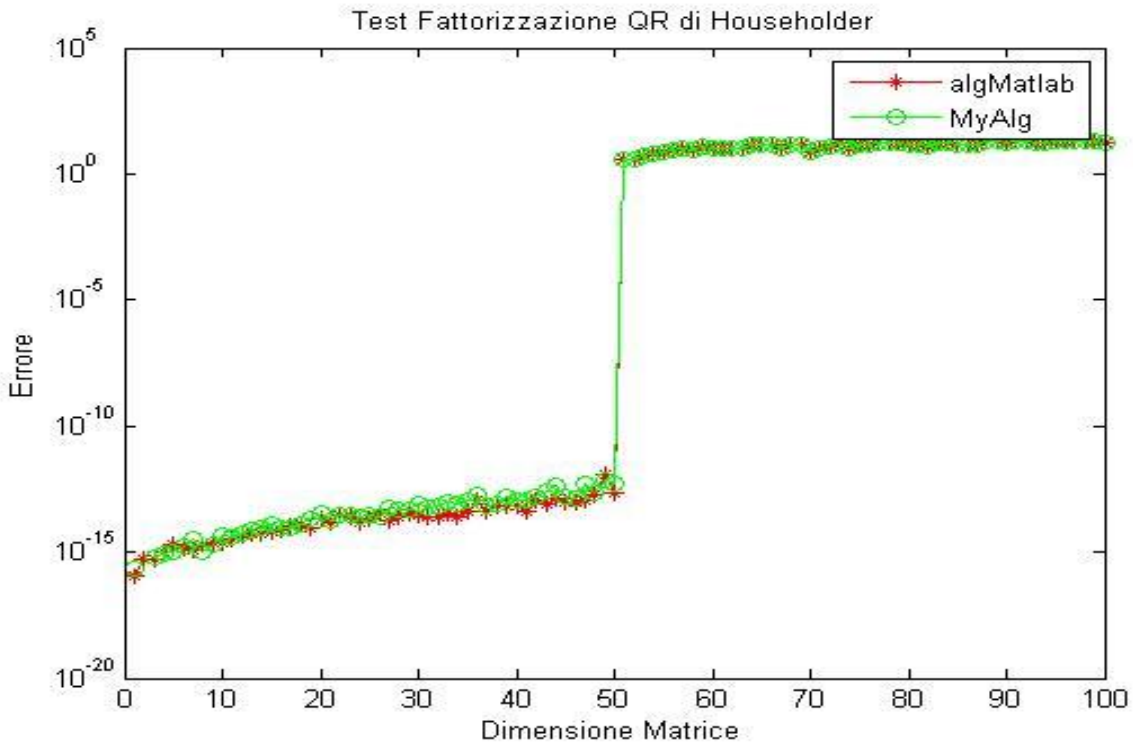
```
%Risoluzione con algoritmo di Matlab
[Q,R] = qr(A);
y = Q'*b;
x= R\y;

%Errore usando l'algoritmo di matlab
err1(i) = norm(x-e);

%Risoluzione con l'algoritmo implementato
[Q,R]= myQR(A);
x_my=soluzionesistemaQR(A,Q,R,b);
err2(i) = norm(x_my-e);
end

%Visualizza vettore degli errori(in norma 2).
t = linspace(1,n,n);
semilogy(t,err1,'r-*',t,err2,'g-o');
legend('algMatlab','MyAlg');
xlabel('Dimensione Matrice');
ylabel('Errore');
title('Test Fattorizzazione QR di Householder');
```



Il primo test riguarda un sistema sovradeterminato e dal grafico sull'andamento dell'errore si vede che l'implementazione di MatLab è migliore di quella fornita da noi anche se gli errori commessi sono minimi. Cambiando i valori della dimensione della matrice possiamo considerare gli altri due casi, per esempio per una matrice quadrata di dimensione n=100 otteniamo il seguente grafico:

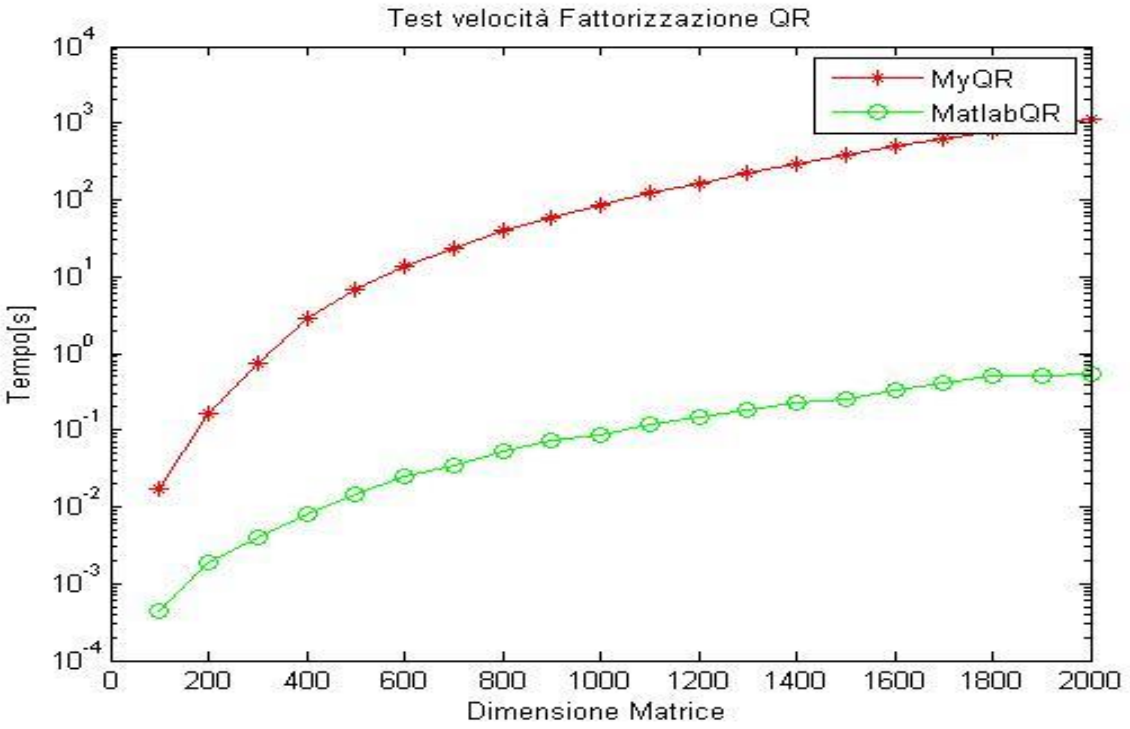


Per cui valgono le stesse considerazioni del caso precedente. Infine possiamo analizzare il sistema sottodeterminato:



Il nostro algoritmo in questo caso ha un comportamento più vicino a quello di MatLab, ma gli errori raggiungono valori nettamente superiori ai casi precedenti, con una differenza evidentemente maggiore con l'aumentate delle dimensioni della matrice.

Oltre al test sulle prestazioni abbiamo fatto anche un confronto sulle velocità di calcolo tra il nostro algoritmo che effettua la fattorizzazione QR e l'algoritmo già presente in Matlab. Il test è stato effettuato aumentando di volta in volta la dimensione della matrice, a partire da matrici quadrate con n=100 fino a n=2000; dimensioni maggiori richiedevano tempi di elaborazione troppo elevati.
Abbiamo ottenuto il seguente grafico:



Il nostro algoritmo determina la fattorizzazione in tempi decisamente più alti rispetto a quello di Matlab, e la differenza è ancora più evidente per dimensioni elevate della matrice. Questo è, però, un risultato aspettato considerando che l'algoritmo di Matlab è scritto in linguaggio Fortran e compilato all'interno del programma stesso.

Bibliografia

G. Rodriguez.
Algoritmi Numerici.
Pitagora Editrice, Bologna, 2008.