Analisi sulla Risoluzione di Sistemi Lineari ai Minimi Quadrati

Carlo Cabras - matr. 60/73/65113

Luglio 2019

Indice

Sommario

In questa mia tesina per l'esame di *Computational Mathematics*, analizzo metodi diretti per la risoluzione di sistemi lineari, con particolare riguardo per la risoluzione ai minimi quadrati.

Conduco dei test su sistemi quadrati, poi sovradimensionati e sottodimensionati, confrontando degli algoritmi scritti da me con gli algoritmi forniti da MATLAB .

Nel repository GitHub [1] sono presenti tutti i file utilizzati per il progetto, sia i sorgenti che i risultati salvati dopo ogni test effettuato.

1 Metodi diretti per la risoluzione di sistemi lineari

I sistemi di equazioni lineari sono i problemi numerici che si incontrano più spesso nelle applicazioni della matematica: essi sono risolvibili tramite algoritmi finiti, chiamati metodi diretti. Esistono diversi tipi di metodi diretti ma tutti seguono la stessa strategia di base: trasformare, con un numero finito di passi, un sistema lineare generico in uno equivalente dotato di una struttura che lo renda più semplice da risolvere [2].

1.1 Condizionamento di un sistema lineare

Un sistema di n equazioni in n incognite assume la forma:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \quad\vdots \qquad\quad \vdots \qquad\qquad\quad \vdots \qquad \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n \end{cases}$$

indicato usando la forma compatta

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

dove $A = (a_{ij})_{i,j=1}^{n}$ è la matrice dei coefficienti, $\mathbf{b} = (b_1, b_2, \ldots, b_n)^T$ il vettore dei termini noti e $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$ la soluzione.

Un sistema lineare ammette una e una sola soluzione se e solo se è verificata una delle seguenti proprietà equivalenti:

1. il determinante di A è diverso da zero;
2. il rango di A è uguale a n;
3. il sistema omogeneo $A\mathbf{x} = 0$ ammette la sola soluzione banale ($x_i = 0, i = 1, \ldots, n$).

Considerato un sistema lineare $A\mathbf{x} = \mathbf{b}$, se perturbiamo i termini noti con un vettore $\delta\mathbf{b}$ di errori anche la soluzione del sistema risulterà perturbata. Detta $\delta\mathbf{x}$ tale perturbazione, il sistema soddisfatto dai vettori perturbati sarà

$$A(\mathbf{x} + \delta\mathbf{x}) = (\mathbf{b} + \delta\mathbf{b}).$$

Da questo, utilizzando la relazione $A\mathbf{x} = \mathbf{b}$, si ottiene il sistema verificato dagli errori

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

la cui soluzione, introdotta una norma matriciale consistente, soddisfa la diseguaglianza

$$||\delta\mathbf{x}|| \leq ||A^{-1}|| \cdot ||\delta\mathbf{b}||. \tag{1}$$

Questa relazione dà informazioni sul condizionamento assoluto del problema, ma è possibili ottenere un'espressione anche per il condizionamento relativo. Infatti, il sistema iniziale implica $||\mathbf{b}|| \leq ||A|| \cdot ||\mathbf{x}||$, ossia

$$\frac{1}{||\mathbf{x}||} \leq ||A|| \cdot \frac{1}{||\mathbf{b}||}.$$

Moltiplicando membro a membro questa diseguaglianza per la (1), si ottiene

$$\frac{||\delta\mathbf{x}||}{||\mathbf{x}||} \leq ||A||\ ||A^{-1}|| \cdot \frac{||\delta\mathbf{b}||}{||\mathbf{b}||}. \tag{2}$$

Si definisce *numero di condizionamento* di una matrice, relativamente alla risoluzione di un sistema lineare, la quantità

$$k(A) = ||A||\ ||A^{-1}||. \tag{3}$$

Il numero di condizionamento misura il massimo fattore di amplificazione dell'errore relativo sulla soluzione rispetto all'errore relativo sui dati.

1.2 Sistemi lineari "facili"

Prima di parlare di algoritmi generici per la risoluzione di sistemi lineari non dotati di una particolare struttura, è opportuno studiare alcun algoritmi specifici per determinate classi di sistemi strutturati.

- *Sistemi diagonali*: un sistema diagonale assume la forma $D\mathbf{x} = \mathbf{b}$, dove $D = diag(d_1, \ldots, d_n)$ con $d_i \neq 0, i = 1, \ldots, n$. La soluzione è data semplicemente dividendo il coefficiente del termine noto per il corrispettivo coefficiente della diagonale. Il calcolo ha complessità $O(n)$;

- *Sistemi ortogonali*: hanno la forma $Q\mathbf{x} = \mathbf{b}$, dove $Q^TQ = QQ^T = I$. La soluzione è data da $\mathbf{x} = Q^T\mathbf{b}$ e richiede $O(n^2)$ sia di moltiplicazioni che di addizioni;

- *Sistemi triangolari*: la matrice dei coefficienti è triangolare inferiore ($a_{ij} = 0$ per $i < j$) o superiore ($a_{ij} = 0$ per $i > j$). Nel primo caso $L\mathbf{x} = \mathbf{b}, l_{ij} = 0, i < j$, si risolve dapprima la prima equazione, che risulta essere in una sola variabile. Il valore calcolato può quindi essere sostituto nella seconda equazione per calcolare la seconda variabile, e così via. Nel secondo caso $U\mathbf{x} = \mathbf{b}, u_{ij} = 0, i > j$, si risolve per prima l'ultima equazione e si procede a ritroso verso la prima equazione. Entrambi gli algoritmi richiedono $O(\frac{1}{2}n^2)$ moltiplicazioni.

1.3 Fattorizzazione QR

Ogni matrice A $m \times n$ è fattorizzabile nella forma

$$A = QR,$$

con Q matrice $m \times m$ ortogonale, cioè tale che $Q\mathbf{x} = \mathbf{b}$, dove $Q^TQ = QQ^T = I$, e R triangolare superiore con le stesse dimensioni di A.

Quando A è una matrice quadrata non singolare, la fattorizzazione QR può essere utilizzata per la risoluzione del sistema lineare $A\mathbf{x} = \mathbf{b}$. Sostituendo alla matrice A la sua fattorizzazione QR, infatti, si ottengono i due sistemi

$$\begin{cases} Q\mathbf{c} = \mathbf{b} \\ R\mathbf{x} = \mathbf{c}. \end{cases}$$

La soluzione del primo è data da $\mathbf{c} = Q^T\mathbf{b}$, mentre il secondo può essere risolto per sostituzione all'indietro. Fattorizzare risulta vantaggioso perché i due sistemi risultanti sono più facili da risolvere rispetto al sistema originale.

Un teorema importante sostiene che $k_2(A) = k_2(R)$, cioè la matrice R ha lo stesso numero di condizionamento rispetto alla norma-2 della matrice A. Ciò comporta che la fattorizzazione QR trasforma un sistema lineare in un sistema triangolare superiore ad esso equivalente e dotato dello stesso numero di condizionamento. Questa proprietà rende preferibile la fattorizzazione QR nella risoluzione di sistemi lineari malcondizionati.

Esistono diversi algoritmi per il calcolo della fattorizzazione.

1.3.1 Matrici elementari di Householder

Una matrice elementare di Householder reale è del tipo

$$H = I - 2\mathbf{w}\mathbf{w}^T, \ \mathbf{w} \in \mathbb{R}^n, ||\mathbf{w}|| = 1,$$

dove la norma utilizzata qui e nel seguito è quella euclidea. Osserviamo che $2\mathbf{w}\mathbf{w}^T$ è una matrice $n \times n$ di rango 1. La matrice H è simmetrica ed ortogonale.

Assegnato un vettore $\mathbf{x}$, vogliamo determinare $\mathbf{w}$ in modo tale che

$$H\mathbf{x} = k\mathbf{e}_1, \tag{4}$$

essendo $\mathbf{e}_1$ il primo versore della base canonica di $\mathbb{R}^n$ e $k \in \mathbb{R}$.

Distinguiamo tre fasi per la costruzione di H.

1. Dalla (4), essendo H ortogonale, segue che $||H\mathbf{x}|| = ||\mathbf{x}|| = |k|$, da cui $k = \pm\sigma$, con $\sigma = ||\mathbf{x}||$.

2. Applicando la definizione di H si ha

$$H\mathbf{x} = \mathbf{x} - 2\mathbf{w}\mathbf{w}^T\mathbf{x} = \mathbf{x} - 2(\mathbf{w}^T\mathbf{x})\mathbf{w} = k\mathbf{e}_1,$$

   che implica

$$\mathbf{w} = \frac{\mathbf{x} - k\mathbf{e}_1}{2\mathbf{w}^T\mathbf{x}} = \frac{\mathbf{x} - k\mathbf{e}_1}{||\mathbf{x} - k\mathbf{e}_1||}$$

   in quanto $\mathbf{w}$ ha norma unitaria. Restano alcune considerazioni numeriche.

3. Dalla relazione

$$\mathbf{x}^T H\mathbf{x} = \mathbf{x}^{\mathbf{x}} - 2(\mathbf{w}^T\mathbf{x})^2 = kx_1$$

   otteniamo

$$(\mathbf{w}^T\mathbf{x})^2 = \frac{\sigma^2 - kx_1}{2}.$$

   L'espressione trovata per $\mathbf{w}$ implica dunque che

$$||\mathbf{x} - k\mathbf{e}_1|| = 2\mathbf{w}^T\mathbf{x} = \sqrt{2(\sigma^2 - kx_1)}.$$

   Usando questa espressione riduciamo il carico computazionale per la normalizzazione di $\mathbf{w}$. Inoltre, per evitare pericoli di cancellazione, è opportuno scegliere il segno di k, finora arbitrario, in modo tale che $-kx_1$ sia positivo. Questo porta alle due espressioni

$$k = -\text{sign}(x_1)\sigma,\ ||\mathbf{x} - k\mathbf{e}_1|| = \sqrt{2\sigma(\sigma + |x_1|)},$$

   definendo sign(y) pari a $+1$ o -1 a seconda che sia $y \geq 0$ oppure $y < 0$.

1.3.2 Fattorizzazione QR di Householder

Le matrici elementari di Householder consentono di costruire la fattorizzazione QR di una matrice A. Supponiamo che A sia una matrice quadrata. Genereremo una successione di matrici $A^{(i)}, i = 1, \ldots, n$, tale che $A^{(n)}$ sia triangolare superiore.

Al **passo 1** dell'algoritmo, scriviamo la matrice A in termini delle sue colonne

$$A = A^{(1)} = \left[\mathbf{a}_1^{(1)} \quad \mathbf{a}_2^{(1)} \quad \ldots \quad \mathbf{a}_n^{(1)}\right].$$

Aplicando la tecnica illustrata nella sezione precedente è possibile costruire la matrice elementare di Householder H_1 tale che

$$H_1\mathbf{a}_1^{(1)} = k_1\mathbf{e}_1.$$

Applichiamo tale matrice a sinistra alla matrice $A^{(1)}$ per generare la prossima matrice della successione

$$A^{(2)} = H_1A^{(1)} = \left[\mathbf{a}_1^{(2)} \quad \mathbf{a}_2^{(2)} \quad \ldots \quad \mathbf{a}_n^{(2)}\right].$$

con

$$\mathbf{a}_1^{(2)} = k_1\mathbf{e}_1,\ \mathbf{a}_j^{(2)} = H_1\mathbf{a}_j^{(1)},\ j = 2,\dots,n.$$

La matrice $A^{(2)}$ ha la seguente struttura

$$A^{(2)} = \begin{bmatrix} k_1 & a_{12}^{(2)} & \cdots & a_{1n}^{(2)} \\ 0 & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & a_{n2}^{(2)} & \cdots & a_{nn}^{(2)} \end{bmatrix} = \begin{bmatrix} k_1 & \mathbf{v}_1^T \\ \mathbf{0} & \hat{A}^{(2)} \end{bmatrix},$$

essendo $\mathbf{0}$ un vettore colonna di dimensioni appropriate avente tutte le componenti nulle e $A^{(2)}$ una sottomatrice di dimensione $n-1$.

Al **passo 2** consideriamo inizialmente la sottomatrice

$$\hat{A}^{(2)} = \begin{bmatrix} \hat{\mathbf{a}}_2^{(2)} & \hat{\mathbf{a}}_3^{(2)} & \dots & \hat{\mathbf{a}}_n^{(2)} \end{bmatrix}.$$

Costruiamo, quindi, la matrice elementare di Householder $\hat{H}_2$ di dimensioni $n-1$ tale che

$$\hat{H}_2\hat{\mathbf{a}}_2^{(2)} = k_2\mathbf{e}_1,$$

dove il vettore e_1 ha questa volta lunghezza $n-1$. A questo punto, *orliamo* la matrice $\hat{H}_2$ con una riga e una colonna della matrice identità per farle raggiungere la dimensione n

$$H_2 = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & & & \\ \vdots & & \hat{H}_2 & \\ 0 & & & \end{bmatrix} = \begin{bmatrix} 1 & \mathbf{0}^T \\ \mathbf{0}^T & \hat{H}_2 \end{bmatrix}$$

e moltiplichiamo a sinistra questa matrice per $A^{(2)}$, ottenendo

$$A^{(3)} = H_2A^{(2)} = \begin{bmatrix} 1 & \mathbf{0}^T \\ \mathbf{0} & \hat{H}_2 \end{bmatrix}\begin{bmatrix} k_1 & \mathbf{v}_1^T \\ \mathbf{0} & \hat{A}_2^{(2)} \end{bmatrix} = \begin{bmatrix} k_1 & \mathbf{v}_1^T \\ \mathbf{0} & \hat{H}_2\hat{A}_2^{(2)} \end{bmatrix},$$

ovvero

$$\hat{A}^{(3)} = \begin{bmatrix} k_1 & * & \cdots & \cdots & * \\ 0 & k_2 & * & \cdots & * \\ \vdots & 0 & & & \\ \vdots & \vdots & & \hat{A}^{(3)} & \\ 0 & 0 & & & \end{bmatrix}.$$

Il passo successivo opererrà sulla sottomatrice $\hat{A}^{(3)}$ di dimensione $n-2$.

Prendiamo ora in esame il generico **i** dell'algoritmo. La matrice prodotta dall'iterazione precedente ha la forma

$$\hat{A}^{(i)} = \begin{bmatrix} k_1 & * & \cdots & \cdots & \cdots & * \\ 0 & \ddots & \ddots & & & \vdots \\ \vdots & \ddots & k_{i-1} & * & \cdots & * \\ \vdots & & 0 & & & \\ \vdots & & \vdots & & \hat{A}^{(i)} & \\ \vdots & 0 & 0 & & & \end{bmatrix} = \begin{bmatrix} A_{11}^{(1)} & A_{12}^{(1)} \\ 0 & \hat{A}^{(i)} \end{bmatrix},$$

dove gli asterischi indicano elementi della matrice che non verranno più modificati, mentre la sottomatrice $\hat{A}^{(i)}$ è del tipo

$$\hat{A}^{(i)} = \begin{bmatrix} \hat{\mathbf{a}}_i^{(i)} & \hat{\mathbf{a}}_{i+1}^{(i)} & \cdots & \hat{\mathbf{a}}_n^{(i)} \end{bmatrix}.$$

Creiamo ora una matrice $\hat{H}_i$ di dimensione $n - i + 1$ tale che

$$\hat{H}_i \hat{\mathbf{a}}_i^{(i)} = k_i \mathbf{e}_1.$$

Dopo aver orlato $\hat{H}_i$ con $i - 1$ righe e colonne dell'identità si avrà

$$A^{(i+1)} = H_i A^{(i)} = \begin{bmatrix} I_{i-1} & 0 \\ 0 & \hat{H}_i \end{bmatrix} \begin{bmatrix} A_{11}^{(i)} & A_{12}^{(i)} \\ 0 & \hat{A}^{(i)} \end{bmatrix} = \begin{bmatrix} A_{11}^{(i)} & A_{12}^{(i)} \\ 0 & \hat{H}_i \hat{A}^{(i)} \end{bmatrix},$$

ossia

$$\hat{A}^{(i+1)} = \begin{bmatrix} k_1 & * & \cdots & \cdots & \cdots & * \\ 0 & \ddots & \ddots & & & \vdots \\ \vdots & \ddots & k_i & * & \cdots & * \\ \vdots & & 0 & & & \\ \vdots & & \vdots & & \hat{A}^{(i+1)} & \\ \vdots & 0 & 0 & & & \end{bmatrix}.$$

Se la matrice A è quadrata, l'algoritmo termina al **passo n-1** con la matrice triangolare superiore

$$A^{(n)} = H_{n-1} A^{(n-1)} = \begin{bmatrix} k_1 & * & \cdots & * \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & * \\ 0 & \cdots & 0 & k_n \end{bmatrix} = R.$$

Possiamo, a questo punto, riformulare l'intero algoritmo in forma matriciale

$$R = A^{(n)} = H_{n-1} A^{(n-1)} = H_{n-2} H_{n-1} A^{(n-2)} = \cdots$$

$$H_{n-2}H_{n-1}\cdots H_1 A^{(1)} = Q^T A.$$

Poiché la matrice

$$Q = H_1 H_2 \cdots H_{n-1}$$

è ortogonale, in quanto prodotto di matrici ortogonali, la precedente relazione implica

$$A = QR,$$

che costituisce la **fattorizzazione QR** della matrice A. Se implementato in maniera ottimizzata, l'algoritmo richiede $O(\frac{2}{3}n^3)$ moltiplicazioni.

Se A è una matrice rettangolare $m \times n$ $(m > n)$, l'algoritmo di tringolarizzazione procede esattamente allo stesso modo mediante prodotti a sinistra per matrici elementari di Householder di dimensione m. L'unica differenza consiste nel fatto che è necessario un passo ulteriore per azzerare gli elementi dell'ultima colonna di A. Dopo n passi otterremo la matrice

$$R = A^{(n+1)} = \begin{bmatrix} k_1 & * & \cdots & * \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & * \\ \vdots & & \ddots & k_n \\ \vdots & & & 0 \\ \vdots & & & \vdots \\ 0 & \cdots & \cdots & 0 \end{bmatrix} = \begin{bmatrix} R_1 \\ 0 \end{bmatrix},$$

con R_1 matrice $n \times n$ triangolare superiore, non singolare se la matrice A è a rango pieno. Il legame tra R e A è

$$R = H_n H_{n-1} \cdots H_1 A,$$

da cui

$$A = QR, \text{ con } Q = H_1 H_2 \cdots H_n.$$

In questo caso l'algoritmo richiede $O(n^2(m - \frac{1}{3}n))$ moltiplicazioni.

1.3.3 Fattorizzazione QR di Givens

Nello spazio $\mathbb{R}^2$ una rotazione piana è rappresentata dalla matrice

$$G(\theta) = \begin{bmatrix} cos(\theta) & sin(\theta) \\ -sin(\theta) & cos(\theta) \end{bmatrix}.$$

Se moltiplicata per un vettore $\mathbf{x}$, tale matrice lo ruota di un angolo θ in senso orario.

La *matrice elementare di Givens* G_{ij} $(i < j)$ opera in $\mathbb{R}^m$ una rotazione che lascia invariato il sottospazio ortogonale al piano definito dai vettori $\mathbf{e}_i$ e $\mathbf{e}_j$ della base canonica. Essa è definita come

$$G_{ij} = \begin{bmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & c & \cdots & s & & \text{(riga i)} \\ & & \vdots & & \vdots & & \\ & & -s & \cdots & c & & \text{(riga j)} \\ & & & & & \ddots & \\ & & & & & & 1 \end{bmatrix}$$

con $c^2 + s^2 = 1$.

Osserviamo che per applicarla ad un vettore non è necessario eseguire esplicitamente il prodotto matriciale, in quanto

$$\mathbf{y} = G_{ij}\mathbf{x} \implies y_k = \begin{cases} cx_i + sx_j, & k = i, \\ -sx_i + cx_j, & k = j, \\ x_k, & k \neq i, j. \end{cases}$$

È possibile determinare i parametri c e s in modo tale da annullare la componente j-esima del vettore $\mathbf{x}$, modificando solo la componente i-esima e lasciando inalterate tutte le altre. Infatti, imponendo $y_j = 0$ si ottiene $c = \frac{x_i}{x_j}s$ che, sostituita nella relazione $c^2 + s^2 = 1$, fornisce

$$\left(\frac{x_i^2}{x_j^2} + 1\right)s^2 = 1$$

da cui discendono

$$s^2 = \frac{x_j^2}{x_i^2 + x_j^2};\ c^2 = \frac{x_i^2}{x_i^2 + x_j^2}.$$

Le espressioni di c e s sono quindi

$$c = \frac{x_i}{\sigma} \quad \text{e} \quad s = \frac{x_j}{\sigma}, \text{ con } \sigma = \sqrt{x_i^2 + x_j^2}.$$

Sono necessarie 6 operazioni in virgola mobile per il calcolo di c e s, e altre 6 per effettuare il prodotto $G_{ij}\mathbf{z}$ per un generico vettore $\mathbf{z}$. Operando in un sistema in virgola mobile, tuttavia, è indispensabile calcolare c e s in modo da evitare *underflows* e *overflows*, come mostrato nello snippet mostrato poco dopo.

Mediante una opportuna successione di matrici elementari di Givens è possibile trasformare un vettore di $\mathbb{R}^m$ in modo analogo all'azione di una matrice elementare di Householder. Infatti, ponendo

$$G = G_{1m} \cdots G_{13}G_{12},$$

▷ Calcolo sicuro di c e s.

1: **if** $x_j = 0$ **then**
2: $\quad c = 1,\ s = 0$
3: **else if** $|x_j| > |x_i|$ **then**
4: $\quad t = x_i/x_j,\ z = \sqrt{1+t^2}$
5: $\quad s = 1/z,\ c = ts$
6: **else**
7: $\quad t = x_j/x_i,\ z = \sqrt{1+t^2}$
8: $\quad c = 1/z,\ s = tc$

dove i parametri c e s di ciascuna matrice sono determinati mediante l'algoritmo visto sopra, si ha che

$$G\mathbf{x} = k\mathbf{e}_1.$$

L'eventuale vantaggio, rispetto all'approccio di Householder, sta nel fatto che se $\mathbf{x}$ ha molti elementi nulli possiamo operare selettivamente solo su quelli diversi da zero, riducendo così il numero delle matrici di Givens utilizzate e, conseguentemente, la complessità computazionale.

Se si moltiplica a sinistra la matrice G_{ij} per una matrice A, si appica la medesima trasformazione a tutte le colonne di A

$$B = G_{ij}A \implies b_{jk} = \begin{cases} ca_{ik} + sa_{jk}, & l = i,\ k = 1, \dots, n, \\ -sa_{ik} + ca_{jk}, & l = j,\ k = 1, \dots, n, \\ a_{lk}, & l \neq i, j,\ k = 1, \dots, n. \end{cases}$$

Se si moltiplica a destra G_{ij} per A, vengono trasformate in modo analogo tutte le righe di A.

L'*algoritmo di fattorizzazione QR di Givens* consiste nell'azzerare tutti gli elementi del triangolo inferiore di A a cominciare dal secondo elemento della prima colonna e proseguendo dapprima verso il basso con gli altri elementi della prima colonna, e poi con le colonne successive.

Se la matrice è quadrata ci si ferma con l'ultimo elemento della penultima colonna, e il procedimento può essere rappresentato in forma matriciale nel modo seguente

$$G_{n-1,n}G_{n-2,n}G_{n-2,n-1} \cdots G_{2n} \cdots G_{23}G_{1n} \cdots G_{13}G_{12}A = R.$$

Se invece la matrie è rettangolare con m righe e n colonne ($m > n$) è necessario azzerare anche gli elementi subidiagonali dell'ultima colonna, e l'algoritmo diventa

$$G_{n,n} \cdots G_{n,n+1} \cdots G_{2m} \cdots G_{23}G_{1m} \cdots G_{13}G_{12}A = R = \begin{bmatrix} R_1 \\ 0 \end{bmatrix},$$

con R_1 matrice triangolare superiore $n \times n$.

In entrambi i casi il prodotto delle matrici di Givens G_{ij} fornisce il fattore Q (trasposto) della fattorizzazione QR di A.

1.4 Problemi ai minimi quadrati

Nelle applicazioni si incontrano frequentemente sistemi lineari con un numero di equazioni differente dal numero delle incognite, cioè del tipo $A\mathbf{x} = \mathbf{b}$ con A matrice $m \times n$, $\mathbf{b} \in \mathbb{R}^m$ e $\mathbf{x} \in \mathbb{R}^n$. Ipotizzando che A sia a rango pieno, si possono presentare due casi:

1. $m > n$: sono disponibili più equazioni che incognite, per cui il problema potrebbe non avere soluzioni: il sistema si dice *sovradeterminato*;

2. $m < n$: vi sono più incognite che equazioni: il sistema potrebbe avere infinite soluzioni, e viene dette *sottodeterminato*.

Il caso a rango non pieno, incluso quello in cui la matrice A è quadrata e singolare, può essere ricondotto ad uno dei due casi precedenti.

Tutti questi problemi sono mal posti e non hanno quindi una soluzione in senso classico. Nel primo caso si sta studiando un fenomeno che dipende da un certo numero di variabili, ma il numero di equazioni lineari a disposizione per descrivere il fenomeno è superiore al numero delle incognite. Possiamo avere il caso ideale di dati esatti in cui $m-n$ equazioni sono linearmente dipendenti dalle altre, ma potrebbe non essere noto quali. Oppure i dati sono affetti da errore e può capitare che nessuna delle equazioni sia verificata esattamente, sarebbe quindi preferibile utilizzarne un numero sovrabbondante per cercare di ripulire la soluzione dagli errori sperimentali.

Essendo il problema mal posto, è necessario riformularlo in uno ben posto. Se tutte le equazioni non possono essere verificate contemporaneamente, è ragionevole richiedere che lo scarto quadratico medio (varianza) tra il primo e il secondo membro del sistema sia minimo. Si passa così ad un **problema ai minimi quadrati** (*least squares problem*). La condizione di varianza minima si può esprimere nella forma

$$\min_{x \in \mathbb{R}^n} ||A\mathbf{x} - \mathbf{b}||_2. \tag{5}$$

Se il minimo risulta essere zero, vuol dire che il sistema originale ammette una soluzione in senso classico. In caso contrario si ottiene la *soluzione nel senso dei minimi quadrati* del sistema lineare sovradeterminato $A\mathbf{x} = \mathbf{b}$.

1.4.1 Il metodo delle equazioni normali

Equazioni normali del primo tipo.

Esistono varie tecniche per risolvere un problema ai minimi quadrati. La più classica è quella che opera risolvendo il *sistema normale* associato al problema.

Richiamiamo qua le formule per il calcolo del gradiente

$$\nabla f(\mathbf{x}) = \left(\frac{\delta f}{\delta x_1}, \ldots, \frac{\delta f}{\delta x_n} \right)$$

di alcune funzioni da $\mathbb{R}^n$ in $\mathbb{R}$ espresse in forma matriciale.

Se $f(\mathbf{x}) = \mathbf{b}^T\mathbf{x}$, con $\mathbf{x}, \mathbf{b} \in \mathbb{R}^n$ (o, equivalentemente, se $f(\mathbf{x}) = \mathbf{x}^T\mathbf{b}$) è facile verificare che

$$\nabla(\mathbf{b}^T\mathbf{x}) = \mathbf{b}.$$

In modo analogo si può mostrare che se A è una matrice $n \times n$

$$\nabla(\mathbf{x}^T A\mathbf{x}) = A\mathbf{x} + A^T\mathbf{x}.$$

Da quest'ultima formula segue che se A è simmetrica si ha

$$\nabla(\mathbf{x}^T A\mathbf{x}) = 2A\mathbf{x}.$$

Dal momento che vogliamo minimizzare una quantità non negativa, non è restrittivo considerare il quadrato della norma del residuo, che esprimiamo in forma matriciale

$$\begin{aligned} ||A\mathbf{x} - \mathbf{b}||^2 &= (A\mathbf{x} - \mathbf{b})^T A\mathbf{x} - \mathbf{b} \\ &= \mathbf{x}^T A^T A\mathbf{x} - \mathbf{x}^T A^T\mathbf{b} - \mathbf{b}^T A\mathbf{x} + \mathbf{b}^T\mathbf{b}. \\ &= \mathbf{x}^T A^T A\mathbf{x} - 2\mathbf{x}^T A^T\mathbf{b} + \mathbf{b}^T\mathbf{b} \end{aligned}$$

Per minimizzare la norma euclidea del residuo imponiamo l'annullarsi del gradiente

$$\frac{1}{2}\nabla(||A\mathbf{x} - \mathbf{b}||^2) = A^T A\mathbf{x} - A^T\mathbf{b} = 0,$$

giungendo così al *sistema normale*

$$A^T A\mathbf{x} = A^T\mathbf{b}.$$

Se A è a rango pieno, la matrice $A^T A \in \mathbb{R}^{n\times n}$ è invertibile, e la soluzione del sistema lineare è unica

$$\mathbf{x}_{LS} = (A^T A)^{-1} A^T\mathbf{b}.$$

La matrice

$$A^\dagger = (A^T A)^{-1} A^T$$

viene detta *pseudo-inversa* di A. Essa è un'inversa sinistra di A, ma non un'inversa destra.

Nel caso a rango pieno la matrice $A^T A$ è simmetrica definita positiva e per risolvere il sistema normale si può fare ricorso alla fattorizzazione di Cholesky (sez. 1.4.2). Calcolata la fattorizzazione $A^T A = R^T R$, con R matrice triangolare superiore, il vettore $\mathbf{x}_{LS}$ può essere calcolato risolvendo in successione i due sistemi triangolari

$$\begin{cases} R^T\mathbf{y} = A^T\mathbf{b} \\ R\mathbf{x} = \mathbf{y}. \end{cases}$$

Il costo computazionale di questo metodo dipende essenzialmente dal calcolo del prodotto $A^T A$ e della fattorizzaione di Cholesky, ed è pari a $O(\frac{1}{2}n^2(m+\frac{1}{3}n))$

moltiplicazioni. Un possibile svantaggio di questo approccio riguarda la stabilità, dal momento che la matrice A^TA ha un numero di condizionamento pari al quadrato di quello di A.

Equazioni normali del secondo tipo.
Nel caso di sistema $A\mathbf{x} = \mathbf{b}$ sottodeterminato, studiamo il problema *duale*

$$A^T\mathbf{y} = \mathbf{c},$$

in modo che le dimensioni di A siano ugualmente $m \times n,\ m > n$. Il sistema ammette infinite soluzioni, che diventa unica se prendiamo quella di norma minima:

$$\begin{cases} \min ||\mathbf{y}||_2 \\ A^T\mathbf{y} = \mathbf{c}. \end{cases}$$

Per risolvere il problema, ci serviamo del metodo dei moltiplicatori lagrangiani. Vogliamo minimizzare la norma di $\mathbf{y}$ sotto il vincolo del sistema $A^T\mathbf{y} = \mathbf{c}$, espresso come $\mathbf{c} - A^T\mathbf{y} = 0$. La funzione lagrangiana è quindi scritta così

$$\begin{aligned} L(\mathbf{y}, \mathbf{z}) &= \frac{1}{2}||\mathbf{y}||^2 + \sum_{i=1}^{n} z_i(c_i - (A^T\mathbf{y})_i) \\ &= \frac{1}{2}\mathbf{y}^T\mathbf{y} + \mathbf{z}^T(\mathbf{c} - A^T\mathbf{y}) \\ &= \frac{1}{2}\mathbf{y}^T\mathbf{y} + \mathbf{z}^T\mathbf{c} - \mathbf{z}^TA^T\mathbf{y}, \end{aligned}$$

dove $\mathbf{y} \in \mathbb{R}^m$ è il vettore della soluzione, $\mathbf{z} \in \mathbb{R}^n$ è il vettore dei moltiplicatori, e $c_i - (A^T\mathbf{y})_i$ sono i vincoli.

Non ci resta che imporre il gradiente a zero

$$\frac{\delta L}{\delta y} = \mathbf{y} - A\mathbf{z} = 0$$

$$\frac{\delta L}{\delta z} = \mathbf{c} - A^T\mathbf{y} = 0,$$

da cui ricaviamo il sistema

$$\begin{cases} \mathbf{y} = A\mathbf{z} \\ A^T\mathbf{y} = \mathbf{c} \end{cases} \quad \begin{cases} \mathbf{y} = A\mathbf{z} \\ A^TA\mathbf{z} = \mathbf{c}, \end{cases}$$

dove $\mathbf{y}$ è la soluzione.

1.4.2 Fattorizzazione di Cholesky

Nel caso di matrice C simmetrica definita positiva, essa si può fattorizzare

$$C = R^TR,$$

con R matrice triangolare superiore. Questo genere di fattorizzazione si può applicare al caso delle equazioni normali del primo tipo se prendiamo $C = A^TA$, visto che è sempre simmetrica definita positiva. Il sistema $A^TA\mathbf{x} = A^T\mathbf{b}$ diventa, una volta fattorizzato $A^TA = R^TR$, così:

$$R^TR\mathbf{x} = A^T\mathbf{b},$$

da cui, chiamando $R\mathbf{x} = \mathbf{z}$, otteniamo

$$\begin{cases} R^TR\mathbf{z} = A^T\mathbf{b} \\ R\mathbf{x} = \mathbf{z}. \end{cases}$$

Per ottenere la fattorizzazione di Cholesky si prosegue nel seguente modo. Sia $R^TR = A$, con r_{ij} elementi di riga i e colonna j della matrice R. Abbiamo che

$$\begin{bmatrix} r_{11} & & & & & & \\ r_{12} & r_{22} & & & & & \\ r_{13} & r_{23} & r_{33} & & & & \\ \vdots & \vdots & \vdots & \ddots & & & \\ r_{1i} & r_{2i} & r_{3i} & \cdots & r_{ii} & & \\ \vdots & & & & & \ddots & \\ r_{1n} & r_{2n} & & & \cdots & & r_{nn} \end{bmatrix} \times \begin{bmatrix} r_{11} & r_{12} & r_{13} & \cdots & r_{1j} & \cdots & r_{1n} \\ & r_{22} & r_{23} & \cdots & r_{2j} & \cdots & r_{2n} \\ & & r_{33} & \cdots & r_{3j} & \cdots & r_{3n} \\ & & & \ddots & & & \vdots \\ & & & & r_{jj} & & \vdots \\ & & & & & \ddots & \\ & & & & & & r_{nn} \end{bmatrix}$$

$$= \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1j} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2j} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3j} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ a_{i1} & a_{i2} & a_{i3} & \cdots & a_{ij} & \cdots & a_{in} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nj} & \cdots & a_{nn} \end{bmatrix}$$

In base al prodotto, possiamo osservare che

$$a_{11} = r_{11}^2;\ r_{11} = \sqrt{a_{11}},$$

ottenendo così la prima colonna di R. Per la seconda invece

$$a_{12} = r_{11}r_{12};\ r_{12} = \frac{1}{r_{11}}a_{12},$$

$$a_{22} = r_{12}^2 + r_{22}^2;\ r_{22} = \sqrt{a_{22} - r_{12}^2}.$$

Mentre per quanto riguarda la terza colonna

$$a_{13} = r_{11}r_{13};\ r_{13} = \frac{1}{r_{11}}a_{13},$$

$$a_{23} = r_{12}r_{13} + r_{22}r_{23};\ r_{23} = \frac{1}{r_{22}}(a_{23} - r_{12}r_{13}),$$
$$a_{33} = r_{13}^2 + r_{23}^2 + r_{33}^2;\ r_{33} = \sqrt{a_{13} - r_{13}^2 - r_{23}^2}.$$

Osserviamo come non appena troviamo un nuovo elemento r, lo usiamo per trovare il successivo.

In generale, gli elementi sopra-diagonale, ovvero quando $i < j$:

$$\begin{aligned} a_{ij} &= r_{1i}r_{1j} + r_{2i}r_{2j} + \cdots + r_{ii}r_{ij} \\ &= \sum_{k=1}^{i-1} r_{ki}r_{kj} + r_{ii}r_{ij}, \end{aligned}$$

da cui ricaviamo

$$r_{ij} = \frac{1}{r_{ii}}\left(a_{ij} - \sum_{k=1}^{i-1} r_{ki}r_{kj}\right)$$

Mentre per gli elementi in diagonale, ovvero quando $i = j$:

$$\begin{aligned} a_{jj} &= a_{1j}^2 + a_{2j}^2 + \cdots + a_{jj}^2 \\ &= \sum_{k=1}^{j-1} r_{kj}^2 + r_{jj}, \end{aligned}$$

da cui

$$r_{jj} = \sqrt{a_{jj} - \sum_{k=1}^{j-1} r_{kj}^2}.$$

La complessità computazionale del metodo è $O(\frac{1}{2}mn^2 + \frac{1}{6}n^3)$.

1.4.3 Risoluzione mediante la fattorizzazione QR

LA fattorizzazione QR forinsce un metodo alternativo, stabile ed efficiente, per calcolare la soluzione del problema (5). Anche in questo caso è conveniente considerare il quadrato della norma del residuo. Sostituendo al suo interno la fattorizzazione $A = QR$ otteniamo

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

ponendo $\mathbf{c} = Q^T\mathbf{b}$ e ricordando che una matrice ortogonale non modifica la norma-2 di un vettore.

La matrice R ha la forma

$$\begin{bmatrix} R_1 \\ 0 \end{bmatrix},$$

con R_1 triangolare superiore, quadrata e, se A è rango pieno, non singolare. Partizionando in modo coerente anche il vettore $\mathbf{c}$

$$\mathbf{c} = \begin{bmatrix} \mathbf{c}_1 \\ \mathbf{c}_2 \end{bmatrix},\ \mathbf{c}_1 \in \mathbb{R}^n,\ \mathbf{c}_2 \in \mathbb{R}^{m-n},$$

otteniamo

$$||A\mathbf{x} - \mathbf{b}||^2 = ||R\mathbf{x} - \mathbf{c}||^2 = \left|\left| \begin{bmatrix} R_1\mathbf{x} \\ 0 \end{bmatrix} - \begin{bmatrix} \mathbf{c}_1 \\ \mathbf{c}_2 \end{bmatrix} \right|\right|^2 = ||R_1\mathbf{x} - \mathbf{c}_1||^2 + ||\mathbf{c}_2||^2.$$

Se $\det(R_1) \neq 0$, il sistema

$$R_1\mathbf{x} = \mathbf{c}_1$$

ammette una e una sola soluzione, per la quale si ha $||R_1\mathbf{x} - \mathbf{c}_1||^2 = 0$. In corrispondenza a tale soluzione

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

perciò l'eventuale annullarsi del vettore $\mathbf{c}_2$ significa che $\mathbf{x}$ è la soluzione classica del sistema $A\mathbf{x} - \mathbf{b}$. In caso contrario, essa è soluzione nel senso dei minimi quadrati, e la norma di $\mathbf{c}_2$ fornisce la misura del residuo.

Rispetto all'approccio basato sulle equazioni normali questo metodo ha il vantaggio di operare direttamente sulla matrice A, il cui condizionamento è pari alla radice quadrata di quello di A^TA. La complessità computazionale coincide con quella della fattorizzazione QR. Qualora si utilizzasse l'algoritmo di Householder, per esempio, il metodo richiederebbe $O(n^2(m - \frac{1}{3}n))$ moltiplicazioni.

2 Testing

Analizzo e confronto gli algoritmi per la risoluzione di sistemi lineari scritti da me (alcuni in laboratorio, altri per conto mio) con le versioni di MATLAB , distinguendo diversi casi e ordini di grandezza. Sistema dalla forma $A\mathbf{x} - \mathbf{b}$ dove le matrici dei coefficienti sono a rango pieno.

L'approccio è il seguente: creo una matrice dei coefficienti casuale, fisso la soluzione (vettore di uni) e produco il vettore dei termini noti. In questo modo posso controllare di quanto differiscono la soluzione vera con quella calcolata. Testo quindi l'errore relativo tra le due soluzioni ($||x - sol||/||sol||$), i tempi di risoluzione del sistema, gli errori di fattorizzazione ed ortogonalizzazione, comparando il tutto al condizionamento del sistema.

Gli algoritmi scritti da me sono la fattorizzazione QR (triangolarizzazione con Householder o con Givens) e la fattorizzazione di Cholesky. La fattorizzazione QR è implementata nella funzione `myqr` a cui si possono passare due parametri: il primo è la matrice dei coefficienti, mentre il secondo è il tipo dell'algoritmo usato per la triangolarizzazione, che può essere: `householder` per usare l'algoritmo "semplice", ovvero quello che prevede la moltiplicazione tra matrici; `householder-light` che è come il precedente ma si evitano prodotti matriciali e si sfrutta la natura della matrice H; `givens` dove anche qua si fattorizza creando le matrici elementari e poi moltiplicando; e infine `givens-light` dove si usano in modo appropriato i coefficienti c ed s di Givens e si evitano i prodotti matriciali. Ho inoltre implementato la fattorizzazione di Cholesky nella funzione `mychol` come è stata spiegata a lezione e riportato qua.

Tutti i test sono presenti nel file `tesina.m` e sono da lanciare uno alla volta per poter osservare i singoli risultati. Qua presento i grafici tratti dai dati che per enormità non riporto qua ma sono visionabili attraverso i file di nome `test_i_j.mat` (presenti nel repository [1]) dove gli indici rispettano le stesse numerazioni usate nel codice.

2.1 Parte 1: testing su sistema quadrato

La prima parte del test riguarda le matrici quadrata di ordine n. Il range di n varia in base agli algoritmi usati, con quelli più efficienti posso permettermi di usare ordini più grandi.

2.1.1 **Test** 1.1

Confronto gli algoritmi: `householder`, `householder-light`, `givens`, `givens-light`; sono anche presenti `qr` e `backslash` di MATLAB , con $n = 50 : 10 : 200$. Grafici in fig. 1.

Ad una prima analisi si nota come ci sia qualche picco nel condizionamento che porta ad un aumentare degli errori, ma non affligge particolarmente i tempi di calcolo. Si nota poi che l'algoritmo migliore in termini di errori relativi e tempo di calcolo è il `backslash` di MATLAB . Givens risulta il peggiore sia in

termini di tempi di calcolo (c'era da aspettarselo) che come errori di fattorizzazione. Per quanto riguarda gli errori di fattorizzazione, il `qr` di MATLAB vince sui miei.

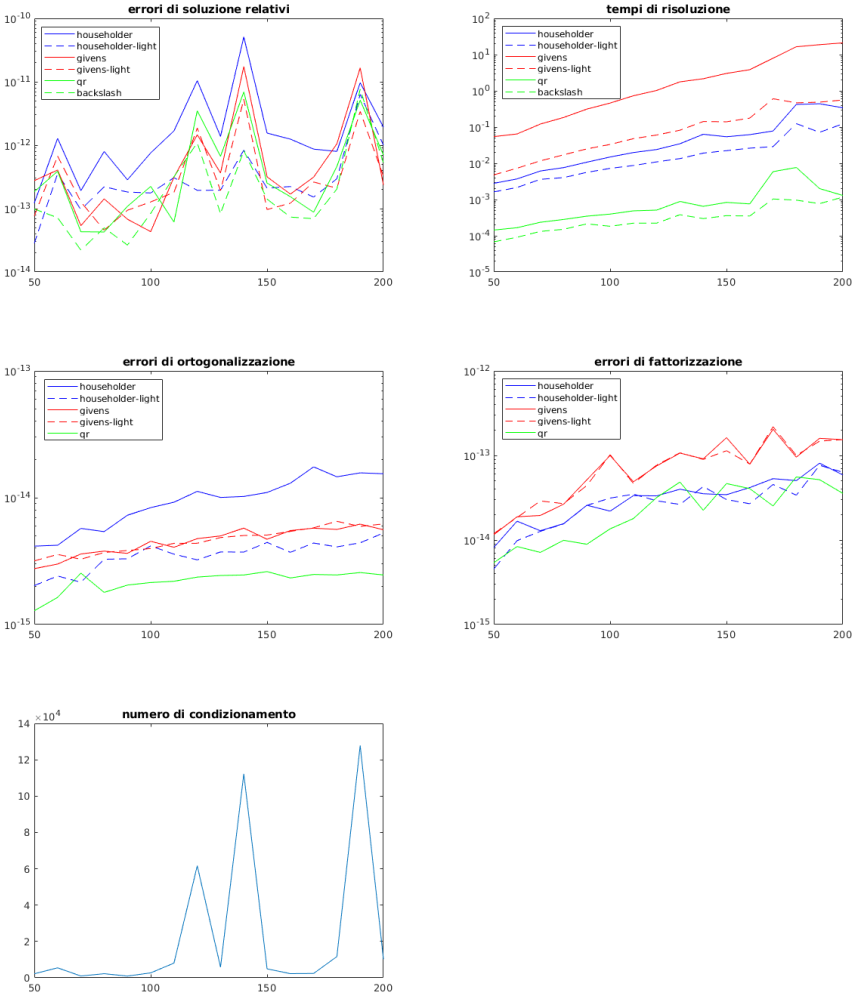

Figura 1: Test 1.1. Sistema quadrato, test di `householder`, `householder-light`, `givens`, `givens-light`, `qr`, `backslash`. $n = 50 : 10 : 200$.

2.1.2 Test 1.2

Visti i tempi più alti per Householder e Givens, uso solo le loro versioni light. Così posso usare n più grandi. $n = 50 : 50 : 500$.

Come si vede nei grafici in fig. 2, gli algoritmi di MATLAB sono i migliori. Scarse prestazioni da parte di `givens-light`, dovute al grande numero di ope-

razioni. Si nota inoltre una crescita più rapida dei tempi/errori al crescere di n, specialmente per i miei algoritmi. Errori fortemente condizionati dal numero di condizionamento, come si evince dalla simile forma delle curve.

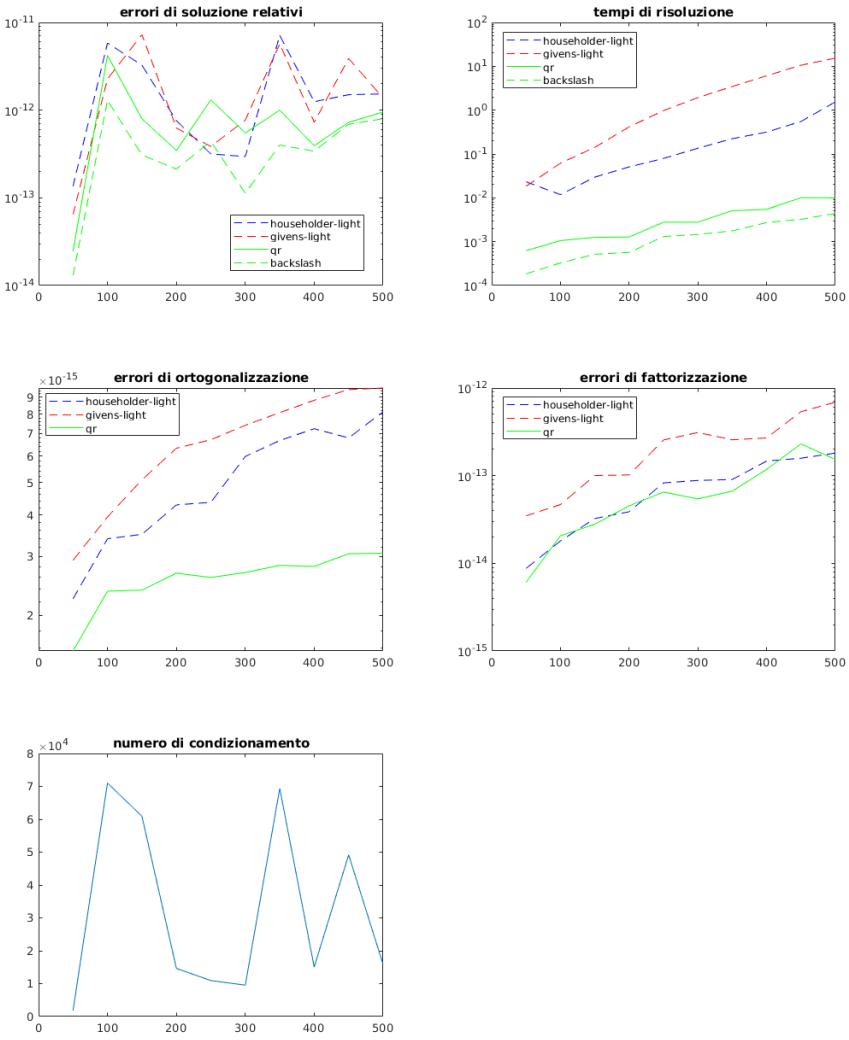

Figura 2: Test 1.2. Sistema quadrato, test di `householder-light` `givens-light`, `qr`, `backslash`. $n = 50 : 50 : 500$.

2.1.3 Test 1.3

Matrice triangolare superiore con sotto-diagonali in più. Dimensioni di A fissate a 100×100, cambia il numero delle sottodiagonali, pari a $n = 4 : 2 : 20$.

Confrontro tra `householder-light`, `givens-light` e `qr` di MATLAB . Grafici in fig. 3.

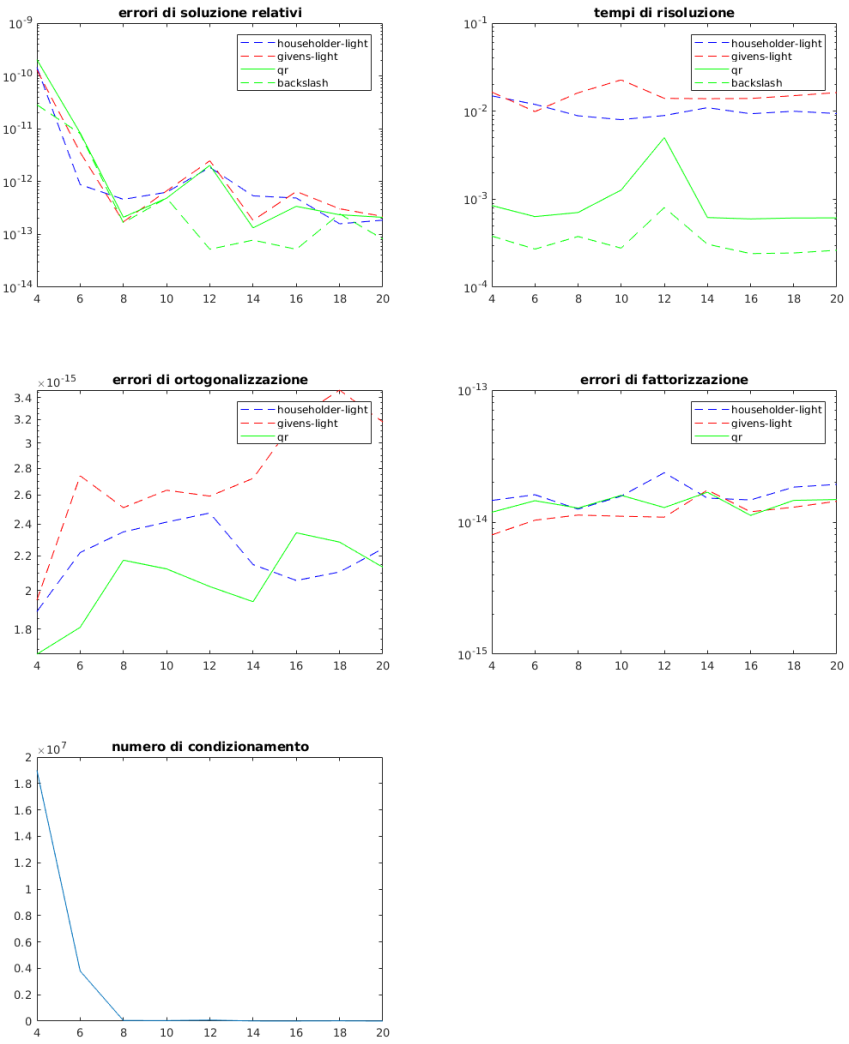

Figura 3: Test 1.3. Sistema quadrato quasi triangolare, test sulle sotto-diagonali. Test di `householder-light` `givens-light`, `qr`, `backslash`. Numero di sottodiagonali $n = 4 : 2 : 20$.

Nonostante mi aspettassi che `givens-light` funzionasse meglio, visto che se un elemento è già zero non calcolo nulla, ciò non accade ed inoltre non si notano particolari differenze negli errori relativi, che scendono man mano che il numero delle sotto-diagonali aumenta. Diversi invece i tempi dove ancora gli algoritmi di MATLAB vincono.

Funziona male per matrici quasi triangolari, perché mi dà l'errore: *Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND = 6.550039e-22.* quindi parto con 4 sottodiagonali. Infatti il numero di condizionamento decresce rapidamente rispetto al valore iniziale.

2.2 Parte 2: testing su sistema sovradimensionato

In questa parte studio i sistemi la cui matrice dei coefficienti A è dalla forma $m \times n,\ m > n = rank(A)$. Questo genere di sistemi ha più equazioni che incognite e potrebbe non ammettere soluzione. Quello che si fa è cercare la soluzione più vicina mediante l'utilizzo delle equazioni normali del primo tipo, giungendo così al *sistema normale*

$$A^T A\mathbf{x} = A^T\mathbf{b}.$$

2.2.1 Test 2.1

In questo test utilizzo la fattorizzazione di Cholesky per fattorizzare $A^T A$. Uso sia il mio `mychol` che il `chol` di MATLAB , che confronto poi con la pseudoinversa sinistra tramite la funzione `pinv` e la risoluzione con il backslash. Il numero m di righe è fissato a 110, mentre il numero n di colonne varia con $n = 10 : 10 : 100$. Grafici in fig. 4.

Le matrici usate sono relativamente piccole perché ricordiamoci che il condizionamento del sistema risulta il quadrato del condizionamento di A, quindi al crescere di n si introducono errori numerici che fanno sì che MATLAB non veda più la matrice come simmetrica definita positiva, un requisito della fattorizzazione di Cholesky che se manca non ci permette di usarla.

2.2.2 Test 2.2

Un secondo approccio è usare la fattorizzazione QR, dove uso la mia versione confrontandola con quella di MATLAB e con il `backslash`. Il numero di righe m è fissato a 500 mentre varia il numero n di colonne $n = 300 : 25 : 475$. Grafici in fig. 5.

È abbastanza chiaro come MATLAB performi decisamente meglio. Inoltre possimo notare che usa proprio la fattorizzazione QR per risolvere con il backslash, visto che le due curve si sovrappongono.

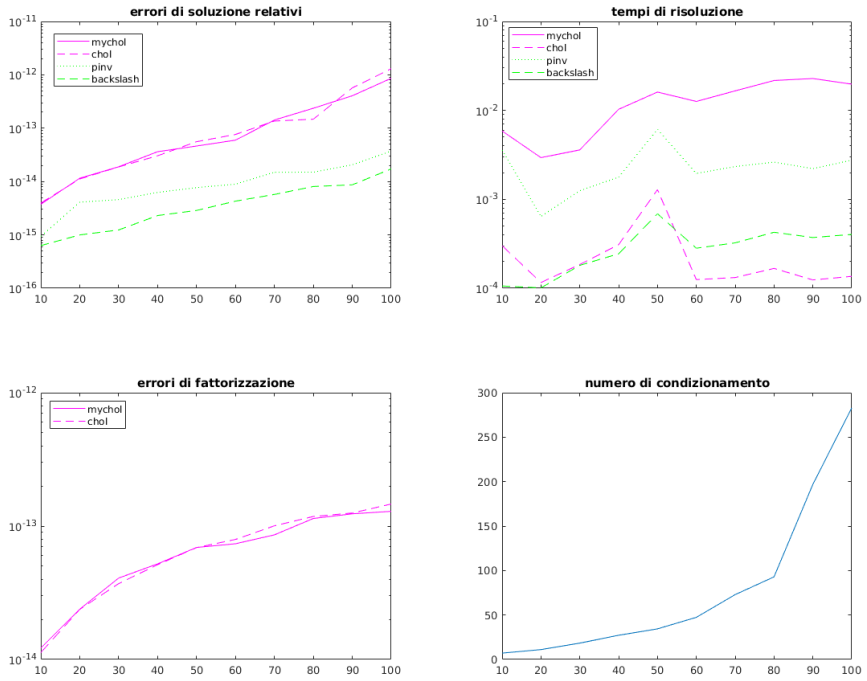

Figura 4: Test 2.1. Sistema sovradeterminato, test sulle equazioni normali e confronto con algoritmi di MATLAB . Test di `mychol` `chol`, `pinv`, `backslash`. Righe m pari a 110, mentre il numero di colonne varia $n = 10 : 10 : 100$.

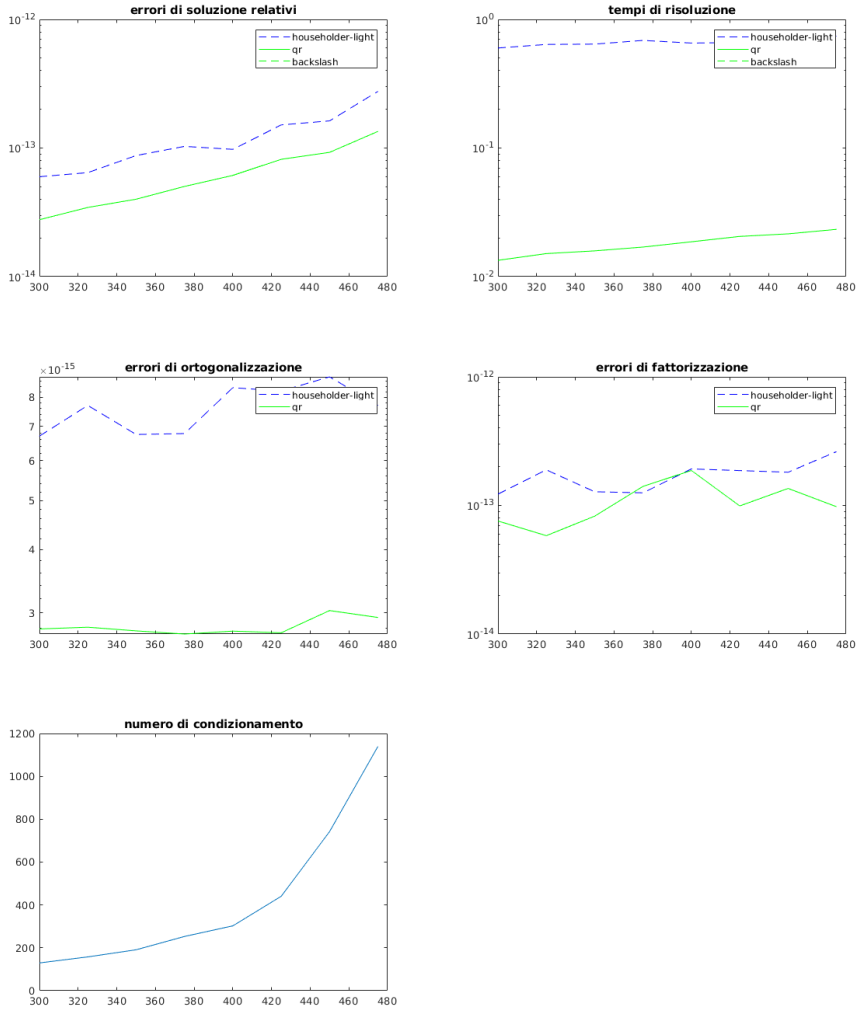

Figura 5: Test 2.2. Sistema sovradeterminato, test su QR mio, di MATLAB e risoluzione con `backslash`. Righe m pari a 500, mentre il numero di colonne varia $n = 300 : 25 : 475$.

2.3 Parte 3: Testing su sistema sottodimensionato

In questi test tratto sistemi dove il numero delle incognite è maggiore del numero del numero delle equazioni. Ciò comporta un numero infinito di soluzioni, che divetano una se si prende quella di minima norma. Sia A una matrice $m \times n$, $m > n$, il sistema sottodimensionato è visto come $A^T\mathbf{y} = \mathbf{c}$. Per la risoluzione si usano le equazioni normali di secondo tipo, da cui ricaviamo il sistema

$$\begin{cases} \mathbf{y} = A\mathbf{z} \\ A^T\mathbf{y} = \mathbf{c} \end{cases} \quad \begin{cases} \mathbf{y} = A\mathbf{z} \\ A^TA\mathbf{z} = \mathbf{c}, \end{cases}$$

dove $\mathbf{y}$ è la soluzione.

2.3.1 Test 3.1

Come primo approccio fattorizzo la matrice A^TA tramite Cholesky, confrontando la mia versione con quella di MATLAB e l'utilizzo della pseudoinversa destra. Anche qua le dimensioni della matrice non possono crescere tanto, per via del condizionamento che è il quadrato di quello della matrice A. Numero di colonne fissato a 120 mentre le righe variano con $30 : 10 : 110$. In questo caso ho voluto usare come vettore di soluzioni un vettore di tutti 20, che comunque non cambia le prestazioni degli algoritmi. Grafici in fig. 6.

Per quanto riguarda gli errori relativi, la fattorizzazione di Cholesky si comporta decisamente meglio del semplice `backslash`, in particolare la versione di MATLAB che lo risolve in un tempo minore. Si evince come sia utile cercare una soluzione particolare piuttosto che una qualsiasi. Cholesky e pseudoinversa si comportano allo stesso modo.

2.3.2 Test 3.2

In questo secondo test fattorizzo con QR e uso matrici di dimensione superiore. Numero di colonne fissato a 510 mentre le righe variano con $300 : 20 : 500$.

Anche qua si nota come la ricerca di una soluzione particolare, ovvero quella di minima norma, dia dei risultati più accurati rispetto ad una soluzione qualsiasi trovata dal `backslash`.

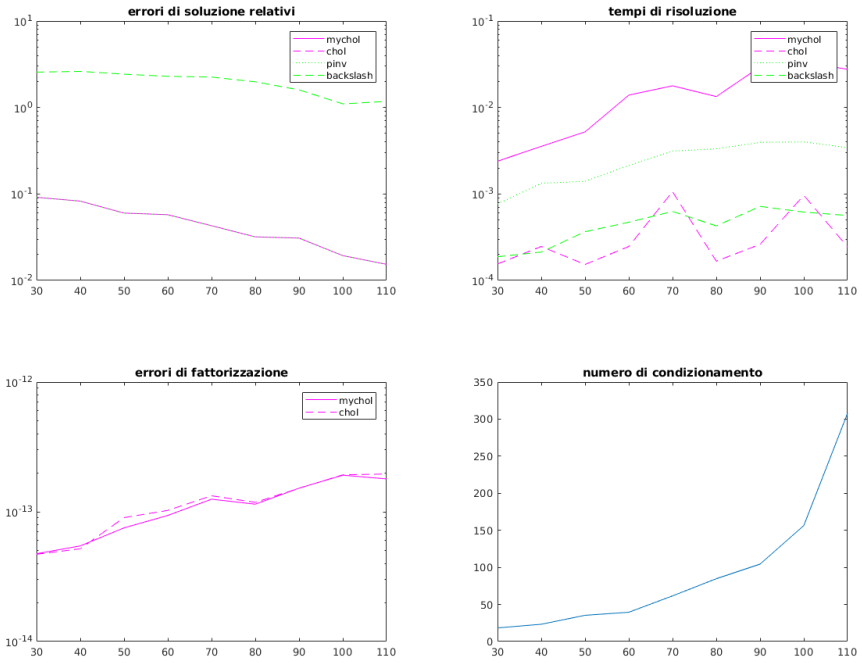

Figura 6: Test 3.1. Sistema sottodeterminato. Test tra le versioni di Cholesky mia e di MATLAB , con la pseudoinversa destra e con il backslash. Numero di colonne fissato a 120 mentre le righe variano con $30 : 10 : 110$.

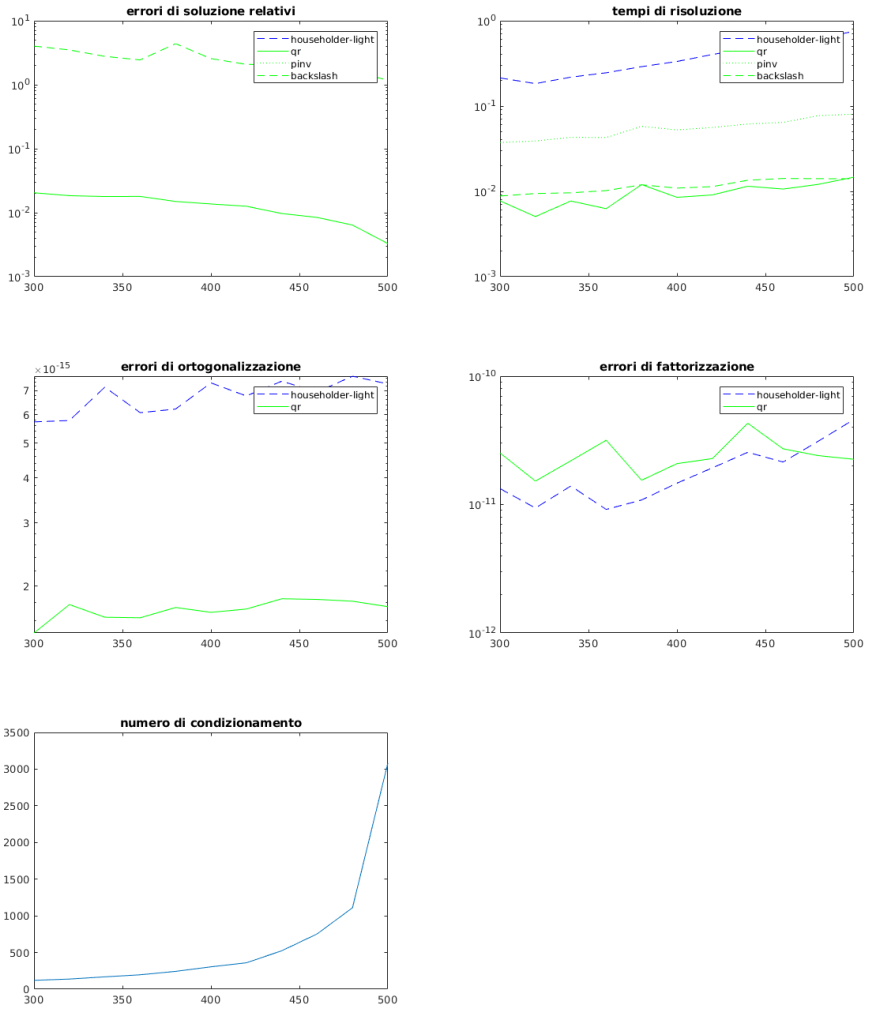

Figura 7: Test 3.2. Sistema sottodeterminato. Test tra le versioni di qr mia e di MATLAB , con la pseudoinversa destra e con il backslash. Numero di colonne fissato a 510 mentre le righe variano con $300 : 20 : 500$.

2.4 Parte 4: testing su sistema mal condizionato

Per studiare gli algoritmi in caso di sistema mal condizionato, uso come matrice dei coefficienti una matrice di Hilbert, dalla forma:

$$H = \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \cdots \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \\ \vdots & & & \ddots \end{bmatrix}$$

che è notoriamente mal condizionata.

2.4.1 Test 4.1

Confronto la fattorizzazione di Cholesky con la QR, aspettandomi dalla prima dei pessimi risultati al crescere del numero delle dimensioni della matrice. Matrici quadrate di ordine $n = 3 : 1 : 9$. Grafici in fig. 8.

Notiamo una simpaticissima crescita esponenziale da parte del numero di condizionamento, che porta a Cholesky ad avere risultati che differiscono fino a 10 volte dalla soluzione. QR e backslash si comportano meglio, e si nota come QR sia appunto quello da usare nel caso di sistemi mal condizionati.

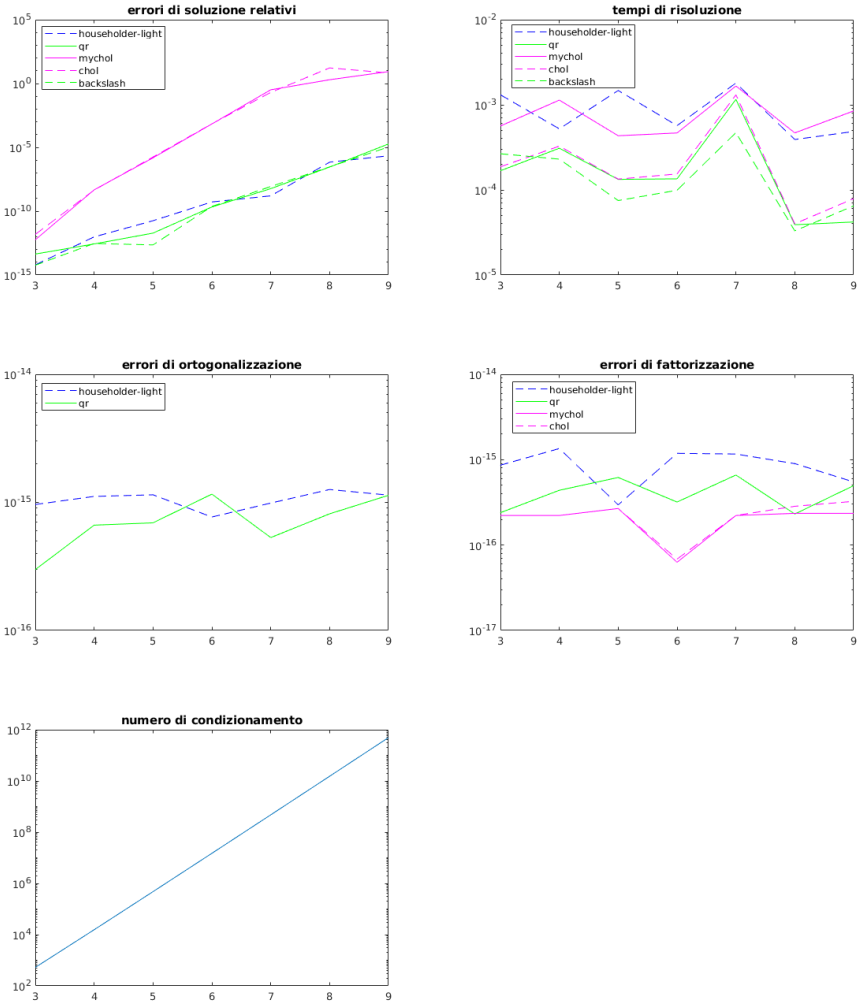

Figura 8: Test 4.1. Sistema malcondizionato. Test tra le versioni di qr mia e di MATLAB , Cholesky con il backslash. Matrici quadrate di ordine $n = 3 : 1 : 9$.

3 Conclusione

Con questo progetto ritengo di avere acquisito una buona conoscenza per quanto riguarda gli algoritmi visti. Si è potuto notare il loro comportamento, che si attestava con quanto già si sapeva dalla teoria.

Gli algoritmi di MATLAB sono in generale migliori; anche se un caso particolare è quello del sistema sottodeterminato dove un semplice utilizzo dell'operatore `backslash` non è stato in grado di dare dei buoni risultati.

Riferimenti bibliografici

[1] Carlo Cabras. Repository github https://github.com/carlocabras21/lss-testing.

[2] G. Rodriguez. *Algoritmi Numerici*. Pitagora Editrice Bologna, 2008.