Metodi diretti per sistemi lineari: implementazione al calcolatore e confronto prestazionale.

Facoltà d'Ingegneria Elettronica dell'Università di Cagliari
Calcolo Numerico 2
Docente Prof.Giuseppe Rodriguez <rodriguez@unica.it>
Studente Riccardo Colamatteo <rcolamatteo@tiscali.it>

Indice

Capitolo 1

Generalità.

1.1 Problemi lineare e loro risoluzione.

Un'applicazione f è detta lineare se additiva ed omogenea, ossia se

$$f(a+b) = f(a) + f(b)$$

ed anche

$$f(kx) = k * f(x);$$

da quest'ultima proprietà abbiamo anche la continuità in quanto $\forall x$ abbiamo

$$\begin{aligned}\lim_{x\to x_0} f(x) &= \lim_{x\to x_0} f\left(\frac{x}{x_0} * x_0\right) \\ &= \lim_{x\to x_0}\left[\frac{x}{x_0} * f(x_0)\right] \\ &= \left(\lim_{x\to x_0}\frac{x}{x_0}\right) * f(x_0) = f(x_0).\end{aligned}$$

Definiamo ben posto ciascun problema dotato d'una unica soluzione data in continuità con i dati.
Un problema lineare, per quanto visto, è quindi ben posto semplicemente quando ammette una ed una sola soluzione; sappiamo che ciò accade quando il sistema d'equazioni che esprime il problema dato ammette tante equazioni indipendenti quante sono le variabili del problema.

La composizione di più applicazioni lineari origina una nuova applicazione lineare; per risolvere quindi un generico sistema lineare possiamo:

- Lasciare invariato il sistema e considerare applicazioni che si annullano per i valori soluzione; ci stiamo qui riferendo alle applicazioni (multi) lineari alternanti su cui si basa il metodo di Cramer; tale via è però impraticabile già per piccole dimensioni del sistema, visto l'altissimo numero d'operazioni necessarie ad ogni passo.

- Comporre più applicazioni lineari tra loro e con il sistema stesso, così da modificare questo sino a renderlo triangolare, e dunque immediatamente risolubile; il metodo di Gauss (semplice e con pivoting pariale o totale) e le fattorizzazioni $A = QR$ (con matrici di Householder e di Givens) si comportano in tal senso, e permettono di risolvere sistemi di grandi dimensioni.

1.2 Implementazione al calcolatore.

Per l'implementazione al calcolatore ci serviamo del linguaggio C++; questo non è un linguggio dedicato espressamente alle applicazioni matematiche[1], e quindi si ha lo svantaggio di dover implementare creazione e gestione di matrice, incognite, termini noti, etc.

Creiamo dunque una classe matrix che contiene in sè la matrice degli elementi caratteristici dell'applicazione lineare considerata, i termini noti, i numeri delle equazioni e delle incognite, ed anche le incognite stesse; ad inizio programma istanziamo dunque due volte la classe matrix, per avere quindi due oggetti che usiamo, rispettivamente, come matrice sorgente e matrice d'uso.
La matrice sorgente contiene le incognite create in origine e quindi, per ciascuna dimensione considerata, calcoliamo i termini noti relativi; per ciascuna dimensione, e per ciascun metodo, copiamo quindi matrice e termini noti nella matrice d'uso e risolviamo; a fine processo verifichiamo quindi il comportamento di ciascun algoritmo semplicemente considerando la norma del vettore differenza tra le incognite calcolate e quelle imposte.

[1] Avremo potuto ad esempio servirci del software gratuito Scilab che permette la programmazione matematica.

Capitolo 2

Fattorizzazione di Gauss.

2.1 Fattorizzazione $A = LU$.

Il metodo di Gauss consiste nell'applicare le operazioni elementari alle varie equazioni del sistema così da triangolarizzare questo e renderlo quindi immediatamente risolubile.

Sia dato il sistema generico

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + .. + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + .. + a_{2n}x_n = b_2 \\ ..... \\ a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + .. + a_{nn}x_n = b_n \end{cases};$$

il primo passo per la triangolarizzazione è annullare i termini nella prima colonna dalla seconda riga in poi: per far questo per ogni equazione calcoliamo il rapporto tra il coefficiente primo ed il coefficiente primo della prima equazione; sottraiamo quindi a ciascuna equazione la prima moltiplicata al rapporto prima calcolato; abbiamo, ad esempio, per la seconda equazione

$$EQ2' = EQ2 - EQ1 * (\frac{a_{21}}{a_{11}}),$$

e quindi

$$EQ2' : \left(a_{21} - a_{11}\frac{a_{21}}{a_{11}}\right)x_1 + \left(a_{22} - a_{12}\frac{a_{21}}{a_{11}}\right)x_2 + .. + \left(a_{2n} - a_{1n}\frac{a_{21}}{a_{11}}\right)x_n = \left(b_2 - b_1\frac{a_{21}}{a_{11}}\right);$$

notiamo dunque che il trasformato ammetterà primo termine nullo.

L'algoritmo di Gauss è dato quindi dal reiterare questa sequenza d'operazioni per ciascuna riga per ciascuna sottoriga.
Le applicazioni lineari che si compongono con il sistema in studio sono dunque le operazioni elementari di somma tra equazioni e prodotto per uno scalare.

Notiamo che eseguendo però le operazioni al calcolatore, per l'aritmetica finita, avremo $a_{21} - a_{11}\frac{a_{21}}{a_{11}} \neq 0$: l'algoritmo di Gauss prima calcola la frazione, probabilmente commettendo quindi un errore non nullo, e poi esegue un prodotto, amplificando quindi l'eventuale errore prima commesso. Scalare i termini del sistema tra 0 ed 1 riduce quindi l'amplificazioe dell'errore.

L'implementazione al calcolatore permette, semplicemente variando i parametri per il preprocessore sintattico (i valori `#define`) nell'intestazione, di animare la risoluzione di un sistema lineare con i vari metodi; a tale fine annulliamo forzatamente le componenti del triangolo inferiore che dovrebbero esser nulle ma tali non sono appunto per l'aritmetica finita; non annullarle non comporterebbe alcuna differenza nelle soluzioni perché il metodo non le considera ed è per tale motivo che nemmeno le calcoliamo, bensì annulliamo.

```
//consideriamo d come numerazione in diagonale
for(d=0; d<dim-1; d++)
{
    //pivoting
    if(piv) pivoting(dim, d, mat, piv);
    //controlla se la matrice è singolare
    if((*mat).pop(d,d)==0) return 1;
    //triangolarizzazione
    for(r=d+1; r<dim; r++)
    {
        elemento_temporaneo=((*mat).pop(r,d) / (*mat).pop(d,d));
        //termini matrice
        for(c=d+1; c<dim; c++)
            (*mat).push(r, c, (*mat).pop(r, c)-((*mat).pop(d, c)*
elemento_temporaneo));
        //valenza estetica
        for(c=0; c<=d; c++)
            (*mat).push(r, c, 0);
        //termini noti
        for(c=termini_noti_min_col; c<termini_noti_max_col; c++)
```

```
            (*mat).push(r, c, (*mat).pop(r, c)-(*mat).pop(d, c)*
elemento_temporaneo);
    }//fine for r
}//fine for d
//se la matrice è singolare l'algoritmo non ha motivo d'andare avanti
if((*mat).pop(dim -1,dim -1) == 0) return 1;
//risolve richiamando un metodo della classe matrix
(*mat).risolvi_sistema_triangolare_superiore(dim);
```

2.2 Pivoting e fattorizzazioni $PA = LU$ e $PAQ = LU$.

Il metodo di Gauss, nella forma prima data detta semplice, non necessariamente è in grado di risolvere un sistema seppur questo sia compatibile; ciò è conseguenza del fatto che, ad ogni ciclo dell'algoritmo, si estrae l'elemento appartenente alla diagonale che andrà poi a dividere i termini delle sottorighe successive ad eguale colonna; non vi è però alcuna garanzia che tale termine sia non nullo, e questa ovviamente è condizione necessaria per il poter eseguire la divisione stessa.
Per ovviare a tale problema potremo, nell'eventualità appunto che l'algoritmo si blocchi per termine in diagonale nullo, permutare tra loro la linea dell'elemento in diagonale nullo (riga oppure colonna) con un'altra che "si comporta bene"; questa è però una strategia inaccettabile perché è data per tentativi non a priori conoscibili; l'effetto sarebbe in effetti di "inondare" l'algoritmo di verifiche per elemento nullo in diagonale.

Al fine ulteriore di abbassare il condizionamento del problema, ci è data però la scelta di comporre permutazioni così da massimizzare l'elemento in diagonale; in tal modo i termini vengono ulteriormente scalati ed il condizionamento ridotto sensibilmente.

```
void pivoting(int dim, int curr_dim, matrix *mat, int piv)
{
    int r, c,
    r_di_piv=curr_dim,
    c_di_piv=curr_dim;
    double elemento_temporaneo, elemento_temporaneo_2;
    elemento_temporaneo=(*mat).pop(r_di_piv, c_di_piv);
    if(elemento_temporaneo<0)
        elemento_temporaneo=elemento_temporaneo*(-1);
```

```
    for(r=curr_dim; r<dim; r++)
    for(c=curr_dim; c<dim; c++)
    {
        elemento_temporaneo_2=(*mat).pop(r, c);
        if(elemento_temporaneo_2<0)
            elemento_temporaneo_2=elemento_temporaneo_2*(-1);
        if(elemento_temporaneo_2>elemento_temporaneo)
        {
            elemento_temporaneo=(*mat).pop(r, c);
            r_di_piv=r;
            c_di_piv=c;
        }
        if(piv == 1) break;
        //per pivoting parziale (piv ==1) analizza 1 sola colonna per
riga
    }
    if(r_di_piv != curr_dim)
        (*mat).permuta_righe(r_di_piv, curr_dim, dim);
    if(c_di_piv != curr_dim)
        (*mat).permuta_colonne(c_di_piv, curr_dim, dim);
}//fine piv
```

Capitolo 3

Fattorizzazione $A = QR$.

Matrici ortogonali Q per definizione sono tali che

$$QQ^T = Q^TQ = I;$$

abbiamo dunque per ciascuna generica matrice A l'equivalenza

$$A = QQ^TA = QR.$$

Una proprietà importante delle matrici ortogonali è di non variere la norma euclidea $\|\cdot\|_2$; abbiamo quindi stesso numero di condizionamento per le matrici A ed R.

La ricerca di matrici ortogonali che triangolarizzino la matrice è centrale, ed a tale compito assolvono ad esempio le matrici di Householder e delle rotazioni di Givens.

3.1 Matrici di Householder.

Una matrice di Householder è del tipo $H = I - 2\mathbf{w}\mathbf{w}^T$ con $\|\mathbf{w}\| = 1$; richiedendo che, per ciascun vettore colonna $\mathbf{x}$ della matrice del sistema, sia

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

abbiamo:

- $\|H\mathbf{x}\| = \|\mathbf{x}\| = \sigma$, dove con σ abbiamo dunque indicato il modulo del vettore colonna considerato.

- $k = \pm\sigma$, che quindi possiamo scegliere positivo o negativo a nostro piacimento.
- $\mathbf{w} = \dfrac{\mathbf{x} - k\mathbf{e}_1}{2\left(\mathbf{w}^T\mathbf{x}\right)} = \dfrac{\mathbf{x} - k\mathbf{e}_1}{\|\mathbf{x} - k\mathbf{e}_1\|}$, che ci porta a preferire $-k$ come quantità positiva per evitare che la quantità $\mathbf{x} - k\mathbf{e}_1$ diventi nulla o comunque tanto piccola da risultare confrontabile con l'errore di macchina (pericolo di cancellazione); una o più componenti del vettore, nel primo caso, sarebbero incalcolabili[1] e, nel secondo, sarebbero affetti da un'errore relativo enorme.
- $\mathbf{x}^T H\mathbf{x} = \mathbf{x}^T\mathbf{x} - 2\mathbf{x}^T\mathbf{w}\mathbf{w}^T\mathbf{x} = \mathbf{x}^T\mathbf{x} - 2\left(\mathbf{w}^T\mathbf{x}\right)^2 = kx_1$ e dunque $\left(\mathbf{w}^T\mathbf{x}\right) = \sqrt{\dfrac{\mathbf{x}^T\mathbf{x} - kx_1}{2}} = \sqrt{\dfrac{\sigma^2 - kx_1}{2}}$; ricordando l'espressione di $\mathbf{w}$ e la scelta di segno per k, abbiamo inoltre $\|\mathbf{x} - k\mathbf{e}_1\| = \sqrt{2\sigma\left(\sigma + |x_1|\right)}$.

Con l'intenzione di semplificare l'algoritmo, introduciamo il vettore

$$\mathbf{v} = \left(\mathbf{x} - k\mathbf{e}_1\right),$$

così da ottenere

$$2\mathbf{w}\mathbf{w}^T = \frac{\left(\mathbf{x} - k\mathbf{e}_1\right)\left(\mathbf{x} - k\mathbf{e}_1\right)^T}{\sigma\left(\sigma + |x_1|\right)} = \frac{\mathbf{v}\mathbf{v}^T}{\sigma\left(\sigma + |x_1|\right)}$$

da cui, per sostituzione, abbiamo

$$H\mathbf{x} = \mathbf{x} - \frac{\mathbf{v}\mathbf{v}^T}{\sigma\left(\sigma + |x_1|\right)}\mathbf{x} = \mathbf{x} - \mathbf{v}\frac{\mathbf{v}^T\mathbf{x}}{\sigma\left(\sigma + |x_1|\right)}.$$

```
for(d=0; d<dim-1; d++)
{
    //estraggo vettore prima colonna sottomatrice corrente x
    for(r=d; r<dim; r++)
        vettore_temporaneo[r]=(*mat).pop(r, d);
    //modulo sigma
    sigma=0;
    for(r=d; r<dim; r++)
        sigma += (vettore_temporaneo[r]*vettore_temporaneo[r]);
    sigma=sqrt(sigma);
```

[1] I calcolatori di ultima generazione non si bloccano di fronte a calcoli del tipo $\dfrac{k}{0}$, restituendo infatti un valore INF, o del genere $\dfrac{0}{0}$, restituendo un NAN (acronimo di not a number).

```
    //controlla che il vettore (e dunque il suo modulo) non sia nullo
    if(sigma == 0) return 1;
    //kappa
    if(vettore_temporaneo[d]>0) kappa=sigma*(-1);
    else kappa=sigma;
    //vettore v
    //prima componente
    vettore_v[d]=(vettore_temporaneo[d]-kappa);
    //componenti restanti
    for(r=d+1; r<dim; r++)
        vettore_v[r]=vettore_temporaneo[r];
    //modulo beta
    elemento_temporaneo=
vettore_temporaneo[d]*((vettore_temporaneo[d]>0) ?  1 :  (-1));
    beta=sigma*(sigma+elemento_temporaneo);
    //applico a tutte le colonne della sottomatrice corrente
    for(c=d; c<dim; c++)
    {
        //delta
        delta=0;
        for(r=d; r<dim; r++)
            delta += ((*mat).pop(r,c)*vettore_v[r]);
        delta=delta / beta;
        //sostituisci nella matrice
        for(r=d; r<dim; r++)
            (*mat).push(r, c, ((*mat).pop(r,c)-(delta*vettore_v[r])));
    }
    //applico a termini noti
    for(c=termini_noti_min_col; c<termini_noti_max_col; c++)
    {
        //delta
        delta=0;
        for(r=d; r<dim; r++)
            delta += ((*mat).pop(r,c)*vettore_v[r]);
        delta=delta / beta;
        //sostituisci nella matrice
        for(r=d; r<dim; r++)
            (*mat).push(r, c, ((*mat).pop(r,c)-delta*vettore_v[r]));
    }
    if(stampa_termini) (*mat).stampa_matrice(dim);
}//fine for d
//controlla se la matrice è singolare
for(d=0; d<dim; d++)
```

```
    if(!(*mat).pop(d,d)) return 1;
//risolve richiamando un metodo della classe matrix
(*mat).risolvi_sistema_triangolare_superiore(dim);
```

3.2 Matrici di rotazione di Givens.

Una matrice di Givens rappresenta una rotazione degli assi capace, per composizione con l'applicazione in studio, di annullare selettivamente le componenti desiderate e quindi di triangolarizzare il sistema. Per matrici sparse questo algoritmo è maggiormente auspicabile del metodo di Householder per il fatto che si opererà selettivamente per soli termini non nulli.

Una matrice di rotazione ortogonale, quindi espressa da una matrice ortogonale, è data da

$$G(\theta) = \begin{vmatrix} \cos(\theta) & \operatorname{sen}(\theta) \\ -\operatorname{sen}(\theta) & \cos(\theta) \end{vmatrix}$$

e tale quindi che

$$G(\theta) \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x' = x\cos(\theta) + y\operatorname{sen}(\theta) \\ y' = -x\operatorname{sen}(\theta) + y\cos(\theta) \end{pmatrix}.$$

Per annullare selettivamente le componenti sotto la diagonale ricerchiamo la rotazione capace di far giacere su un asse (ascissa od ordinata nulla) il punto di coordinate x, y dove x ed y sono appunto i termini estratti dalla matrice, ossia l'elemento appartenente alla diagonale e l'elemento della stessa colonna ma d'una riga inferiore; ovviamente andremo ad annullare il termine inferiore alla diagonale.
Volendo annullare la componente y' del trasformato prima ottenuto

$$\begin{pmatrix} x' = x\cos(\theta) + y\operatorname{sen}(\theta) \\ y' = -x\operatorname{sen}(\theta) + y\cos(\theta) \end{pmatrix}$$

poniamo appunto

$$-x\operatorname{sen}(\theta) + y\cos(\theta) = 0;$$

da qui possiamo infine ricavare il seno ed il coseno della rotazione cercata, infatti

$$
\begin{aligned}
& y\cos(\theta) = x\,\mathrm{sen}(\theta), \\
& \cos(\theta) = \frac{x}{y}\,\mathrm{sen}(\theta), \\
& \cos^2(\theta) + \mathrm{sen}^2(\theta) = 1, \\
& \frac{x^2}{y^2}\,\mathrm{sen}^2(\theta) + \mathrm{sen}^2(\theta) = 1, \\
& \left(\frac{x^2}{y^2} + 1\right)\mathrm{sen}^2(\theta) = 1, \\
& \mathrm{sen}(\theta) = \left(\frac{x^2+y^2}{y^2}\right)^{-1/2} = \frac{y}{\sqrt{x^2+y^2}}, \\
& \cos(\theta) = \frac{x}{\sqrt{x^2+y^2}}.
\end{aligned}
$$

Quanto detto viene però adattato secondo sia $x > y$ od $y > x$ per evitare *underflows* (superamenti limite inferiore consentito) od *owerflows* (straripamenti).

```
for(d=0; d<dim-1; d++)
{
    for(r=d+1; r<dim; r++)
    {
        //parametri di rotazione
        if((*mat).pop(r, d) == 0)
            {cos=1; sen=0;}
        else
            if(((*mat).pop(r,d)*(*mat).pop(r,d))>
((*mat).pop(d,d)*(*mat).pop(d,d)))
            {
                t=(*mat).pop(d, d)/(*mat).pop(r, d);
                z=sqrt(1+t*t);
                sen=(double) 1 / z;
                cos=t*sen;
            }
            else
            {
                t=(*mat).pop(r, d)/(*mat).pop(d, d);
                z=sqrt(1+t*t);
                cos=(double) 1 / z;
                sen=t*cos;
            }
```

```
        //termini matrice
        for(c=d; c<dim; c++)
        {
            t=cos*(*mat).pop(d, c)+sen*(*mat).pop(r, c);
            (*mat).push(r, c, (-1)*sen*(*mat).pop(d, c)+
cos*(*mat).pop(r, c));
            (*mat).push(d, c, t);
        }//fine for c

        //termini noti
        for(c=termini_noti_min_col; c<termini_noti_max_col; c++)
        {
            t=cos*(*mat).pop(d, c)+sen*(*mat).pop(r, c);
            (*mat).push(r, c, (-1)*sen*(*mat).pop(d, c)+
cos*(*mat).pop(r, c));
            (*mat).push(d, c, t);
        }//fine for c
    }//fine for r
}//fine for d
//controlla se la matrice è singolare
for(d=0; d<dim; d++)
    if(!(*mat).pop(d,d)) return 1;
//risolve richiamando un metodo della classe matrix
(*mat).risolvi_sistema_triangolare_superiore(dim);
```

Capitolo 4

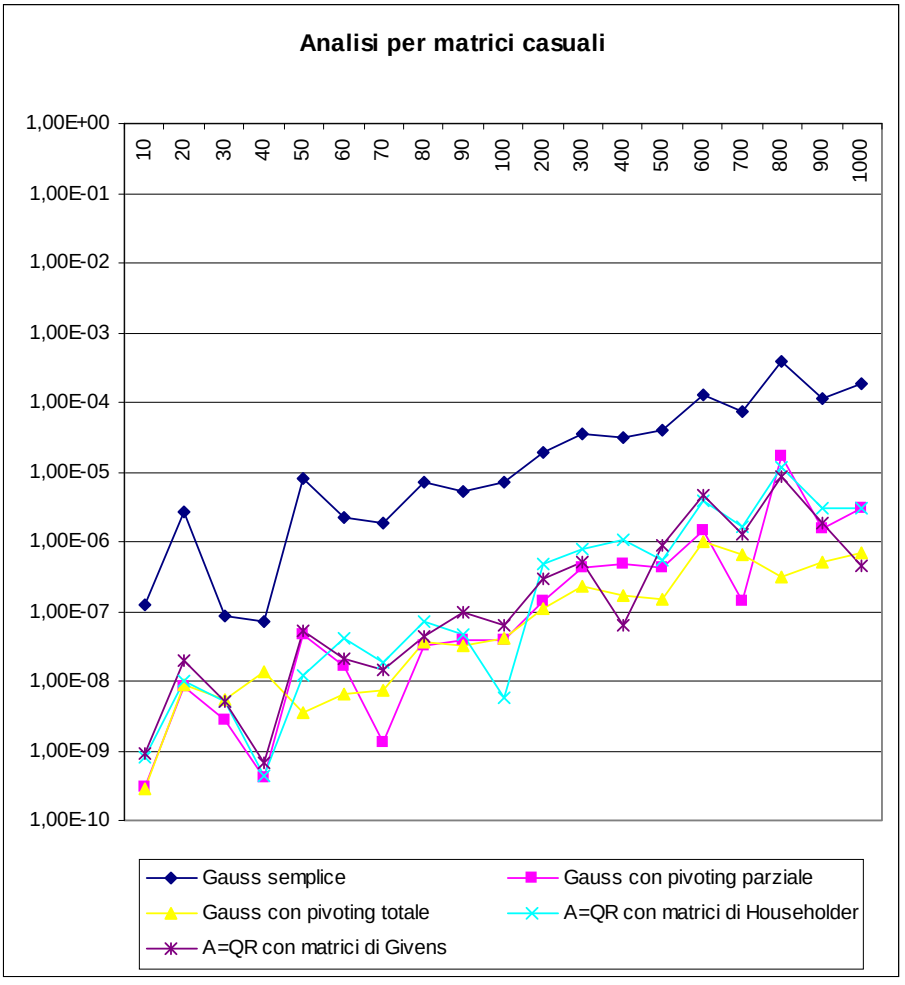
Confronto prestazionale.

Di seguito mostriamo il codice implementato per creare le matrici casuali, sparse ed hilbertiane; presentiamo poi gli errori commessi dai vari metodi implementati accompagnati anche da un grafico in scala logaritmica.

4.1 Matrici casuali

```
void matrice_casuale()
{
    int n, r, c;
    //elementi matrice
    for(r=elementi_matrice_min_row; r<elementi_matrice_max_row; r++)
        for(c=elementi_matrice_min_col; c<elementi_matrice_max_col;
c++)
            elemento[r][c]=(double) rand() / 65536;
    //incognite
    for(n=0; n<num_volte_analisi; n++)
        for(c=incognite_min_col; c<incognite_max_col; c++)
            elemento[incognite_min_row+n][c]=(double) rand() / 65536;
    //numeri incognite
    for(c=elementi_matrice_min_col; c<elementi_matrice_max_col; c++)
        elemento[num_incognite_row][c]=c;
    //numeri equazioni
    for(r=elementi_matrice_min_row; r<elementi_matrice_max_row; r++)
        elemento[r][num_equazioni_col]=r;
};
```

dim	$A = LU$	$PA = LU$	$PAQ = LU$	$A = Q_{\mathrm{H}} R$	$A = Q_{\mathrm{G}} R$
10	1,21E-07	3,09E-10	2,80E-10	8,05E-10	9,40E-10
20	2,64E-06	8,18E-09	8,74E-09	9,74E-09	2,02E-08
30	8,70E-08	2,81E-09	5,39E-09	4,98E-09	4,96E-09
40	7,11E-08	4,20E-10	1,37E-08	4,45E-10	6,90E-10
50	8,25E-06	4,71E-08	3,51E-09	1,20E-08	5,22E-08
60	2,19E-06	1,63E-08	6,66E-09	4,04E-08	2,06E-08
70	1,82E-06	1,30E-09	7,31E-09	1,83E-08	1,46E-08
80	7,04E-06	3,14E-08	3,56E-08	7,31E-08	4,49E-08
90	5,17E-06	3,98E-08	3,24E-08	4,78E-08	9,89E-08
100	6,94E-06	3,91E-08	4,04E-08	5,69E-09	6,26E-08
200	1,89E-05	1,36E-07	1,10E-07	4,70E-07	2,93E-07
300	3,51E-05	4,12E-07	2,33E-07	7,75E-07	4,96E-07
400	3,08E-05	4,74E-07	1,72E-07	1,04E-06	6,16E-08
500	4,09E-05	4,21E-07	1,47E-07	5,53E-07	8,89E-07
600	1,25E-04	1,45E-06	1,01E-06	3,76E-06	4,70E-06
700	7,35E-05	1,37E-07	6,45E-07	1,61E-06	1,29E-06
800	3,86E-04	1,73E-05	3,16E-07	1,20E-05	8,58E-06
900	1,14E-04	1,50E-06	5,12E-07	3,04E-06	1,84E-06
1000	1,83E-04	2,94E-06	7,02E-07	3,04E-06	4,59E-07

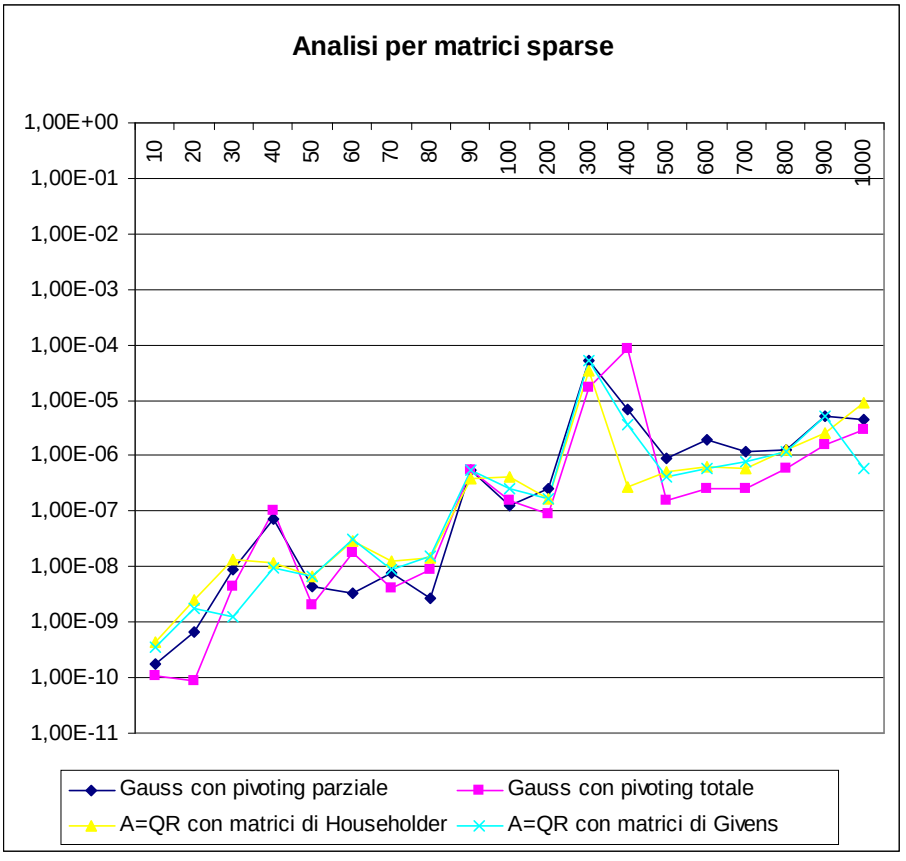


4.2 Matrici sparse.

```
void matrice_sparsa()
{
    int n, r, c;
    //elementi matrice
    for(r=elementi_matrice_min_row; r<elementi_matrice_max_row; r++)
        for(c=elementi_matrice_min_col; c<elementi_matrice_max_col;
c++)
```

```
            if(rand()> 16384) elemento[r][c]=(double) rand() / 65536;
            else elemento[r][c]=0;
    //incognite
    for(n=0; n<num_volte_analisi; n++)
        for(c=incognite_min_col; c<incognite_max_col; c++)
            elemento[incognite_min_row+n][c]=(double) rand() / 65536;
    //numeri incognite
    for(c=elementi_matrice_min_col; c<elementi_matrice_max_col; c++)
        elemento[num_incognite_row][c]=c;
    //numeri equazioni
    for(r=elementi_matrice_min_row; r<elementi_matrice_max_row; r++)
        elemento[r][num_equazioni_col]=r;
}
```

dim	$A = LU$	$PA = LU$	$PAQ = LU$	$A = Q_{\mathrm{H}} R$	$A = Q_{\mathrm{G}} R$
10	—	1,81E-10	1,09E-10	4,38E-10	3,54E-10
20	—	6,49E-10	9,05E-11	2,57E-09	1,75E-09
30	—	8,61E-09	4,47E-09	1,37E-08	1,26E-09
40	—	7,38E-08	1,01E-07	1,15E-08	9,72E-09
50	—	4,26E-09	1,99E-09	6,63E-09	6,85E-09
60	—	3,28E-09	1,79E-08	2,98E-08	3,11E-08
70	—	7,46E-09	4,14E-09	1,26E-08	9,00E-09
80	—	2,72E-09	8,56E-09	1,47E-08	1,58E-08
90	—	5,33E-07	5,68E-07	3,89E-07	5,40E-07
100	—	1,26E-07	1,60E-07	4,04E-07	2,51E-07
200	—	2,58E-07	8,77E-08	1,72E-07	1,67E-07
300	—	5,37E-05	1,71E-05	3,37E-05	5,16E-05
400	—	6,69E-06	8,42E-05	2,77E-07	3,63E-06
500	—	9,24E-07	1,51E-07	5,10E-07	4,30E-07
600	—	1,93E-06	2,56E-07	6,50E-07	5,94E-07
700	—	1,20E-06	2,60E-07	5,85E-07	7,95E-07
800	—	1,24E-06	5,72E-07	1,31E-06	1,15E-06
900	—	5,00E-06	1,55E-06	2,64E-06	5,08E-06
1000	—	4,64E-06	3,02E-06	9,08E-06	5,89E-07

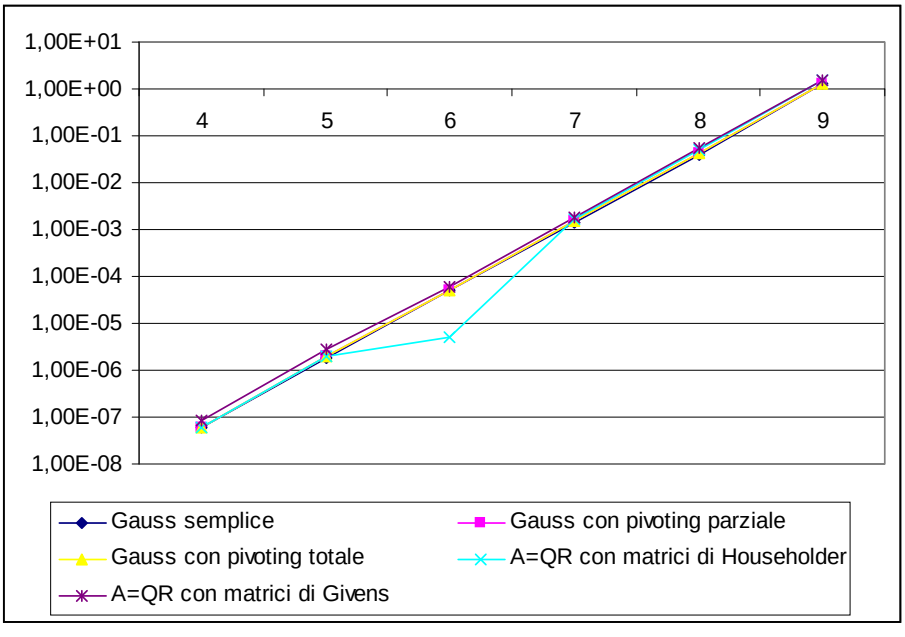


4.3 Matrici di Hilbert.

```
void matrice_hilbert()
{
    int n, r, c;
    //elementi matrice
    for(r=elementi_matrice_min_row; r<elementi_matrice_max_row; r++)
        for(c=elementi_matrice_min_col; c<elementi_matrice_max_col;
c++)
            elemento[r][c]=1 / ((double)(r+c+1));
    //incognite
    for(n=0; n<num_volte_analisi; n++)
```

```
        for(c=incognite_min_col; c<incognite_max_col; c++)
            elemento[incognite_min_row+n][c]=(double) rand() / 65536;
    //numeri incognite
    for(c=elementi_matrice_min_col; c<elementi_matrice_max_col; c++)
        elemento[num_incognite_row][c]=c;
    //numeri equazioni
    for(r=elementi_matrice_min_row; r<elementi_matrice_max_row; r++)
        elemento[r][num_equazioni_col]=r;
};
```

dim	$A = LU$	$PA = LU$	$PAQ = LU$	$A = Q_H R$	$A = Q_G R$
4	5,99E-08	6,13E-08	6,14E-08	5,95E-08	8,53E-08
5	1,89E-06	1,91E-06	1,92E-06	2,05E-06	2,79E-06
6	5,06E-05	5,11E-05	5,16E-05	4,96E-06	6,14E-05
7	1,43E-03	1,52E-03	1,54E-03	1,67E-03	1,83E-03
8	3,90E-02	4,19E-02	4,17E-02	5,22E-02	5,61E-02
9	1,30E+00	1,31E+00	1,26E+00	1,54E+00	1,58E+00



Capitolo 5

Animazione per matrice 4×4.

Incognite impresse.
x0: 4.570007e-002
x1: 1.822205e-001
x2: 7.365417e-002
x3: 8.294678e-002

Matrice d'uso:
(0,0)6.256104e-004 (0,1)2.817841e-001 (0,2)9.664917e-002 (0,3)4.043579e-001
(1,0)2.924957e-001 (1,1)2.399292e-001 (1,2)1.751404e-001 (1,3)4.479675e-001
(2,0)4.114075e-001 (2,1)3.732910e-001 (2,2)8.705139e-002 (2,3)4.294586e-001
(3,0)3.552399e-001 (3,1)2.567596e-001 (3,2)1.519928e-001 (3,3)7.492065e-003

Termine noto num. 1:
b0: 9.203421e-002
b1: 1.071444e-001
b2: 1.288565e-001
b3: 7.483769e-002

Fattorizzazione di Gauss semplice

(0,0)6.256104e-004 (0,1)2.817841e-001 (0,2)9.664917e-002 (0,3)4.043579e-001
(1,0)2.924957e-001 (1,1)2.399292e-001 (1,2)1.751404e-001 (1,3)4.479675e-001
(2,0)4.114075e-001 (2,1)3.732910e-001 (2,2)8.705139e-002 (2,3)4.294586e-001
(3,0)3.552399e-001 (3,1)2.567596e-001 (3,2)1.519928e-001 (3,3)7.492065e-003

(0,0)6.256104e-004 (0,1)2.817841e-001 (0,2)9.664917e-002 (0,3)4.043579e-001
(1,0)0.000000e+000 (1,1)-1.315044e+002 (1,2)-4.501188e+001 (1,3)-1.886041e+002
(2,0)0.000000e+000 (2,1)-1.849307e+002 (2,2)-6.347039e+001 (2,3)-2.654802e+002
(3,0)0.000000e+000 (3,1)-1.597485e+002 (3,2)-5.472823e+001 (3,3)-2.295988e+002

(0,0)6.256104e-004 (0,1)2.817841e-001 (0,2)9.664917e-002 (0,3)4.043579e-001
(1,0)0.000000e+000 (1,1)-1.315044e+002 (1,2)-4.501188e+001 (1,3)-1.886041e+002
(2,0)0.000000e+000 (2,1)0.000000e+000 (2,2)-1.715517e-001 (2,3)-2.520004e-001
(3,0)0.000000e+000 (3,1)0.000000e+000 (3,2)-4.886161e-002 (3,3)-4.868931e-001

(0,0)6.256104e-004 (0,1)2.817841e-001 (0,2)9.664917e-002 (0,3)4.043579e-001
(1,0)0.000000e+000 (1,1)-1.315044e+002 (1,2)-4.501188e+001 (1,3)-1.886041e+002
(2,0)0.000000e+000 (2,1)0.000000e+000 (2,2)-1.715517e-001 (2,3)-2.520004e-001
(3,0)0.000000e+000 (3,1)0.000000e+000 (3,2)0.000000e+000 (3,3)-4.151180e-001

Incognite calcolate:
x0: 4.570007e-002
x1: 1.822205e-001
x2: 7.365417e-002
x3: 8.294678e-002

errore: 3.74053e-014

Fattorizzazione di Gauss con pivoting parziale per righe

(0,0)6.256104e-004 (0,1)2.817841e-001 (0,2)9.664917e-002 (0,3)4.043579e-001
(1,0)2.924957e-001 (1,1)2.399292e-001 (1,2)1.751404e-001 (1,3)4.479675e-001
(2,0)4.114075e-001 (2,1)3.732910e-001 (2,2)8.705139e-002 (2,3)4.294586e-001
(3,0)3.552399e-001 (3,1)2.567596e-001 (3,2)1.519928e-001 (3,3)7.492065e-003

(2,0)4.114075e-001 (2,1)3.732910e-001 (2,2)8.705139e-002 (2,3)4.294586e-001
(1,0)2.924957e-001 (1,1)2.399292e-001 (1,2)1.751404e-001 (1,3)4.479675e-001
(0,0)6.256104e-004 (0,1)2.817841e-001 (0,2)9.664917e-002 (0,3)4.043579e-001
(3,0)3.552399e-001 (3,1)2.567596e-001 (3,2)1.519928e-001 (3,3)7.492065e-003

(2,0)4.114075e-001 (2,1)3.732910e-001 (2,2)8.705139e-002 (2,3)4.294586e-001
(1,0)0.000000e+000 (1,1)-2.546712e-002 (1,2)1.132500e-001 (1,3)1.426381e-001
(0,0)0.000000e+000 (0,1)2.812164e-001 (0,2)9.651679e-002 (0,3)4.037049e-001
(3,0)0.000000e+000 (3,1)-6.556764e-002 (3,2)7.682614e-002 (3,3)-3.633345e-001

(2,0)4.114075e-001 (2,1)3.732910e-001 (2,2)8.705139e-002 (2,3)4.294586e-001

(0,0)0.000000e+000 (0,1)2.812164e-001 (0,2)9.651679e-002 (0,3)4.037049e-001
(1,0)0.000000e+000 (1,1)-2.546712e-002 (1,2)1.132500e-001 (1,3)1.426381e-001
(3,0)0.000000e+000 (3,1)-6.556764e-002 (3,2)7.682614e-002 (3,3)-3.633345e-001

(2,0)4.114075e-001 (2,1)3.732910e-001 (2,2)8.705139e-002 (2,3)4.294586e-001
(0,0)0.000000e+000 (0,1)2.812164e-001 (0,2)9.651679e-002 (0,3)4.037049e-001
(1,0)0.000000e+000 (1,1)0.000000e+000 (1,2)1.219906e-001 (1,3)1.791978e-001
(3,0)0.000000e+000 (3,1)0.000000e+000 (3,2)9.932973e-002 (3,3)-2.692078e-001

(2,0)4.114075e-001 (2,1)3.732910e-001 (2,2)8.705139e-002 (2,3)4.294586e-001
(0,0)0.000000e+000 (0,1)2.812164e-001 (0,2)9.651679e-002 (0,3)4.037049e-001
(1,0)0.000000e+000 (1,1)0.000000e+000 (1,2)1.219906e-001 (1,3)1.791978e-001
(3,0)0.000000e+000 (3,1)0.000000e+000 (3,2)0.000000e+000 (3,3)-4.151180e-001

Vettore incognite num. 1:
x0: 4.570007e-002
x1: 1.822205e-001
x2: 7.365417e-002
x3: 8.294678e-002

errore: 5.23875e-017

Fattorizzazione di Gauss con pivoting totale

(0,0)6.256104e-004 (0,1)2.817841e-001 (0,2)9.664917e-002 (0,3)4.043579e-001
(1,0)2.924957e-001 (1,1)2.399292e-001 (1,2)1.751404e-001 (1,3)4.479675e-001
(2,0)4.114075e-001 (2,1)3.732910e-001 (2,2)8.705139e-002 (2,3)4.294586e-001
(3,0)3.552399e-001 (3,1)2.567596e-001 (3,2)1.519928e-001 (3,3)7.492065e-003

(1,3)4.479675e-001 (1,1)2.399292e-001 (1,2)1.751404e-001 (1,0)2.924957e-001
(0,3)4.043579e-001 (0,1)2.817841e-001 (0,2)9.664917e-002 (0,0)6.256104e-004
(2,3)4.294586e-001 (2,1)3.732910e-001 (2,2)8.705139e-002 (2,0)4.114075e-001
(3,3)7.492065e-003 (3,1)2.567596e-001 (3,2)1.519928e-001 (3,0)3.552399e-001

(1,3)4.479675e-001 (1,1)2.399292e-001 (1,2)1.751404e-001 (1,0)2.924957e-001
(0,3)0.000000e+000 (0,1)6.521196e-002 (0,2)-6.144130e-002 (0,0)-2.633957e-001
(2,3)0.000000e+000 (2,1)1.432751e-001 (2,2)-8.085262e-002 (2,0)1.309969e-001
(3,3)0.000000e+000 (3,1)2.527469e-001 (3,2)1.490636e-001 (3,0)3.503480e-001

(1,3)4.479675e-001 (1,0)2.924957e-001 (1,2)1.751404e-001 (1,1)2.399292e-001
(3,3)0.000000e+000 (3,0)3.503480e-001 (3,2)1.490636e-001 (3,1)2.527469e-001

(2,3)0.000000e+000 (2,0)1.309969e-001 (2,2)-8.085262e-002 (2,1)1.432751e-001
(0,3)0.000000e+000 (0,0)-2.633957e-001 (0,2)-6.144130e-002 (0,1)6.521196e-002

(1,3)4.479675e-001 (1,0)2.924957e-001 (1,2)1.751404e-001 (1,1)2.399292e-001
(3,3)0.000000e+000 (3,0)3.503480e-001 (3,2)1.490636e-001 (3,1)2.527469e-001
(2,3)0.000000e+000 (2,0)0.000000e+000 (2,2)-1.365883e-001 (2,1)4.877170e-002
(0,3)0.000000e+000 (0,0)0.000000e+000 (0,2)5.062647e-002 (0,1)2.552300e-001

(1,3)4.479675e-001 (1,0)2.924957e-001 (1,1)2.399292e-001 (1,2)1.751404e-001
(3,3)0.000000e+000 (3,0)3.503480e-001 (3,1)2.527469e-001 (3,2)1.490636e-001
(0,3)0.000000e+000 (0,0)0.000000e+000 (0,1)2.552300e-001 (0,2)5.062647e-002
(2,3)0.000000e+000 (2,0)0.000000e+000 (2,1)4.877170e-002 (2,2)-1.365883e-001

(1,3)4.479675e-001 (1,0)2.924957e-001 (1,1)2.399292e-001 (1,2)1.751404e-001
(3,3)0.000000e+000 (3,0)3.503480e-001 (3,1)2.527469e-001 (3,2)1.490636e-001
(0,3)0.000000e+000 (0,0)0.000000e+000 (0,1)2.552300e-001 (0,2)5.062647e-002
(2,3)0.000000e+000 (2,0)0.000000e+000 (2,1)0.000000e+000 (2,2)-1.462625e-001

errore: 1.96262e-017

Vettore incognite num. 1:
x3: 8.294678e-002
x0: 4.570007e-002
x1: 1.822205e-001
x2: 7.365417e-002

Fattorizzazione QR di Householder

(0,0)6.256104e-004 (0,1)2.817841e-001 (0,2)9.664917e-002 (0,3)4.043579e-001
(1,0)2.924957e-001 (1,1)2.399292e-001 (1,2)1.751404e-001 (1,3)4.479675e-001
(2,0)4.114075e-001 (2,1)3.732910e-001 (2,2)8.705139e-002 (2,3)4.294586e-001
(3,0)3.552399e-001 (3,1)2.567596e-001 (3,2)1.519928e-001 (3,3)7.492065e-003

(0,0)-6.172565e-001 (0,1)-5.105503e-001 (0,2)-2.285853e-001 (0,3)-5.032357e-001
(1,0)0.000000e+000 (1,1)-1.351495e-001 (1,2)2.117947e-002 (1,3)1.832692e-002
(2,0)0.000000e+000 (2,1)-1.542729e-001 (2,2)-1.295011e-001 (2,3)-1.748489e-001
(3,0)0.000000e+000 (3,1)-1.987783e-001 (3,2)-3.499473e-002 (3,3)-5.143120e-001

(0,0)-6.172565e-001 (0,1)-5.105503e-001 (0,2)-2.285853e-001 (0,3)-5.032357e-001
(1,0)0.000000e+000 (1,1)2.856192e-001 (1,2)8.428110e-002 (1,3)4.437084e-001
(2,0)0.000000e+000 (2,1)-1.713039e-017 (2,2)-1.063652e-001 (2,3)-1.888474e-002

(3,0)0.000000e+000 (3,1)-2.206351e-017 (3,2)-5.184445e-003 (3,3)-3.133546e-001

(0,0)-6.172565e-001 (0,1)-5.105503e-001 (0,2)-2.285853e-001 (0,3)-5.032357e-001
(1,0)0.000000e+000 (1,1)2.856192e-001 (1,2)8.428110e-002 (1,3)4.437084e-001
(2,0)0.000000e+000 (2,1)-1.713039e-017 (2,2)1.064914e-001 (2,3)3.411774e-002
(3,0)0.000000e+000 (3,1)-2.206351e-017 (3,2)1.151118e-018 (3,3)-3.120636e-001

errore: 4.16334e-017

Vettore incognite num. 1:
x0: 4.570007e-002
x1: 1.822205e-001
x2: 7.365417e-002
x3: 8.294678e-002

Fattorizzazione QR di Givens

(0,0)6.256104e-004 (0,1)2.817841e-001 (0,2)9.664917e-002 (0,3)4.043579e-001
(1,0)2.924957e-001 (1,1)2.399292e-001 (1,2)1.751404e-001 (1,3)4.479675e-001
(2,0)4.114075e-001 (2,1)3.732910e-001 (2,2)8.705139e-002 (2,3)4.294586e-001
(3,0)3.552399e-001 (3,1)2.567596e-001 (3,2)1.519928e-001 (3,3)7.492065e-003

(0,0)6.172565e-001 (0,1)5.105503e-001 (0,2)2.285853e-001 (0,3)5.032357e-001
(1,0)-1.079967e-020 (1,1)-2.812702e-001 (1,2)-9.627435e-002 (1,3)-4.033988e-001
(2,0)-5.027988e-018 (2,1)2.026573e-002 (2,2)-9.246810e-002 (2,3)-1.169552e-001
(3,0)-2.168404e-018 (3,1)-4.532854e-002 (3,2)2.499257e-002 (3,3)-3.449866e-001

(0,0)6.172565e-001 (0,1)5.105503e-001 (0,2)2.285853e-001 (0,3)5.032357e-001
(1,0)-1.079967e-020 (1,1)-2.856192e-001 (1,2)-8.428110e-002 (1,3)-4.437084e-001
(2,0)-5.027988e-018 (2,1)-3.881105e-018 (2,2)-9.914772e-002 (2,3)-1.456428e-001
(3,0)-2.168404e-018 (3,1)-2.205674e-018 (3,2)3.886071e-002 (3,3)-2.780933e-001

(0,0)6.172565e-001 (0,1)5.105503e-001 (0,2)2.285853e-001 (0,3)5.032357e-001
(1,0)-1.079967e-020 (1,1)-2.856192e-001 (1,2)-8.428110e-002 (1,3)-4.437084e-001
(2,0)-5.027988e-018 (2,1)-3.881105e-018 (2,2)-1.064914e-001 (2,3)-3.411774e-002
(3,0)-2.168404e-018 (3,1)-2.205674e-018 (3,2)1.768605e-018 (3,3)-3.120636e-001

Vettore incognite num. 1:
x0: 4.570007e-002
x1: 1.822205e-001
x2: 7.365417e-002
x3: 8.294678e-002

errore: 5.19259e-017