RELAZIONE QUARTA ESERCITAZIONE

1) RISOLUZIONE DI UN SISTEMA LINEARE LX=B CON MATRICE TRIANGOLARE INFERIORE.

CODICE MATLAB, ANALISI E RISULTATI:

```
function [ x ] = avanti( L,b )
%il vettore colonna x ha la stessa dimensione di b
x = zeros(size(b));
%implementazione pseudo codice
for i=1:1:max(size(b))
    x(i,1) = b(i,1);
    for j=1:1:i-1
        x(i,1) = x(i,1) - L(i,j) * x(j,1);
    end
    x(i,1) = x(i,1) / L(i,i);
end
end
```

Esempio di utilizzo:

```
L =    1     0     0
       2    -1     0
       2     1     3

b =    1
      -1
       3

avanti(L,b)

ans =   1.0000
        3.0000
       -0.6667
```

Analisi:

L'algoritmo di sostituzione in avanti calcola la soluzione del sistema lineare LX=B, dove L è una matrice triangolare inferiore. La prima incognita che si calcola è la x1, per calcolarla infatti basta fare il rapporto fra il termine noto e il coefficiente di x1.
Dopo si calcola x2, andando a sostituire ad x1 il valore trovato prima e modificando il valore del termine noto, poi si rifà il rapporto fra il termine noto e il coefficiente di x2. Questi passi si fanno finché non si arriva all'ultima riga della matrice. Finita la sequenza di passi, il sistema lineare è stato risolto.

2) RISOLUZIONE DI UN SISTEMA LINEARE RX=B CON MATRICE TRIANGOLARE SUPERIORE.

CODICE MATLAB, ANALISI E RISULTATI:

```
function [ x ] = indietro( R,b )
%il vettore colonna x ha la stessa dimensione di b
x = zeros(size(b));
%implementazione pseudo codice
for i=max(size(b)):-1:1
    x(i,1) = b(i,1);
    for j=i+1:1:max(size(b))
        x(i,1) = x(i,1) - R(i,j) * x(j,1);
    end
    x(i,1) = x(i,1) / R(i,i);
end
end
```

Esempio di utilizzo:

```
R =    2    1    3
       0    1    2
       0    0    4

b =     1
        2
       -1

indietro(R,b)

ans =    -0.3750
          2.5000
         -0.2500
```

Analisi:

L'algoritmo di sostituzione all'indietro calcola la soluzione del sistema lineare RX=B, dove R è una matrice triangolare superiore. La prima incognita che si calcola è la xn, per calcolarla infatti basta fare il rapporto fra il termine noto e il coefficiente di xn.
Dopo si calcola x(n - 1), andando a sostituire ad xn il valore trovato prima e modificando il valore del termine noto, poi si rifà il rapporto fra il termine noto e il coefficiente di x(n- 1). Questi passi si fanno finché non si arriva alla prima riga della matrice. Finita la sequenza di passi, il sistema lineare è stato risolto.

3) FATTORIZZAZIONE DI GAUSS DI UNA MATRICE

CODICE MATLAB, ANALISI E RISULTATI:

```
function [ L,R ] = gauss( A )
%inizializza L alla matrice identità
L = eye(max(size(A)));
%inizializza R uguale ad A
R = A;
n = size(A)
%implementazione pseudo codice
for k=1:1:n - 1
    for i = k + 1:1:n
        if(R(k,k) == 0)
            break;
        end
        L(i,k) = R(i,k) / R(k,k);
        for j=k+1:1:n
            R(i,j) = R(i,j) - L(i,k) * R(k,j);
        end
    end
end
R = triu(R);
end
```

Esempio di utilizzo:

$$A = \begin{matrix} 1 & 2 & 1 \\ 2 & 4 & 3 \\ -5 & 4 & 3 \end{matrix}$$

[L,R] = gauss(A)

$$L = \begin{matrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -5 & 0 & 1 \end{matrix}$$

$$R = \begin{matrix} 1 & 2 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 8 \end{matrix}$$

Analisi:

L'algoritmo di Gauss ha al suo interno una fattorizzazione LR. La matrice R è il risultato finale dell'algoritmo sulla matrice di partenza.
La matrice L all'inizio parte come la matrice identità di ordine n (stessa dimensione della matrice passata come parametro), poi si forma mettendo negli elementi sotto la diagonale i moltiplicatori m che rendono gli elementi della matrice R sotto la diagonale uguali a zero.
Alla fine dell'algoritmo la matrice R diventa la matrice triangolare superiore, ma ha ancora gli elementi soto la diagonale diversi da zero. Per renderli zero e quindi rendere R una mtrice triangola superiore, si usa la funzione triu(R) di MatLab.

4) VERIFICA DELLA STABILITÀ DELL'ALGORITMO DI FATTORIZZAZIONE DI GAUSS

CODICE MATLAB, ANALISI E RISULTATI:

```
W = (matrice di Wilkinson)

     1     0     0     0     0     0     0    -1
     1     1     0     0     0     0     0     1
    -1     1     1     0     0     0     0    -1
     1    -1     1     1     0     0     0     1
    -1     1    -1     1     1     0     0    -1
     1    -1     1    -1     1     1     0     1
    -1     1    -1     1    -1     1     1    -1
     1    -1     1    -1     1    -1     1     1


[L,R] = gauss(W)

L =

     1     0     0     0     0     0     0     0
     1     1     0     0     0     0     0     0
    -1     1     1     0     0     0     0     0
     1    -1     1     1     0     0     0     0
    -1     1    -1     1     1     0     0     0
     1    -1     1    -1     1     1     0     0
    -1     1    -1     1    -1     1     1     0
     1    -1     1    -1     1    -1     1     1


R =

     1     0     0     0     0     0     0    -1
     0     1     0     0     0     0     0     2
     0     0     1     0     0     0     0    -4
     0     0     0     1     0     0     0     8
     0     0     0     0     1     0     0   -16
     0     0     0     0     0     1     0    32
     0     0     0     0     0     0     1   -64
     0     0     0     0     0     0     0   128

R(8,8) = 128
```

P = (matrice di perturbazione)

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0.5

R = R + P

R =

1	0	0	0	0	0	0	-1
0	1	0	0	0	0	0	2
0	0	1	0	0	0	0	-4
0	0	0	1	0	0	0	8
0	0	0	0	1	0	0	-16
0	0	0	0	0	1	0	32
0	0	0	0	0	0	1	-64
0	0	0	0	0	0	0	128.5

L'errore relativo che si commette ad R(8,8) è:

err = (128.5 - 128)/128

err =   0.0039 (pari a 1/256, cioè lo 0,39%)

$\hat{A} = A + \delta A = L \cdot (R + \delta R)$

A = L*R

A =

1	0	0	0	0	0	0	-1
1	1	0	0	0	0	0	1
-1	1	1	0	0	0	0	-1
1	-1	1	1	0	0	0	1
-1	1	-1	1	1	0	0	-1
1	-1	1	-1	1	1	0	1
-1	1	-1	1	-1	1	1	-1
1	-1	1	-1	1	-1	1	1.5

L'errore relativo che si commette ad A(8,8) è:

```
err = (1.5 - 1) / 1

err =    0.5000  (pari al 50%)
```

A(8,8) dovrebbe essere uguale a 1.

Da questo esempio si capisce che se si ha una piccola perturbazione sulla matrice R, la matrice di partenza ha una perturbazione molto maggiore. La soluzione del sistema quindi sarà diversa rispetto alla soluzione teorica.

Da questo esempio si capisce anche che l'algoritmo ad eliminazione di Gauss con pivottaggio è stabile in senso debole.