```
//SISTEMI LINEARI
creiamo la matrice dei coefficienti e la matrice dei termini noti
A = rand(6)

A =

    0.8147    0.2785    0.9572    0.7922    0.6787    0.7060
    0.9058    0.5469    0.4854    0.9595    0.7577    0.0318
    0.1270    0.9575    0.8003    0.6557    0.7431    0.2769
    0.9134    0.9649    0.1419    0.0357    0.3922    0.0462
    0.6324    0.1576    0.4218    0.8491    0.6555    0.0971
    0.0975    0.9706    0.9157    0.9340    0.1712    0.8235

b = rand(6,1)

b =

    0.6948
    0.3171
    0.9502
    0.0344
    0.4387
    0.3816

//usiamo il comando backslash \ per risolvere il sistema lineare A^-1*b
x = A\b

x =

   -0.6617
   -0.0122
   -0.4586
   -0.2055
    1.7327
    0.9390

%x = inv(A)*b
Quindi posso dire che A*x è uguale a b
A*x

ans =

    0.6948
    0.3171
    0.9502
    0.0344
    0.4387
    0.3816

b

b =

    0.6948
    0.3171
    0.9502
    0.0344
    0.4387
    0.3816
//per verificare l'uguaglianza di due matrici, è necessario fare la
norma(infinito) della differenza
norm(A*x - b,inf)
```

```
ans =

   1.1102e-16//restituisce un valore molto vicino a zero, quindi i valori sono
uguali

//ANALISI DELL'ERRORE IN AVANTI: conoscendo la soluzione di un problema(in
questo caso di un sistema lineare), io la impongo (b è la nostra soluzione) e
vedo quanto la soluzione differisce dai dati.
Numero di condizionamento: dice quanto un errore sui dati influisce sulla
soluzione - è un approccio diverso, si chiama ANALISI DELL'ERRORE ALL'INDIETRO
(quindi un'analisi di un problema fortemente affetto da un numero di
condizionamento) che andiamo ad esaminare
//costruisco un sistema lineare di cui conosco la soluzione, e poi farò la
differenza tra la soluzione che so, e quella che mi trovo

//A è la matrice dei coefficienti
x = A\b

x =

   -0.6617
   -0.0122
   -0.4586
   -0.2055
    1.7327
    0.9390
//scriviamo la soluzione: questa soluzione conviene scriverla come vettore di
tutti uno perché i numeri interi vengono memorizzati esatti dal calcolatore e
poi gli uno sono abbastanza piccoli, e non vado incontro ad errori di overflow
x = ones(6,1)

x =

     1
     1
     1
     1
     1
     1
//mi calcolo il b di conseguenza
b = A*x

b =

    4.2274
    3.6871
    3.5606
    2.4943
    2.8135
    3.9125
//ho costruito un sistema lineare ad hoc: conoscendo la matrice dei
coefficienti(A), ho imposto una soluzione (x) e da qui mi son ricavata b
//adesso faccio finta di non conoscere la soluzione x, e me lo ricalcolo con la
strategia precedente,utilizzando A e b

//calcoliamo quindi la nuova soluzione
x1 = A\b

x1 =

    1.0000
    1.0000
    1.0000
```

```
    1.0000
    1.0000
    1.0000

//noto che dopo gli 1 ho degli zeri dopo la virgola. Questo significa che ci
sono altri numeri diversi da 0 oltre la virgola.

//calcoliamo la norma infinito della differenza tra la soluzione vera e la
soluzione ricavata
norm(x-x1,inf)

ans =

   8.4377e-15
//notiamo che il valore della norma è molto vicino a 0, e va bene. Con gli
stessi dati, usando l'errore in avanti,la norma mi aveva dato un ordine di
grandezza in più (1.1102e-16).
//calcoliamo il numero di condizionamento
cond(A,2) //indico tra parentesi (matrice,indice di condizionamento)

ans =

        91.4705 //ricordiamo che la situazione ottimale del numero di
                condizionamento è circa 1

//se creo una matrice random molto grande, impongo la soluzione ad 1 e calcolo
il vettore dei termini noti
A = rand(100);
x = ones(100,1);
b = A*x;
//se provo quindi a trovare la soluzione tramite calcolo e calcolo la norma
x1 = A\b;
norm(x-x1,inf)
//posso notare che ho perso altri 2 ordini di grandezza.
ans =

   1.0658e-13
//il numero di condizionamento è molto elevato
cond(A,2)

ans =

   1.1478e+03
//se vado a fare l'analisi in avanti invece (vado a vedere quanto si discosta
a*x1 da essere uguale a b) (errore sul termine noto), ho sempre quell'ordine di
grandezza in meno rispetto all'analisi all'indietro
norm(A*x1 - b,inf)

ans =

   4.2633e-14
//vediamo gli altri 2 numeri di condizionamento - condizionamento in norma 2 di
default
cond(A)

ans =

   1.1478e+03
//condizionamento in norma 1
cond(A,1)

ans =
```

```
   3.3102e+03
//condizionamento in norma infinito
cond(A,inf)

ans =

   3.0865e+03
//l'ordine di grandezza rimane lo stesso per tutti i numeri di condizionamento

//creo uno script come base di controllo del funzionamento degli algoritmi.
Scritto un algoritmo per il calcolo della soluzione di un sistema lineare, e
volendone verificare il funzionamento, questo è lo scheletro generale che si
usa:
        -Si costruisce una matrice random di dimensione a scelta,
        -si impone la soluzione,
        -si calcola il vettore dei termini noti corrispondente,
        -si testa l'algoritmo(nel nostro caso l'algoritmo che stavamo testando
        è il backslash),
        -si calcola l'errore all'indietro(la differenza tra la soluzione trovata
         col metodo e la soluzione imposta)

//lo script testa la risoluzione dei sistemi diagonali
-----------------SCRIPT: test  diag-----------
//devo creare una matrice dei coefficienti diagonale:
n = 10;
A = randn(n);
D = diag(diag(A));  %creo mat diag da A
x = ones(n,1);      %impongo soluzione
b = D*x;            %creo vettore termini noti

//Devo risolvere il sistema diagonale usando l'algoritmo che conosco:
calcolo ogni xi come bi/Di

//verificare prima che il sistema ammetta soluzione!
Restituisco errore se il determinante è 0 (ovviamente non esiste lo zero come
numero perfetto, quindi usiamo un numero MOLTO piccolo)
if det(D)<1e-10
        error('Determinante nullo');
else

x1=zeros(n,1); //prealloco il vettore x1,altrimenti ho un warning

faccio un for per calcolare tutti gli elementi di x1

for i=1:n
        x1(i)=b(i)/D(i,i);
    end

norm(x-x1,inf)

-----------------------------------------------
clear,clc
test_diag
{Error using <a
href="matlab:matlab.internal.language.introspective.errorDocCallback('test_diag'
, 'C:\Users\LabT_Pal_Sc\Desktop\CSMN_2223\Lezione5\test_diag.m', 8)"
style="font-weight:bold">test_diag</a> (<a href="matlab:
opentoline('C:\Users\LabT_Pal_Sc\Desktop\CSMN_2223\Lezione5\test_diag.m',8,0)">l
ine 8</a>)

Determinante nullo!

}
```

```
test_diag
{Error using <a
href="matlab:matlab.internal.language.introspective.errorDocCallback('test_diag'
, 'C:\Users\LabT_Pal_Sc\Desktop\CSMN_2223\Lezione5\test_diag.m', 8)"
style="font-weight:bold">test_diag</a> (<a href="matlab:
opentoline('C:\Users\LabT_Pal_Sc\Desktop\CSMN_2223\Lezione5\test_diag.m',8,0)">l
ine 8</a>)
Determinante nullo!
}
//questa matrice creata ha troppi valori vicini allo 0, perché sto usando una
gaussiana con centro 0. Modifico la generazione della matrice A aggiungengo un
coefficiente che sposti il centro della gaussiana:
-----------------SCRIPT: test  diag-----------
n = 10;
A = 3+10*randn(n);
D = diag(diag(A));  %creo mat diag da A
x = ones(n,1);      %impongo soluzione
b = D*x;            %creo vettore termini noti

if det(D)<1e-10
        error('Determinante nullo');
else

x1=zeros(n,1);

for i=1:n
        x1(i)=b(i)/D(i,i);
    end

norm(x-x1,inf)

-----------------------------------------------


test_diag

ans =

     0
//questa è la differenza tra la soluzione calcolata e la soluzione imposta.
Queste due soluzioni in questo caso sono perfettamente uguali

-----------------SCRIPT: test  diag-----------
//modifichiamo ulteriormente lo script mettendo due valori casuali come
coefficienti di spostamento della gaussiana di A

n = 100;

a = randi([0,10],1);
c = randi([0,10],1);

A = a+c*randn(n);
D = diag(diag(A));  %creo mat diag da A
x = ones(n,1);      %impongo soluzione
b = D*x;            %creo vettore termini noti

if det(D)<1e-10
        error('Determinante nullo');
else

x1=zeros(n,1);

for i=1:n
```

```
        x1(i)=b(i)/D(i,i);
    end

norm(x-x1,inf)

-----------------------------------------------
//a seonda della dimensione della matrice A, il determinante tende ad essere
nullo sempre meno spesso

test_diag

ans =

     0

test_diag

ans =

     0

test_diag
{Error using <a
href="matlab:matlab.internal.language.introspective.errorDocCallback('test_diag'
, 'C:\Users\LabT_Pal_Sc\Desktop\CSMN_2223\Lezione5\test_diag.m', 11)"
style="font-weight:bold">test_diag</a> (<a href="matlab:
opentoline('C:\Users\LabT_Pal_Sc\Desktop\CSMN_2223\Lezione5\test_diag.m',11,0)">
line 11</a>)
Determinante nullo!
}
test_diag
{Error using <a
href="matlab:matlab.internal.language.introspective.errorDocCallback('test_diag'
, 'C:\Users\LabT_Pal_Sc\Desktop\CSMN_2223\Lezione5\test_diag.m', 11)"
style="font-weight:bold">test_diag</a> (<a href="matlab:
opentoline('C:\Users\LabT_Pal_Sc\Desktop\CSMN_2223\Lezione5\test_diag.m',11,0)">
line 11</a>)
Determinante nullo!
}

-----------------SCRIPT: test  diag-----------

n = 100;

a = randi([0,10],1);
c = randi([0,10],1);

A = a+c*randn(n);
d=diag(A);
D = diag(d);  %creo mat diag da A
x = ones(n,1);      %impongo soluzione
b = D*x;            %creo vettore termini noti

if det(D)<1e-10
        error('Determinante nullo');
else

x1=zeros(n,1);

//possiamo sostituire il ciclo for con una semplice operazione tra matrici:
prendo il vettore b, prendo la diagonale e le divido elemento per elemento. Uso
il punto per effettuare la divisione!
questo permette di eliminare l'allocazione di x1 precedente
```

```
x1=b./d
    end

norm(x-x1,inf)

-------------------------------------------

test_diag

ans =

     0

test_diag

ans =

     0

test_diag
{Error using <a
href="matlab:matlab.internal.language.introspective.errorDocCallback('test_diag'
, 'C:\Users\LabT_Pal_Sc\Desktop\CSMN_2223\Lezione5\test_diag.m', 11)"
style="font-weight:bold">test_diag</a> (<a href="matlab:
opentoline('C:\Users\LabT_Pal_Sc\Desktop\CSMN_2223\Lezione5\test_diag.m',11,0)">
line 11</a>)
Determinante nullo!
}
test_diag

ans =

     0

edit backslash
test_diag
{Error using <a
href="matlab:matlab.internal.language.introspective.errorDocCallback('test_diag'
, 'C:\Users\LabT_Pal_Sc\Desktop\CSMN_2223\Lezione5\test_diag.m', 12)"
style="font-weight:bold">test_diag</a> (<a href="matlab:
opentoline('C:\Users\LabT_Pal_Sc\Desktop\CSMN_2223\Lezione5\test_diag.m',12,0)">
line 12</a>)
Determinante nullo!
}
test_diag

ans =

     0

//creo una funzione per poter applicare l'algoritmo nell'ambiente di test (test
diag)
-----------------FUNCTION: Sist  diag-----------
//la prima parola di una funzione è "function". Per poter funzionare, ha bisogno
di input e output. Nel nostro caso l'algoritmo è il sistema lineare, quindi ho
bisogno di inserire nella prima riga i seguenti parametri:
INPUT: matrice D, vettore termini noti b
NOME: sist_diag
OUTPUT: la soluzione x1,

function [x1,d] = sist_diag(D,b) //intestazione funzione
```

```
d = diag(D);

if det(D) < 1e-10
    error('Determinante nullo!')
else
    x1 = b./d;
end
//la function non si chiude con un end!
Lo script della function deve avere lo stesso nome della function!
Tutto ciò che non metto in output in una function non posso utilizzarlo nella
command window!
-------------------------------------------
//adesso posso modificare test_diag
----------------SCRIPT: test  diag----------
n = 100;

a = randi([0,10],1);
c = randi([0,10],1);

A = a+c*randn(n);
D = diag(diag(A));
x = ones(n,1);
b = D*x;

[x1,d] = sist_diag(D,b); //modifico lo script inserendo la funzione
//d deve stare tra gli output se voglio vederlo al di fuori nella command window
norm(x-x1,inf)
-------------------------------------------



clear,clc
test_diag

ans =

     0

test_diag
{Error using <a
href="matlab:matlab.internal.language.introspective.errorDocCallback('sist_diag'
, 'C:\Users\LabT_Pal_Sc\Desktop\CSMN_2223\Lezione5\sist_diag.m', 6)"
style="font-weight:bold">sist_diag</a> (<a href="matlab:
opentoline('C:\Users\LabT_Pal_Sc\Desktop\CSMN_2223\Lezione5\sist_diag.m',6,0)">l
ine 6</a>)
Determinante nullo!
//adesso l'errore 'Determinante nullo!
' si trova nella funzione sist_diag e non più su test_diag!
Su test_diag troviamo solo un errore generico di non poter continuare lo script
Error in <a
href="matlab:matlab.internal.language.introspective.errorDocCallback('test_diag'
, 'C:\Users\LabT_Pal_Sc\Desktop\CSMN_2223\Lezione5\test_diag.m', 11)"
style="font-weight:bold">test_diag</a> (<a href="matlab:
opentoline('C:\Users\LabT_Pal_Sc\Desktop\CSMN_2223\Lezione5\test_diag.m',11,0)">
line 11</a>)
x1 = sist_diag(D,b);
}
test_diag

ans =

     0
```

```
//l'intestazione della function è l'intestazione generale, che deve
corrispondere alle variabili usate ALL'INTERNO della funzione. Gli output e gli
input devono avere gli stessi nomi di quelli all'interno della funzione.
Se però richiamiamo la funzione con variabili con diverso nome, matlab si basa
sulla posizione dei parametri.
clear,clc
test_diag
{Error using <a
href="matlab:matlab.internal.language.introspective.errorDocCallback('sist_diag'
, 'C:\Users\LabT_Pal_Sc\Desktop\CSMN_2223\Lezione5\sist_diag.m', 6)"
style="font-weight:bold">sist_diag</a> (<a href="matlab:
opentoline('C:\Users\LabT_Pal_Sc\Desktop\CSMN_2223\Lezione5\sist_diag.m',6,0)">l
ine 6</a>)
Determinante nullo!

Error in <a
href="matlab:matlab.internal.language.introspective.errorDocCallback('test_diag'
, 'C:\Users\LabT_Pal_Sc\Desktop\CSMN_2223\Lezione5\test_diag.m', 11)"
style="font-weight:bold">test_diag</a> (<a href="matlab:
opentoline('C:\Users\LabT_Pal_Sc\Desktop\CSMN_2223\Lezione5\test_diag.m',11,0)">
line 11</a>)
x1 = sist_diag(D,b);
}
test_diag

ans =

     0

d
{Unrecognized function or variable 'd'.
}
clear,clc
test_diag

ans =

     0

d
{Unrecognized function or variable 'd'.
}
test_diag
{Error using <a
href="matlab:matlab.internal.language.introspective.errorDocCallback('sist_diag'
, 'C:\Users\LabT_Pal_Sc\Desktop\CSMN_2223\Lezione5\sist_diag.m', 6)"
style="font-weight:bold">sist_diag</a> (<a href="matlab:
opentoline('C:\Users\LabT_Pal_Sc\Desktop\CSMN_2223\Lezione5\sist_diag.m',6,0)">l
ine 6</a>)
Determinante nullo!

Error in <a
href="matlab:matlab.internal.language.introspective.errorDocCallback('test_diag'
, 'C:\Users\LabT_Pal_Sc\Desktop\CSMN_2223\Lezione5\test_diag.m', 11)"
style="font-weight:bold">test_diag</a> (<a href="matlab:
opentoline('C:\Users\LabT_Pal_Sc\Desktop\CSMN_2223\Lezione5\test_diag.m',11,0)">
line 11</a>)
x1 = sist_diag(D,b);
}
test_diag

ans =
```

```
     0
//se non mettessi il vettore d negli output dello script, troverei questo
messaggio di errore

d
{Unrecognized function or variable 'd'.
}
clear,clc
test_diag

ans =

     0
//mentre invece, se inserisco d negli output ottendo il vettore contentente i
valori della diagonale
d

d =

    8.5463
    7.3709
    9.7228
    7.0662
   11.6544
   11.6450
    8.1889
   13.7600
    9.7736
   15.0636
   11.6379
   12.7635
    9.4525
   12.0744
   10.2121
    9.4357
    7.5806
    6.9343
   10.2905
    9.7262
    9.6124
   10.0991
    9.7812
    9.5227
   11.3031
    9.9001
    9.7403
    9.2754
    7.3775
    9.2667
    9.3320
   14.1790
   14.1423
    9.4616
   14.1380
    8.5556
    8.9997
   12.2414
    8.9367
   11.3092
    9.8393
    9.5546
   10.6565
    9.6193
```

```
   11.1907
    8.2178
    7.8146
    7.5563
   12.1959
    9.0073
   10.5213
   12.5124
    7.9966
    9.1458
    7.0207
    7.1231
   10.8252
    9.1283
   10.5555
    8.1121
   13.2174
    9.1379
    9.1390
   12.9553
   10.8880
    7.1416
    6.1033
    9.5050
    6.5457
   15.5358
   14.5926
    7.8836
   10.5750
   10.4301
   12.3530
   11.4202
   10.8235
   13.9931
   15.0091
   11.0820
    7.9854
    8.2867
    7.6607
    8.1460
   12.4956
    9.7792
   10.4165
   12.3074
    7.7038
   10.1831
   12.6204
   12.3702
    9.9369
   12.4091
    6.1404
   11.0821
    8.1677
   10.5904
   12.2154
    8.5215

diary off
```

//ESERCIZIO: implementare la function che fa la risoluzione all'indietro o in avanti di un sistema triangolare, sup o inf. L'algoritmo si trova sul libro, bisogna solo capire input/output