```
//creiamo una matrice casuale - usiamo randn cosi abbiamo anche num negativi
A = randn(10)

A =

  Columns 1 through 7

    0.5377   -1.3499    0.6715    0.8884   -0.1022   -0.8637   -1.0891
    1.8339    3.0349   -1.2075   -1.1471   -0.2414    0.0774    0.0326
   -2.2588    0.7254    0.7172   -1.0689    0.3192   -1.2141    0.5525
    0.8622   -0.0631    1.6302   -0.8095    0.3129   -1.1135    1.1006
    0.3188    0.7147    0.4889   -2.9443   -0.8649   -0.0068    1.5442
   -1.3077   -0.2050    1.0347    1.4384   -0.0301    1.5326    0.0859
   -0.4336   -0.1241    0.7269    0.3252   -0.1649   -0.7697   -1.4916
    0.3426    1.4897   -0.3034   -0.7549    0.6277    0.3714   -0.7423
    3.5784    1.4090    0.2939    1.3703    1.0933   -0.2256   -1.0616
    2.7694    1.4172   -0.7873   -1.7115    1.1093    1.1174    2.3505

  Columns 8 through 10

   -0.6156    1.4193   -1.1480
    0.7481    0.2916    0.1049
   -0.1924    0.1978    0.7223
    0.8886    1.5877    2.5855
   -0.7648   -0.8045   -0.6669
   -1.4023    0.6966    0.1873
   -1.4224    0.8351   -0.0825
    0.4882   -0.2437   -1.9330
   -0.1774    0.2157   -0.4390
   -0.1961   -1.1658   -1.7947
//verifichiamo se la matrice è simmetrica: il comando dice se è simmetrica con 1
e se non lo è con 0
issymetric(A)
{Unrecognized function or variable 'issymetric'.
}
issymmetric(A)

ans =

  <a href="matlab:helpPopup logical" style="font-weight:bold">logical</a>

   0
//forziamo la simmetria: moltiplico la matrice trasposta per la matrice

A = A'*A

A =

  Columns 1 through 7

   32.0897   13.1740   -4.8130   -2.9651    6.0853    2.2018    2.6580
   13.1740   18.3444   -5.2602   -9.4638    3.0731    2.1855    3.9010
   -4.8130   -5.2602    7.7173    2.1597   -0.3544   -3.3951   -0.7576
   -2.9651   -9.4638    2.1597   20.1232    1.1670    0.8156  -12.3116
    6.0853    3.0731   -0.3544    1.1670    3.8644    0.6464    0.5127
    2.2018    2.1855   -3.3951    0.8156    0.6464    7.8445    2.9060
    2.6580    3.9010   -0.7576  -12.3116    0.5127    2.9060   14.5232
    3.4379    3.0226   -2.9107   -2.4223    0.9321   -1.2132    1.7516
   -1.8501   -3.5204    5.3209    5.5468   -0.3289   -4.2227   -4.8958
   -7.4523   -4.3176    5.5138    2.1305   -1.9678   -5.0253    1.2902

  Columns 8 through 10
```

```
    3.4379   -1.8501   -7.4523
    3.0226   -3.5204   -4.3176
   -2.9107    5.3209    5.5138
   -2.4223    5.5468    2.1305
    0.9321   -0.3289   -1.9678
   -1.2132   -4.2227   -5.0253
    1.7516   -4.8958    1.2902
    6.6480   -0.7609    2.7944
   -0.7609    7.9543    5.7160
    2.7944    5.7160   16.1720
//verifichiamo la simmetria
issymmetric(A)

ans =

  <a href="matlab:helpPopup logical" style="font-weight:bold">logical</a>

   1
//calcolo AUTOVALORI: eig(), genera un vettore colonna
lambda = eig(A)

lambda = //genera valori reali perché A è simmetrica,ma sono anche positivi
percgè ho fatto A'*A

    0.1989
    0.8898
    1.9214
    3.1655
    6.6788
    9.3638
   10.5014
   20.3824
   29.3344
   52.8445

//se mi interessano gli autovettori chiedo 2 output ad eig. In questo caso gli
autovalori vengono messi in una matrice diagonale, e gli autovettori vengono
messi nella colonna degli autovettori

[V,D] = eig(A)
//v contiene in ogni colonna gli autovettori
V =

  Columns 1 through 7

    0.0321   -0.0678   -0.0661    0.2447    0.1799    0.0719   -0.3507
   -0.2556    0.0316   -0.0587   -0.0362   -0.5742   -0.0038    0.5939
   -0.2993   -0.6264    0.0148   -0.2179   -0.2747    0.4511   -0.2592
   -0.4549    0.1875   -0.1243   -0.1059   -0.2334   -0.3781   -0.1194
    0.4974    0.0865   -0.1642   -0.8057   -0.1371   -0.0397   -0.0903
    0.3124   -0.3431    0.3952    0.1998   -0.4813   -0.3439   -0.2790
   -0.3736    0.4004    0.0762   -0.1616   -0.1361   -0.1451   -0.5566
   -0.2221   -0.3475    0.4298   -0.3069    0.4175   -0.5151    0.2072
    0.1446    0.3982    0.7157    0.0234   -0.1528    0.2616    0.0586
    0.2868   -0.0452   -0.3010    0.2649   -0.2036   -0.4168   -0.0249

  Columns 8 through 10

    0.3676   -0.5030   -0.6176
    0.1344    0.0331   -0.4768
    0.2744    0.0375    0.2179
   -0.0655   -0.6250    0.3530
    0.0660   -0.1483   -0.1185
```

```
   -0.3784   -0.0344   -0.1282
    0.0744    0.5044   -0.2426
    0.2331    0.0566   -0.1044
    0.3761   -0.1662    0.2084
    0.6467    0.2131    0.2748

//D è una matrice diagonale che contiene gli autovalori
D =

  Columns 1 through 7

    0.1989         0         0         0         0         0         0
         0    0.8898         0         0         0         0         0
         0         0    1.9214         0         0         0         0
         0         0         0    3.1655         0         0         0
         0         0         0         0    6.6788         0         0
         0         0         0         0         0    9.3638         0
         0         0         0         0         0         0   10.5014
         0         0         0         0         0         0         0
         0         0         0         0         0         0         0
         0         0         0         0         0         0         0

  Columns 8 through 10

         0         0         0
         0         0         0
         0         0         0
         0         0         0
         0         0         0
         0         0         0
         0         0         0
   20.3824         0         0
         0   29.3344         0
         0         0   52.8445

lambda

lambda =

    0.1989
    0.8898
    1.9214
    3.1655
    6.6788
    9.3638
   10.5014
   20.3824
   29.3344
   52.8445
//se dovessi calcolare V*D*V'(che si ottiene con la fattorizzazione spettrale),
dovrei ottenere A

V*D*V'

ans =

  Columns 1 through 7

   32.0897   13.1740   -4.8130   -2.9651    6.0853    2.2018    2.6580
   13.1740   18.3444   -5.2602   -9.4638    3.0731    2.1855    3.9010
   -4.8130   -5.2602    7.7173    2.1597   -0.3544   -3.3951   -0.7576
   -2.9651   -9.4638    2.1597   20.1232    1.1670    0.8156  -12.3116
    6.0853    3.0731   -0.3544    1.1670    3.8644    0.6464    0.5127
```

```
    2.2018    2.1855   -3.3951    0.8156    0.6464    7.8445    2.9060
    2.6580    3.9010   -0.7576  -12.3116    0.5127    2.9060   14.5232
    3.4379    3.0226   -2.9107   -2.4223    0.9321   -1.2132    1.7516
   -1.8501   -3.5204    5.3209    5.5468   -0.3289   -4.2227   -4.8958
   -7.4523   -4.3176    5.5138    2.1305   -1.9678   -5.0253    1.2902

  Columns 8 through 10

    3.4379   -1.8501   -7.4523
    3.0226   -3.5204   -4.3176
   -2.9107    5.3209    5.5138
   -2.4223    5.5468    2.1305
    0.9321   -0.3289   -1.9678
   -1.2132   -4.2227   -5.0253
    1.7516   -4.8958    1.2902
    6.6480   -0.7609    2.7944
   -0.7609    7.9543    5.7160
    2.7944    5.7160   16.1720

A

A =

  Columns 1 through 7

   32.0897   13.1740   -4.8130   -2.9651    6.0853    2.2018    2.6580
   13.1740   18.3444   -5.2602   -9.4638    3.0731    2.1855    3.9010
   -4.8130   -5.2602    7.7173    2.1597   -0.3544   -3.3951   -0.7576
   -2.9651   -9.4638    2.1597   20.1232    1.1670    0.8156  -12.3116
    6.0853    3.0731   -0.3544    1.1670    3.8644    0.6464    0.5127
    2.2018    2.1855   -3.3951    0.8156    0.6464    7.8445    2.9060
    2.6580    3.9010   -0.7576  -12.3116    0.5127    2.9060   14.5232
    3.4379    3.0226   -2.9107   -2.4223    0.9321   -1.2132    1.7516
   -1.8501   -3.5204    5.3209    5.5468   -0.3289   -4.2227   -4.8958
   -7.4523   -4.3176    5.5138    2.1305   -1.9678   -5.0253    1.2902

  Columns 8 through 10

    3.4379   -1.8501   -7.4523
    3.0226   -3.5204   -4.3176
   -2.9107    5.3209    5.5138
   -2.4223    5.5468    2.1305
    0.9321   -0.3289   -1.9678
   -1.2132   -4.2227   -5.0253
    1.7516   -4.8958    1.2902
    6.6480   -0.7609    2.7944
   -0.7609    7.9543    5.7160
    2.7944    5.7160   16.1720

//per verificare che A e V*D*V' siano uguali, faccio la differenza tra i due e
calcolo una norma(in questo caso infinito). Dato che la norma infinito calcola
le somme riga e prende il massimo tra queste somme. Quindi la norma infinito
della differenza, dovrebbe essere molto vicino a zero se le due matrici sono
uguali
norm(A-V*D*V',inf)

ans =

   6.5059e-14 //questo numero è molto piccolo, va bene! Ricordiamo che in matlab
lo zero non esiste, se non in casi particolari
```

```
//calcolo norma 2, è il caso di default

norm(A,2)

ans =

   52.8445
//norma 2
norm(A)

ans =

   52.8445
//norma 1
norm(A,1)

ans =

   76.7272
//inizializziamo una matrice NON simmetrica,gli autovalori di una matrice simile
sono complessi
B = randn(5)

B =

    0.8404   -0.6003   -2.1384    0.1240    2.9080
   -0.8880    0.4900   -0.8396    1.4367    0.8252
    0.1001    0.7394    1.3546   -1.9609    1.3790
   -0.5445    1.7119   -1.0722   -0.1977   -1.0582
    0.3035   -0.1941    0.9610   -1.2078   -0.4686

eig(B)

ans =

   3.0999 + 0.0000i
   2.0100 + 0.0000i
  -2.0700 + 0.0000i
  -0.5106 + 0.4354i
  -0.5106 - 0.4354i
//possiamo notare che i numeri del vettore sono scritti tutti nella stessa
maniera. I numeri reali vengono visualizzati con la parte immaginaria a 0.
I numeri complessi in questo caso sono sempre complessi coniugati

//calcoliamo il raggio spettrale:
        abs calcola il valore assoluto
        max calcola il massimo
rho = max(abs(ans)) //ans è answer in matlab,si usa quando non salvo un valore
su una variabile

rho =

    3.0999

lambdaB = eig(B);
abs(lambdaB)

ans =

    3.0999
    2.0100
    2.0700
    0.6711
```

```
    0.6711
//calcoliamo il raggio spettrale di A
rhoA = max(abs(lambda))

rhoA =

   52.8445
//calcoliamo la norma 2 di A
norm(A)

ans =

   52.8445
//A è una matrice simmetrica, quindi il raggio spettrale e la norma 2 coincidono

x = rand(7,1)

x =

    0.2599
    0.8001
    0.4314
    0.9106
    0.1818
    0.2638
    0.1455
//la funzione max calcola il massimo, e se applicato ad un vettore mi
restituisce il valore dell'elemento massimo
max(x)

ans =

    0.9106
//a me interessa però sapere la posizione in cui si trova il massimo: mi faccio
restituire due output:
        mx è il valore max
        imx è la posizione nel vettore - 4 elemento nel nostro caso
[mx,imx] = max(x)

mx =

    0.9106


imx =

     4

B

B =

    0.8404   -0.6003   -2.1384    0.1240    2.9080
   -0.8880    0.4900   -0.8396    1.4367    0.8252
    0.1001    0.7394    1.3546   -1.9609    1.3790
   -0.5445    1.7119   -1.0722   -0.1977   -1.0582
    0.3035   -0.1941    0.9610   -1.2078   -0.4686

//se applico max ad una matrice, mi restituisce il massimo di ogni colonna
max(B)

ans =
```

```
    0.8404    1.7119    1.3546    1.4367    2.9080
//se chiamo max con il doppio output, restituisce il max di ogni colonna e la
posizione di ogni colonna
[mxB,imxB] = max(B)

mxB =

    0.8404    1.7119    1.3546    1.4367    2.9080


imxB =

     1     4     3     2     1

0.8404    1.7119    1.3546    1.4367    2.9080
 0.8404    1.7119    1.3546    1.4367    2.9080
           ↑
{Error: Invalid expression. Check for missing multiplication
operator, missing or unbalanced delimiters, or other syntax error.
To construct matrices, use brackets instead of parentheses.
}

//per trovare il massimo tra i massimi di colonna, è sufficiente fare il max dei
valori massimi
mmx = max(mxB)

mmx =

    2.9080
//vediamo con il doppio output vedo sia il valore che dove si trova il massimo
dei massimi
[mmx,immx] = max(mxB)

mmx =

    2.9080


immx =

     5
//quindi, il massimo di una matrice si ottiene con questa formula
max(max(B))

ans =

    2.9080
//per fare il massimo di ogni riga, devo fare il max riguardo la trasposta di B
max(B')
ans =

    2.9080    1.4367    1.3790    1.7119    0.9610

//ATTENZIONE: il vettore risultante è sempre il vettore riga, quindi se vogliamo
il vettore colonna va trasposto!
(max(B'))'

ans =

    2.9080
    1.4367
    1.3790
    1.7119
```

```
    0.9610
//MATRICI PARTICOLARI
//abbiamo visto issymmetric per le matrici simmetrica
//abbiamo ishermitian per una matrice hermitiana, restituisce 1/0
help ishermitian
 <strong>ishermitian</strong> Determine whether a matrix is real symmetric or
complex Hermitian.
    <strong>ishermitian</strong>(X) returns true if a square matrix X is
Hermitian.
    That is, X equals to X'. Otherwise, it returns false.

    <strong>ishermitian</strong>(X,'skew') returns true if a square matrix X is
skew-Hermitian.
    That is, X equals to -X'. Otherwise, it returns false.

    <strong>ishermitian</strong>(X, 'nonskew') is same as
<strong>ishermitian</strong>(X).

    X must be a double or single matrix.

    See also <a href="matlab:help issymmetric">issymmetric</a>.

    <a href="matlab:doc ishermitian">Documentation for ishermitian</a>
//genero una matrice complessa con numero casuali. Genero in maniera casuale la
parte reale e la parte immaginaria

C = rand(3)+i*rand(3)

C =

   0.9561 + 0.6491i   0.2348 + 0.4509i   0.0154 + 0.7447i
   0.5752 + 0.7317i   0.3532 + 0.5470i   0.0430 + 0.1890i
   0.0598 + 0.6477i   0.8212 + 0.2963i   0.1690 + 0.6868i
//verifico che sia hermitiana
ishermitian(C)

ans =

  <a href="matlab:helpPopup logical" style="font-weight:bold">logical</a>

   0
//faccio la matrice aggiunta di C:
        -cambio il segno da + a -, creando una matrice di elementi complessi
        coniugati alla prima matrice
        -faccio la trasposizione elemento su elemento C.'
C = rand(3)-i*rand(3)

C =

   0.1835 - 0.4468i   0.7802 - 0.5108i   0.7757 - 0.6443i
   0.3685 - 0.3063i   0.0811 - 0.8176i   0.4868 - 0.3786i
   0.6256 - 0.5085i   0.9294 - 0.7948i   0.4359 - 0.8116i

C = C.'

C =

   0.1835 - 0.4468i   0.3685 - 0.3063i   0.6256 - 0.5085i
   0.7802 - 0.5108i   0.0811 - 0.8176i   0.9294 - 0.7948i
   0.7757 - 0.6443i   0.4868 - 0.3786i   0.4359 - 0.8116i

//l'alternativa è di fare direttamente la matrice aggiunta con C'
C = rand(3)+i*rand(3)
```

```
C =

   0.9234 + 0.5949i   0.9049 + 0.7112i   0.1111 + 0.2967i
   0.4302 + 0.2622i   0.9797 + 0.2217i   0.2581 + 0.3188i
   0.1848 + 0.6028i   0.4389 + 0.1174i   0.4087 + 0.4242i
//otteniamo infatti la matrice aggiunta
C'

ans =

   0.9234 - 0.5949i   0.4302 - 0.2622i   0.1848 - 0.6028i
   0.9049 - 0.7112i   0.9797 - 0.2217i   0.4389 - 0.1174i
   0.1111 - 0.2967i   0.2581 - 0.3188i   0.4087 - 0.4242i
//se quindi faccio la moltiplicazione tra una matrice aggiunta e la matrice
originale, ottengo una matrice hermitiana

E = C'*C

E =

   1.8579 + 0.0000i   1.8902 - 0.2860i   0.8049 + 0.1093i
   1.8902 + 0.2860i   2.5401 + 0.0000i   0.8643 + 0.5827i
   0.8049 - 0.1093i   0.8643 - 0.5827i   0.6156 + 0.0000i

ishermitian(E)

ans =

  <a href="matlab:helpPopup logical" style="font-weight:bold">logical</a>

   1

//facciamo gli autovalori di questa matrice hermitiana
eig(E)

ans =

    0.0239
    0.4169
    4.5728
//gli autovalori di una matrice hermitiana(quindi con valori complessi), ha
comunque valori reali, come per le matrici simmetriche

//MATRICI ORTOGONALI: la trasposta coincide con l'inversa
B

B =

    0.8404   -0.6003   -2.1384    0.1240    2.9080
   -0.8880    0.4900   -0.8396    1.4367    0.8252
    0.1001    0.7394    1.3546   -1.9609    1.3790
   -0.5445    1.7119   -1.0722   -0.1977   -1.0582
    0.3035   -0.1941    0.9610   -1.2078   -0.4686
//per ottenerla, chiedo a matlab di ortogonalizzare una matrice che ho già
//ortogonalizzare una matrice significa trovare una base ortogonale nello spazio
generato dalla matrice
//procedimento di graham-shmidth: si parte dalla prima colonna della matrice e
la si normalizza ->la prima colonna della matrice è un vettore, la divido con la
sua norma, e ottengo un vettore con norma 1
Il secondo vettore sarà diverso dal primo. Per ortogonalizzarlo rispetto al
primo sottraggo la componente del primo vettore sul secondo, e vado avanti così
//questo è il metodo che usa matlab per generare la matrice ortogonale
```

```
Q = orth(B)

Q =

   -0.8748    0.2990   -0.0883   -0.3573    0.0992
   -0.3376   -0.3359   -0.1969    0.7223    0.4612
    0.1334    0.7447   -0.5087    0.3760   -0.1659
    0.1104   -0.3882   -0.8318   -0.3793    0.0371
    0.3011    0.3043    0.0532   -0.2557    0.8652
//quindi in questo caso dovrei ottenere la matrice identità
Q*Q'

ans =

    1.0000   -0.0000   -0.0000    0.0000    0.0000
   -0.0000    1.0000    0.0000   -0.0000    0.0000
   -0.0000    0.0000    1.0000    0.0000   -0.0000
    0.0000   -0.0000    0.0000    1.0000    0.0000
    0.0000    0.0000   -0.0000    0.0000    1.0000

//noto che ci sono un sacco di 0 decimali, significa che non sono zero assoluti,
perché sono ottenuti da operazioni che non restituiscono mai zero in matlab
//per sapere quanto la matrice ottenuta si discosta dalla matrice identità,
faccio la norma infinito tra la differenza della matrice con la matrice identità
generata da matlab
forseI = ans;
norm(forseI - eye(5),inf)

ans =

   1.0591e-15 //la norma è molto vicina allo zero, le due matrici sono molto
simili
//se dovessi cambiare un solo valore nella matrice ottenuta, allora la norma si
discosterebbe molto da 0
forseI(1,5) = 2

forseI =

    1.0000   -0.0000   -0.0000    0.0000    2.0000
   -0.0000    1.0000    0.0000   -0.0000    0.0000
   -0.0000    0.0000    1.0000    0.0000   -0.0000
    0.0000   -0.0000    0.0000    1.0000    0.0000
    0.0000    0.0000   -0.0000    0.0000    1.0000

norm(forseI - eye(5),inf)

ans =

    2.0000 //infatti 2>>0
//PROPRIETA' MATRICI ORTOGONALI

//calcoliamo il determinante di Q: le matrici ortogonali hanno modulo del
determinante 1 (quindi nel caso reale il determinante può valere +-1)
det(Q)

ans =

    1.0000 //det di Q è 1

x = randn(5,1) //creo un vettore colonna casuale

x =
```

```
    0.0125
   -3.0292
   -0.4570
    1.2424
   -1.0667
//se calcolo la norma di un vettore che ottengo facendo la moltiplicazione con
una matrice ortogonale, la norma è uguale a quella del vettore di partenza
norm_x = norm(x,2) //calcolo la norma 2 del vettore creato

norm_x =

    3.4737

norm_Qx = norm(Q*x,2) //faccio la norma del vettore moltiplicato per Q
ortogonale

norm_Qx =

    3.4737//identica alla norma di x

B

B =

    0.8404   -0.6003   -2.1384    0.1240    2.9080
   -0.8880    0.4900   -0.8396    1.4367    0.8252
    0.1001    0.7394    1.3546   -1.9609    1.3790
   -0.5445    1.7119   -1.0722   -0.1977   -1.0582
    0.3035   -0.1941    0.9610   -1.2078   -0.4686
//MATRICI TRIANGOLARI: vengono generate da una matrice di partenza

//generazione matrice triangolare superiore
U = triu(B)

U =

    0.8404   -0.6003   -2.1384    0.1240    2.9080
         0    0.4900   -0.8396    1.4367    0.8252
         0         0    1.3546   -1.9609    1.3790
         0         0         0   -0.1977   -1.0582
         0         0         0         0   -0.4686
//generazione matrice triangolare inferiore
L = tril(B)

L =

    0.8404         0         0         0         0
   -0.8880    0.4900         0         0         0
    0.1001    0.7394    1.3546         0         0
   -0.5445    1.7119   -1.0722   -0.1977         0
    0.3035   -0.1941    0.9610   -1.2078   -0.4686
//posso usare triu e tril con un secondo input. Il secondo input indica quante
diagonali voglio mantenere nella mia matrice sopra(o sotto) la diagonale
principale
L = tril(B,0) //in questo caso (base) tutto ciò che sta sopra la diagonale
principale è = 0

L =

    0.8404         0         0         0         0
   -0.8880    0.4900         0         0         0
    0.1001    0.7394    1.3546         0         0
   -0.5445    1.7119   -1.0722   -0.1977         0
```

```
    0.3035   -0.1941    0.9610   -1.2078   -0.4686
```

L = tril(B,1) //in questo caso, sopra la diagonale principale c'è un'altra diagonale diversa da zero.

```
L =

    0.8404   -0.6003         0         0         0
   -0.8880    0.4900   -0.8396         0         0
    0.1001    0.7394    1.3546   -1.9609         0
   -0.5445    1.7119   -1.0722   -0.1977   -1.0582
    0.3035   -0.1941    0.9610   -1.2078   -0.4686
```

L = tril(B,2) //in questo caso le diagonali diverse da 0 sopra la diagonale principale sono 2

```
L =

    0.8404   -0.6003   -2.1384         0         0
   -0.8880    0.4900   -0.8396    1.4367         0
    0.1001    0.7394    1.3546   -1.9609    1.3790
   -0.5445    1.7119   -1.0722   -0.1977   -1.0582
    0.3035   -0.1941    0.9610   -1.2078   -0.4686
```

//…e così via

L = tril(B,3)

```
L =

    0.8404   -0.6003   -2.1384    0.1240         0
   -0.8880    0.4900   -0.8396    1.4367    0.8252
    0.1001    0.7394    1.3546   -1.9609    1.3790
   -0.5445    1.7119   -1.0722   -0.1977   -1.0582
    0.3035   -0.1941    0.9610   -1.2078   -0.4686
```

L = tril(B,4)

```
L =

    0.8404   -0.6003   -2.1384    0.1240    2.9080
   -0.8880    0.4900   -0.8396    1.4367    0.8252
    0.1001    0.7394    1.3546   -1.9609    1.3790
   -0.5445    1.7119   -1.0722   -0.1977   -1.0582
    0.3035   -0.1941    0.9610   -1.2078   -0.4686
```

//vale anche per numeri negativi, per eliminare diagonali visibili. Nel caso -1 ho la diagonale principale a 0

L = tril(B,-1)

```
L =

         0         0         0         0         0
   -0.8880         0         0         0         0
    0.1001    0.7394         0         0         0
   -0.5445    1.7119   -1.0722         0         0
    0.3035   -0.1941    0.9610   -1.2078         0
```

L = tril(B,-2)

```
L =

         0         0         0         0         0
         0         0         0         0         0
    0.1001         0         0         0         0
   -0.5445    1.7119         0         0         0
```

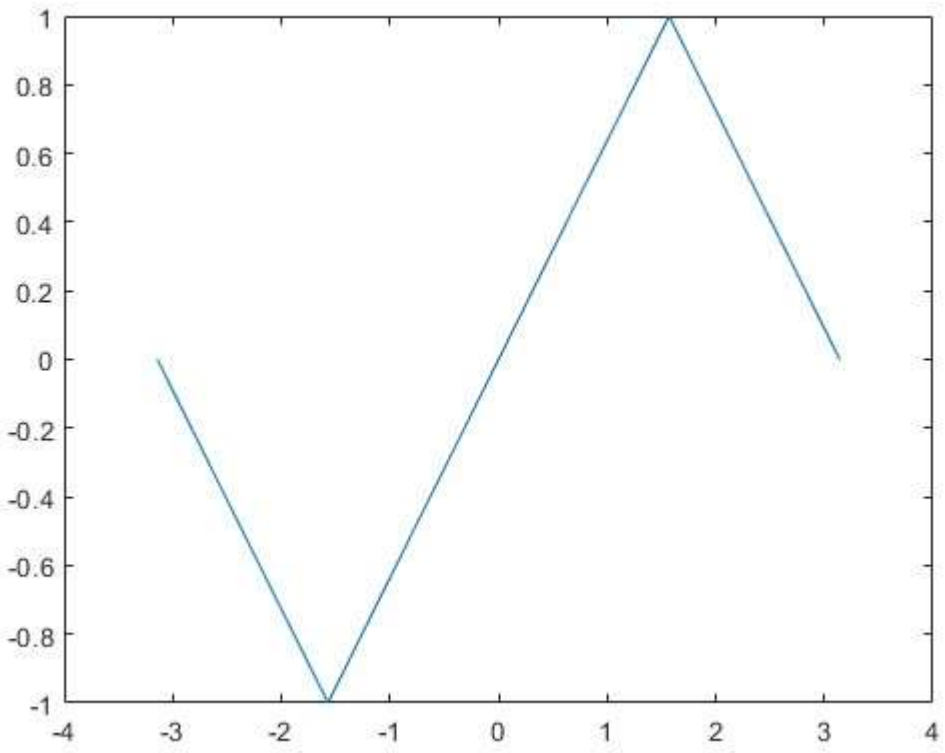
```
    0.3035   -0.1941    0.9610         0         0
//comando per verificare che la matrice sia triangolare o inf/sup, con output
1/0
istriu(U)

ans =

  <a href="matlab:helpPopup logical" style="font-weight:bold">logical</a>

   1

istril(L)

ans =

  <a href="matlab:helpPopup logical" style="font-weight:bold">logical</a>

   1

U = triu(B)

U =

    0.8404   -0.6003   -2.1384    0.1240    2.9080
         0    0.4900   -0.8396    1.4367    0.8252
         0         0    1.3546   -1.9609    1.3790
         0         0         0   -0.1977   -1.0582
         0         0         0         0   -0.4686

istriu(U)

ans =

  <a href="matlab:helpPopup logical" style="font-weight:bold">logical</a>

   1
//ATTENZIONE: se ottengo una matrice trami calcolo, non avrà mai gli elementi a
0 perfetto, ma avrà un numero molto vicino allo 0!
infatti, se almeno uno dei termini è un numero vicino allo zero e non 0, istriu
e istril restituiscono 0
Quindi il verificatore di triangolarità lo riscriveremo a mano
U(5,1) = 1e-16

U =

    0.8404   -0.6003   -2.1384    0.1240    2.9080
         0    0.4900   -0.8396    1.4367    0.8252
         0         0    1.3546   -1.9609    1.3790
         0         0         0   -0.1977   -1.0582
    0.0000         0         0         0   -0.4686

istriu(U)

ans =

  <a href="matlab:helpPopup logical" style="font-weight:bold">logical</a>

   0
//MATRICI DIAGONALI
B

B =
```

```
    0.8404   -0.6003   -2.1384    0.1240    2.9080
   -0.8880    0.4900   -0.8396    1.4367    0.8252
    0.1001    0.7394    1.3546   -1.9609    1.3790
   -0.5445    1.7119   -1.0722   -0.1977   -1.0582
    0.3035   -0.1941    0.9610   -1.2078   -0.4686
//il comando diag si comporta in maniera diversa se applicato alle matrici o ai
vettori
-matrice: diag crea un vettore che contiene gli elementi diagonali
-vettore: diag crea una matrice con il vettore come diagonale
diag(B)

ans =

    0.8404
    0.4900
    1.3546
   -0.1977
   -0.4686

diag(ans)

ans =

    0.8404         0         0         0         0
         0    0.4900         0         0         0
         0         0    1.3546         0         0
         0         0         0   -0.1977         0
         0         0         0         0   -0.4686

//quindi per ricavare la matrice con la diagonale basta innestare i 2 diag
diag(diag(B))

ans =

    0.8404         0         0         0         0
         0    0.4900         0         0         0
         0         0    1.3546         0         0
         0         0         0   -0.1977         0
         0         0         0         0   -0.4686

//proprietà matrici triangolari:
        -autovalori sono gli elementi sulla diagonale
        -determinante è il prodotto degli autovalori
U

U =

    0.8404   -0.6003   -2.1384    0.1240    2.9080
         0    0.4900   -0.8396    1.4367    0.8252
         0         0    1.3546   -1.9609    1.3790
         0         0         0   -0.1977   -1.0582
    0.0000         0         0         0   -0.4686
//ricaviamo la matrice triangolare superiore, inferiore e diagonale di B
U = triu(B)

U =

    0.8404   -0.6003   -2.1384    0.1240    2.9080
         0    0.4900   -0.8396    1.4367    0.8252
         0         0    1.3546   -1.9609    1.3790
         0         0         0   -0.1977   -1.0582
         0         0         0         0   -0.4686
```

```
L = tril(B)

L =

    0.8404         0         0         0         0
   -0.8880    0.4900         0         0         0
    0.1001    0.7394    1.3546         0         0
   -0.5445    1.7119   -1.0722   -0.1977         0
    0.3035   -0.1941    0.9610   -1.2078   -0.4686

D = diag(diag(B))

D =

    0.8404         0         0         0         0
         0    0.4900         0         0         0
         0         0    1.3546         0         0
         0         0         0   -0.1977         0
         0         0         0         0   -0.4686
//controlliamo gli autovalori di U
eig(U)

ans =

    0.8404
    0.4900
    1.3546
   -0.1977
   -0.4686

U

U =

    0.8404   -0.6003   -2.1384    0.1240    2.9080
         0    0.4900   -0.8396    1.4367    0.8252
         0         0    1.3546   -1.9609    1.3790
         0         0         0   -0.1977   -1.0582
         0         0         0         0   -0.4686
//gli stessi autovalori li troviamo per L e D, in ordine di grandezza
eig(L)

ans =

   -0.4686
   -0.1977
    1.3546
    0.4900
    0.8404

eig(D)

ans =

   -0.4686
   -0.1977
    0.4900
    0.8404
    1.3546
//eig nota che la matrice è diagonale o triangolare e invece di calcolare gli
autovalori li prende direttamente dalla diagonale - solo se i valori 0 sono
effettivamente 0, e non un numero molto piccolo. In quel caso fa il calcolo
```

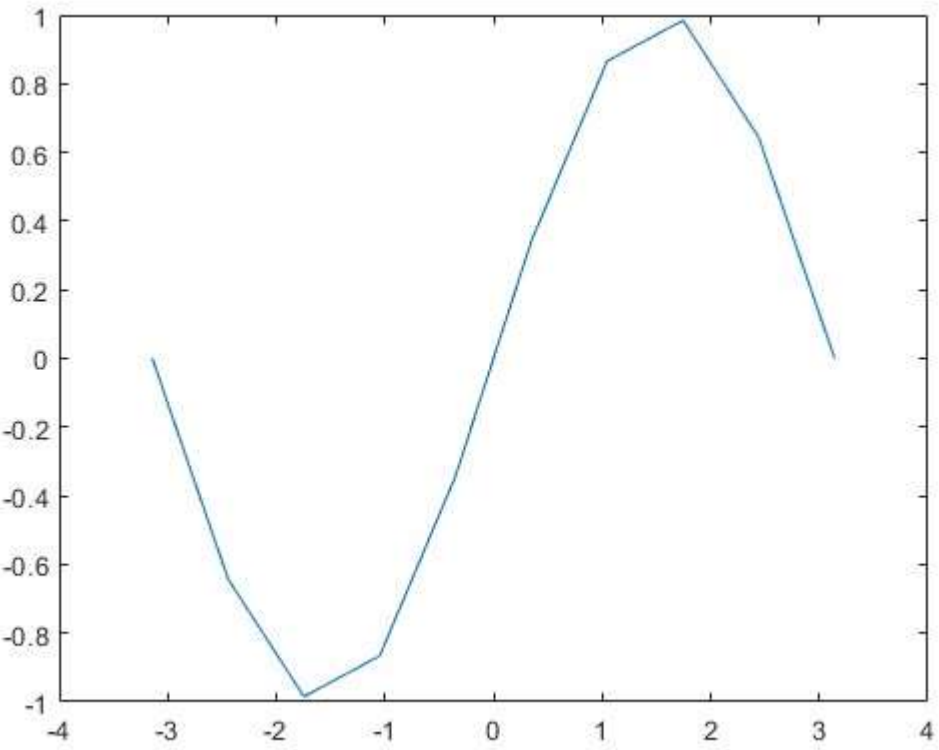
```
//isdiag risponde con 1/0 per verificare se è diagonale o no. Non funziona con
numeri molto piccoli

isdiag(D)

ans =

  <a href="matlab:helpPopup logical" style="font-weight:bold">logical</a>

   1

D

D =

    0.8404         0         0         0         0
         0    0.4900         0         0         0
         0         0    1.3546         0         0
         0         0         0   -0.1977         0
         0         0         0         0   -0.4686

eig(D)

ans =

   -0.4686
   -0.1977
    0.4900
    0.8404
    1.3546

Clear

//GRAFICI

//creo un vettore di ascisse
x = [-pi;-pi/2;0;pi/2;pi]

x =

   -3.1416
   -1.5708
         0
    1.5708
    3.1416
//creo la funzione
y = sin(x)

y =

   -0.0000
   -1.0000
         0
    1.0000
    0.0000
```

```
//creo il grafico con questa funzione, che viene visualizzato con dei
collegamenti diretti
plot(x,y)

---------PLOT: x,y-----------
```



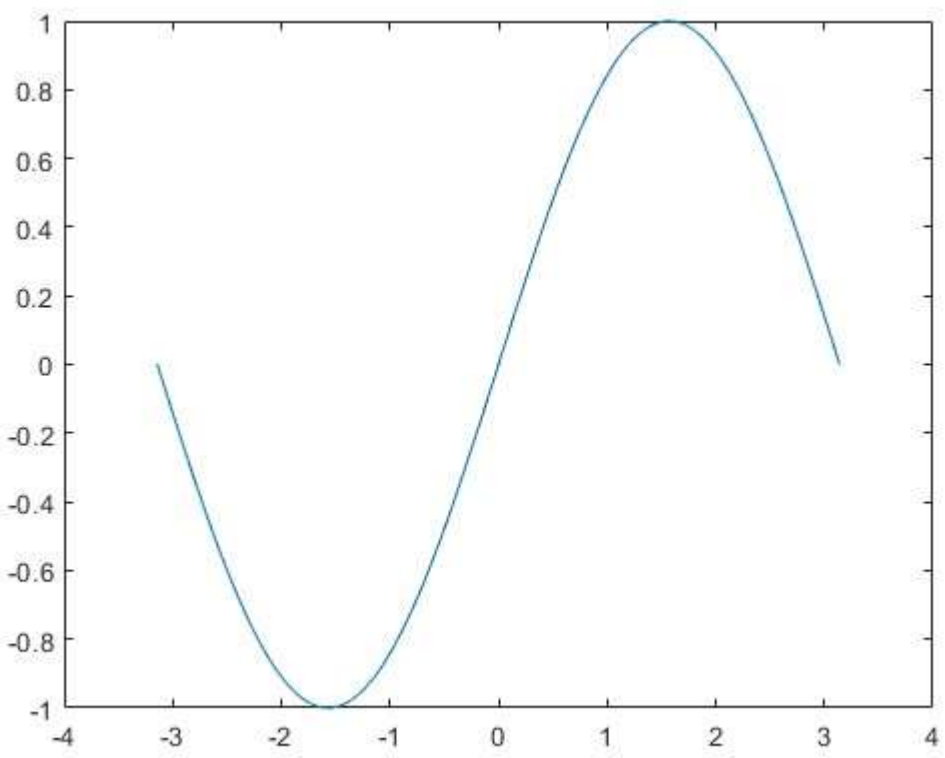

```
-----------------------------

//per forzare l'andamento curvo bisogna fare un campionamento più fitto tramite
la funzione linspace(primo estremo, ultimo estremo, quanti punti di divisione
intervallo)
x = linspace(-pi,pi,10)

x = //attenzione! Se non lo trasponiamo, matlab ci darà un vettore riga, ma a
noi serve un vettore colonna!

  Columns 1 through 6

   -3.1416   -2.4435   -1.7453   -1.0472   -0.3491    0.3491

  Columns 7 through 10

    1.0472    1.7453    2.4435    3.1416
//transpongo il vettore linspace per ottenere il vettore colonna
x = (linspace(-pi,pi,10))'

x =

   -3.1416
   -2.4435
   -1.7453
   -1.0472
   -0.3491
    0.3491
    1.0472
    1.7453
```

```
    2.4435
    3.1416

y = sin(x)

y =

   -0.0000
   -0.6428
   -0.9848
   -0.8660
   -0.3420
    0.3420
    0.8660
    0.9848
    0.6428
    0.0000
//faccio il plot con i nuovi valori: il risultato migliora
plot(x,y)
------------------PLOT: x,y--------------
```




```
----------------------------------------
```

```
//se metto più punti, il grafico migliorerà sempre di più
x = (linspace(-pi,pi,500))';
y = sin(x);
plot(x,y)
----------PLOT:x,y-----------------------
```




```
----------------------------------------
//per modificare le impostazioni, nella finestra del plot:
View-Property editor
        click su linea:
                -marker: vengono posizionati in corrispond delle coordinate
                dei punti
        click su assi
                -X Limits o Y Limits: cambia il range di visualizzazione
                Degli assi

help plot
 <strong>plot</strong>   Linear plot.
    <strong>plot</strong>(X,Y) plots vector Y versus vector X. If X or Y is a
matrix,
    then the vector is plotted versus the rows or columns of the matrix,
    whichever line up.  If X is a scalar and Y is a vector, disconnected
    line objects are created and plotted as discrete points vertically at
    X.

    <strong>plot</strong>(Y) plots the columns of Y versus their index.
    If Y is complex, <strong>plot</strong>(Y) is equivalent to
<strong>plot</strong>(real(Y),imag(Y)).
    In all other uses of <strong>plot</strong>, the imaginary part is ignored.

    Various line types, plot symbols and colors may be obtained with
    <strong>plot</strong>(X,Y,S) where S is a character string made from one
element
    from any or all the following 3 columns:

            b     blue          .     point              -     solid
```

g	green	o	circle	:	dotted
r	red	x	x-mark	-.	dashdot
c	cyan	+	plus	--	dashed
m	magenta	*	star	(none)	no line
y	yellow	s	square		
k	black	d	diamond		
w	white	v	triangle (down)		
		^	triangle (up)		
		<	triangle (left)		
		>	triangle (right)		
		p	pentagram		
		h	hexagram		

For example, plot(X,Y,'c+:') plots a cyan dotted line with a plus
at each data point; plot(X,Y,'bd') plots blue diamond at each data
point but does not draw any line.

plot(X1,Y1,S1,X2,Y2,S2,X3,Y3,S3,...) combines the plots defined by
the (X,Y,S) triples, where the X's and Y's are vectors or matrices
and the S's are strings.

For example, plot(X,Y,'y-',X,Y,'go') plots the data twice, with a
solid yellow line interpolating green circles at the data points.

The plot command, if no color is specified, makes automatic use of
the colors specified by the axes ColorOrder property. By default,
plot cycles through the colors in the ColorOrder property. For
monochrome systems, plot cycles over the axes LineStyleOrder property.

Note that RGB colors in the ColorOrder property may differ from
similarly-named colors in the (X,Y,S) triples. For example, the
second axes ColorOrder property is medium green with RGB [0 .5 0],
while plot(X,Y,'g') plots a green line with RGB [0 1 0].

If you do not specify a marker type, plot uses no marker.
If you do not specify a line style, plot uses a solid line.

plot(AX,...) plots into the axes with handle AX.

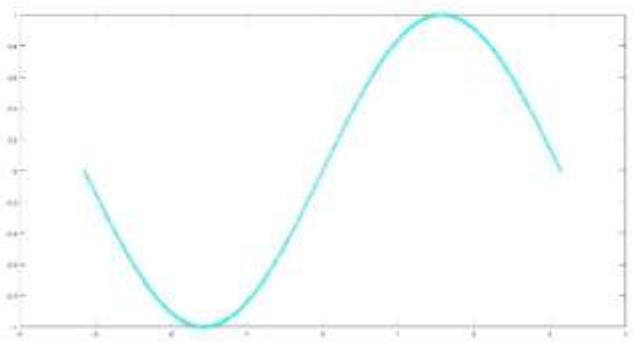
plot returns a column vector of handles to lineseries objects, one
handle per plotted line.

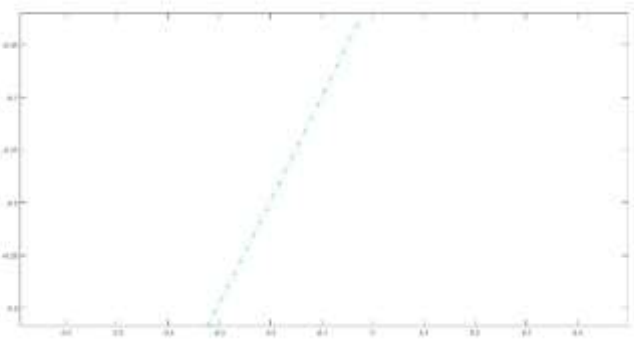
The X,Y pairs, or X,Y,S triples, can be followed by
parameter/value pairs to specify additional properties
of the lines. For example,
plot(X,Y,'LineWidth',2,'Color',[.6 0 0])
will create a plot with a dark red line width of 2 points.

Example

```
x = -pi:pi/10:pi;
y = tan(sin(x)) - sin(tan(x));
plot(x,y,'--rs','LineWidth',2,...
                'MarkerEdgeColor','k',...
                'MarkerFaceColor','g',...
                'MarkerSize',10)
```

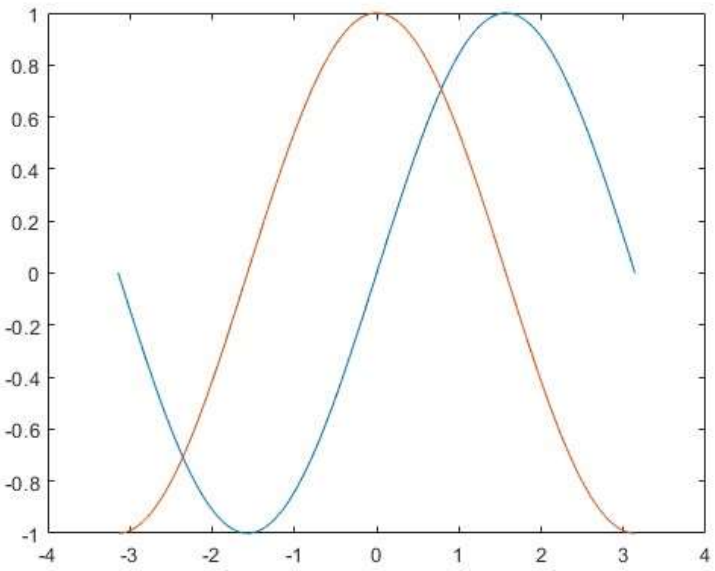
```
    See also <a href="matlab:help plottools">plottools</a>, <a href="matlab:help
semilogx">semilogx</a>, <a href="matlab:help semilogy">semilogy</a>, <a
href="matlab:help loglog">loglog</a>, <a href="matlab:help plotyy">plotyy</a>,
<a href="matlab:help plot3">plot3</a>, <a href="matlab:help grid">grid</a>,
    <a href="matlab:help title">title</a>, <a href="matlab:help
xlabel">xlabel</a>, <a href="matlab:help ylabel">ylabel</a>, <a
href="matlab:help axis">axis</a>, <a href="matlab:help axes">axes</a>, <a
href="matlab:help hold">hold</a>, <a href="matlab:help legend">legend</a>, <a
href="matlab:help subplot">subplot</a>, <a href="matlab:help
scatter">scatter</a>.

    <a href="matlab:doc plot">Documentation for plot</a>
    <a
href="matlab:matlab.internal.language.introspective.overloads.displayOverloads('
plot')">Other functions named plot</a>

//Posso aggiungere le specifiche del plot direttamente in fase di chiamata. Le
specifiche vanno messe tra apici:
//esempio: linea ciano(c) grassetto(+) con stile dotted(:)
plot(x,y,'c+:')

----------PLOT x,y,'c+:'-------
```





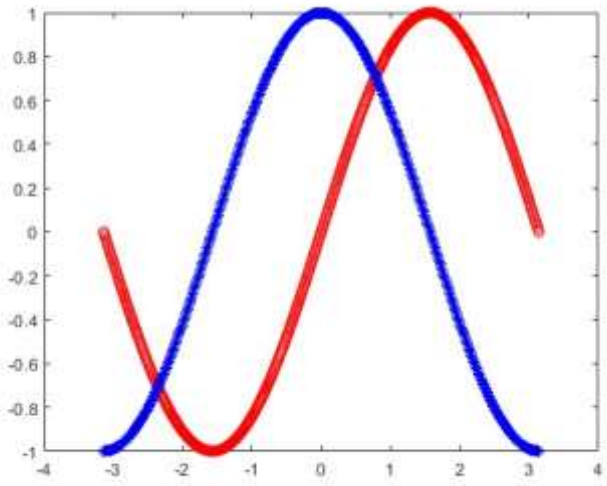
```
--------------------------------
//esempio: due diverse funzioni nello stesso plot: scrivo le coppie (x,y) e
(x,y2)
y2 = cos(x);
plot(x,y,x,y2)
```

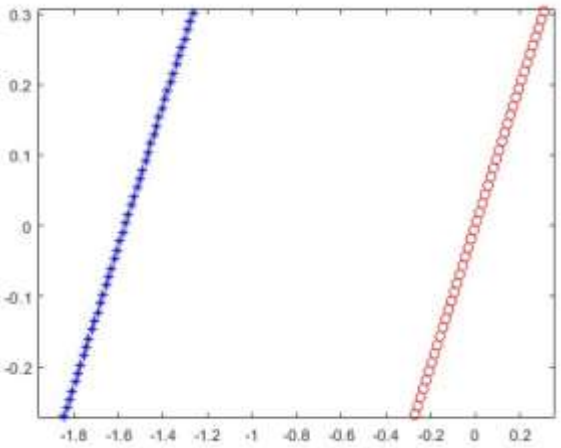
----------PLOT sin-cos----------



```
//disegnamo le due funzioni con colori diversi e con forme diverse
plot(x,y,'ro',x,y2,'b*')
```

-----PLOT: x,y,'ro',x,y2,'b*'----

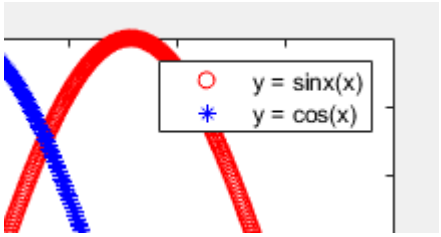




//per capire come distinguere i vari grafici, si scrive una legenda con il plot aperto. Si può spostare trascinandola con il mouse.
la sintassi della legenda e del titolo sono codificate con Latec, e le scrive in maniera ordinata, senza apici

```
legend('y = sinx(x)','y = cos(x)')
```

------PLOT: legenda----------



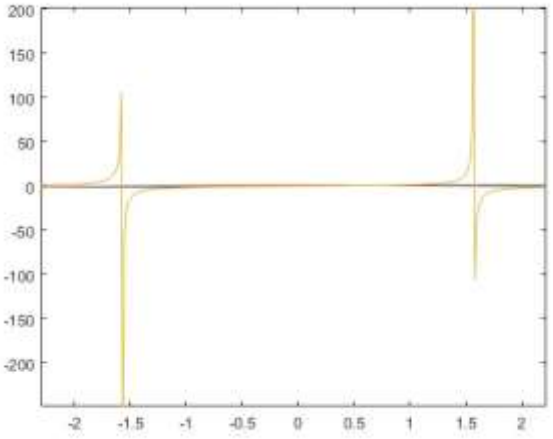
//un altro modo per poter rappresentare due funzioni è far finta che siano due colonne della stessa matrice: Y=[y,y2]. Matlab plotta ogni colonna come se fosse una serie di dati separata

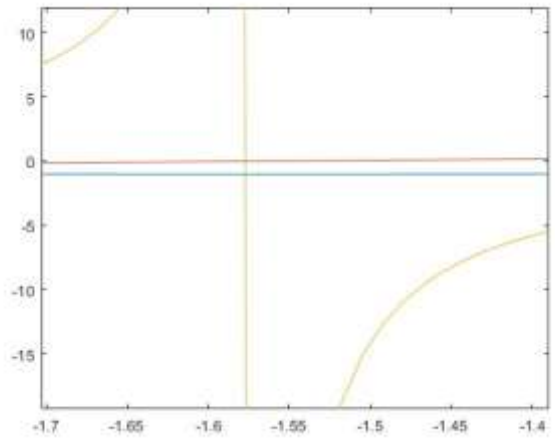
```
plot(x,[y,y2])
Y = [y,y2];
plot(x,Y)
```

//aggiungo una funzione tangente e faccio il plot delle 3 funzioni insieme. Possiamo vedere che la tangente schiaccia sin e cos nella visuale perché punta ad infinito

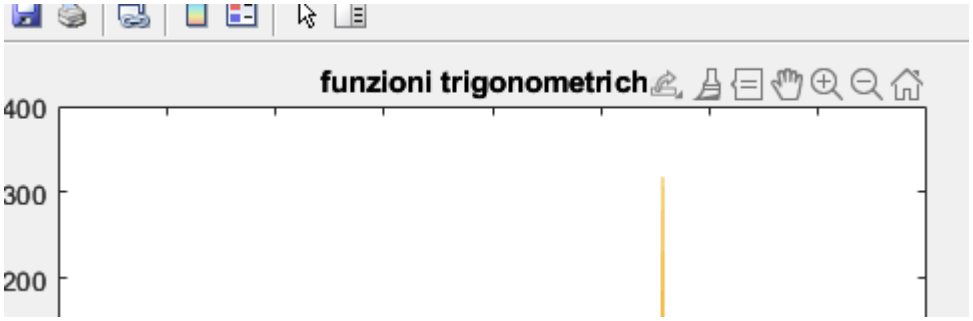
```
y3 = tan(x);
Y = [y,y2,y3];
plot(x,Y)
```

---------PLOT sin,cos,tg----------

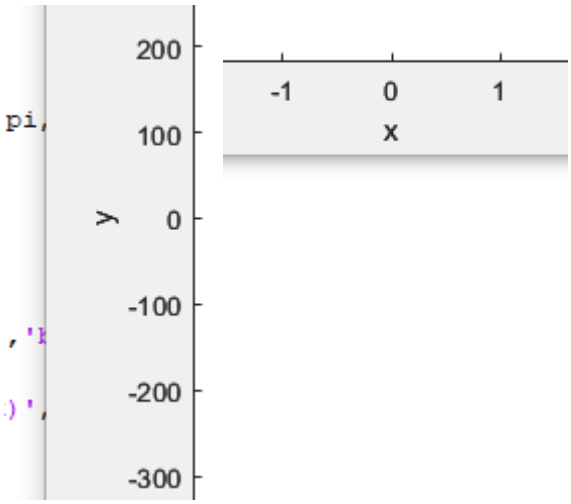




```
//aggiungiamo il titolo al nostro plot
title('funzioni trigonometriche')
plot(x,Y)
title('funzioni trigonometriche')
-------------PLOT:title---------------------
```




```
-----------------------------------
//posso mettere delle etichette per gli assi
xlabel('x')
ylabel('y')
-----------------------------------
```




```
-----------------------------------

diary off
```