

Matlab mette a disposizione, per risolvere sistemi lineari la Routine `pcg`.

Description

`x = pcg(A,b)` attempts to solve the system of linear equations $A*x = b$ for x using the [Preconditioned Conjugate Gradients Method](#). When the attempt is successful, `pcg` displays a message to confirm convergence. If `pcg` fails to converge after the maximum number of iterations or halts for any reason, it displays a diagnostic message that includes the relative residual $\text{norm}(b-A*x)/\text{norm}(b)$ and the iteration number at which the method stopped.

`x = pcg(A,b,tol)` specifies a tolerance for the method. The default tolerance is $1e-6$.

`x = pcg(A,b,tol,maxit)` specifies the maximum number of iterations to use. `pcg` displays a diagnostic message if it fails to converge within `maxit` iterations.

`[x,flag] = pcg(__)` returns a flag that specifies whether the algorithm successfully converged. When `flag = 0`, convergence was successful. You can use this output syntax with any of the previous input argument combinations. When you specify the `flag` output, `pcg` does not display any diagnostic messages.

`[x,flag,relres] = pcg(__)` also returns the relative residual $\text{norm}(b-A*x)/\text{norm}(b)$. If `flag` is 0, then `relres` \leq `tol`.

`[x,flag,relres,iter] = pcg(__)` also returns the iteration number `iter` at which `x` was computed.

`[x,flag,relres,iter,resvec] = pcg(__)` also returns a vector of the residual norm at each iteration, including the first residual $\text{norm}(b-A*x_0)$.



Matlab mette a disposizione, per risolvere sistemi lineari con matrice dei coefficienti simmetrica e definita positiva la Routine pcg.

pcg

Solve system of linear equations — preconditioned conjugate gradients method

Syntax

```
x = pcg(A,b)
x = pcg(A,b,tol)
x = pcg(A,b,tol,maxit)
x = pcg(A,b,tol,maxit,M)
```

```
[x,flag] = pcg(___)
[x,flag,relres] = pcg(___)
[x,flag,relres,iter] = pcg(___)
[x,flag,relres,iter,resvec] = pcg(___)
```

Input Arguments

- > **A** — Coefficient matrix
matrix | function handle
- > **b** — Right-hand side of linear equation
vector
- > **tol** — Method tolerance
[] or 1e-6 (default) | positive scalar
- > **maxit** — Maximum number of iterations
[] or min(size(A,1),20) (default) | positive scalar integer

Matlab mette a disposizione, per risolvere sistemi lineari con matrice dei coefficienti simmetrica e definita positiva la Routine `pcg`.

`pcg`

Solve system of linear equations — preconditioned conjugate gradients method

Syntax

```
x = pcg(A,b)
x = pcg(A,b,tol)
x = pcg(A,b,tol,maxit)
x = pcg(A,b,tol,maxit,M)
```

```
[x,flag] = pcg(___)
[x,flag,relres] = pcg(___)
[x,flag,relres,iter] = pcg(___)
[x,flag,relres,iter,resvec] = pcg(___)

```

Output Arguments

- > **x** — Linear system solution vector
- > **flag** — Convergence flag scalar
- > **relres** — Relative residual error scalar
- > **iter** — Iteration number scalar
- > **resvec** — Residual error vector

Esempio: assegnati A e b rappresentiamo la funzione, le curve di livello, la soluzione calcolata con CG e le direzioni seguite

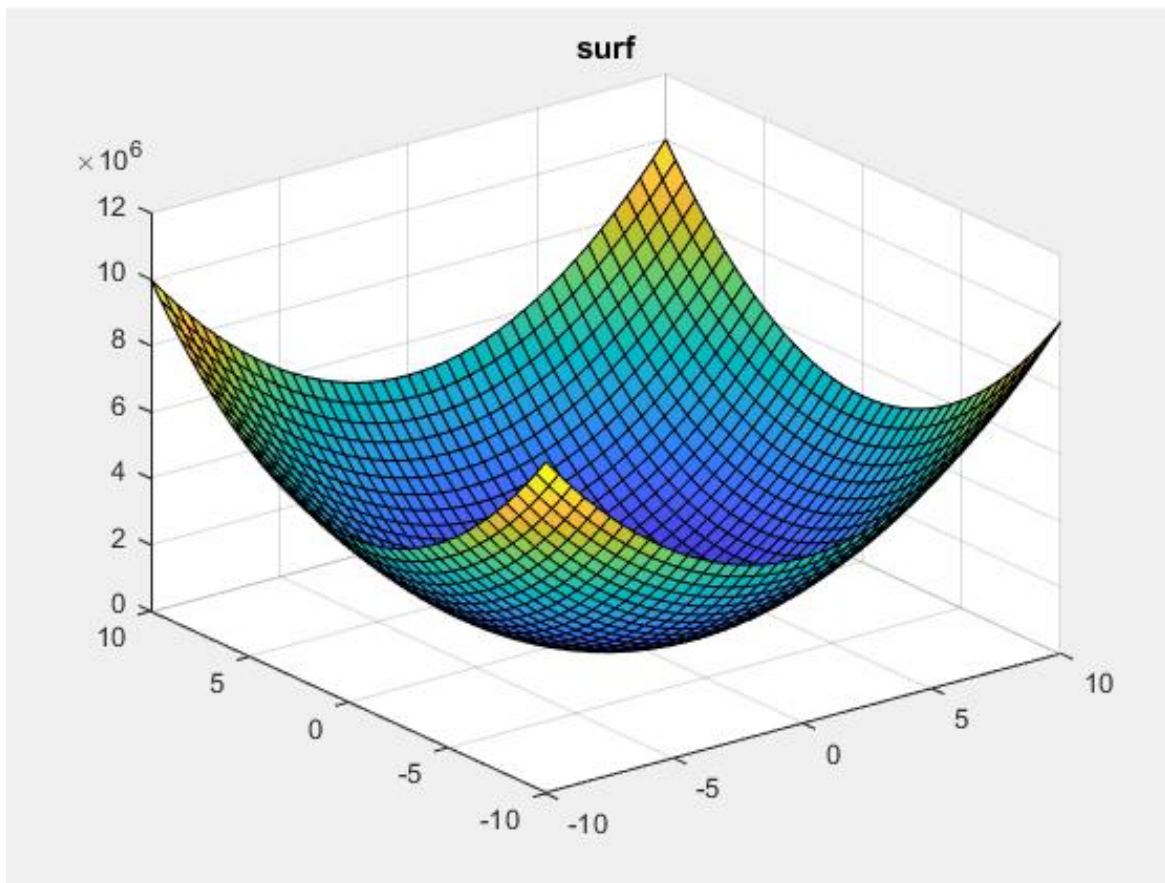
```
a=[10^(5) 2 ; 2 10^(5)];  
%vettore colonna iniziale  
x0=ones(2,1);  
%vettore colonna dei termini noti  
b=[5 3]';  
%massimo numero di iterazioni  
maxit=10;  
%tolleranza  
tol=10^(-4);
```

Esempio: assegnati A e b rappresentiamo la funzione, le curve di livello, la soluzione calcolato con CG e le direzioni seguite

```
x1=(-10:0.5:10);  
y1=(-10:0.5:10);  
[X,Y]=meshgrid(x1,y1) %griglia cartesiana in 2-D  
  
E=(1/2)*(A1*(X).^2+(A2+A3)*(X).*(Y)+A4*(Y).^2)-(b1*X+b2*Y);  
  
%superficie del paraboloide  
figure(1)  
surf(X,Y,E) %traccia la superficie piena  
title('surf')  
%superficie del paraboloide con proiezione delle curve di livello  
  
figure(2)  
surfc(X,Y,E)  
title('surfc')  
figure(4)  
contour(X,Y,E); %curve di livello  
title('contour')
```

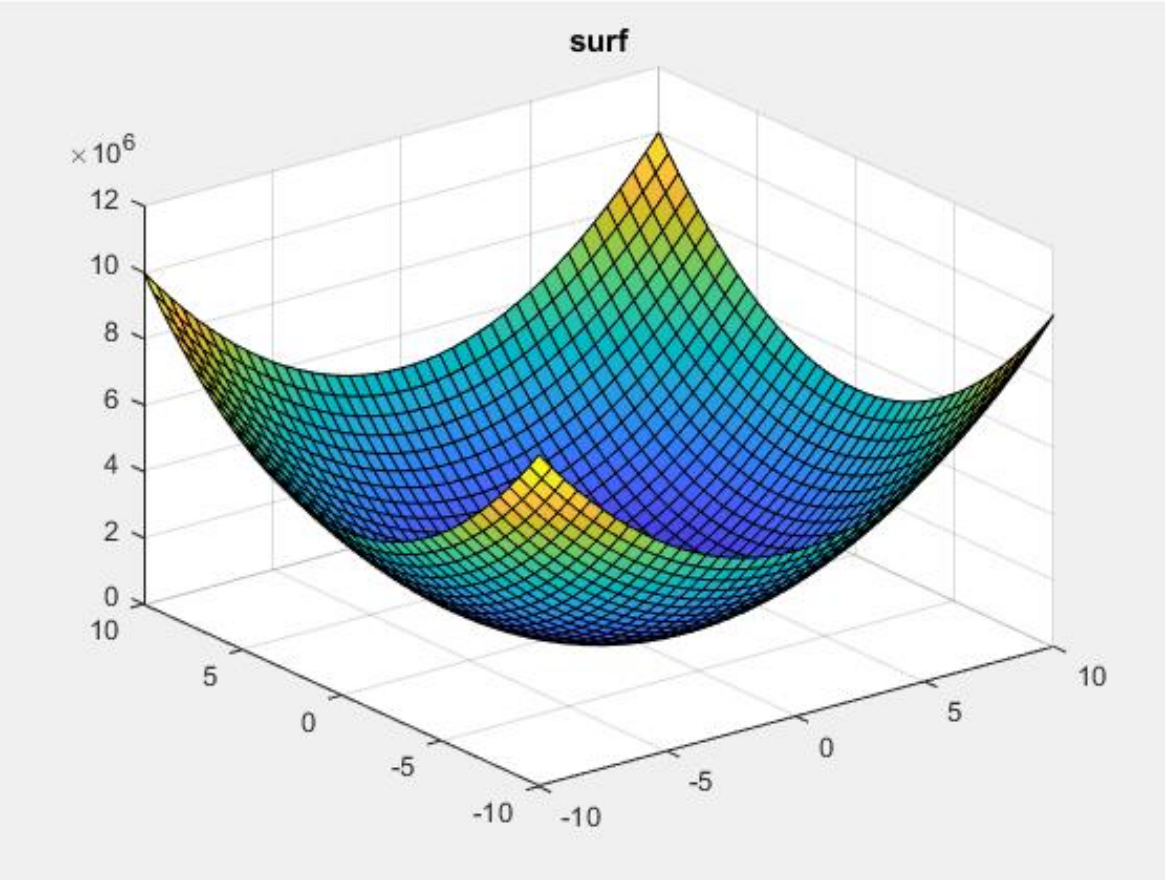
Esempio: assegnati A e b rappresentiamo la funzione, le curve di livello, la soluzione calcolato con CG e le direzioni seguite

Paraboloide

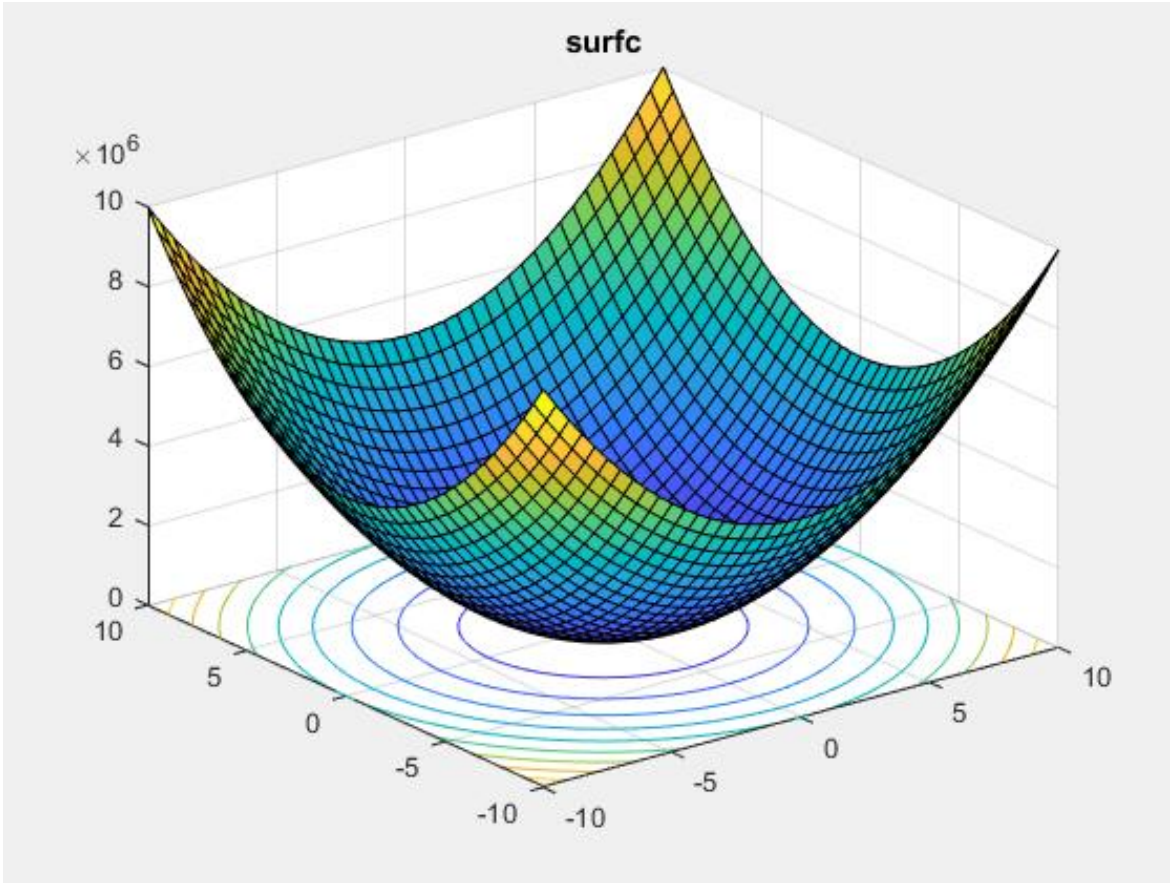


Esempio: assegnati A e b rappresentiamo la funzione, le curve di livello, la soluzione calcolato con CG e le direzioni seguite

Paraboloide

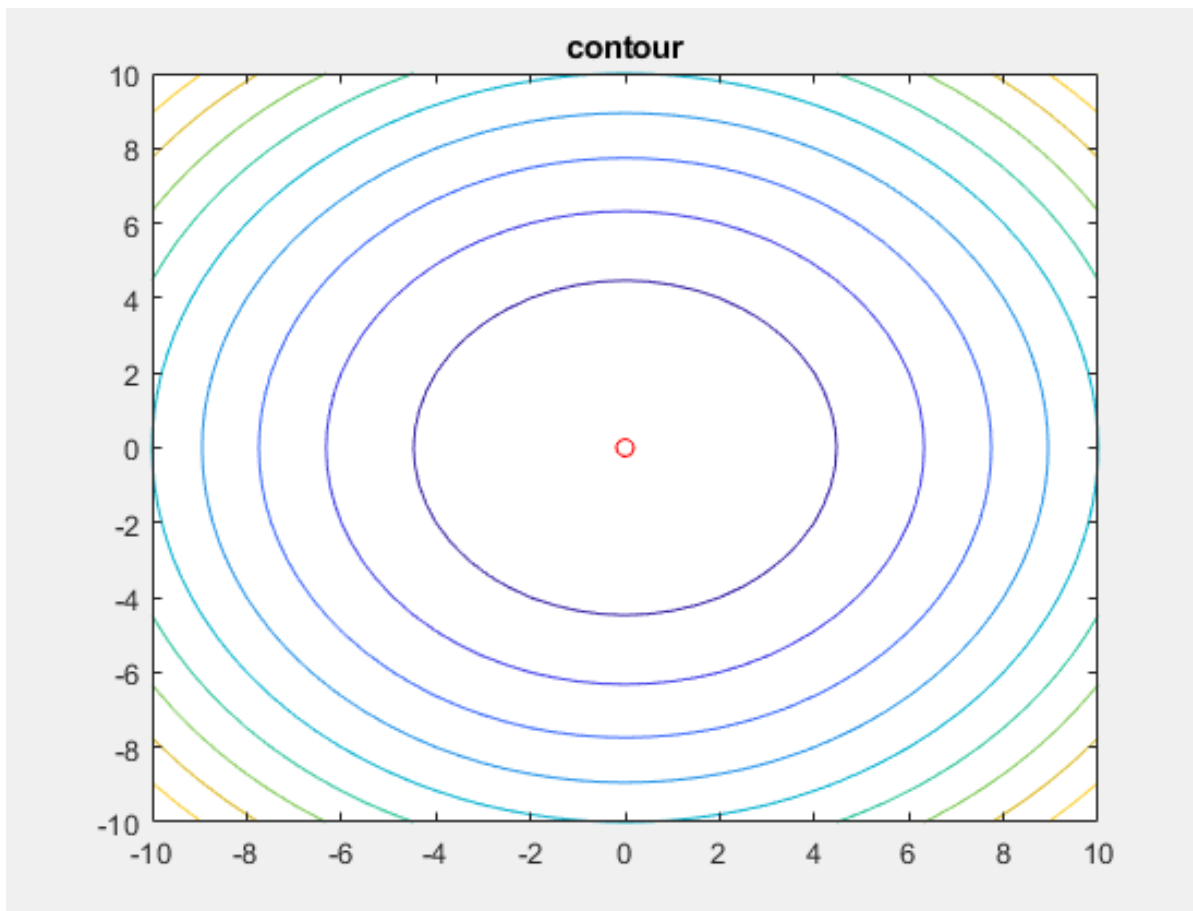


Paraboloide e curve di livello



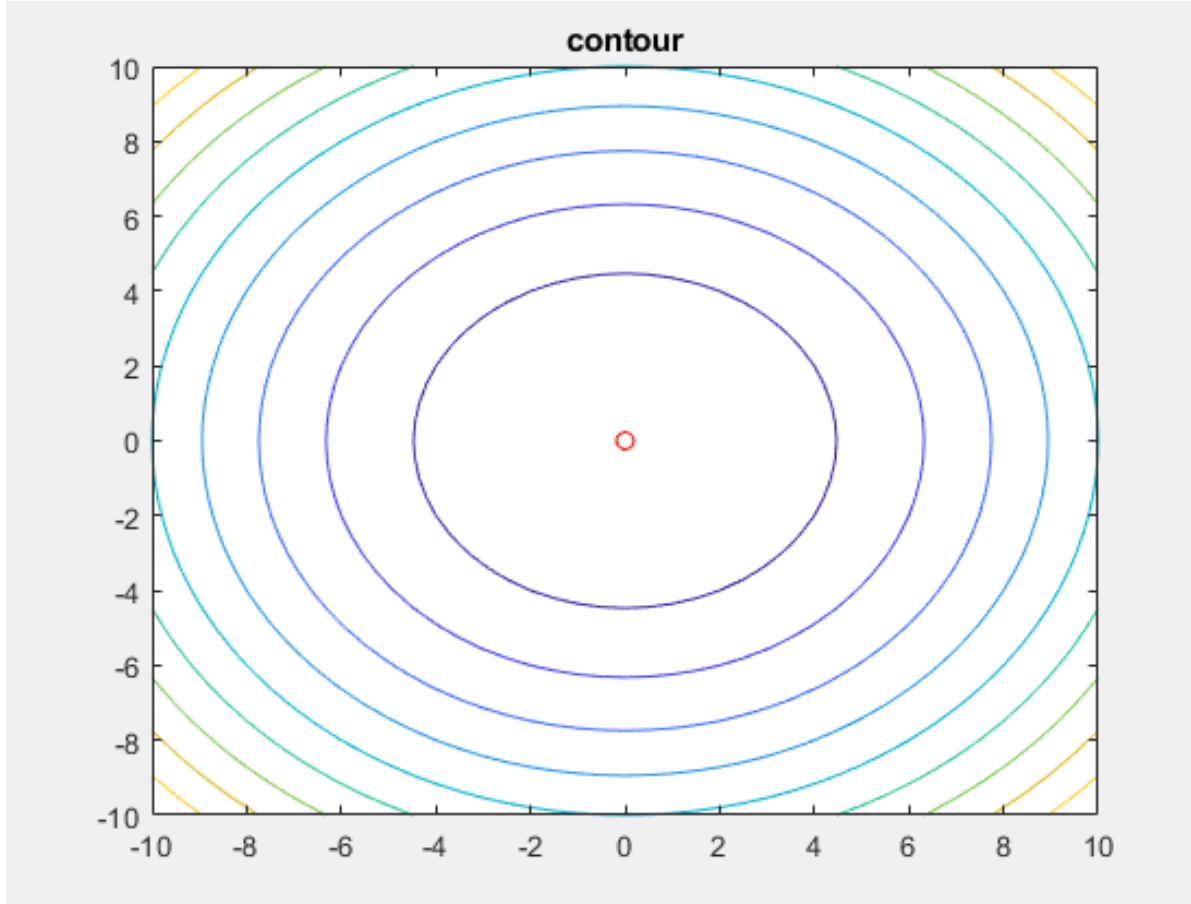
Esempio: assegnati A e b rappresentiamo la funzione, le curve di livello, la soluzione calcolata con CG e le direzioni seguite

Curve di livello e soluzione calcolata con CG



Esempio: assegnati A e b rappresentiamo la funzione, le curve di livello, la soluzione calcolata con CG e le direzioni seguite

Curve di livello e soluzione calcolata con CG



```
[x,flag,RELRES,iter,RESVEC] = pcg(a,b,tol,maxit);
```

flag =

0

RELRES =

9.4116e-06

iter =

1

```
[x,error,niter,flag]=gradiente(a,x0,b,maxit,tol);
```

flag =

0

niter =

1

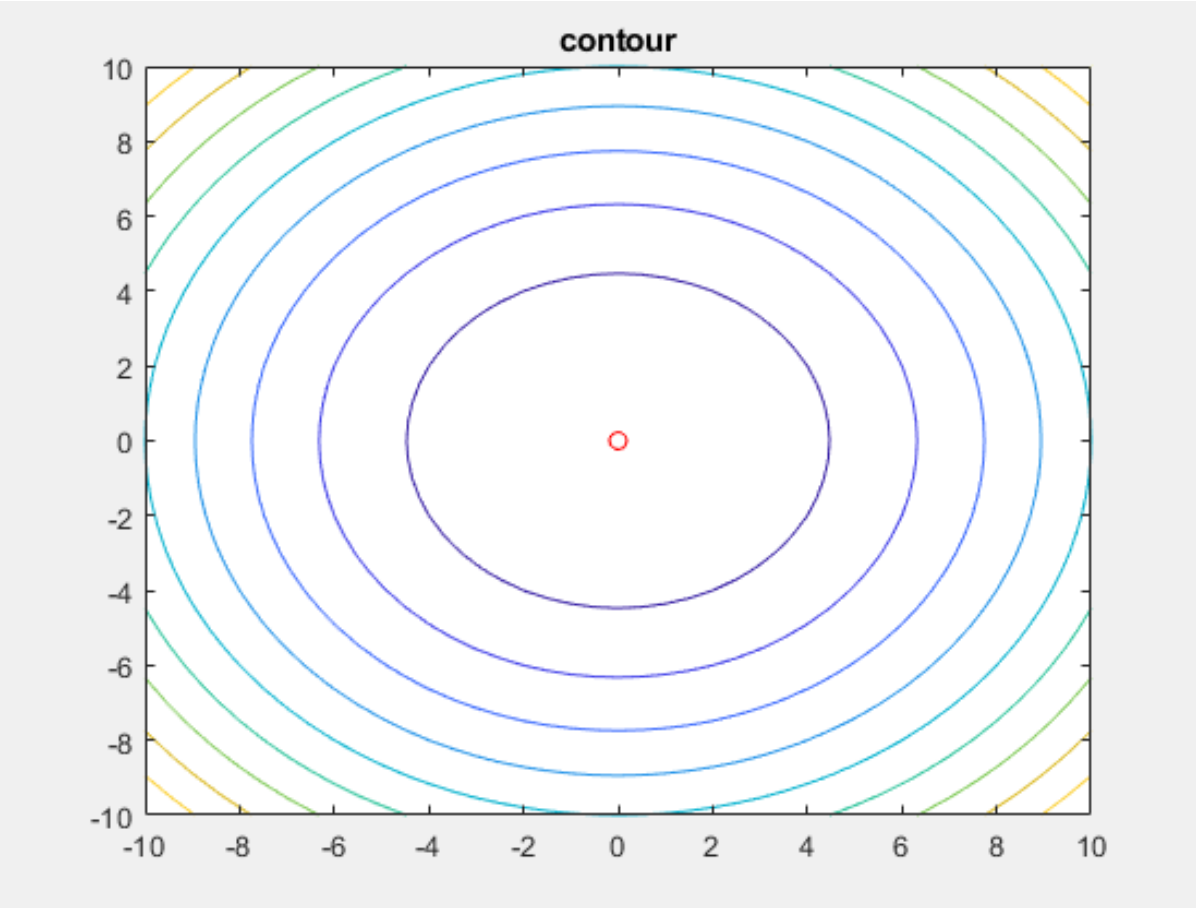
```
>> error
```

error =

9.7012e-06

Esempio: assegnati A e b rappresentiamo la funzione, le curve di livello, la soluzione calcolato con CG e le direzioni seguite

Curve di livello e soluzione calcolata con CG



```
[x,flag,RELRES,iter,RESVEC] = pcg(a,b,tol,maxit);
```

flag =

0

RELRES =

9.4116e-06

iter =

1

>> eig(a)

ans =

99998

100002

>> cond(a)

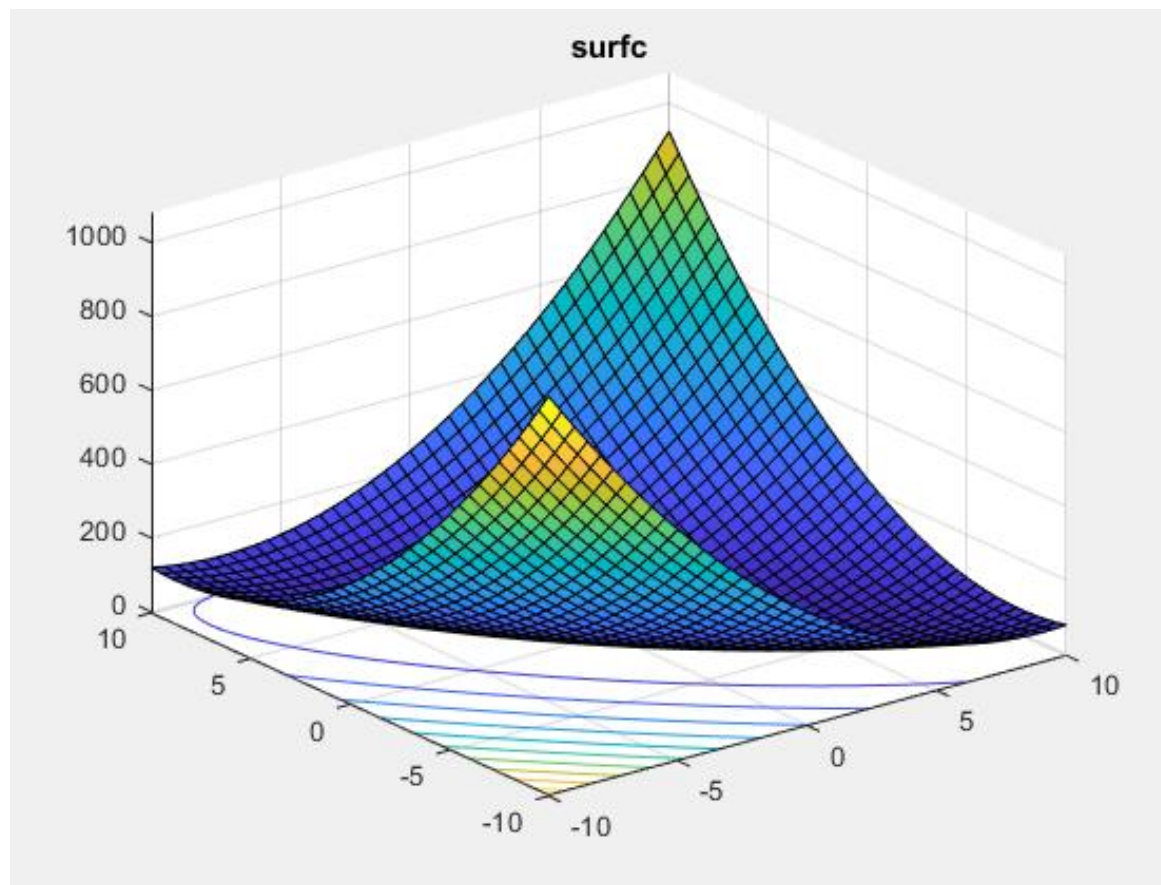
ans =

1.0000

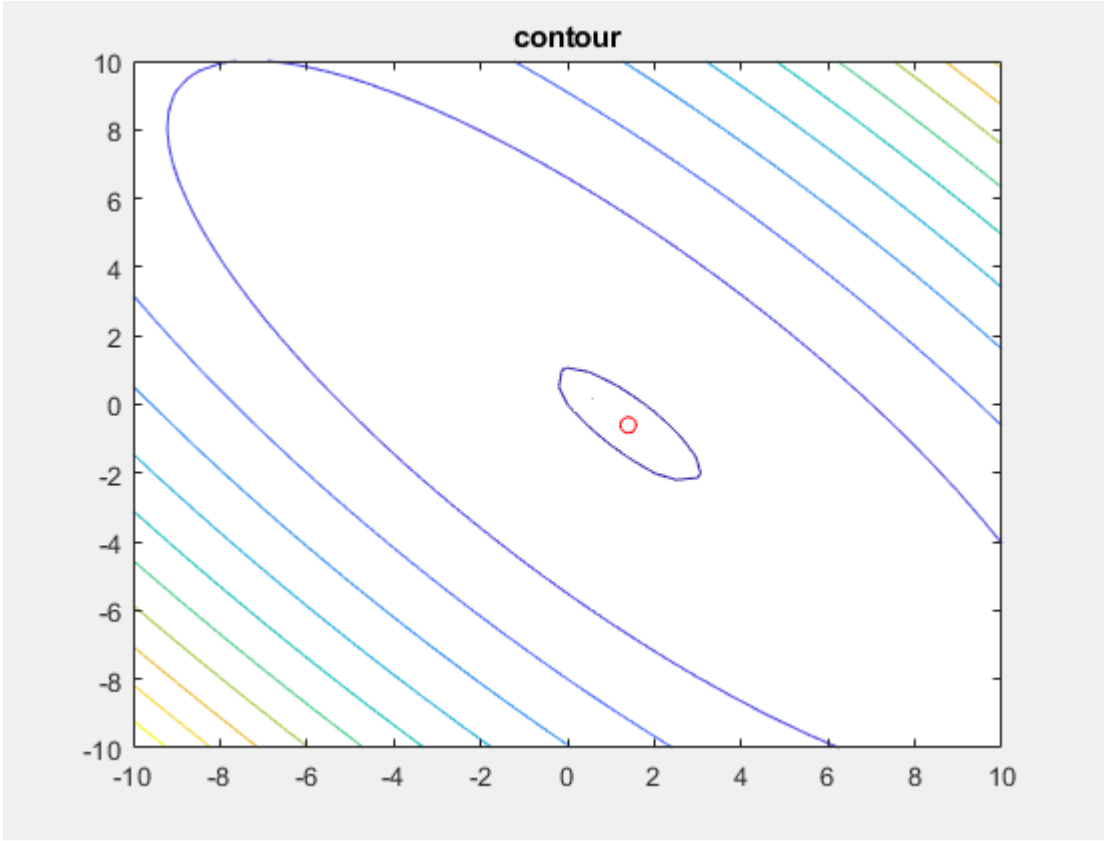
Esempio: assegnati A e b rappresentiamo la funzione, le curve di livello, la soluzione calcolata con CG e le direzioni seguite

```
a=[ 10 1];  
%vettore colonna iniziale  
x0=ones(2,1);  
%vettore colonna dei termini noti  
b=[5 3]';  
%massimo numero di iterazioni  
maxit=100;  
%tolleranza  
tol=10^(-7);  
%chiamata function  
u=gallery('orthog',2);  
a=u'*diag(a)*u;
```

Esempio: assegnati A e b rappresentiamo la funzione, le curve di livello, la soluzione calcolato con CG e le direzioni seguite



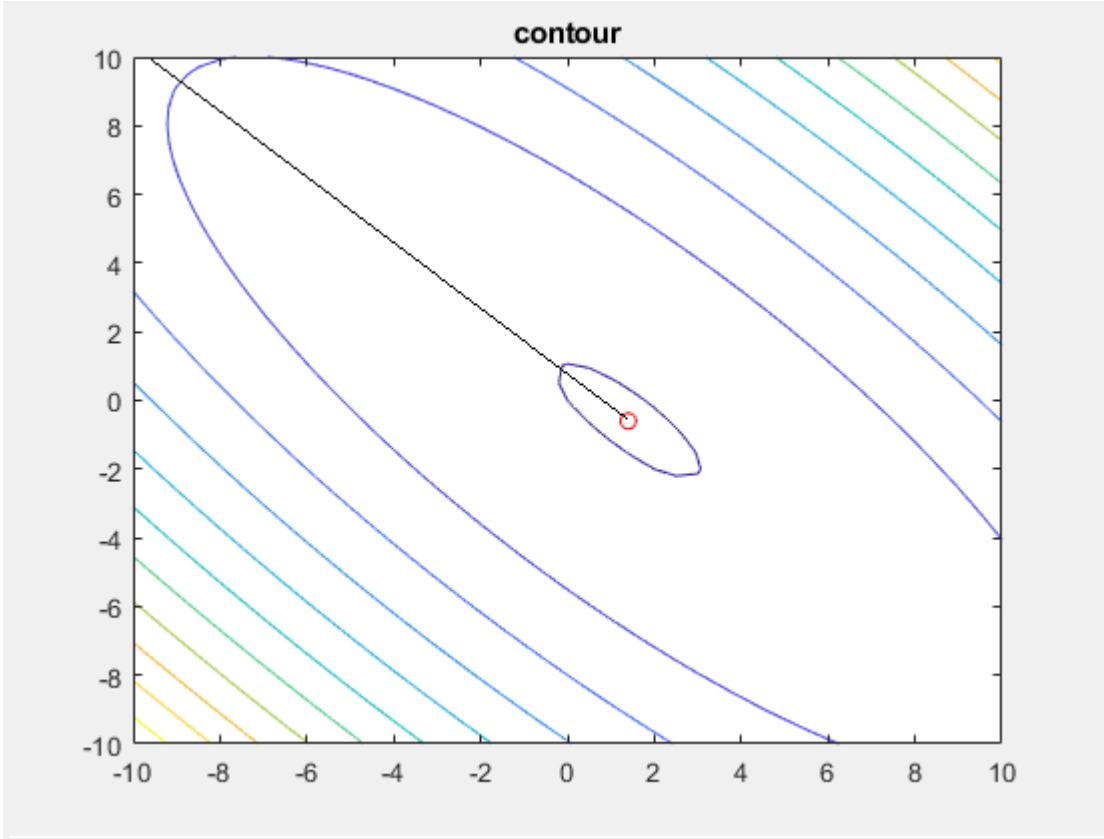
Esempio: assegnati A e b rappresentiamo la funzione, le curve di livello, la soluzione calcolato con CG e le direzioni seguite



metodo	flag	iter	residuo
Metodo iterativo (G)	0	17	1.9950e-07
Metodo CG	0	2	1.5232e-16

Esempio: assegnati A e b rappresentiamo la funzione, le curve di livello, la soluzione calcolato con CG e le direzioni seguite

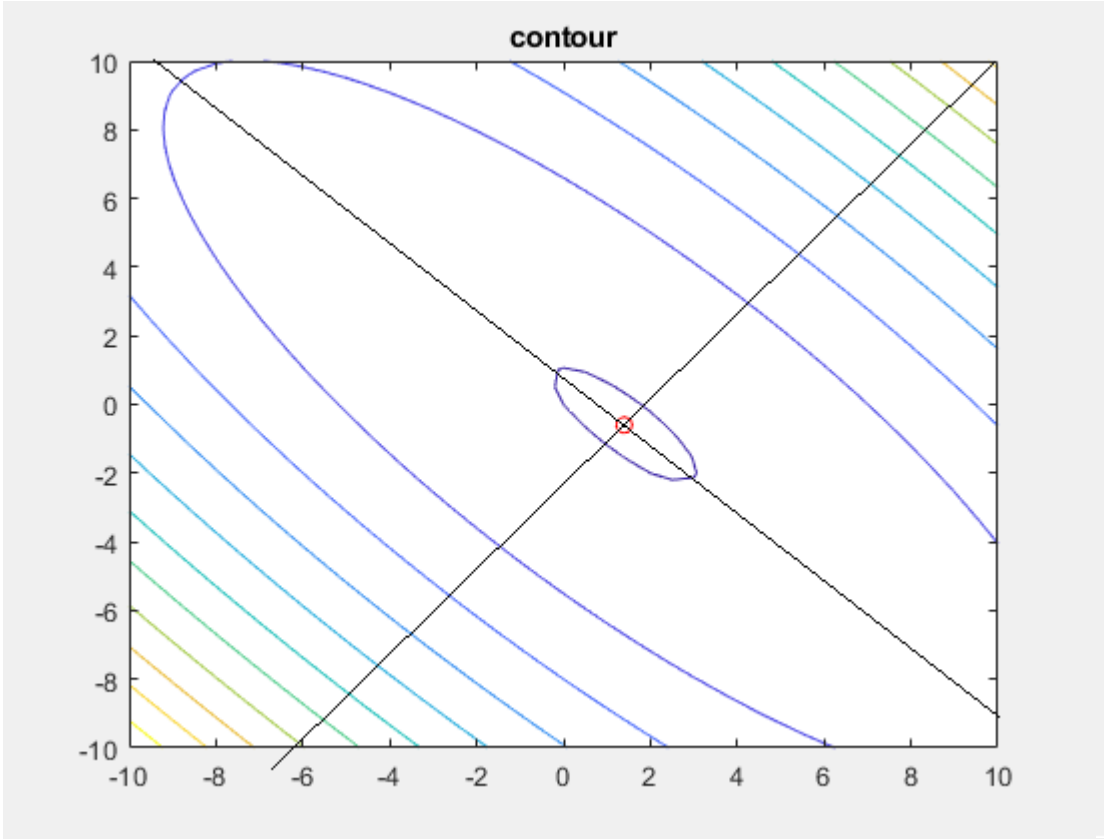
Con metodo CG



metodo	flag	iter	residuo
Metodo iterativo (G)	0	17	1.9950e-07
Metodo CG	0	2	1.5232e-16

Esempio: assegnati A e b rappresentiamo la funzione, le curve di livello, la soluzione calcolato con CG e le direzioni seguite

Con metodo CG



```
>> [V,D]=eig(a)
```

V =

-0.7071	0.7071
0.7071	0.7071

autovettori

D =

1	0
0	10

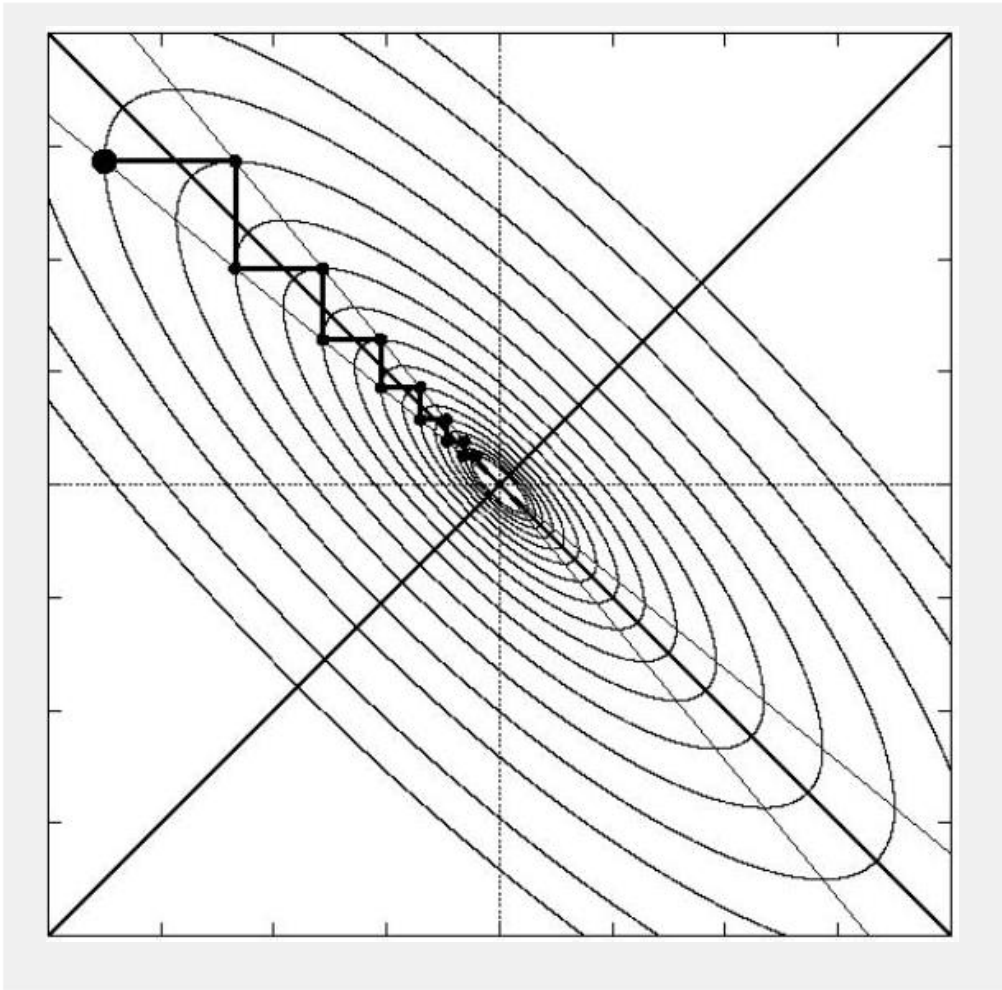
autovalori

Semiassi:
 $-0.7071(x-1.4)+0.7071(y+0.6)=0$
 $0.7071(x-1.4)+0.7071(y+0.6)=0$

Stessa direzione degli autovalori

Esempio: assegnati A e b rappresentiamo la funzione, le curve di livello, la soluzione calcolato con CG e le direzioni seguite

Metodo del gradiente



metodo	flag	iter	residuo
Metodo iterativo (G)	0	17	1.9950e-07
Metodo CG	0	2	1.5232e-16

>> eig(a)

>> cond(a)

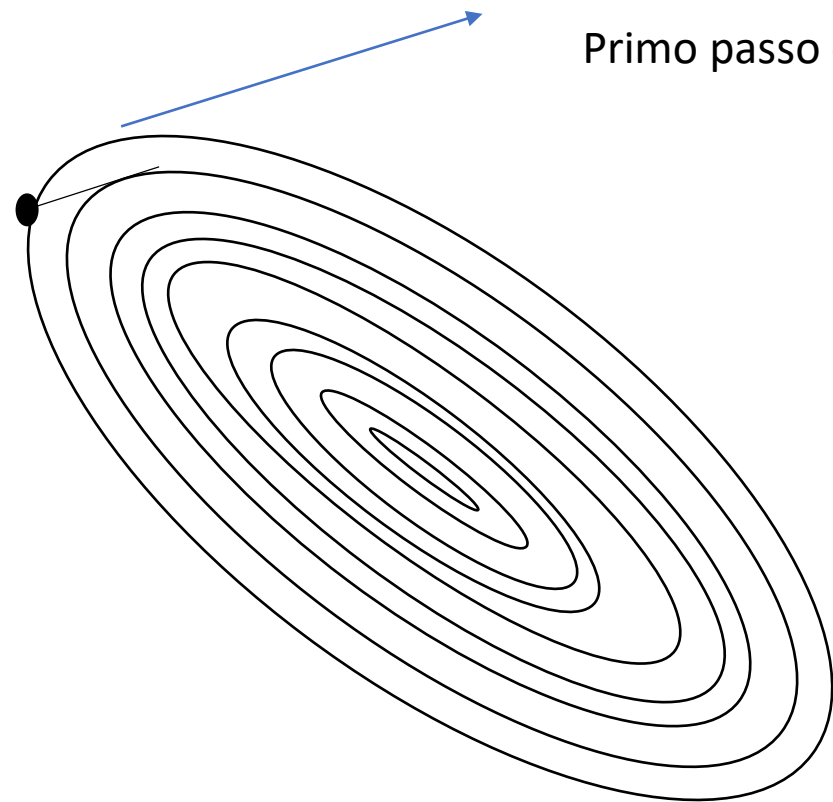
ans =

ans =

1
10

9.9999999999999991

Esempio: assegnati A e b rappresentiamo la funzione, le curve di livello, la soluzione calcolato con CG e le direzioni seguite



Primo passo direzione di discesa: gradiente (residuo) ortogonale alle curve di livello

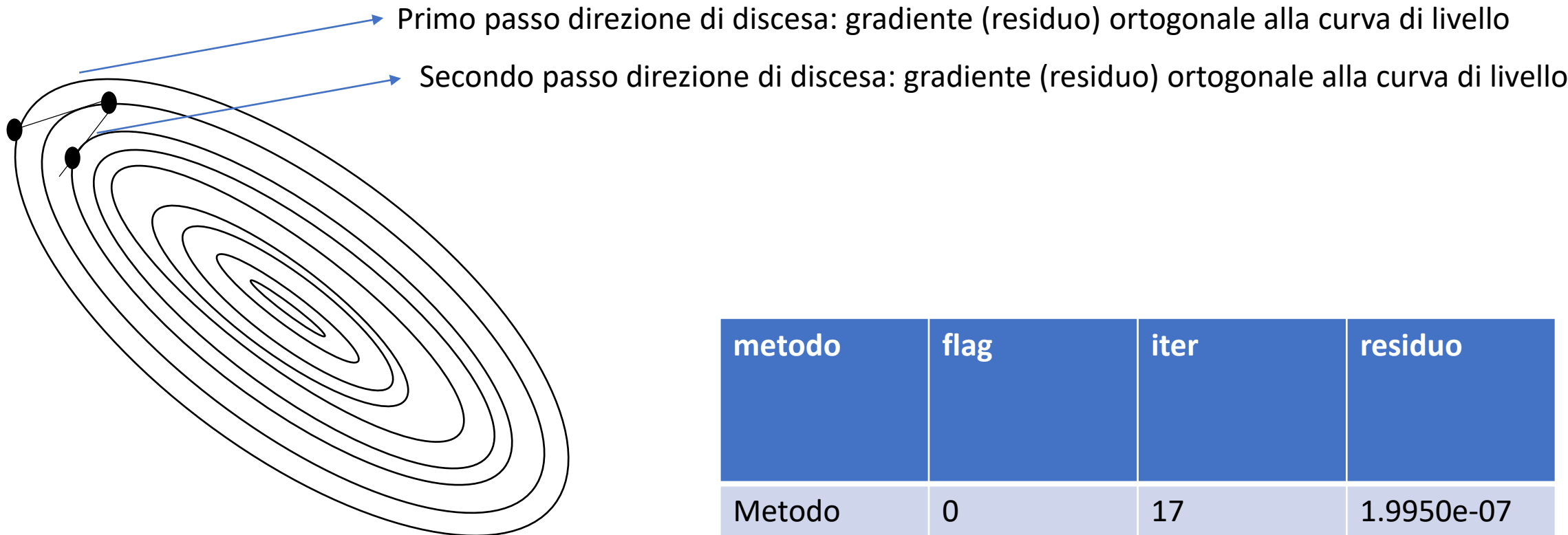
metodo	flag	iter	residuo
Metodo iterativo (G)	0	17	1.9950e-07
Metodo CG	0	2	1.5232e-16

```
>> eig(a)
```

```
ans =
```

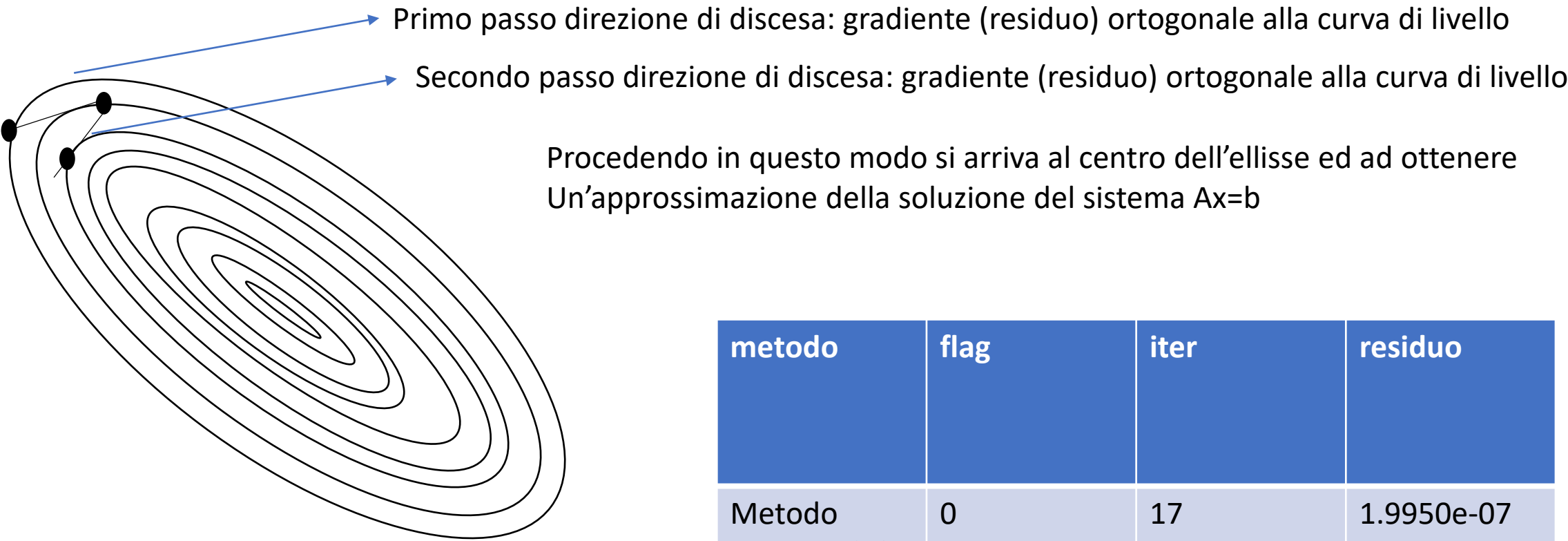
```
1  
10
```

Esempio: assegnati A e b rappresentiamo la funzione, le curve di livello, la soluzione calcolato con CG e le direzioni seguite



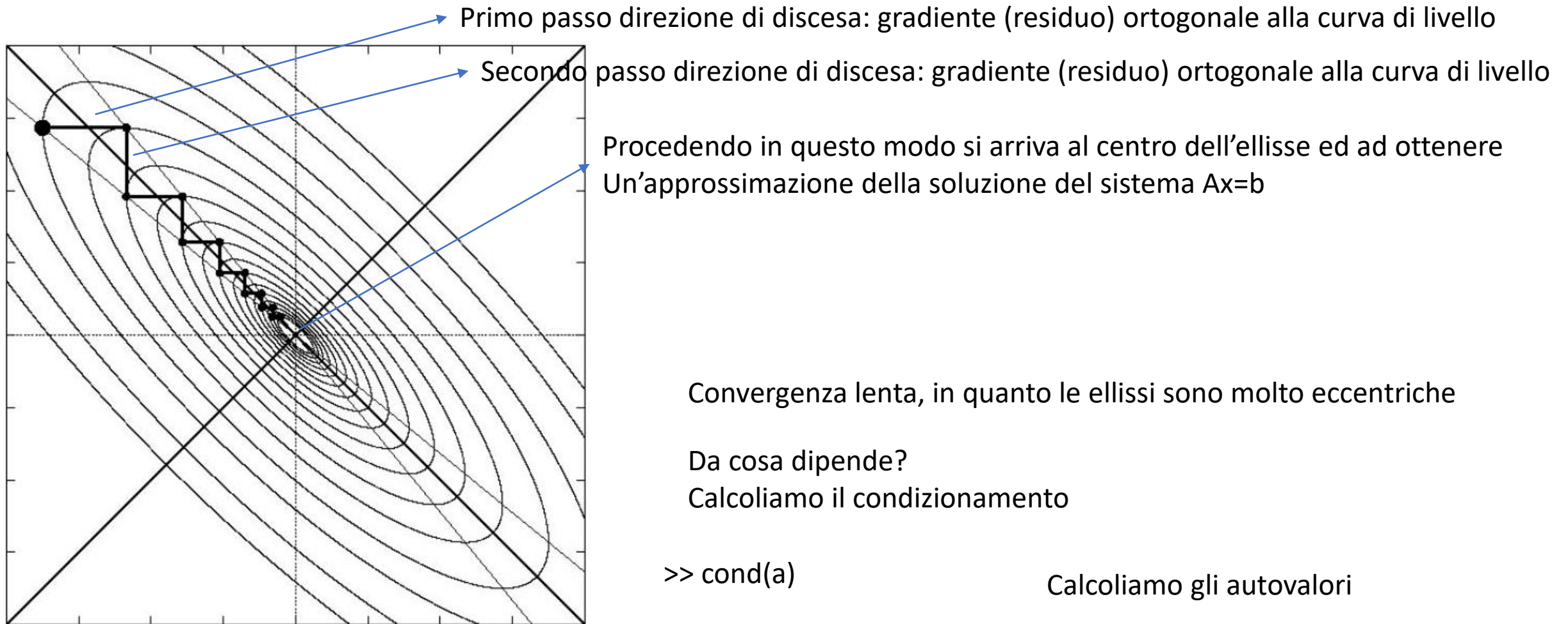
metodo	flag	iter	residuo
Metodo iterativo (G)	0	17	1.9950e-07
Metodo CG	0	2	1.5232e-16

Esempio: assegnati A e b rappresentiamo la funzione, le curve di livello, la soluzione calcolato con CG e le direzioni seguite



metodo	flag	iter	residuo
Metodo iterativo (G)	0	17	1.9950e-07
Metodo CG	0	2	1.5232e-16

Esempio: assegnati A e b rappresentiamo la funzione, le curve di livello, la soluzione calcolato con CG e le direzioni seguite



Convergenza lenta, in quanto le ellissi sono molto eccentriche

Da cosa dipende?

Calcoliamo il condizionamento

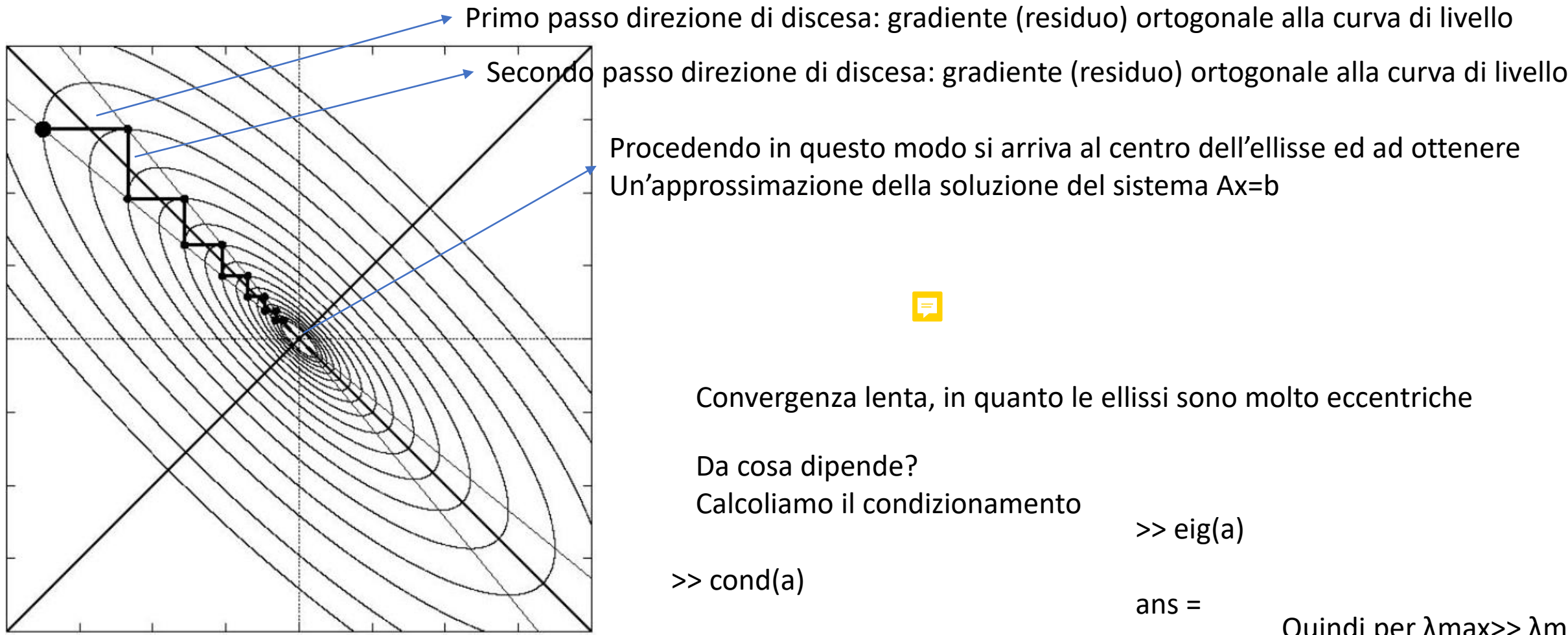
```
>> cond(a)
```

Calcoliamo gli autovalori

```
ans =
```

```
9.999999999999991
```

Esempio: assegnati A e b rappresentiamo la funzione, le curve di livello, la soluzione calcolato con CG e le direzioni seguite



Convergenza lenta, in quanto le ellissi sono molto eccentriche

Da cosa dipende?
Calcoliamo il condizionamento

```
>> eig(a)
ans =
    1
   10
Quindi per  $\lambda_{max} \gg \lambda_{min}$ 
Convergenza molto lenta

>> cond(a)
ans =
    9.999999999999991
```

Esempio d'uso:

A matrice simmetrica definita positiva sparsa con 50% di densità, b vettore random

```
>> A = sprand(400,400,.5) ;
```

```
>> A = A'*A;
```

```
>> b = sum(A,2) ;
```


Esempio d'uso:

A matrice simmetrica definita positiva, b vettore random

```
>> A = sprand(400,400,.5);
```

```
>> A = A'*A;
```

```
>> b = sum(A,2);
```

A è simmetrica e definita positiva, per verificare si posso calcolare gli autovalori e
Verificare che siano positivi

```
>> eig(A)>0
```

```
ans =
```

400×1 logical array

```
1  
1  
1  
1  
.  
.  
.  
1
```

Vero

eig routine matlab calcolo
autovalori

Esempio d'uso:

A matrice simmetrica definita positiva, b vettore random

```
>> A = sprand(400,400,.5);
```

```
>> A = A'*A;
```

```
>> b = sum(A,2);
```

```
>> x = pcg(A,b);
```

```
pcg stopped at iteration 20 without converging to the desired tolerance 1e-06  
because the maximum number of iterations was reached.
```

```
The iterate returned (number 20) has relative residual 4.1e-06.
```

Esempio d'uso:

A matrice simmetrica definita positiva, b vettore random

```
>> A = sprand(400,400,.5);
```

```
>> A = A'*A;
```

```
>> b = sum(A,2);
```

Massimo numero di iterazioni di default

Tolleranza di default è 1e-06

```
>> x = pcg(A,b);
```

```
pcg stopped at iteration 20 without converging to the desired tolerance 1e-06  
because the maximum number of iterations was reached.
```

```
The iterate returned (number 20) has relative residual 4.1e-06.
```

Esempio d'uso:

A matrice simmetrica definita positiva, b vettore random

```
>> A = sprand(400,400,.5);
```

```
>> A = A'*A;
```

```
>> b = sum(A,2);
```

Tolleranza di default è 1e-06

Massimo numero di iterazioni di default

```
>> x = pcg(A,b);
```

```
pcg stopped at iteration 20 without converging to the desired tolerance 1e-06  
because the maximum number of iterations was reached.
```

```
The iterate returned (number 20) has relative residual 4.1e-06.
```

Oppure

Esempio d'uso:
A matrice simmetrica definita positiva, b vettore random

```
>> [x,flag,relres,iter,resvec] = pcg(A,b);  
>> flag
```

```
flag =  
1
```

▼ **flag** — Convergence **flag**
scalar


Convergence flag, returned as one of the scalar values in this table. The convergence flag indicates whether the calculation was successful and differentiates between several different forms of failure.

Flag Value	Convergence
0	Success — pcg converged to the desired tolerance tol within maxit iterations.
1	Failure — pcg iterated maxit iterations but did not converge.
2	Failure — The preconditioner matrix M or M = M1*M2 is ill conditioned.
3	Failure — pcg stagnated after two consecutive iterations were the same.
4	Failure — One of the scalar quantities calculated by the pcg algorithm became too small or too large to continue computing.

Esempio d'uso:

A matrice simmetrica definita positiva, b vettore random

```
>> [x,flag,relres,iter,resvec] = pcg(A,b);  
>> flag  
  
flag =  
  
1  
  
>> relres  
  
relres =  
  
4.0513e-06  
  
>> iter
```



relres — Relative residual error
scalar

Relative residual error, returned as a scalar. The relative residual error $\text{relres} = \text{norm}(b - A*x) / \text{norm}(b)$ is an indication of how accurate the answer is. If the calculation converges to the tolerance `tol` within `maxit` iterations, then $\text{relres} \leq \text{tol}$.

Data Types: double

Esempio d'uso:

A matrice simmetrica definita positiva, b vettore random

```
>> [x,flag,relres,iter,resvec] = pcg(A,b);
```

```
>> flag
```

```
flag =
```

```
1
```

```
>> relres
```

```
relres =
```

```
4.0513e-06
```

```
>> iter
```

```
iter =
```

```
20
```



iter — Iteration number
scalar

Iteration number, returned as a scalar. This output indicates the iteration number at which the computed answer for **x** was calculated.

Data Types: double

Esempio d'uso:

A matrice simmetrica definita positiva, b vettore random

```
>> [x,flag,relres,iter,resvec] = pcg(A,b);
```

```
resvec =
```

```
1.0e+05 *
```

```
1.2553
```

```
0.0009
```

```
0.0003
```

```
0.0001
```

```
0.0001
```

```
0.0001
```

```
0.0000
```

```
0.0001
```

```
0.0000
```

```
0.0000
```

```
0.0000
```

```
0.0000
```

```
0.0000
```

```
0.0000
```

```
0.0000
```

```
0.0000
```

```
0.0000
```

```
0.0000
```

```
0.0000
```

```
0.0000
```

```
0.0000
```

▼ **resvec — Residual error**
vector

Residual error, returned as a vector. The residual error $\text{norm}(b-A*x)$ reveals how close the algorithm is to converging for a given value of x . The number of elements in `resvec` is equal to the number of iterations. You can examine the contents of `resvec` to help decide whether to change the values of `tol` or `maxit`.

Data Types: double

Esempio d'uso:

A matrice simmetrica definita positiva, b vettore random

```
>> [x,flag,relres,iter,resvec] = pcg(A,b);
```

```
resvec =
```

```
1.0e+05 *
```

```
1.2553
```

```
0.0009
```

```
0.0003
```

```
0.0001
```

```
0.0001
```

```
0.0001
```

```
0.0000
```

```
0.0001
```

```
0.0000
```

```
0.0000
```

```
0.0000
```

```
0.0000
```

```
0.0000
```

```
0.0000
```

```
0.0000
```

```
0.0000
```

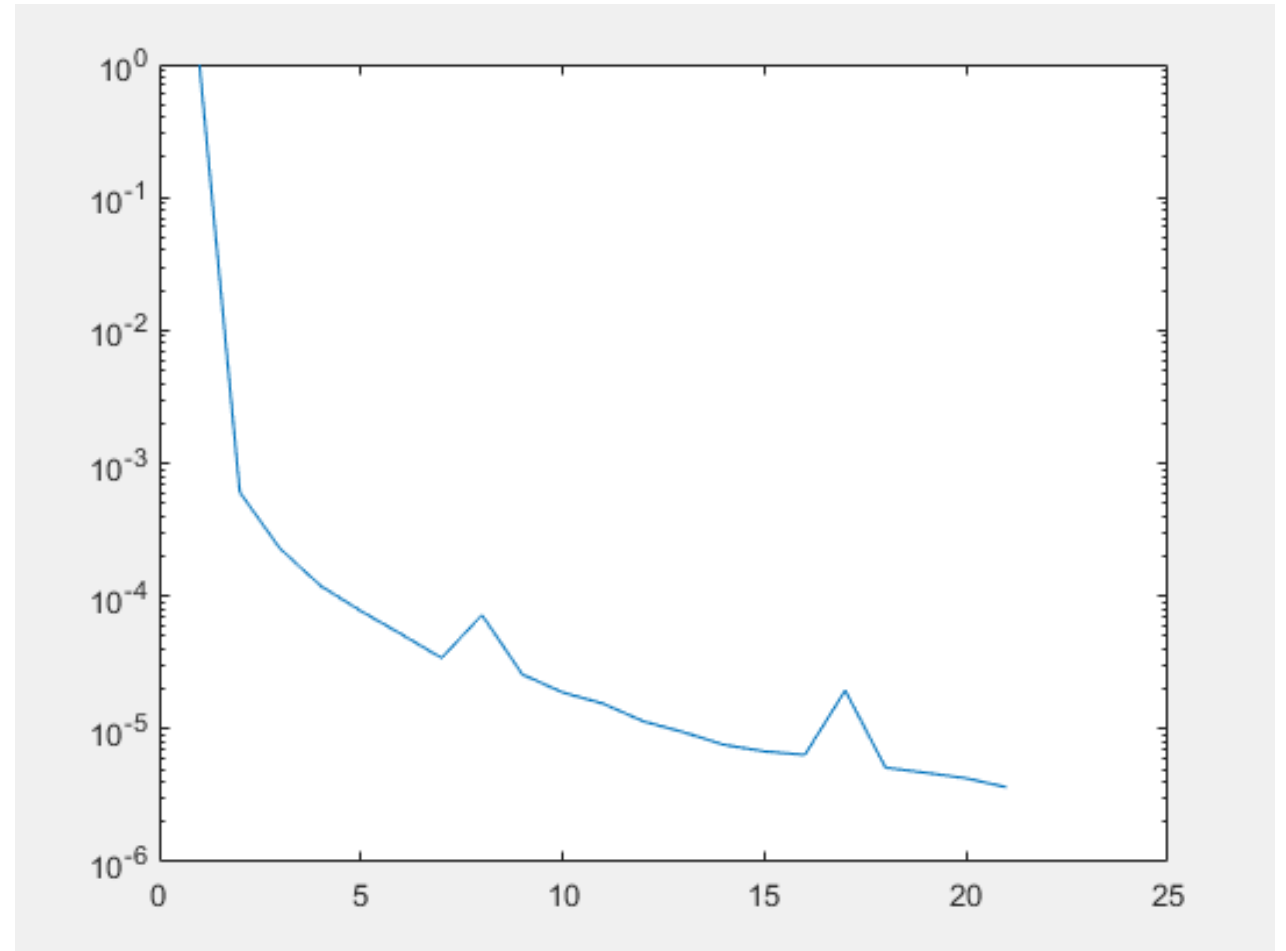
```
0.0000
```

```
0.0000
```

```
0.0000
```

```
0.0000
```

```
0.0000
```



Esempio d'uso:

A matrice simmetrica definita positiva, b vettore random

Se aumentiamo il numero di iterazioni

```
>> x=pcg(A,b,10^-7,400);
```

pcg converged at iteration 141 to a solution with relative residual 9.2e-08.

Oppure

```
>> [x,flag]=pcg(A,b,10^-7,400);
```

```
>> flag
```

flag =

0

Esempio d'uso:

A matrice simmetrica definita positiva, b vettore random

Se aumentiamo il numero di iterazioni

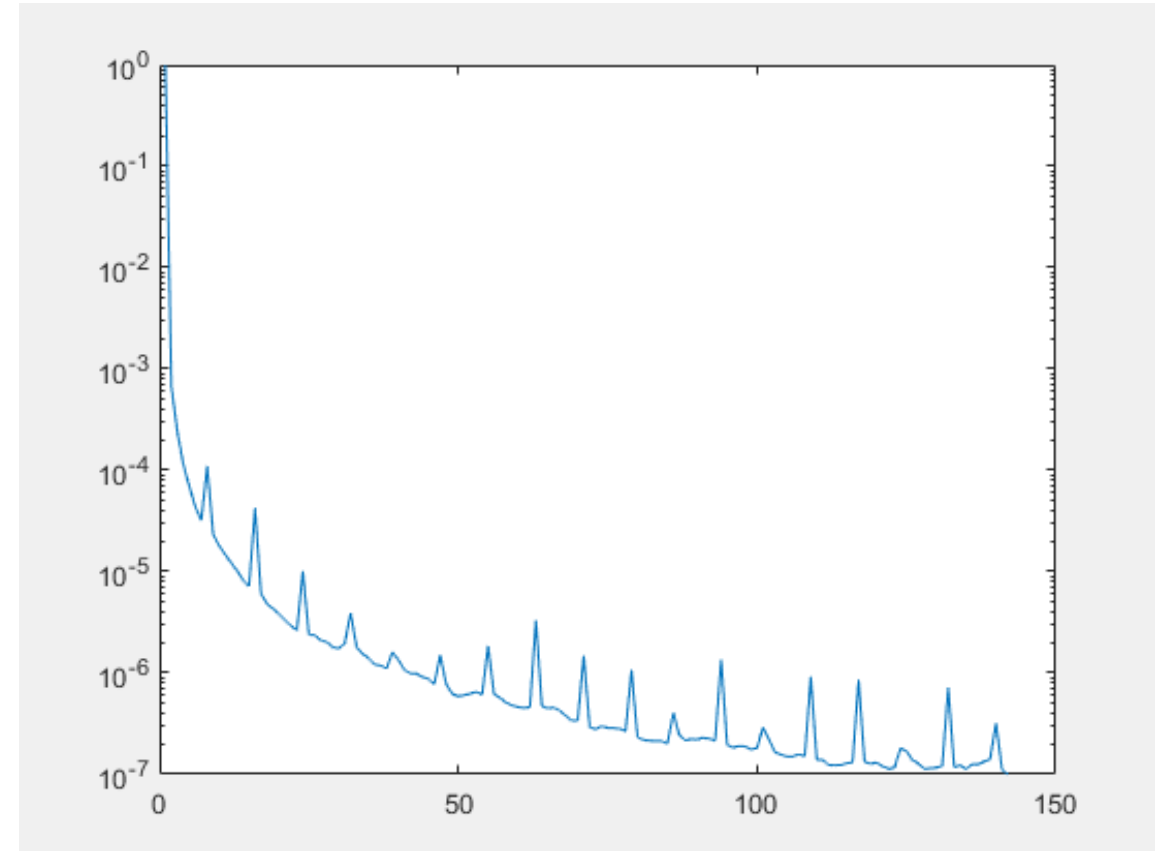
```
>> x=pcg(A,b,10^-7,400);  
pcg converged at iteration 141 to a solution with relative  
residual 9.2e-08.
```

Oppure

```
>> [x,flag]=pcg(A,b,10^-7,400);  
>> flag
```

flag =

0



Esempio: costruire una matrice simmetrica definita positiva, A , di dimensione 100, i cui elementi siano del tipo $A(i,j)=\min(i,j)$. Risolvere il sistema lineare $Ax=b$ con $b=\text{ones}(100,1)$;


Esempio: costruire una matrice simmetrica definita positiva, A , di dimensione 100, i cui elementi siano del tipo $A(i,j)=\min(i,j)$. Risolvere il sistema lineare $Ax=b$ con $b=\text{ones}(100,1)$;

```
%costruzione di una matrice di dimensione 100x100 i cui elementi sono  
%del tipo  $A(i,j)=\min(i,j)$ 
```

```
A=gallery ('minij', 100);
```



```
%vettore termini noti  
b=ones(size(A,1),1);  
%massimo numero di iterazioni  
maxit=size(A,1);
```



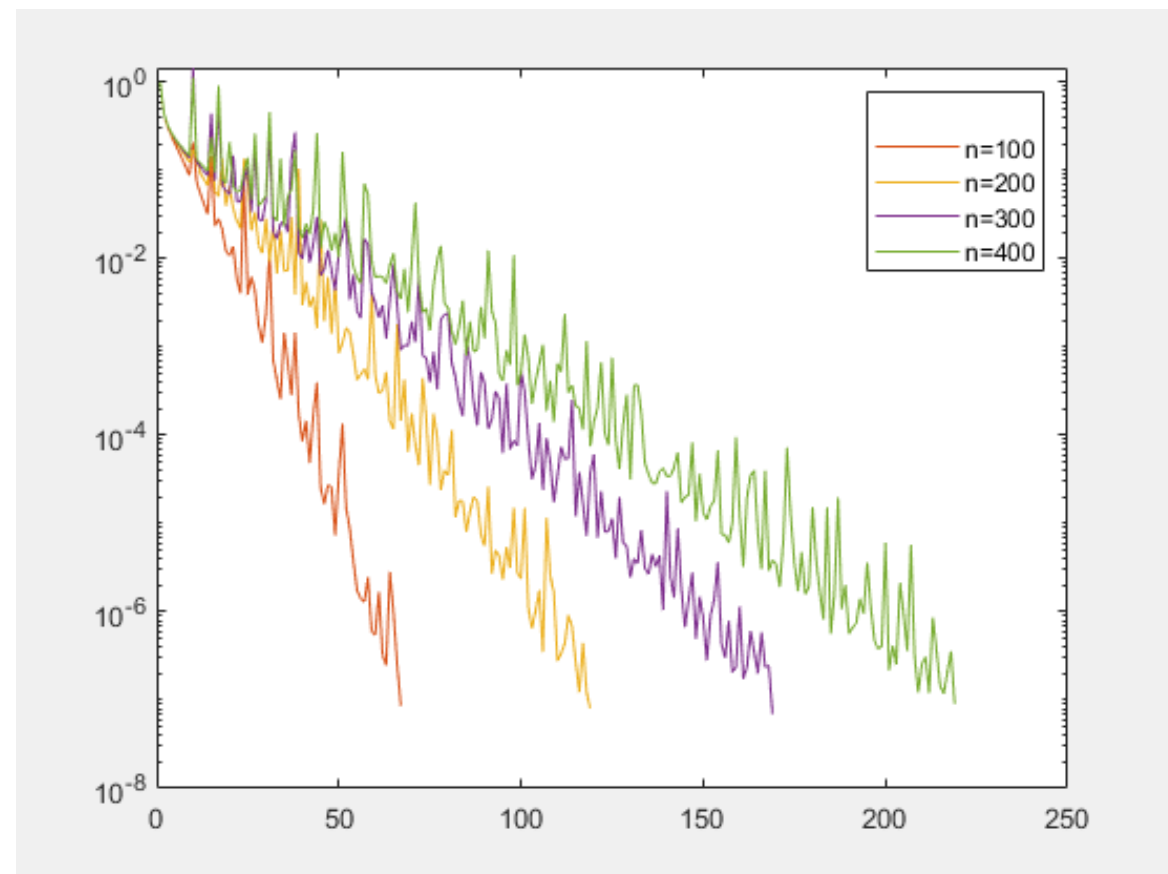
Funzione «gallery» di matlab con l'opzione «minij» permette di memorizzare in A la Matrice richiesta

Massimo numero di iterazioni uguale alla Dimensione della matrice A

Esempio: costruire una matrice simmetrica definita positiva, A , di dimensione n , i cui elementi siano del tipo $A(i,j)=\min(i,j)$. Risolvere il sistema lineare $Ax=b$ con $b=\text{ones}(n,1)$;

```
>> [x,flag,relres,iter,resvec]=pcg(A, b,10^-7,size(A,1));
```

N(dimensione)	flag	iter	residuo
100	0	66	8.5919e-08
200	0	118	8.0621e-08
300	0	168	6.9378e-08
400	0	218	9.0514e-08



Esempio: costruire una matrice simmetrica definita positiva, A , di dimensione n , i cui elementi siano del tipo $A(i,j)=\min(i,j)$. Risolvere il sistema lineare $Ax=b$ con $b=\text{ones}(n,1)$;

`tol=eps;`

`>> [x,flag,relres,iter,resvec]=pcg(A, b,10^-7,1000);`

N(dimensione)	flag	iter	residuo
100	0	66	8.5919e-08
200	0	118	8.0621e-08
300	0	168	6.9378e-08
400	0	218	9.0514e-08

Numero di iterazioni è sempre minore della dimensione della matrice rispetto La tolleranza fissata

Esempio: costruire una matrice simmetrica definita positiva, A , di dimensione 100, i cui elementi siano del tipo $A(i,j)=\min(i,j)$. Risolvere il sistema lineare $Ax=b$ con $b=\text{ones}(100,1)$;

```
tol=10^(-5);
```

```
display('esempio con tol=10^-5')
```

```
[x,flag,RELRES,iter,RESVEC] = pcg(A,b,tol,maxit)
```

```
%cambio tolleranza
```

```
display('esempio con tol=10^-8')
```

```
tol=10^(-8);
```

```
[x,flag,RELRES,iter,RESVEC] = pcg(A,b,tol,maxit)
```

```
%tolleranza
```

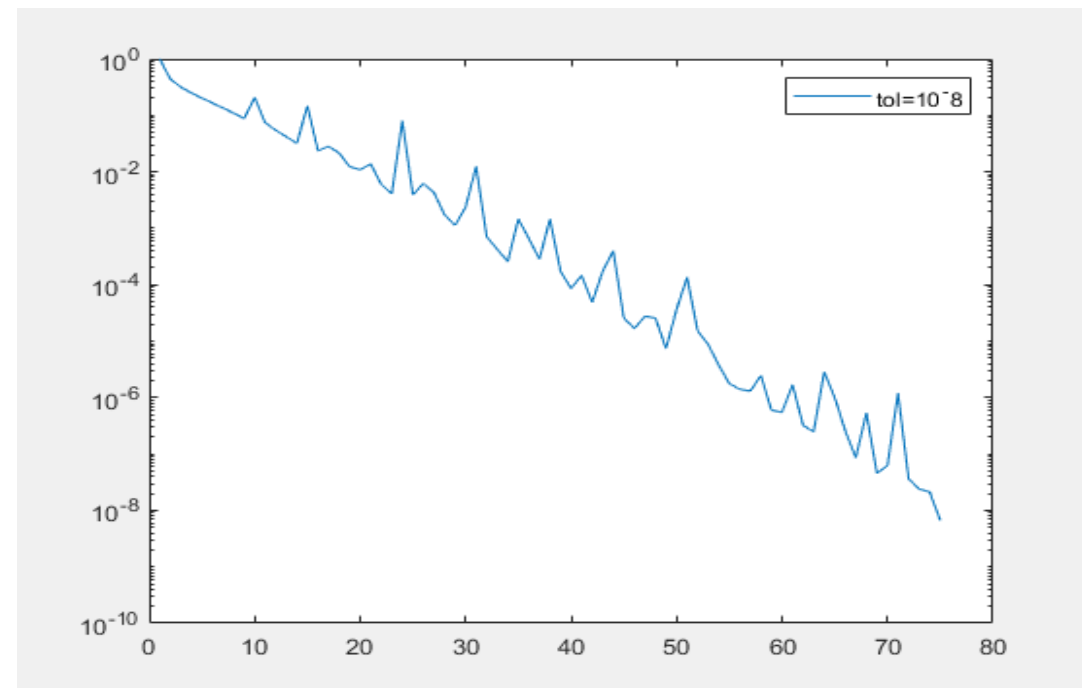
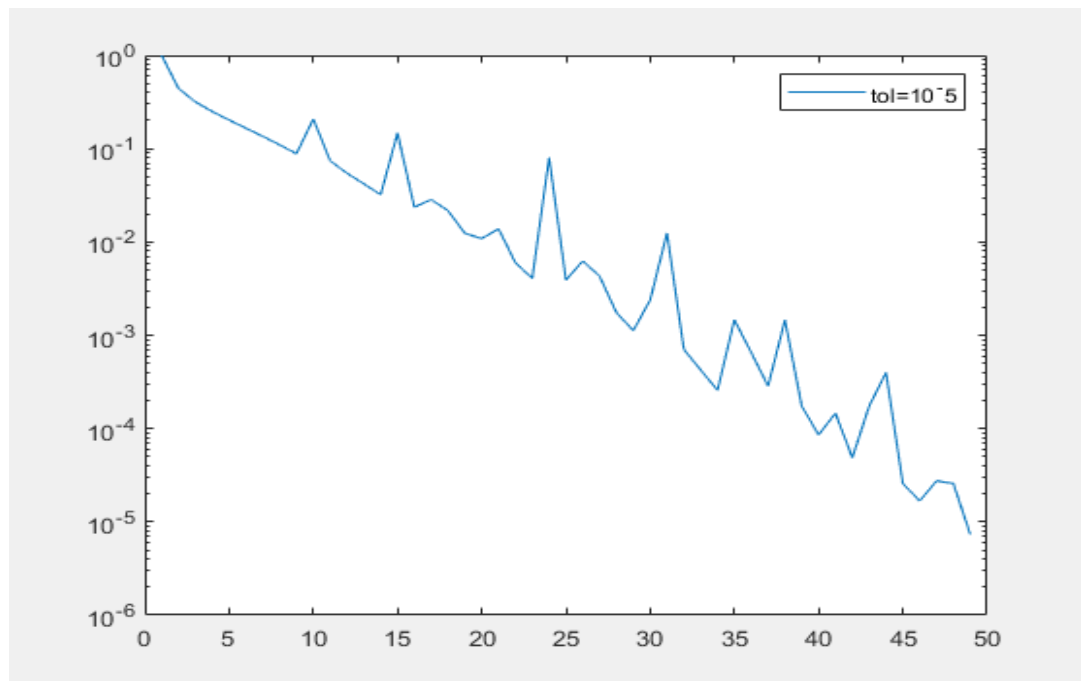
```
tol=10^(-10);
```

```
display('esempio con tol=10^-10')
```

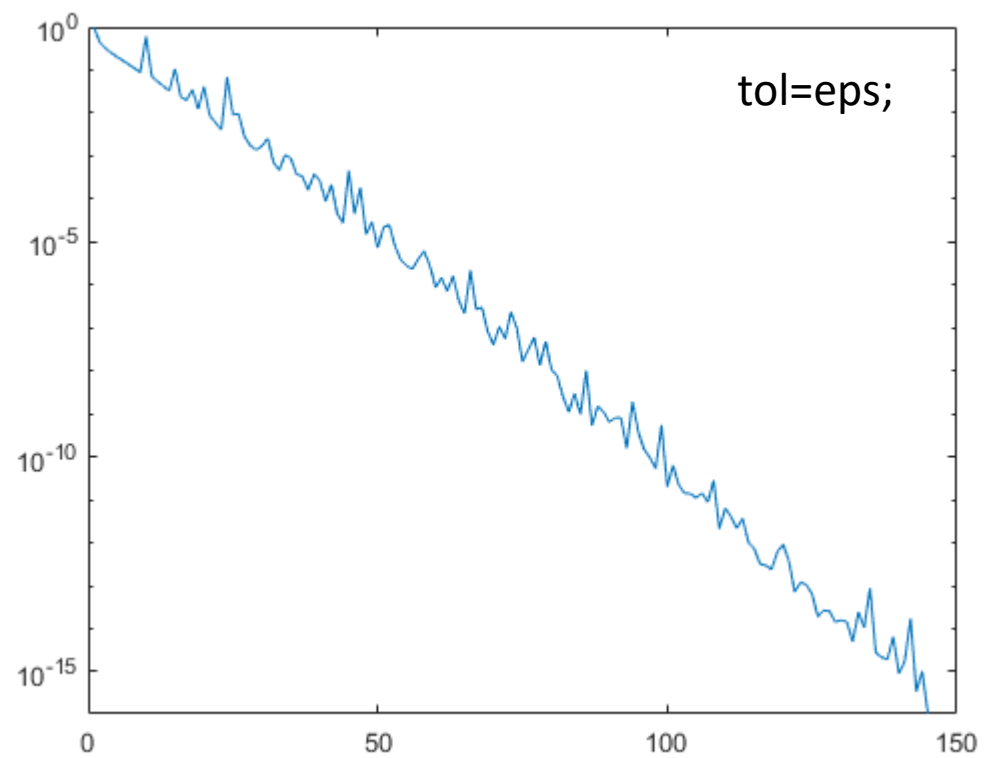
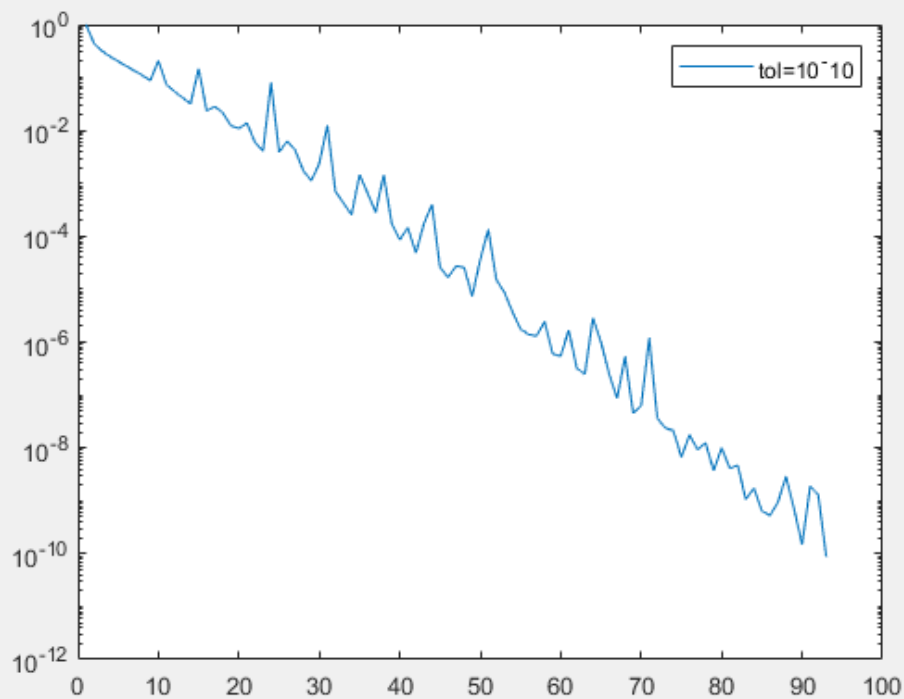
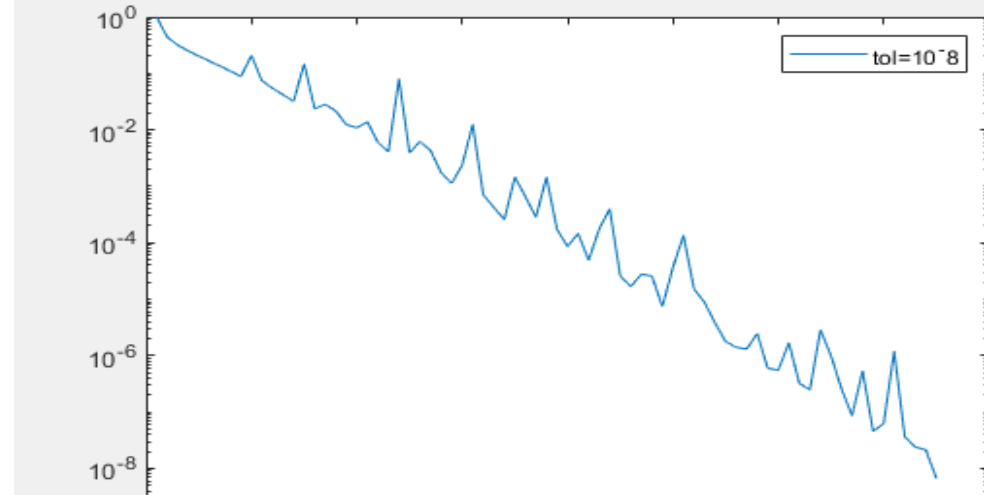
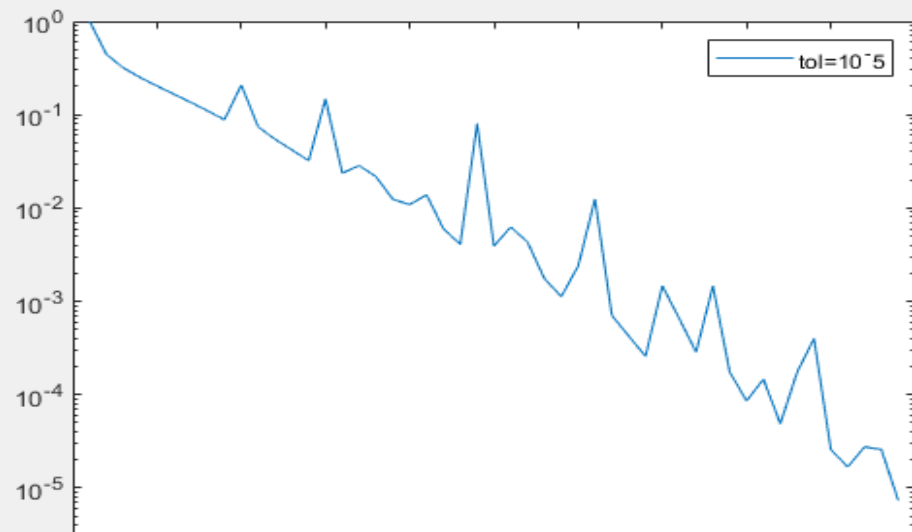
```
[x,flag,RELRES,iter,RESVEC] = pcg(A,b,tol,maxit)
```

```
tol=eps;
```

Esempio: grafici al variare della tolleranza



Esempio: grafici al variare della tolleranza



Esempio: costruire una matrice simmetrica definita positiva, A, di dimensione 100, i cui elementi siano del tipo $A(i,j)=\min(i,j)$. Risolvere il sistema lineare $Ax=b$ con $b=\text{ones}(100,1)$;

TOLLERANZA	FLAG	ITERAZIONI	RESIDUO
10^-5	0	49	8.8490e-06
10^-8	0	73	6.7002e-09
10^-10	0	93	5.7087e-11
eps	0	146	1.0425e-16

Convergenza
rispetto la tolleranza fissata

al decrescere della tolleranza decresce il residuo e aumenta il numero di iterazione

Esempio: costruire una matrice simmetrica definita positiva, A, di dimensione 100, i cui elementi siano del tipo $A(i,j)=\min(i,j)$. Risolvere il sistema lineare $Ax=b$ con $b=\text{ones}(100,1)$;

TOLLERANZA	FLAG	ITERAZIONI	RESIDUO
10^{-5}	0	49	8.8490e-06
10^{-8}	0	73	6.7002e-09
10^{-10}	0	93	5.7087e-11
eps	0	146	1.0425e-16

Convergenza
rispetto la tolleranza fissata

Per $\text{tol}=10^{-5}, 10^{-8}, 10^{-10}$, pcg termina in
Un numero di iterazione
Inferiore a n.

Per $\text{tol}=\text{eps}$ il residuo è
inferiore ad eps ma in 146
iterazioni >n

al decrescere della tolleranza decresce il residuo e aumenta il numero di iterazione

Esempio: costruire una matrice simmetrica definita positiva, A , partendo da una matrice diagonale di autovalori positivi ed una ortogonale.

```
%costruzione matrice A
D=[1 1.0001 1.0002 1.3 1.4 1.5 1.6 1.7 1.8 10 10.001 10.04 60 60.05 60.01 60.0005 500 997 997.001 997.3 1000];
U=gallery('orthog',max(size(D)));
A=U'*diag(D)*U;
```

Costruzione matrice A

```
display('con tol=10^(-5)')
%vettore colonna dei termini noti
b=ones(max(size(D)),1);
%tolleranza
tol=10^(-5);
%massimo numero di iterazioni
maxit=100;
%chiamata function cg

display('con pcg')
%chiamata function pcg
[X1,FLAG1,RELRES,ITER,RESVEC] = pcg(A,b,tol,maxit)

display('con tol=10^(-8)')

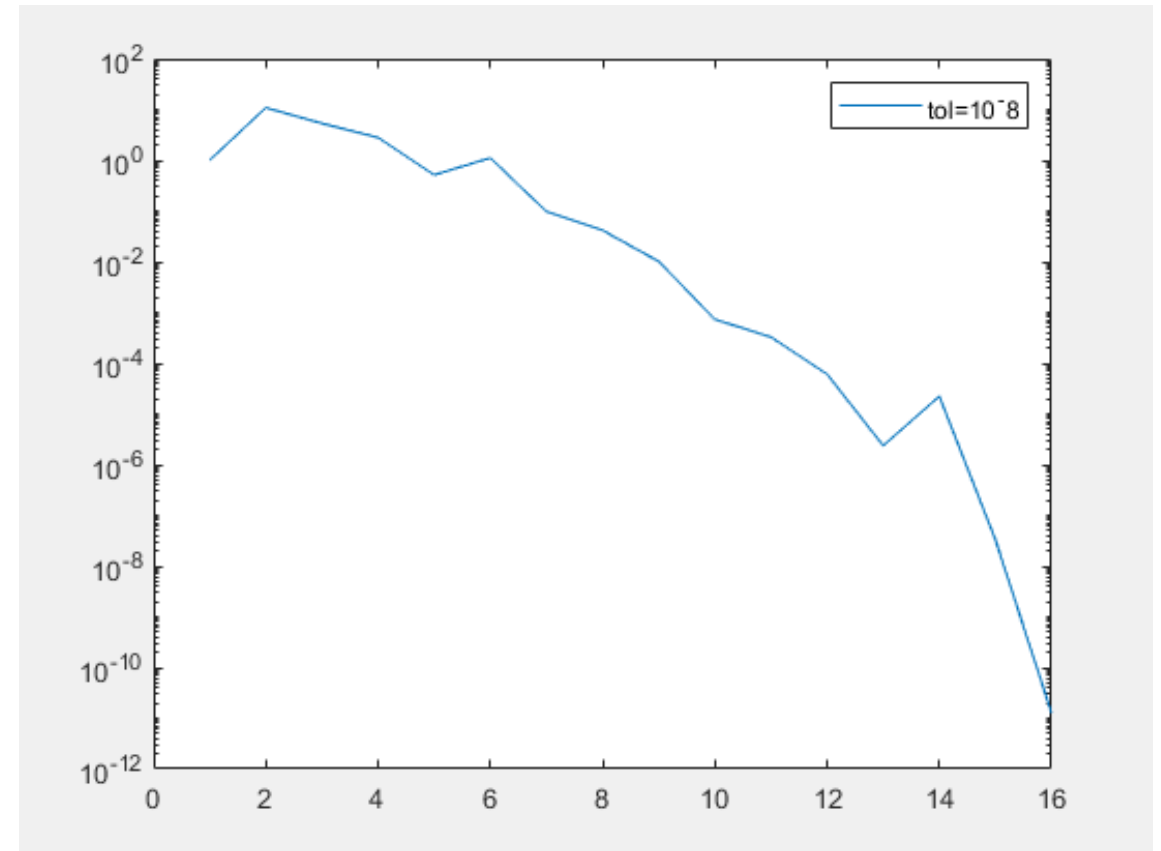
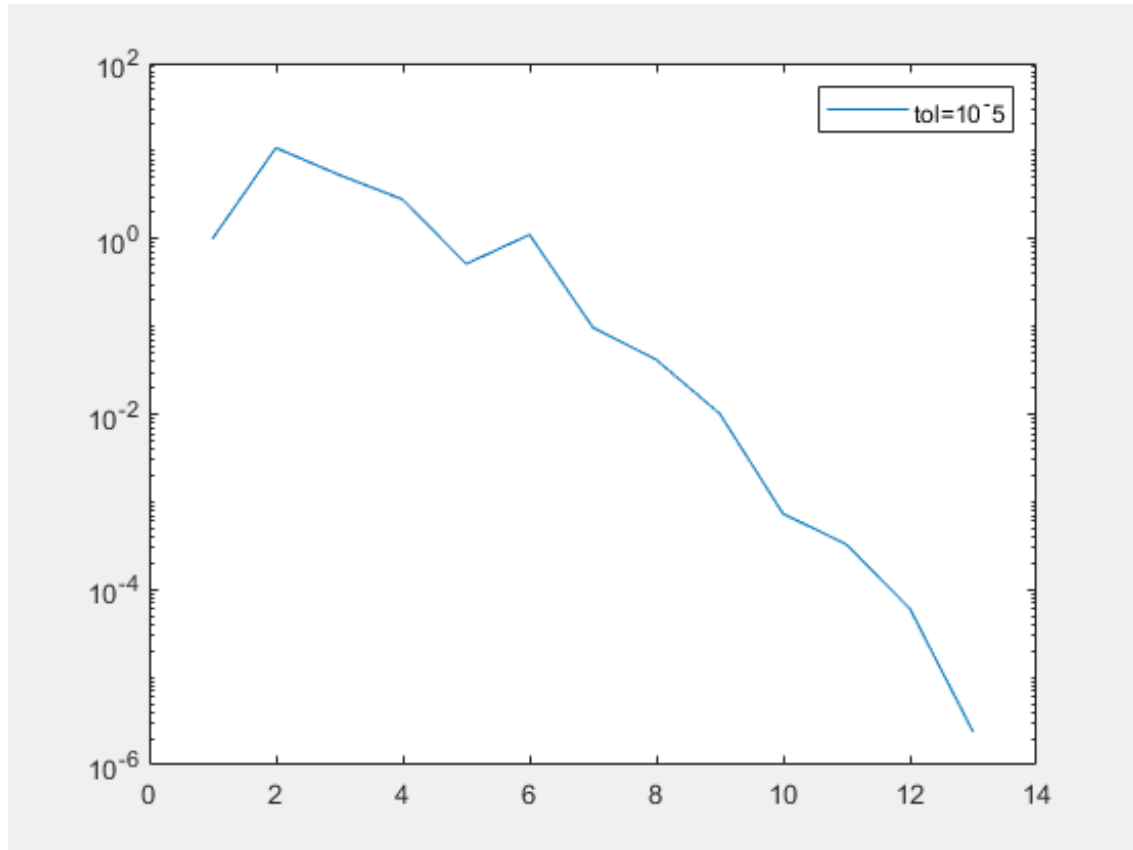
%tolleranza
tol=10^(-8);

[X1,FLAG1,RELRES,ITER,RESVEC] = pcg(A,b,tol,maxit)
```

Function pcg per differenti valori di tol della tolleranza

Esempio: costruire una matrice simmetrica definita positiva, A , partendo da una matrice diagonale di autovalori positivi ed una ortogonale.

Grafici

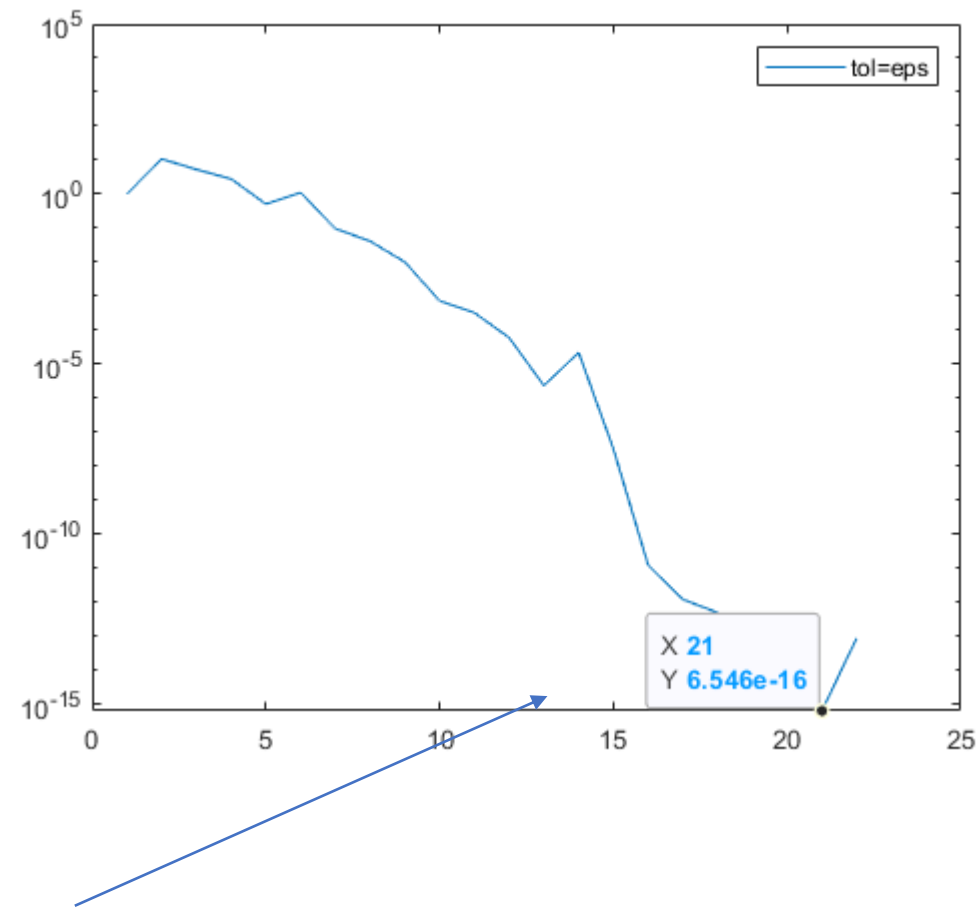


Esempio: costruire una matrice simmetrica definita positiva, A , partendo da una matrice diagonale di autovalori positivi ed una ortogonale.

TOLLERANZA	FLAG	ITERAZIONI	RESIDUO
10^{-5}	0	12	2.3643e-06
10^{-8}	0	15	1.2562e-11

Esempio: costruire una matrice simmetrica definita positiva, A , partendo da una matrice diagonale di autovalori positivi ed una ortogonale.

TOLLERANZ A	FLAG	ITERAZIONI	RESIDUO
eps	3	21	6.546e-16



semiconvergenza

Esempio: costruire una matrice simmetrica definita positiva, A , partendo da una matrice diagonale di autovalori positivi ed una ortogonale.

```
>> auto=sort(eig(A)),
```

```
auto =
```

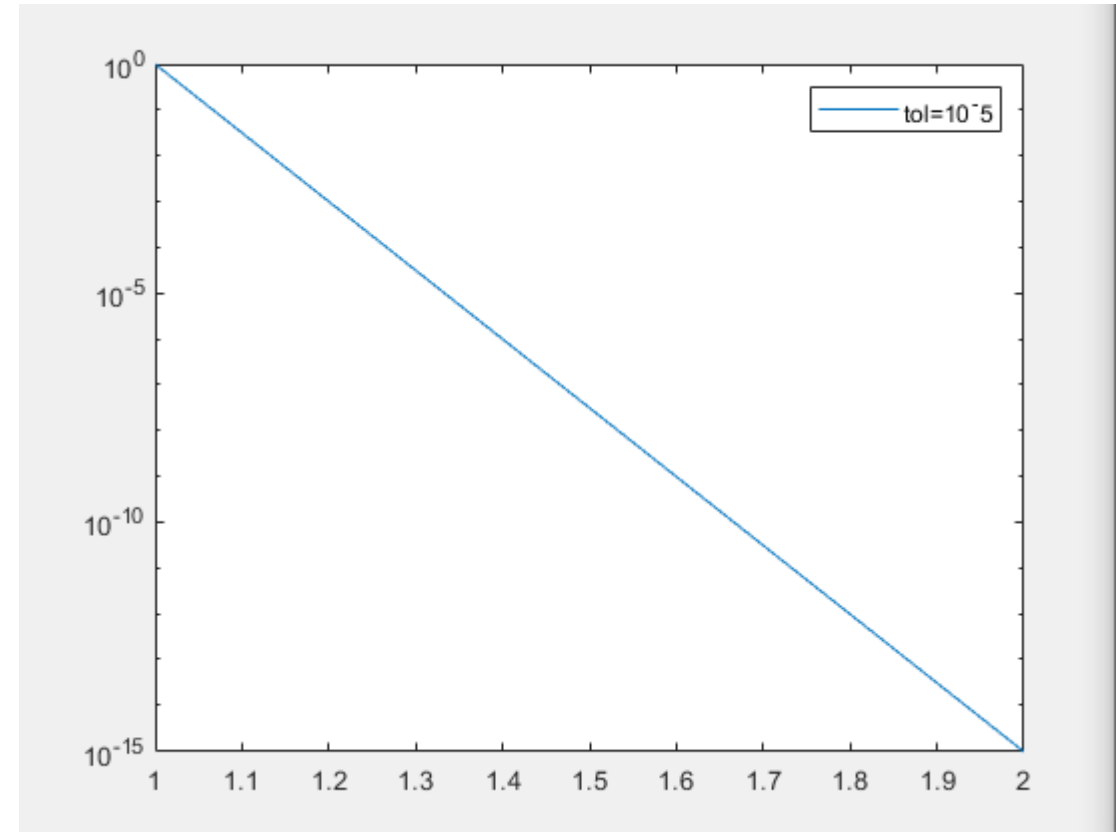
```
69.999999999999758  
69.999999999999758  
69.999999999999886  
69.999999999999915  
69.999999999999943  
69.999999999999943  
69.999999999999957  
69.999999999999972  
69.999999999999972  
70.000000000000000  
70.000000000000000  
70.000000000000000  
70.000000000000014  
70.000000000000028  
70.000000000000043  
70.000000000000043  
70.000000000000071  
70.000000000000085  
70.000000000000099  
70.000000000000185  
70.000000000000213
```

Esempio: costruire una matrice simmetrica definita positiva, A , partendo da una matrice diagonale di autovalori positivi ed una ortogonale.

```
>> auto=sort(eig(A)),
```

```
auto =
```

```
69.999999999999758  
69.999999999999758  
69.999999999999886  
69.999999999999915  
69.999999999999943  
69.999999999999943  
69.999999999999957  
69.999999999999972  
69.999999999999972  
70.000000000000000  
70.000000000000000  
70.000000000000000  
70.000000000000014  
70.000000000000028  
70.000000000000043  
70.000000000000043  
70.000000000000071  
70.000000000000085  
70.000000000000099  
70.000000000000185  
70.000000000000213
```

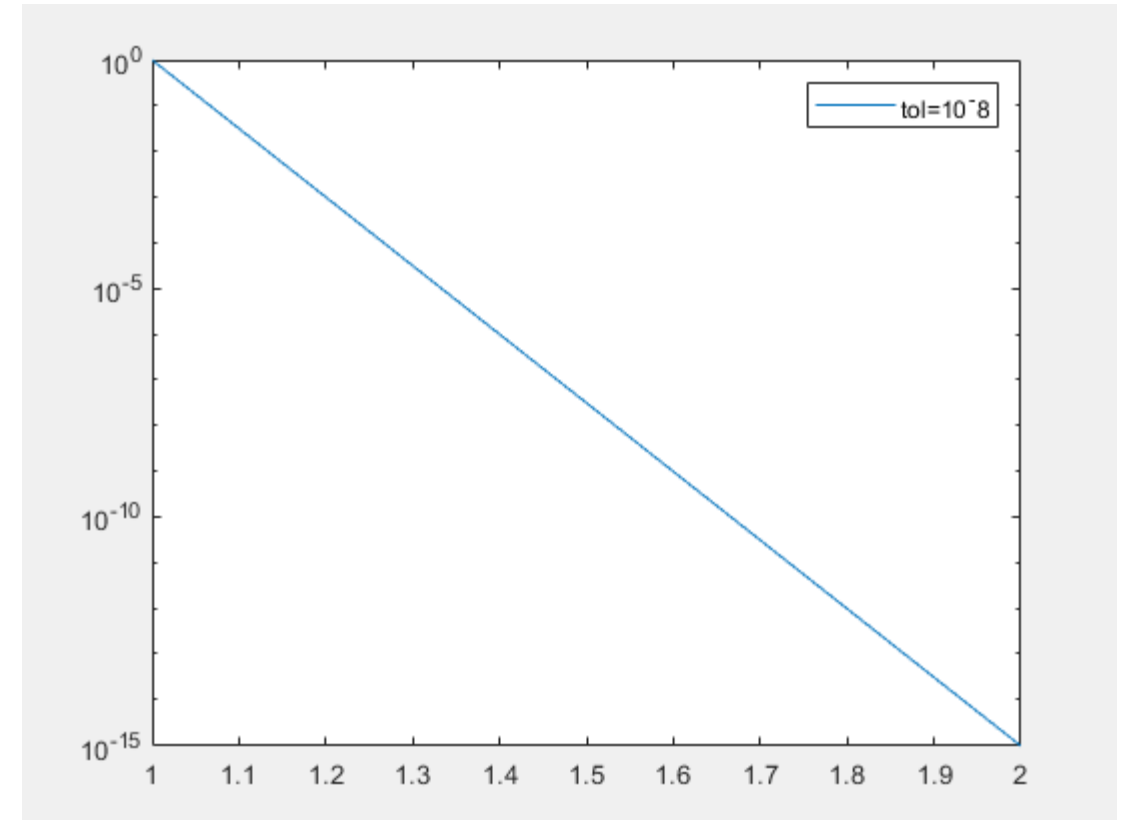


Esempio: costruire una matrice simmetrica definita positiva, A , partendo da una matrice diagonale di autovalori positivi ed una ortogonale.

```
>> auto=sort(eig(A)),
```

```
auto =
```

```
69.999999999999758  
69.999999999999758  
69.999999999999886  
69.999999999999915  
69.999999999999943  
69.999999999999943  
69.999999999999957  
69.999999999999972  
69.999999999999972  
70.000000000000000  
70.000000000000000  
70.000000000000000  
70.000000000000014  
70.000000000000028  
70.000000000000043  
70.000000000000043  
70.000000000000071  
70.000000000000085  
70.000000000000099  
70.000000000000185  
70.000000000000213
```

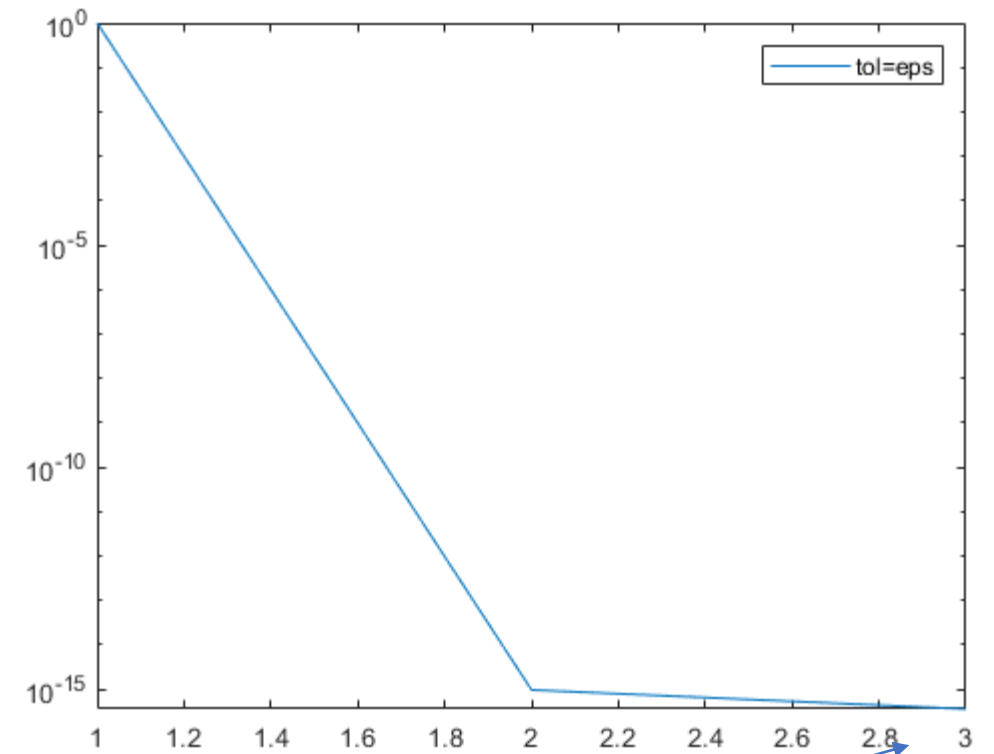


Esempio: costruire una matrice simmetrica definita positiva, A , partendo da una matrice diagonale di autovalori positivi ed una ortogonale.

```
>> auto=sort(eig(A)),
```

```
auto =
```

```
69.999999999999758  
69.999999999999758  
69.999999999999886  
69.999999999999915  
69.999999999999943  
69.999999999999943  
69.999999999999957  
69.999999999999972  
69.999999999999972  
70.000000000000000  
70.000000000000000  
70.000000000000000  
70.000000000000014  
70.000000000000028  
70.000000000000043  
70.000000000000043  
70.000000000000071  
70.000000000000085  
70.000000000000099  
70.000000000000185  
70.000000000000213
```



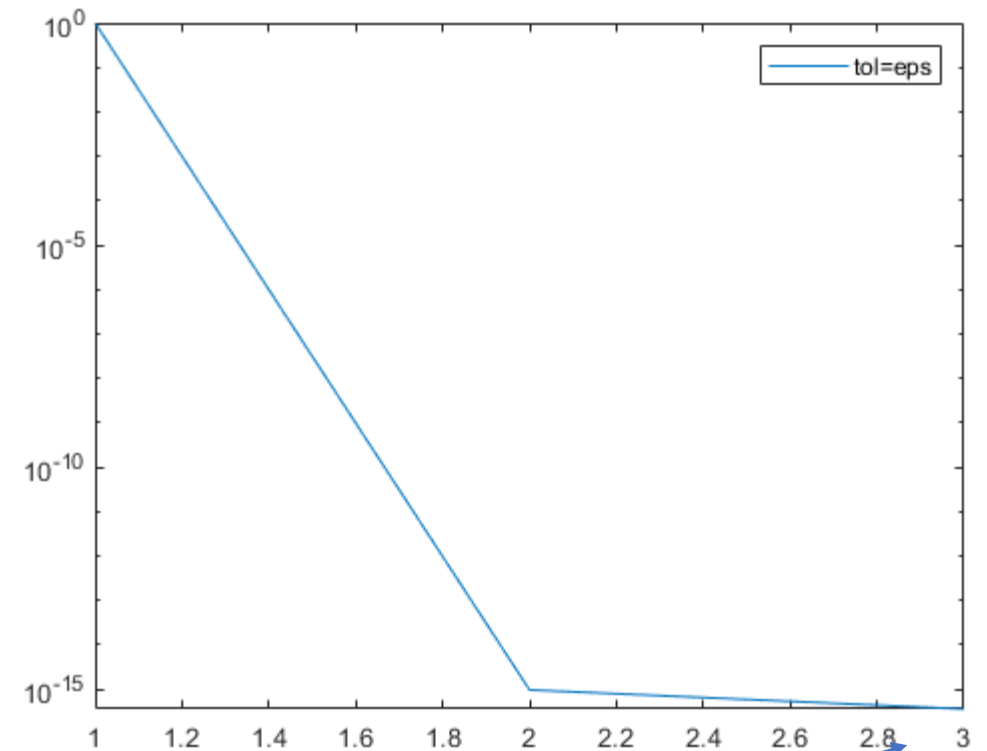
Convergenza numerica in 2 iterazioni

Esempio: costruire una matrice simmetrica definita positiva, A , partendo da una matrice diagonale di autovalori positivi ed una ortogonale.

```
>> auto=sort(eig(A)),
```

```
auto =
```

```
69.999999999999758  
69.999999999999758  
69.999999999999886  
69.999999999999915  
69.999999999999943  
69.999999999999943  
69.999999999999957  
69.999999999999972  
69.999999999999972  
70.000000000000000  
70.000000000000000  
70.000000000000000  
70.000000000000014  
70.000000000000028  
70.000000000000043  
70.000000000000043  
70.000000000000071  
70.000000000000085  
70.000000000000099  
70.000000000000185  
70.000000000000213
```



Convergenza numerica in 2 iterazioni

Notiamo che il numero di autovalori distinti sono maggiori di 2

Esempio: risolvere il sistema con la matrice di poisson (matrice sparsa)

A matrice tridiagonale a blocchi

$$A = \begin{bmatrix} C & D & O \\ D & C & D \\ O & D & C \end{bmatrix}$$

$$C = \begin{bmatrix} 4 & -1 & 0 & 0 \\ -1 & 4 & -1 & 0 \\ 0 & -1 & 4 & -1 \\ 0 & 0 & -1 & 4 \end{bmatrix}$$

$$D = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$$

Caso $(n-1)(m-1)$
Con $n=5$ e $m=4$

poisson — Block tridiagonal matrix from Poisson's equation (sparse)

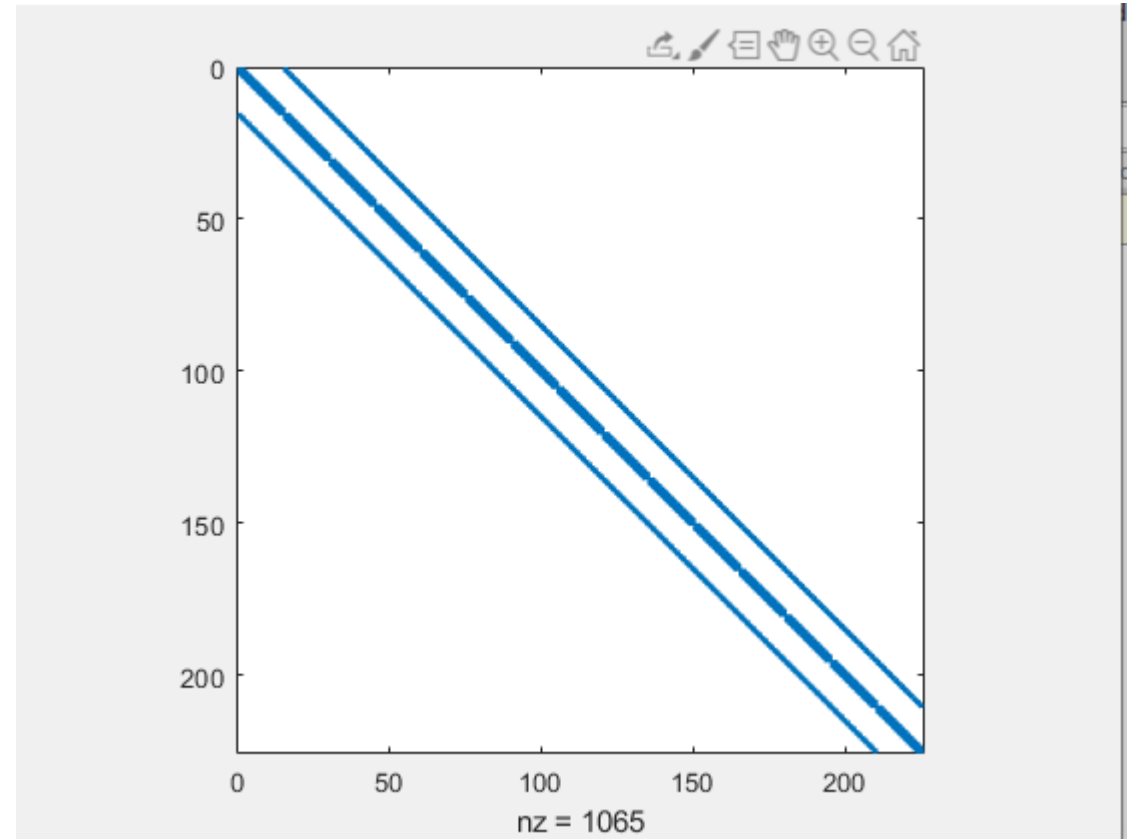
`A = gallery('poisson',n)` returns the block tridiagonal (sparse) matrix of order n^2 resulting from discretizing Poisson's equation.

Esempio: risolvere il sistema con la matrice di poisson (matrice sparsa)

poisson — Block tridiagonal matrix from Poisson's equation (sparse)

`A = gallery('poisson',n)` returns the block tridiagonal (sparse) matrix of order n^2 resulting from discretizing Poisson's equation with the 5-point operator on an n -by- n mesh.

```
>> A=gallery('poisson',15);  
>> spy(A)
```

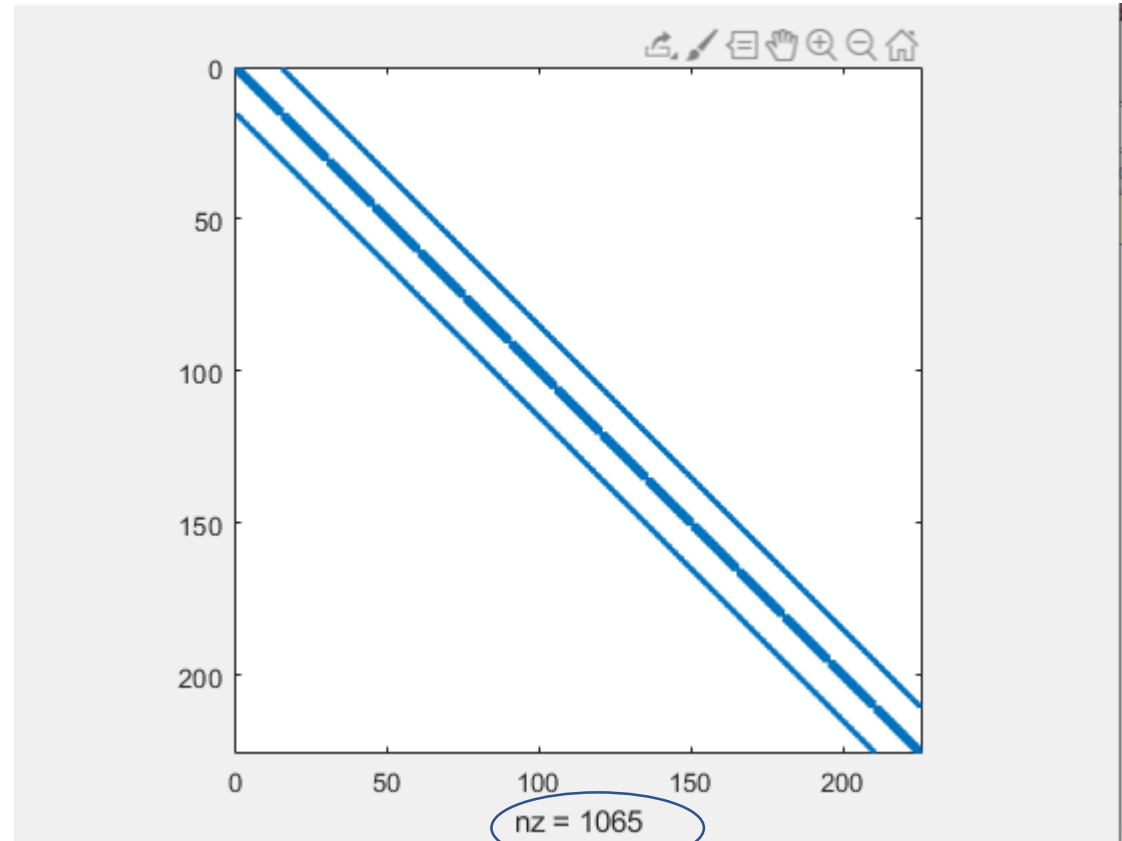


Esempio: risolvere il sistema con la matrice di poisson (matrice sparsa)

```
>> A=gallery('poisson',15);  
>> spy(A)  
>> size(A)
```

ans =

225 225

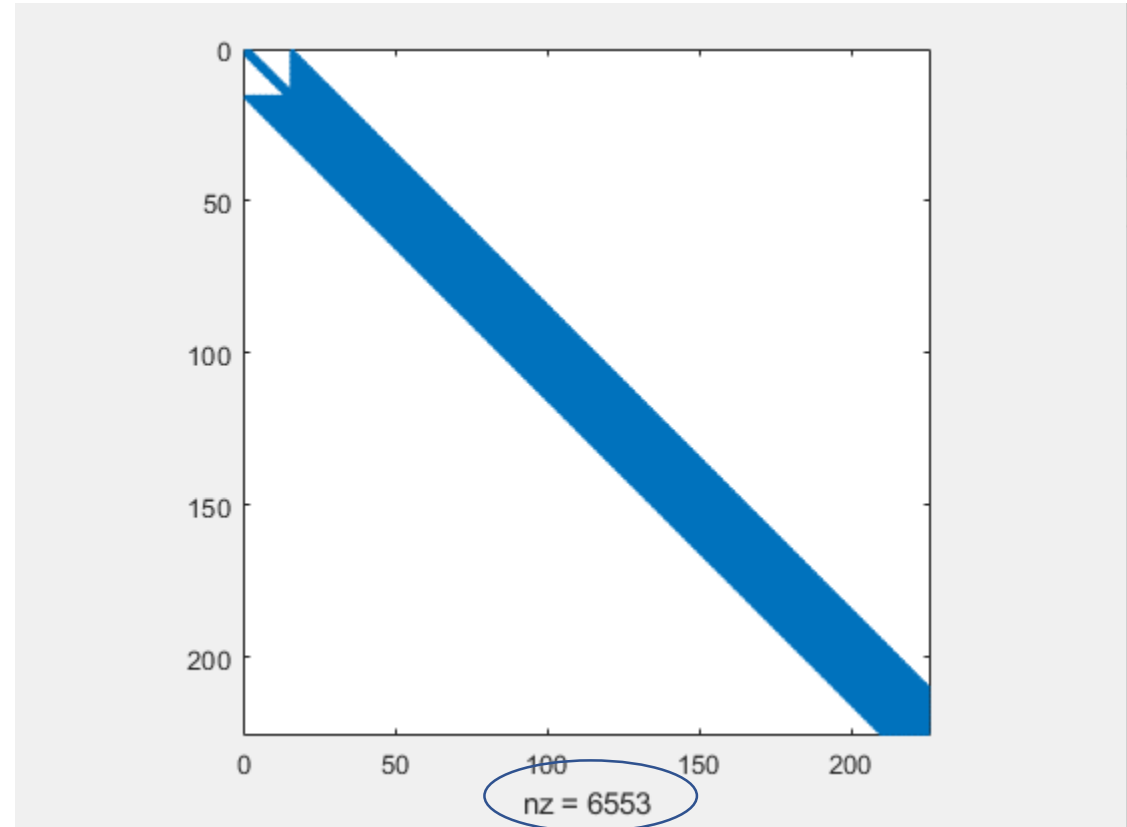


Elementi diversi da 0 su 50625

Esempio: risolvere il sistema con la matrice di poisson (matrice sparsa) con il metodo di Gauss

```
>> A=gallery('poisson',15);  
>> spy(A)  
>> [L,U]=lu(A);  
>> y=L\b;  
>> x=U\y;
```

Fill-in



Elementi diversi da 0 maggiore di 1065
(elementi diversi da 0 della matrice di poisson)

Metodo iterativo: metodo del gradiente

```
1  function [x,error,niter,flag]=gradiente (A,x,b,maxiter,tol)
2  -   flag=0;
3  -   niter=0;
4  -   bnorm2=norm(b);
5  -   r=b-A*x;
6  -   error=norm(r)/bnorm2;
7  -   if (error<tol)
8  -       return
9  -   end
10 -   for niter=1:maxiter
11 -       rho=r'*r;
12 -       q=A*r;
13 -       alpha=rho/(r'*q);
14 -       x=x+alpha*r;
15 -       r=r-alpha*q;
16 -       error=norm(r)/bnorm2;
17 -       if (error<=tol)
18 -           break
19 -       end
20 -   end
21 -   if error>tol
22 -       flag=1;
23 -   end
24
25
26 -   end
```

Esempio: risolvere il sistema con la matrice di poisson (matrice sparsa) con il metodo iterativi e cg

```
>> [x,flag,relres,iter,resvec]=pcg(A,b,10^-7,500);  
>> error_cg=resvec/norm(b);  
>> hold on  
>> semilogy(1:iter,(error_cg))
```

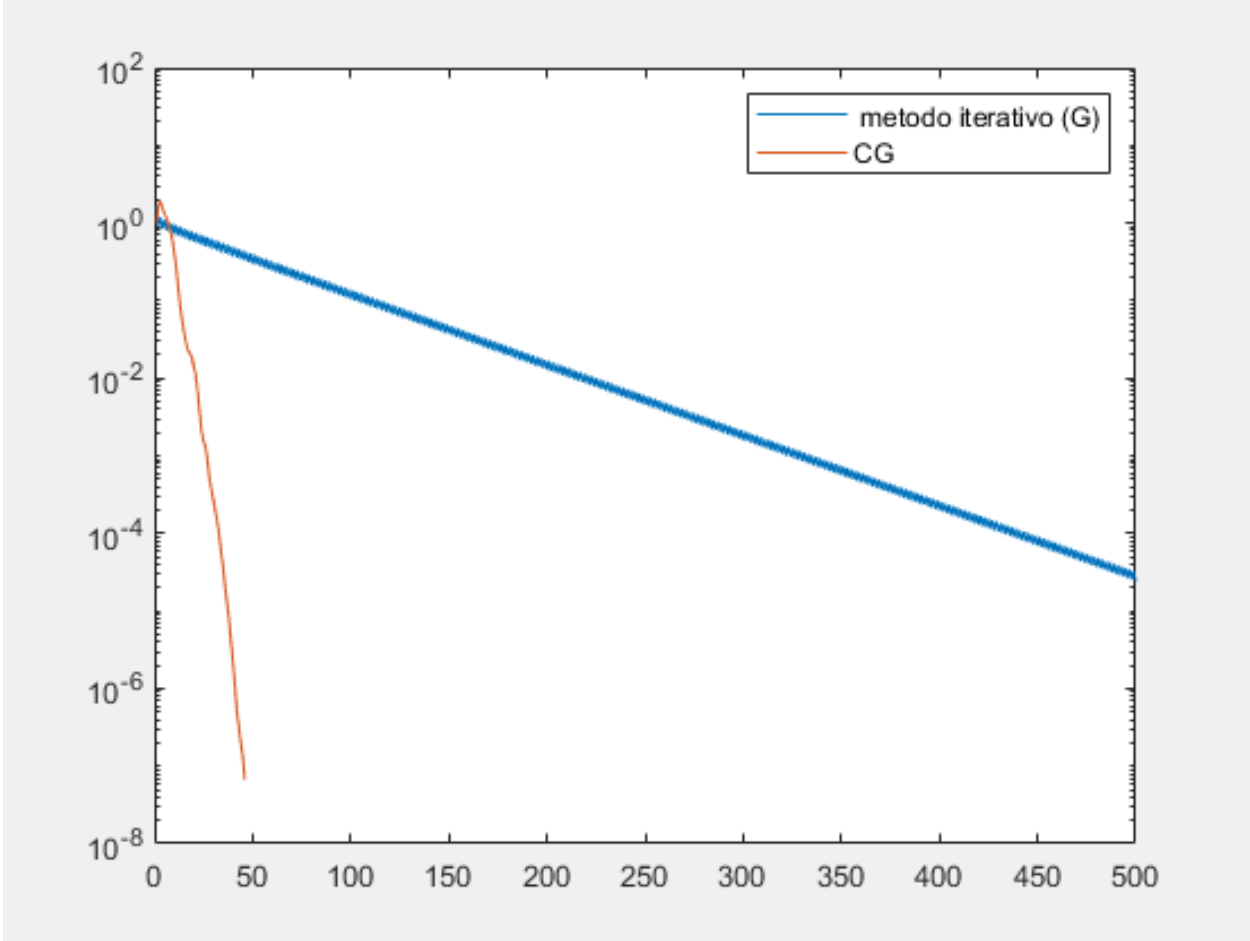
```
>> [x,error,niter,flag]=gradiente(A,zeros(225,1),b,500,10^-7)  
>> semilogy(1:size(error_cg,1),(error_cg))
```

metodo	flag	iter	Residuo (normalizzato)
Metodo iterativo (G)	1	500	2.4407e-05
Metodo CG	0	45	6.6571e-08

Esempio: risolvere il sistema con la matrice di poisson (matrice sparsa) con il metodo iterativi e cg

metodo	flag	iter	Residuo (normalizzato)
Metodo iterativo (G)	1	500	2.4407e-05
Metodo CG	0	45	6.6571e-08

Grafico



Esempio: confronto metodo iterativo (G) e metodo CG

```
>> A=sprand(500,500,.5);
```

```
>> A=A'*A;
```

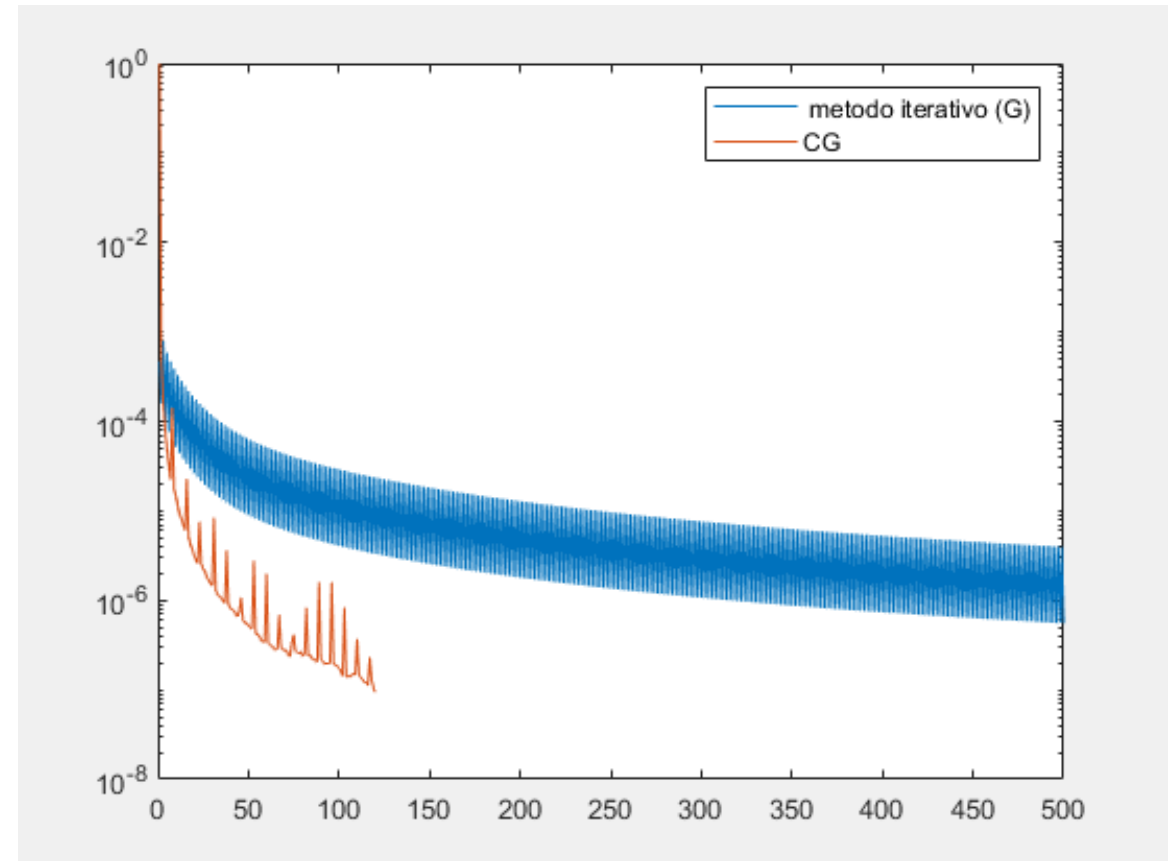
```
>> b=sum(A,2);
```

```
>> [x,flag,relres,iter,resvec]=pcg(A,b,10^-7,size(A,1)); >> [x,error,niter,flag]=gradiente(A,zeros(500,1),b,size(A,1),10^-7)
```

Metodo	flag	iterazioni	Residuo
Metodo iterativo (G)	1	500	5.5300e-07
Metodo CG	0	119	9.3846e-08

Esempio: confronto metodo iterativo (G) e metodo CG

Metodo	flag	iterazioni	Residuo
Metodo iterativo (G)	1	500	5.5300e-07
Metodo CG	0	119	9.3846e-08



Esempio: risolvere il sistema $Ax=b$ con A matrice non simmetrica

```
>> A = west0479;
```

```
>> b = sum(A,2);
```

```
>> [x,flag,relres,iter,resvec]=cgs(A,b,tol,size(A,1));
```

```
>> flag
```

```
flag =
```

```
1
```

```
>> relres
```


```
relres =
```

```
0.5639
```

```
>> iter
```


```
iter =
```

```
454
```



Non c'è convergenza e il condizionamento è $1.4244e+12$

Esempio: risolvere il sistema $Ax=b$ con A matrice non simmetrica

```
>> A = west0479;   
>> b = sum(A,2);
```

```
>> [x,flag,relres,iter,resvec]=cgs(A,b,tol,size(A,1));
```

```
>> flag
```

```
flag =
```

```
1 
```

```
>> relres
```

```
relres =
```

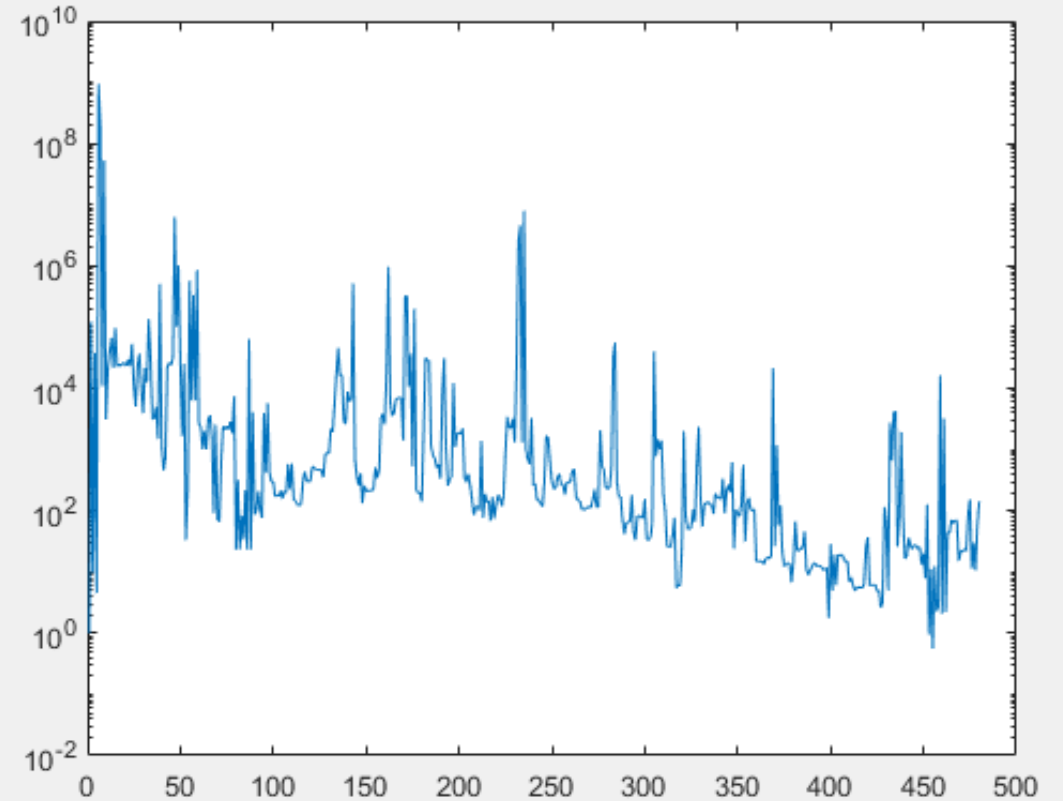
```
0.5639
```

```
>> iter
```

```
iter =
```

```
454
```

Grafico residuo



Esempio: risolvere il sistema $Ax=b$ con A matrice non simmetrica

```
>> A = west0479;  
>> b = sum(A,2);  
>> tol=10^-7;
```

Matrice di preconditionamento

```
[L,U] = ilu(A);
```

Esempio: risolvere il sistema $Ax=b$ con A matrice non simmetrica

```
>> A = west0479;  
>> b = sum(A,2);  
>> tol=10^-7;
```

Matrice di preconditionamento

```
[L,U] = ilu(A);
```

```
>> [x,flag,relres,iter,resvec]=pcg(A,b,tol,size(A,1),L,U);  
>> flag
```

flag =

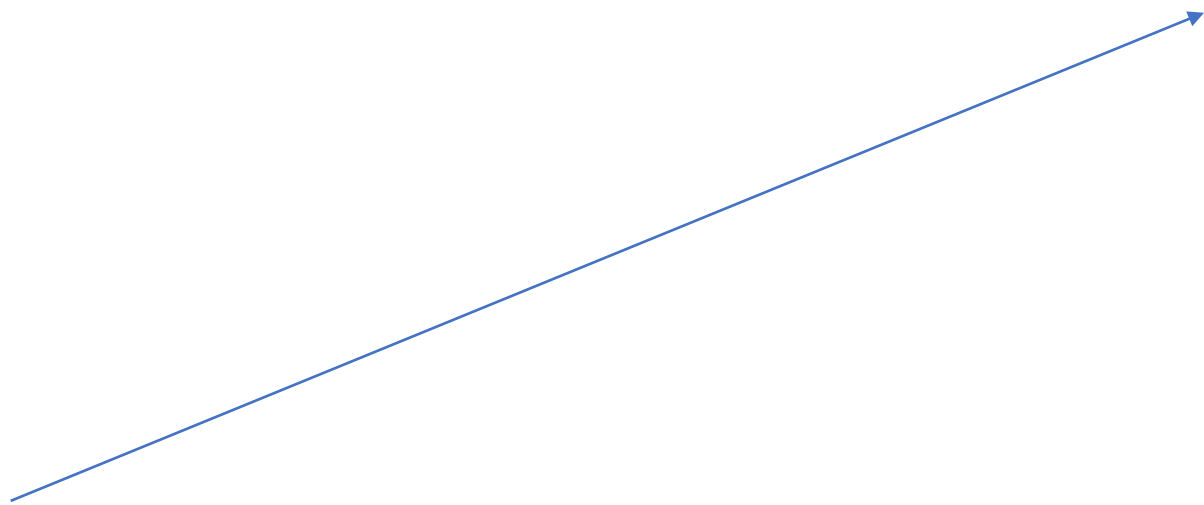
0

```
>> iter
```

iter =

1

Convergenza in 1 iterazione



Esempio: risolvere il sistema $Ax=b$ con A matrice non simmetrica

```
>> A = west0479;  
>> b = sum(A,2);  
>> tol=10^-7;
```

Matrice di preconditionamento

```
[L,U] = ilu(A);
```

```
>> [x,flag,relres,iter,resvec]=pcg(A,b,tol,size(A,1),L,U);  
>> flag
```

flag =

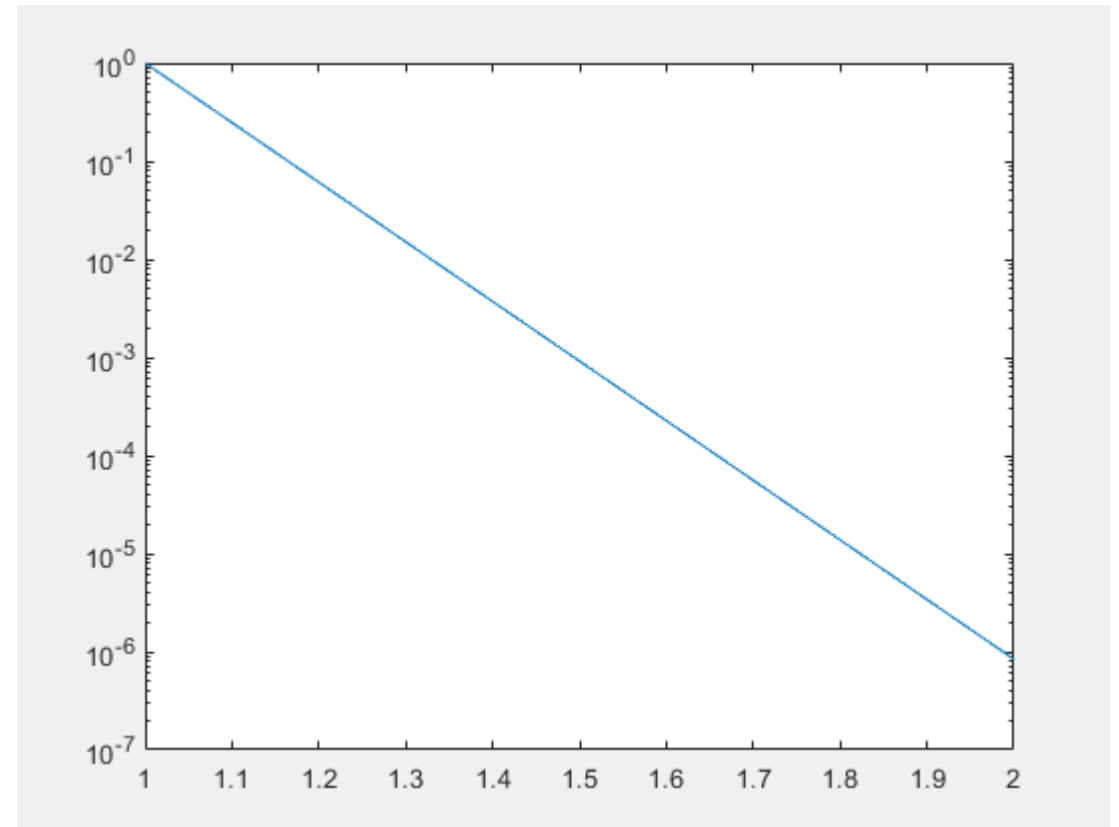
0

```
>> iter
```

iter =

1

Grafico residuo



Esempio: costruire una matrice simmetrica definita positiva, A , partendo da una matrice diagonale di autovalori clusterizzati positivi ed una ortogonale.

```
D=[1 999 999.09 999.1 999.1999 999.2 999.2999 999.4 999.4999 999.49999  
999.5 999.5999 999.6 999.6999 999.7 999.71 999.7999 999.8 999.8999 999.9 1000];  
U=gallery('orthog',max(size(D)));  
A=U'*diag(D)*U;
```

```
display('con tol=10^(-5)')  
%vettore colonna dei termini noti  
b=ones(max(size(D)),1);  
%tolleranza  
tol=10^(-5);  
%massimo numero di iterazioni  
maxit=100;  
%chiamata function cg
```

```
display('con pcg')  
%chiamata function pcg  
[X1,FLAG1,RELRES,ITER,RESVEC] = pcg(A,b,tol,maxit)
```

```
display('con tol=10^(-8)')
```

```
%tolleranza  
tol=10^(-8);
```

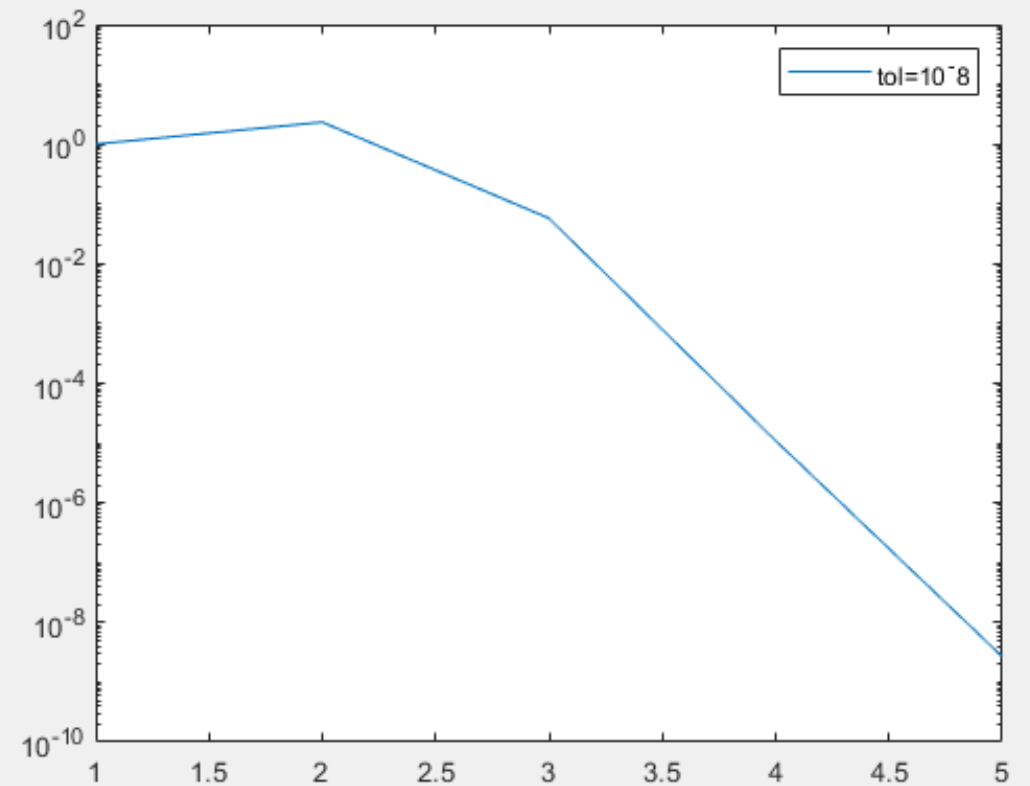
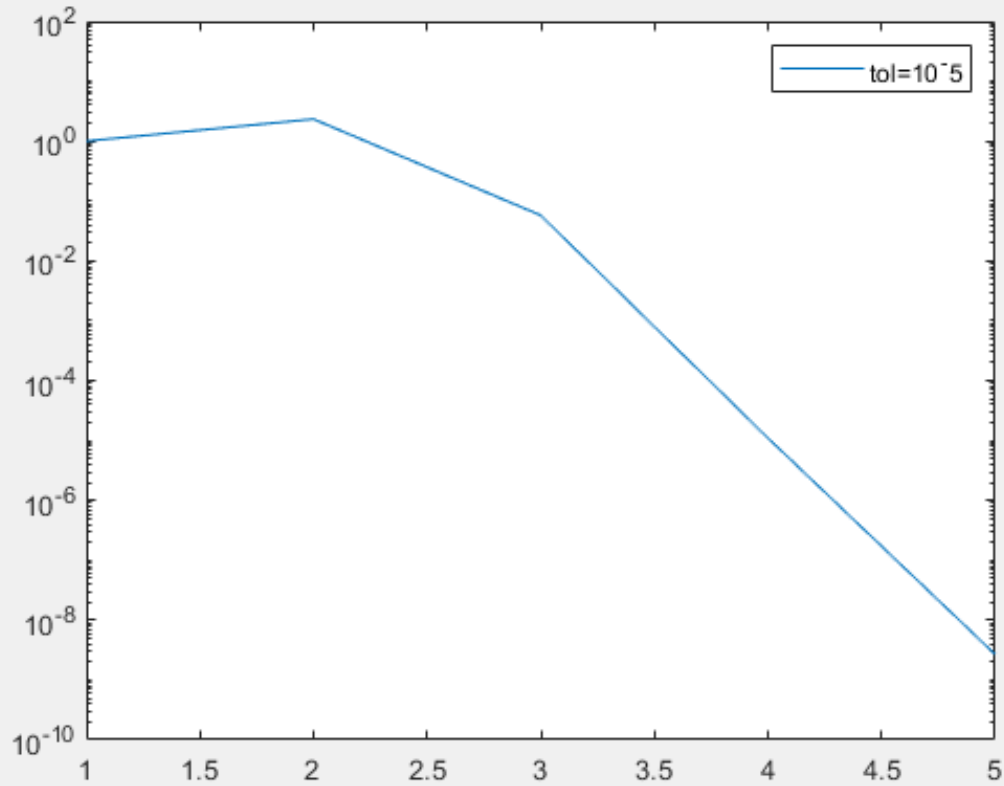
```
[X1,FLAG1,RELRES,ITER,RESVEC] = pcg(A,b,tol,maxit)
```

Costruzione matrice A

Function pcg per differenti valori di tol della tolleranza

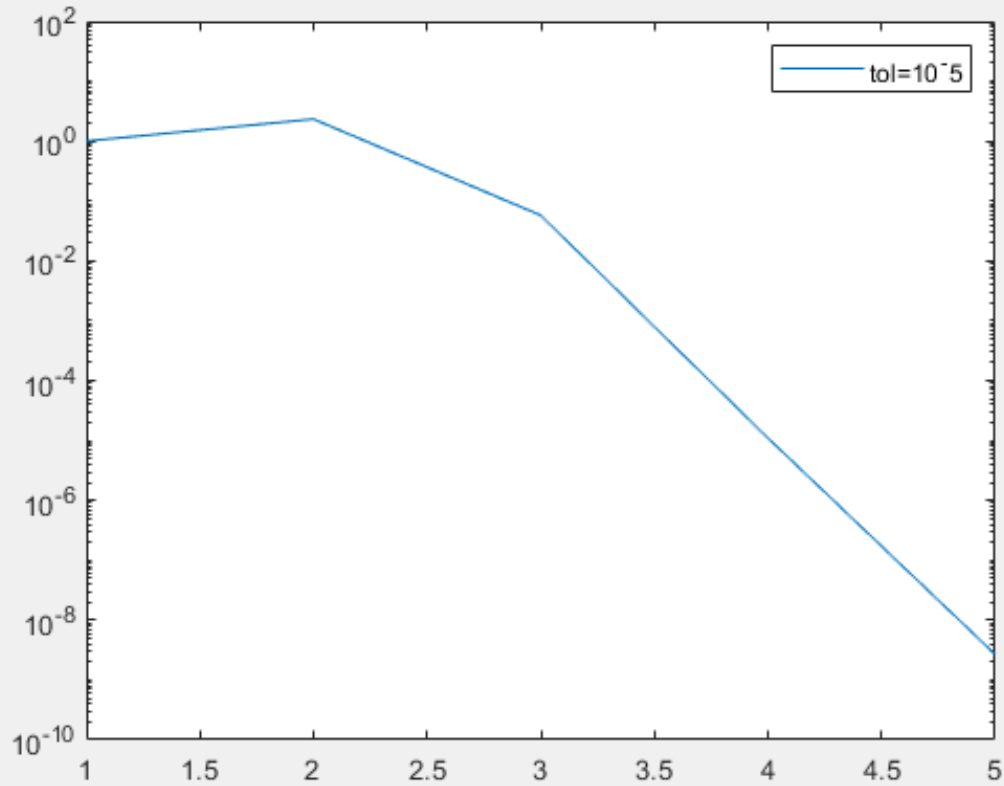
Esempio: costruire una matrice simmetrica definita positiva, A , partendo da una matrice diagonale di autovalori clusterizzati positivi ed una ortogonale.

Grafici residuo

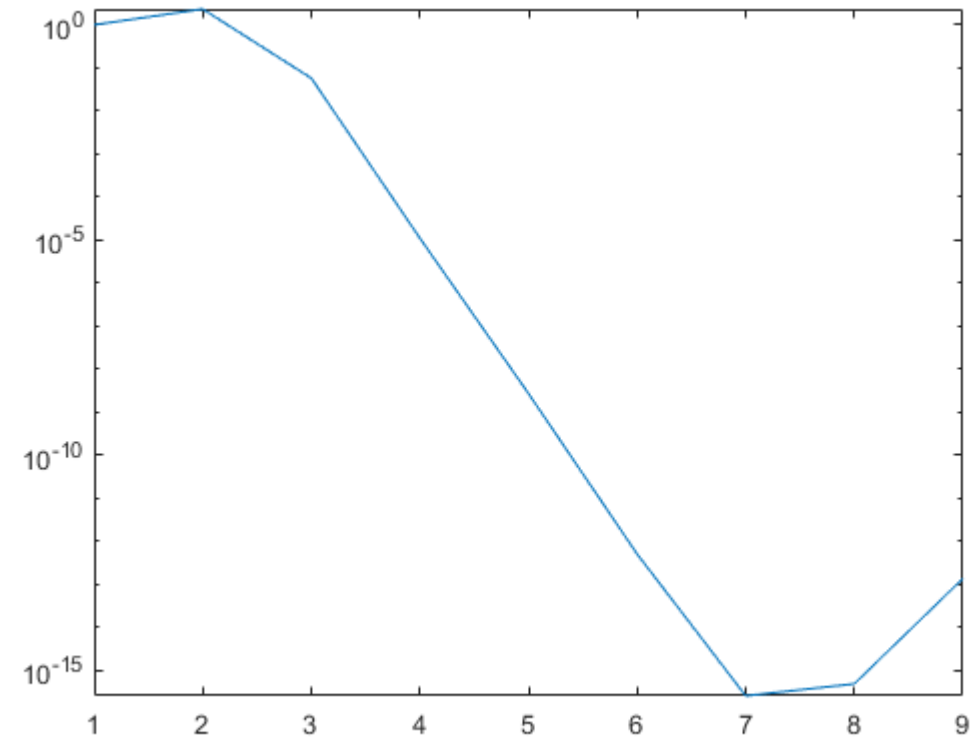


Esempio: costruire una matrice simmetrica definita positiva, A , partendo da una matrice diagonale di autovalori clusterizzati positivi ed una ortogonale.

Grafici residuo



Tol=eps



semiconvergenza

Esempio: costruire una matrice simmetrica definita positiva, A, partendo da una matrice diagonale di autovalori positivi ed una ortogonale.

>> num_elem_dist

il numero di
Autovalori distinti

num_elem_dist =

21

TOLLERANZA	FLAG	ITERAZIONI	RESIDUO
10^-5	0	4	2.6796e-09
10^-8	0	4	2.6796e-09
eps	3	8	1.3369e-13

Numero di iterazione è minore di 21

Esempio: costruire una matrice simmetrica definita positiva, A, partendo da una matrice diagonale di autovalori positivi ed una ortogonale.

```
>> num_elem_dist
num_elem_dist =
```

il numero di
Autovalori distinti

3	Failure — pcg stagnated after two consecutive iterations were the same.		
TOLLERANZA	FLAG	ITERAZIONI	RESIDUO
10^-5	0	4	2.6796e-09
10^-8	0	4	2.6796e-09
eps	3	8	1.3369e-13

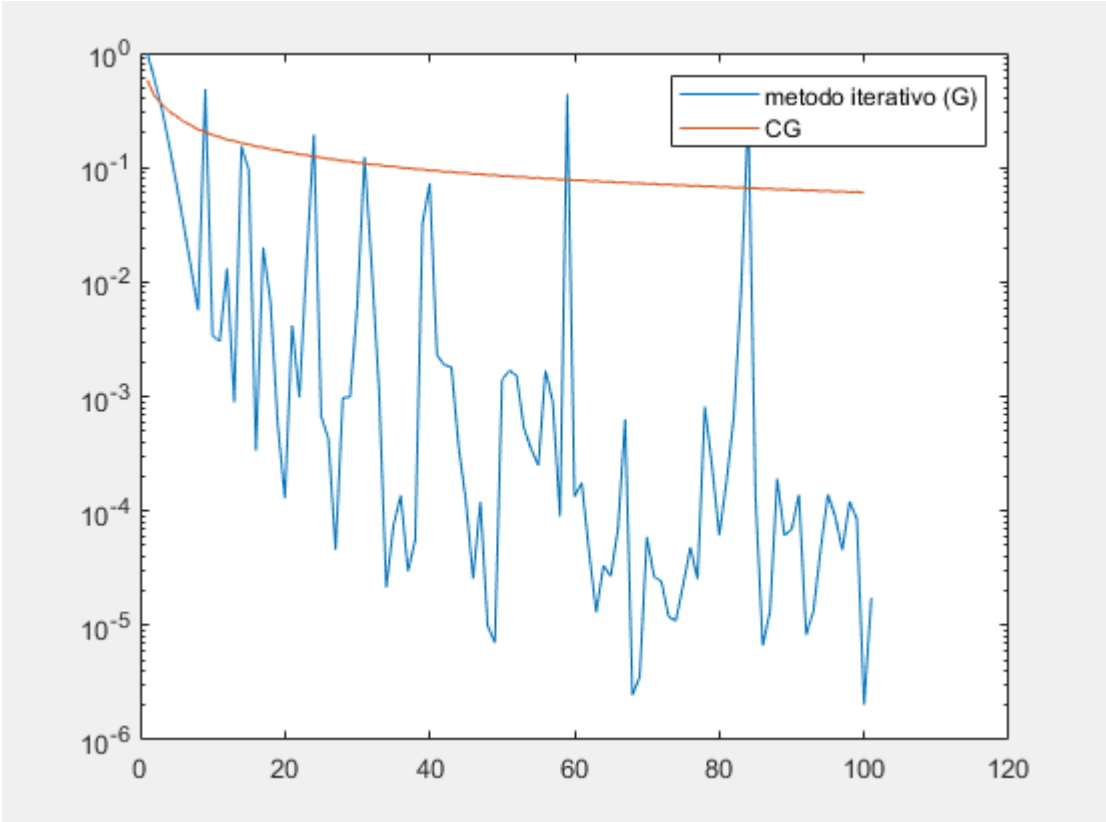
Numero di iterazione è minore di 21

Esempio: risolvere il sistema $Ax=b$ con A matrice malcondizionata

```
>> A=hilb(100);  
>> x0=zeros(100,1);  
>> b=ones(100,1);  
>> [x,error,niter,flag]=gradiente(A,x0,b,maxit,tol)  
>> semilogy(1:500,(error))
```

```
>> [x,flag,relres,iter,resvec]=pcg(A,b,10^-7,size(A,1));  
>> error_cg=resvec/norm(b);  
>> hold on  
  
>> semilogy(1:size(error_cg,1),(error_cg))
```

metodo	iterazione	flag	residuo
Metodo iterativo (G)	100	1	0.0602
Metodo CG	100	1	2.0268e-06

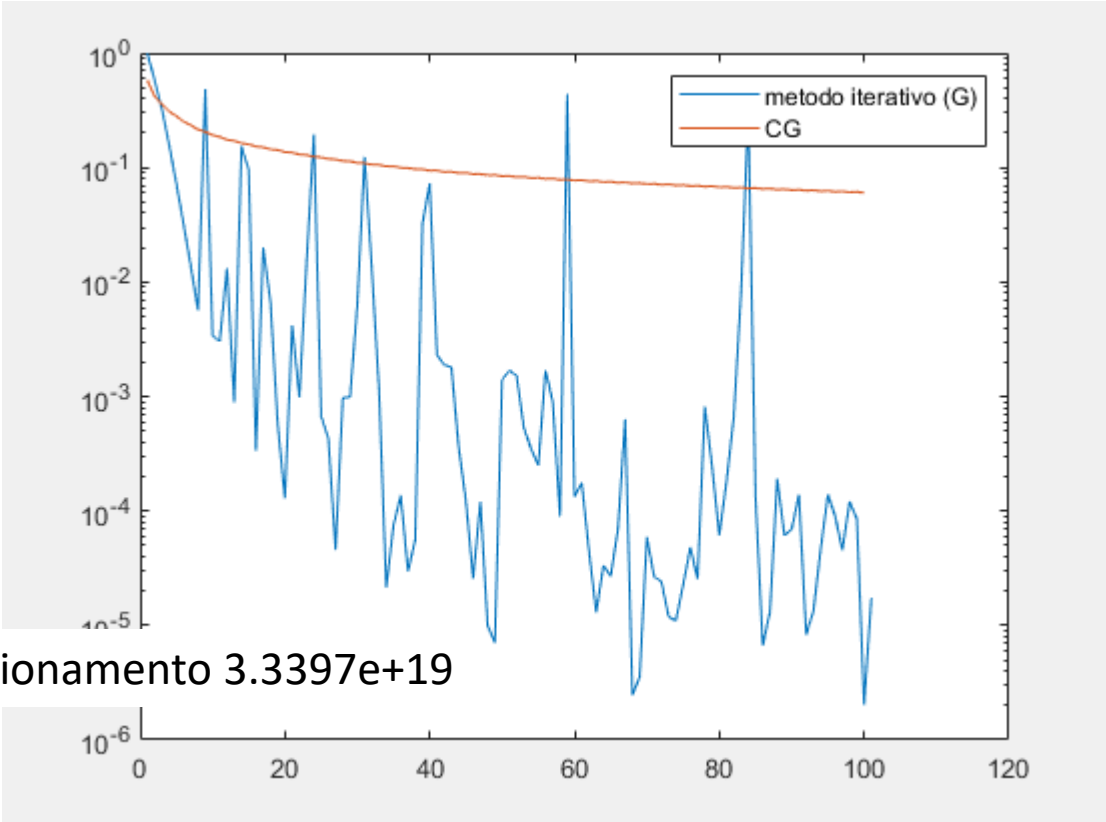


Esempio: risolvere il sistema $Ax=b$ con A matrice malcondizionata

```
>> A=hilb(100);  
>> x0=zeros(100,1);  
>> b=ones(100,1);  
>> [x,error,niter,flag]=gradiente(A,x0,b,maxit,tol)  
>> semilogy(1:500,(error))
```

metodo	iterazione	flag	residuo
Metodo iterativo (G)	100	1	0.0602
Metodo CG	100	1	2.0268e-06

```
>> [x,flag,relres,iter,resvec]=pcg(A,b,10^-7,size(A,1));  
>> error_cg=resvec/norm(b);  
>> hold on  
  
>> semilogy(1:size(error_cg,1),(error_cg))
```



Esempio: risolvere il sistema con il metodo pcg con A matrice di hilbert

```
>> A=hilb(200);  
>> [M]=mat_pre(A);  
>> [x,flag,relres,iter,resvec]=pcg(A,sum(A,2),10^-7,1000,M);
```

Matrice singolare

Convergenza passando ad un sistema
precondizionato

```
>> flag  
flag =  
    0  
>> relres  
relres =  
    5.389364406844510e-08  
>> iter  
iter =
```

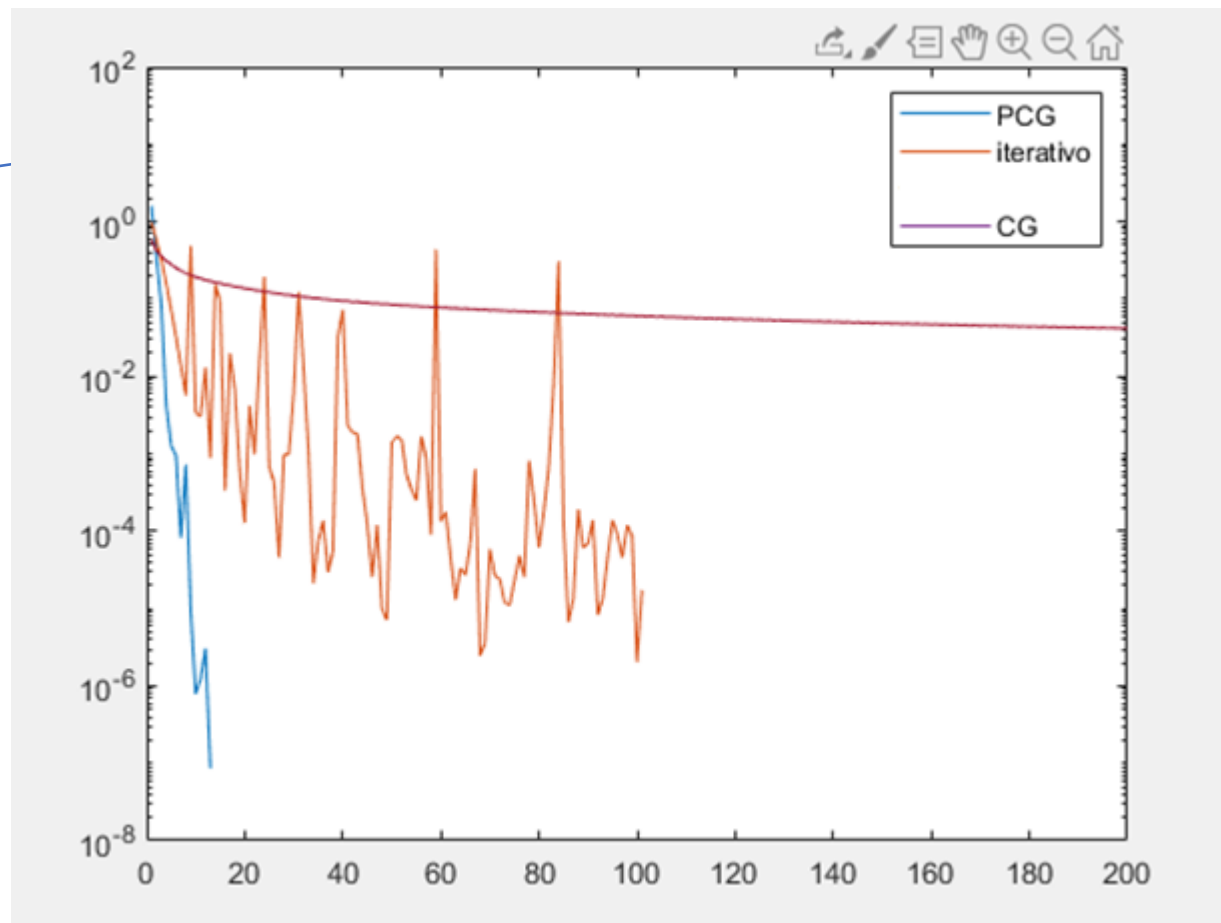
12

Esempio: risolvere il sistema con il metodo pcg con A matrice di hilbert

```
>> A=hilb(200);  
>> [M]=mat_pre(A);  
>> [x,flag,relres,iter,resvec]=pcg(A,sum(A,2),10^-7,1000,M);
```

```
>> flag  
flag =  
    0  
>> relres  
relres =  
    5.389364406844510e-08  
>> iter  
iter =  
  
    12
```

Matrice singolare



Esercizio: risolvere il sistema $Ax=b$ con A associata alla discretizzazione dell'operatore differenziale di Laplace sul dominio $[-1,1] \times [-1,1]$ di dimensione $m=16$ e ripetere per $m=400$, con $\text{tol}=10^{-10}$;

A matrice tridiagonale a blocchi

$$A = \begin{bmatrix} C & D & O \\ D & C & D \\ O & D & C \end{bmatrix} \quad C = \begin{bmatrix} 4 & -1 & 0 & 0 \\ -1 & 4 & -1 & 0 \\ 0 & -1 & 4 & -1 \\ 0 & 0 & -1 & 4 \end{bmatrix} \quad D = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$$

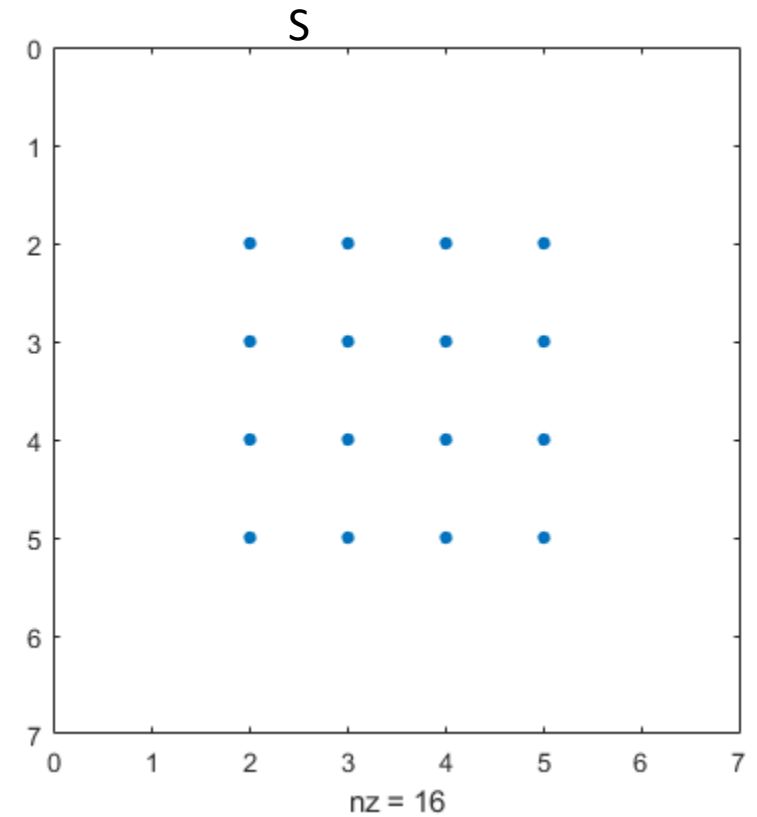
Caso $(n-1)(m-1)$
Con $n=5$ e $m=4$

Esercizio: risolvere il sistema $Ax=b$ con A associata alla discretizzazione dell'operatore differenziale di Laplace sul dominio $[-1,1] \times [-1,1]$ di dimensione $m=16$ e ripetere per $m=400$, con $\text{tol}=10^{-10}$;

```
n=6;  
% mxm dimensione della matrice A  
m=16;  
G=numgrid('S',n);  
%questa matrice di dimensione 16 è  
%associata alla discretizzazione  
%dell'operatore differenziale di Laplace  
%sul dominio  $[-1,1] \times [-1,1]$   
A=delsq(G);  
%massimo numero di iterazioni  
maxit=200;  
%tolleranza  
tol=10^(-10);  
%vettore colonna dei termini noti  
b=[2 1 1 2 1 0 0 1 1 0 0 1 2 1 1 2]';
```


Esercizio: risolvere il sistema $Ax=b$ con A associata alla discretizzazione dell'operatore differenziale di Laplace sul dominio $[-1,1] \times [-1,1]$ di dimensione $m=16$ e ripetere per $m=400$, con $\text{tol}=10^{-10}$;

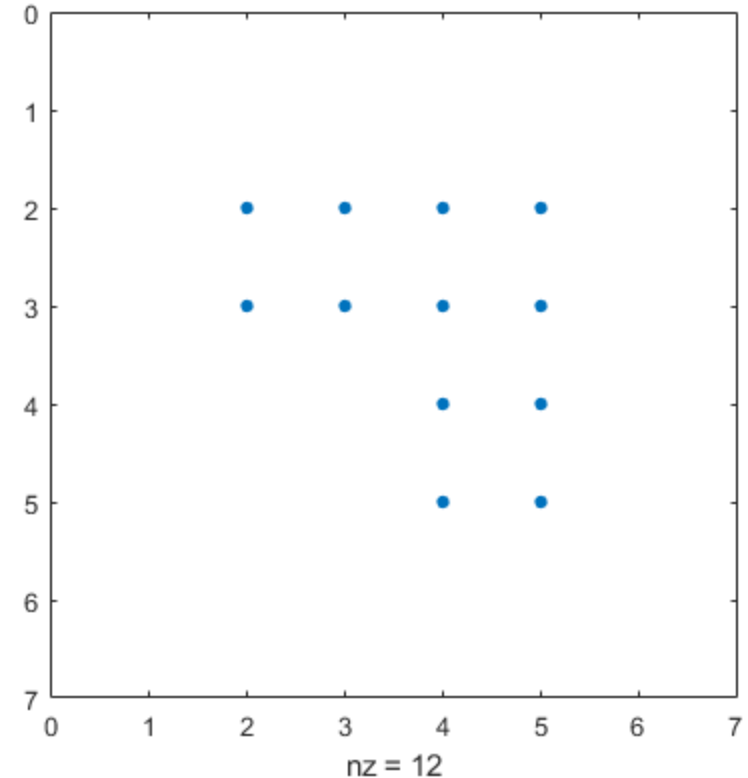
```
n=6;  
% mxm dimensione della matrice A  
m=16;  
G=numgrid('S',n);  
%questa matrice di dimensione 16 è  
%associata alla discretizzazione  
%dell'operatore differenziale di Laplace  
%sul dominio [-1,1]x[-1,1]  
A=delsq(G);  
%massimo numero di iterazioni  
maxit=200;  
%tolleranza  
tol=10^(-10);  
%vettore colonna dei termini noti  
b=[2 1 1 2 1 0 0 1 1 0 0 1 2 1 1 2]';
```



Esercizio: risolvere il sistema $Ax=b$ con A associata alla discretizzazione dell'operatore differenziale di Laplace sul dominio $[-1,1] \times [-1,1]$ di dimensione $m=16$ e ripetere per $m=400$, con $\text{tol}=10^{-10}$;

```
n=6;  
% mxm dimensione della matrice A  
m=16;  
G=numgrid('S',n);  
%questa matrice di dimensione 16 è  
%associata alla discretizzazione  
%dell'operatore differenziale di Laplace  
%sul dominio [-1,1]x[-1,1]  
A=delsq(G);  
%massimo numero di iterazioni  
maxit=200;  
%tolleranza  
tol=10^(-4);  
%vettore colonna dei termini noti  
b=[2 1 1 2 1 0 0 1 1 0 0 1 2 1 1 2]';
```

L

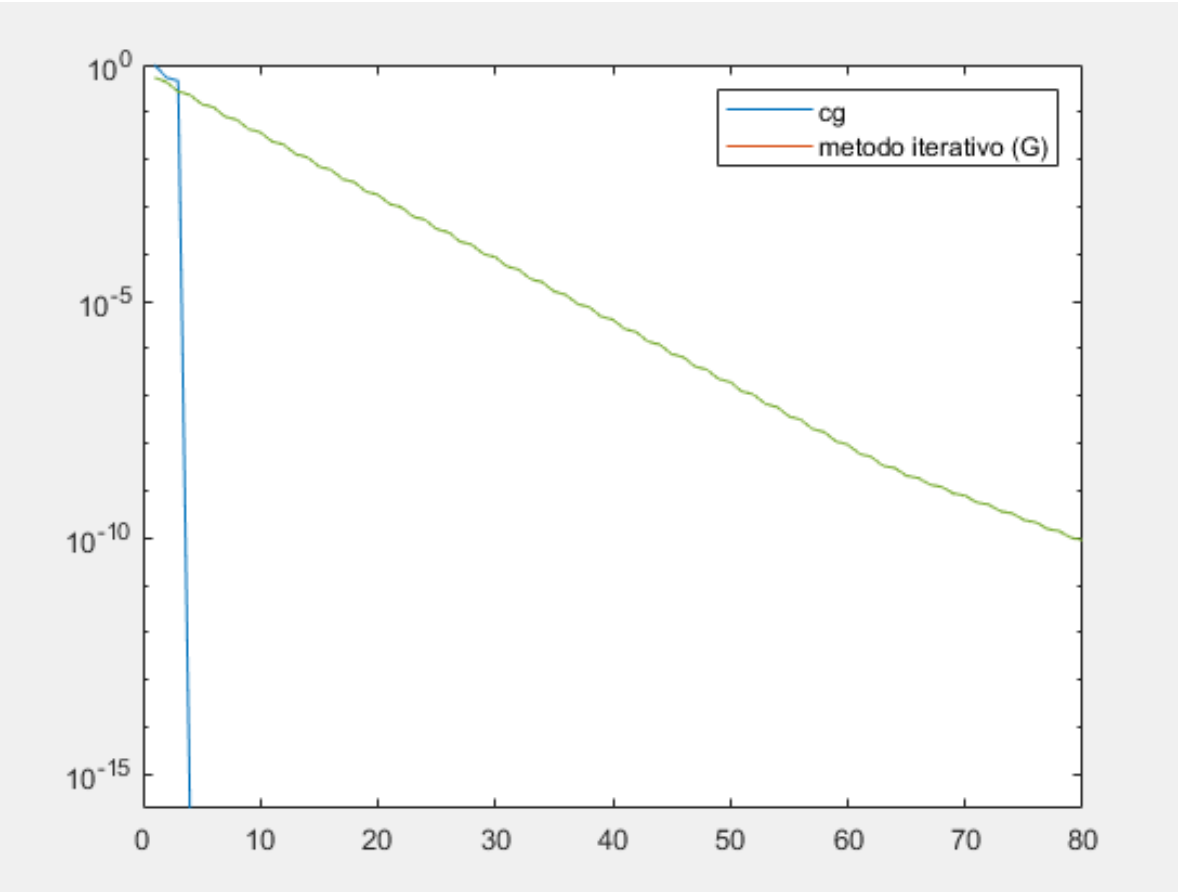


Esercizio: risolvere il sistema $Ax=b$ con A associata alla discretizzazione dell'operatore differenziale di Laplace sul dominio $[-1,1] \times [-1,1]$ di dimensione $m=16$ e ripetere per $m=400$

Grafico del residuo

```
Scegliamo come tol=10^-4;  
  
>> semilogy(1:size(residuo1,2), residuo1)  
>> hold on  
>> semilogy(1:size(residuo2,1), residuo2)
```

metodo	iterazione	flag	residuo
Metodo iterativo (G)	40	0	4.1318e-06
Metodo CG	3	0	9.4206e-16



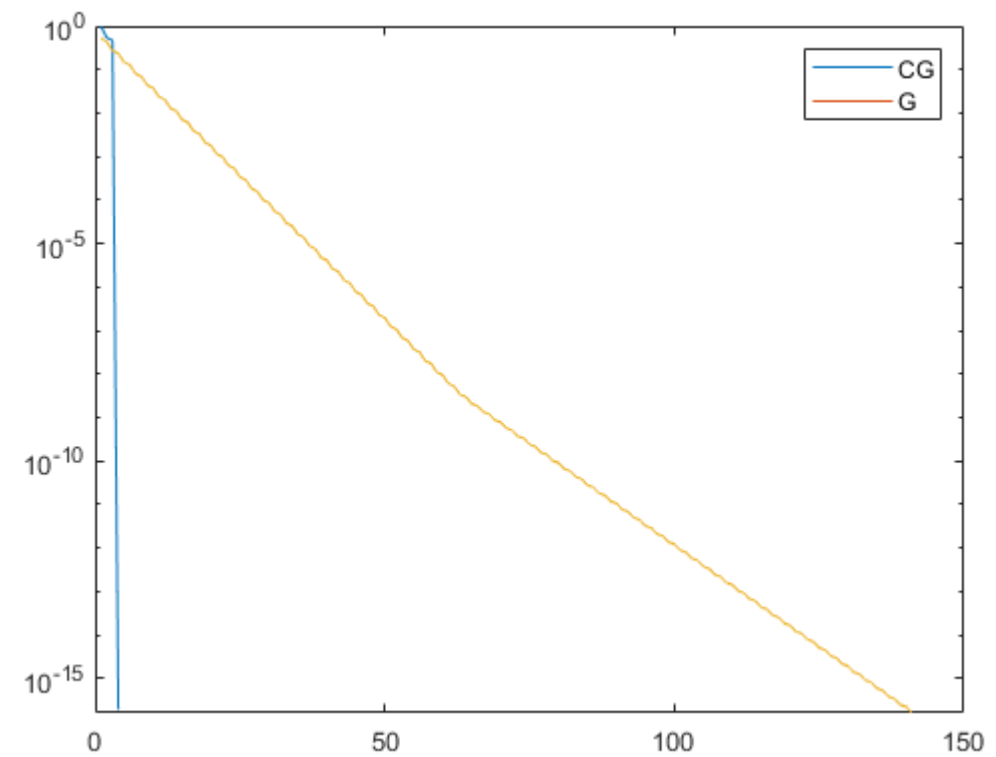
Esercizio: risolvere il sistema $Ax=b$ con A associata alla discretizzazione dell'operatore differenziale di Laplace sul dominio $[-1,1] \times [-1,1]$ di dimensione $m=16$ e ripetere per $m=400$

Grafico del residuo

Scegliamo come $\text{tol}=\text{eps}$;

```
>> semilogy(1:size(residuo1,2), residuo1)
>> hold on
>> semilogy(1:size(residuo2,1), residuo2)
```

metodo	iterazione	flag	residuo
Metodo iterativo (G)	141	0	1.6147e-16
Metodo CG	4	0	1.9230e-16

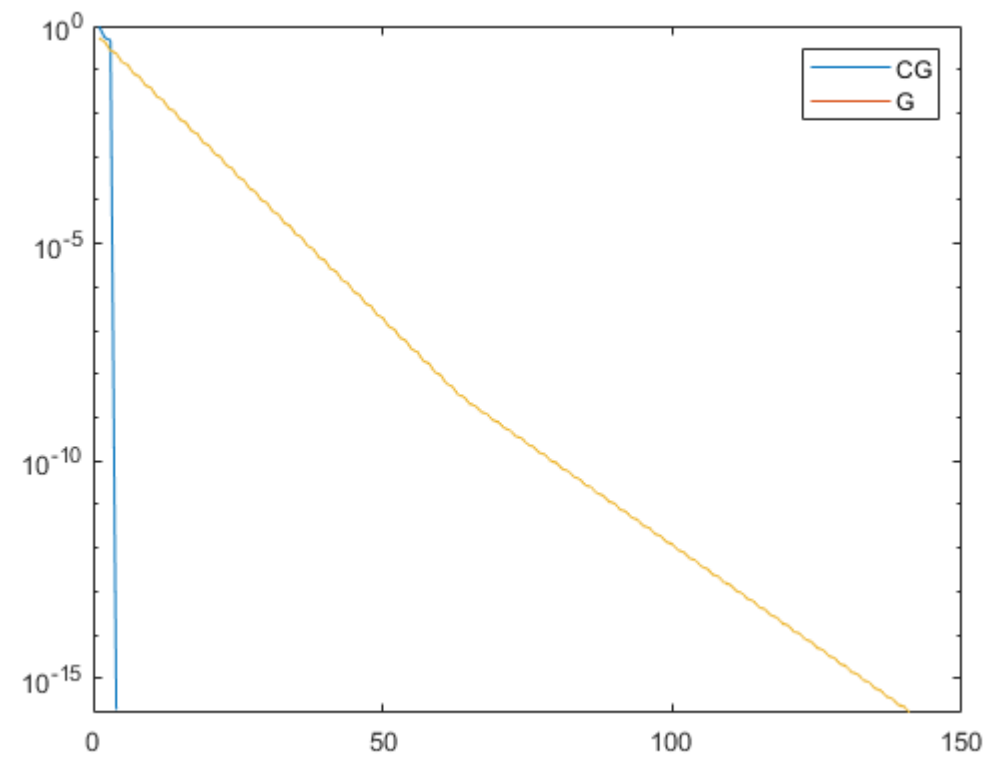


Esercizio: risolvere il sistema $Ax=b$ con A associata alla discretizzazione dell'operatore differenziale di Laplace sul dominio $[-1,1] \times [-1,1]$ di dimensione $m=16$ e ripetere per $m=400$

Grafico del residuo

```
Scegliamo come tol=eps;  
  
>> semilogy(1:size(residuo1,2), residuo1)  
>> hold on  
>> semilogy(1:size(residuo2,1), residuo2)
```

metodo	iterazione	flag	residuo
Metodo iterativo (G)	141	0	1.6147e-16
Metodo CG	4	0	1.9230e-16



Convergenza numerica del metodo CG

Esercizio: risolvere il sistema $Ax=b$ con A associata alla discretizzazione dell'operatore differenziale di Laplace sul dominio $[-1,1] \times [-1,1]$ di dimensione $m=16$ e ripetere per $m=400$

Scegliamo come $tol=eps$;

```
>> semilogy(1:size(residuo1,2), residuo1)
>> hold on
>> semilogy(1:size(residuo2,1), residuo2)
```

```
>> num_elem_dist
```

```
num_elem_dist =
```

15

metodo	iterazione	flag	residuo
Metodo iterativo (G)	141	0	1.6147e-16
Metodo CG	4	0	1.9230e-16

Esercizio: risolvere il sistema $Ax=b$ con A associata alla discretizzazione dell'operatore differenziale di Laplace sul dominio $[-1,1] \times [-1,1]$ di dimensione $m=16$ e ripetere per $m=400$

Scegliamo come $tol=eps$;

```
>> semilogy(1:size(residuo1,2), residuo1)
>> hold on
>> semilogy(1:size(residuo2,1), residuo2)
```

metodo	iterazione	flag	residuo
Metodo iterativo (G)	141	0	1.6147e-16
Metodo CG	4	0	1.9230e-16

```
>> auto=eig(A);
>> num_elem_dist
```

num_elem_dist =

15

Convergenza numerica in un numero di iterazioni
Minore del numero degli autovalori distinti

Esercizio: risolvere il sistema $Ax=b$ con A associata alla discretizzazione dell'operatore differenziale di Laplace sul dominio $[-1,1] \times [-1,1]$ di dimensione $m=16$ e ripetere per $m=400$

Grafico del residuo

```
>> semilogy(1:size(residuo1,2), residuo1)
>> hold on
>> semilogy(1:size(residuo2,1), residuo2)
```

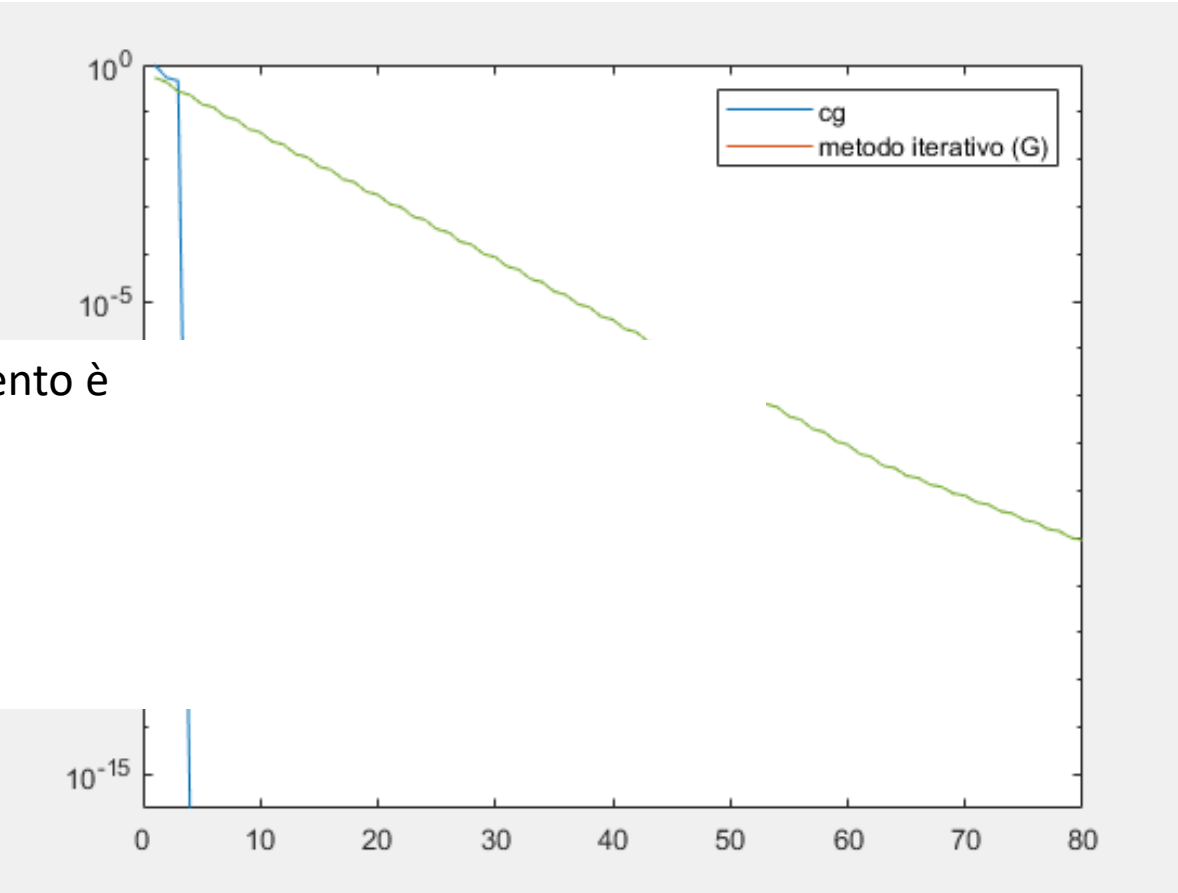
metodo	iterazione	flag	residuo
Metodo iterativo (G)	40	0	
Metodo CG	3	0	

Per $m=16$ il condizionamento è
>> condizionamento_1

condizionamento_1 =

13.3333

9.4206e-16



Esercizio: risolvere il sistema $Ax=b$ con A associata alla discretizzazione dell'operatore differenziale di Laplace sul dominio $[-1,1] \times [-1,1]$ di dimensione $m=16$ e ripetere per $m=400$

```
%matrice con dimensionone 400x400
n=22;
% mxm dimensione della matrice A
m=400;
G=numgrid('S',n);
%questa matrice di dimensione 16 è
associata alla discretizzazione
% dell'operatore differenziale di Laplace
sul dominio [-1,1]x[-1,1]
A=delsq(G);
%massimo numero di iterazioni
maxit=200;
%tolleranza
tol=10^(-10);
%vettore colonna dei termini noti
b=A*ones(400,1);
```

Per $m=400$ il condizionamento è

>> condizionamento_2

condizionamento_2 =

258.4520

Esercizio: risolvere il sistema $Ax=b$ con A associata alla discretizzazione dell'operatore differenziale di Laplace sul dominio $[-1,1] \times [-1,1]$ di dimensione $m=16$ e ripetere per $m=400$

Costruzione matrice di preconditionamento

```
droptol=0;  
%matrice triangolare inferiore generata dalla fattorizzazione  
incompleta di  
%Cholesky di A  
lichol=ichol(A,struct('type','ict','droptol',droptol));  
P=lichol'*lichol;
```

In matlab per PCG basta dare in input la matrice P richiamando la routine pcg:

```
[x,flag,RELRES,iter6,residuo6] = pcg(A,b,tol,maxit,P);
```

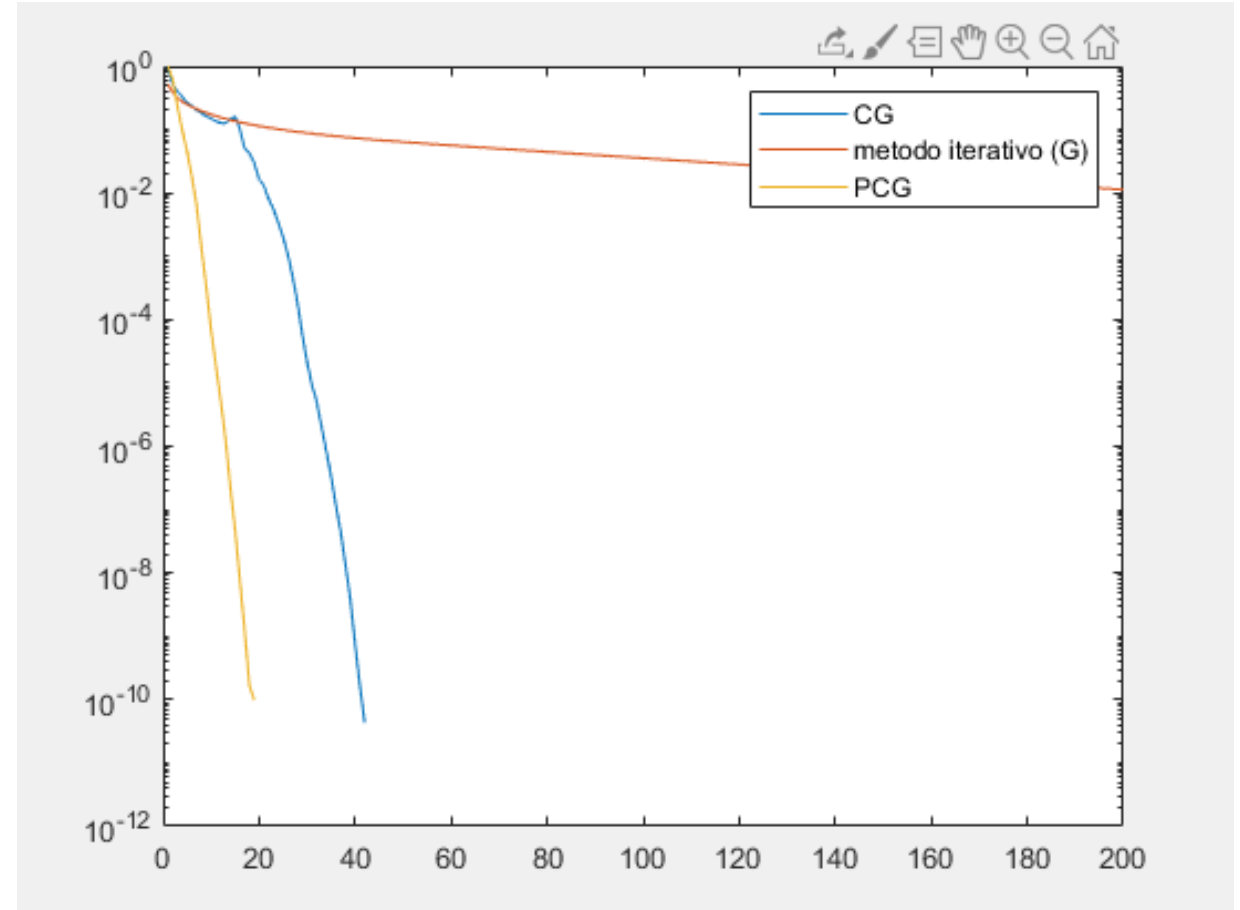
Matrice di preconditionamento



Esercizio: risolvere il sistema $Ax=b$ con A associata alla discretizzazione dell'operatore differenziale di Laplace sul dominio $[-1,1] \times [-1,1]$ di dimensione $m=16$ e ripetere per $m=400$

Grafico del residuo

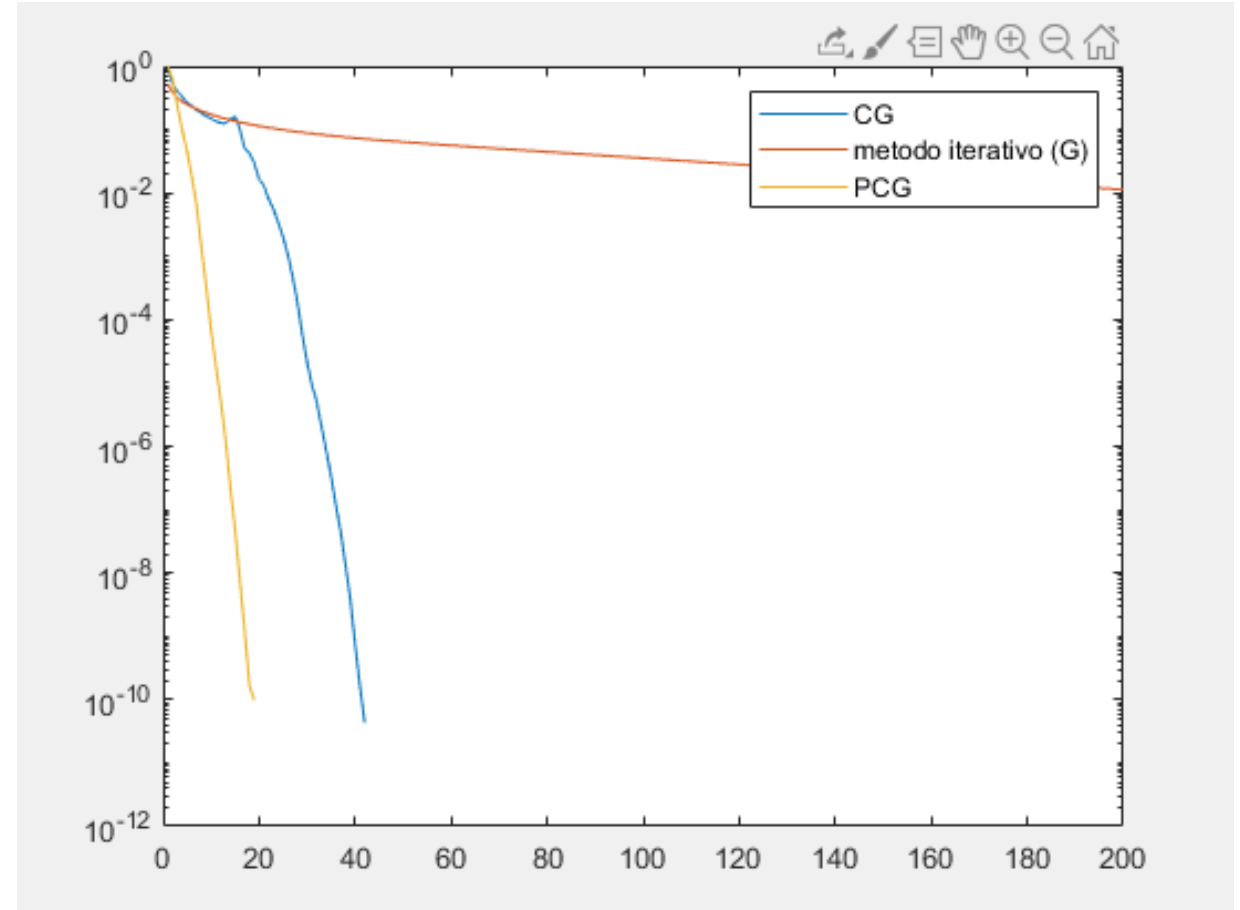
```
>> semilogy(1:size(residuo5,1), residuo5)
>> hold on
>> semilogy(1:size(residuo4,2), residuo4)
>> semilogy(1:size(residuo6,1), residuo6)
>> legend('CG','metodo iterativo (G)','PCG')
```



Esercizio: risolvere il sistema $Ax=b$ con A associata alla discretizzazione dell'operatore differenziale di Laplace sul dominio $[-1,1] \times [-1,1]$ di dimensione $m=16$ e ripetere per $m=400$

Grafico del residuo

```
>> semilogy(1:size(residuo5,1), residuo5)
>> hold on
>> semilogy(1:size(residuo4,2), residuo4)
>> semilogy(1:size(residuo6,1), residuo6)
>> legend('CG','metodo iterativo (G)','PCG')
```



Se invece preconditioniamo il sistema con la matrice P
Il condizionamento del nuovo sistema è circa 38