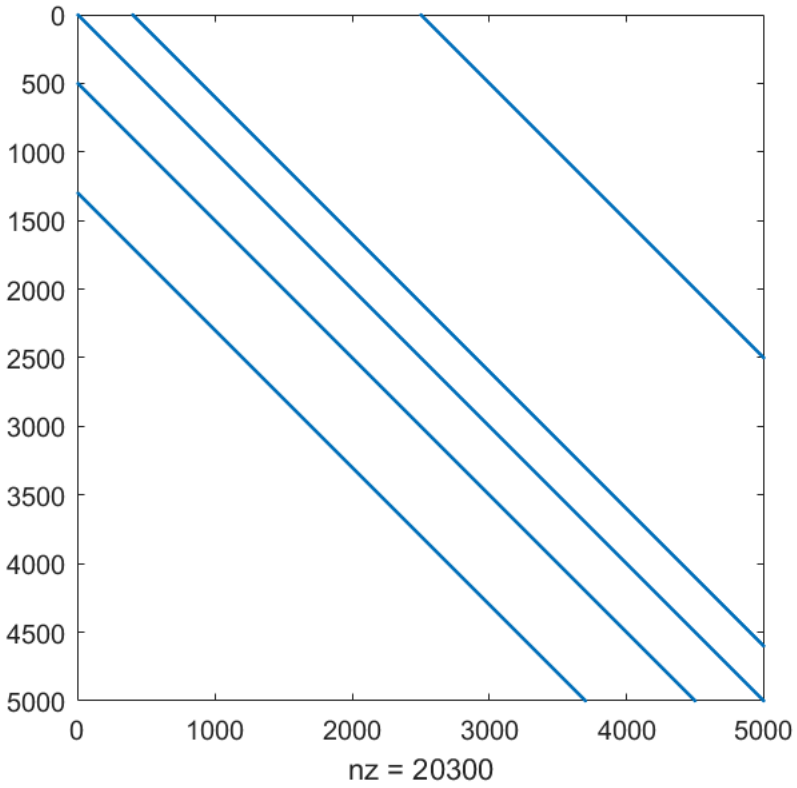
Testing del metodo di Jacobi

Esempi di utilizzo della function Jacobi

Si testa il software su matrici sparse di notevoli dimensioni, generate usando funzioni Matlab non singolari e ben condizionate. Si illustra, attraverso grafico, la struttura delle matrici di test. *Nel paragrafo Test di accuratezza sono effettuati test sulla precisione della function Jacobi*.

Un primo esempio di matrice ben condizionata:

```
n = 5000;
e = ones(n,1);
Z = spdiags([e 3*e 15*e -e e], [-1300 -500 0 400 2500], n, n);
spy(Z)
```




```
b = Z*e;
c = condest(Z)
```

```
c = 2.2693
```

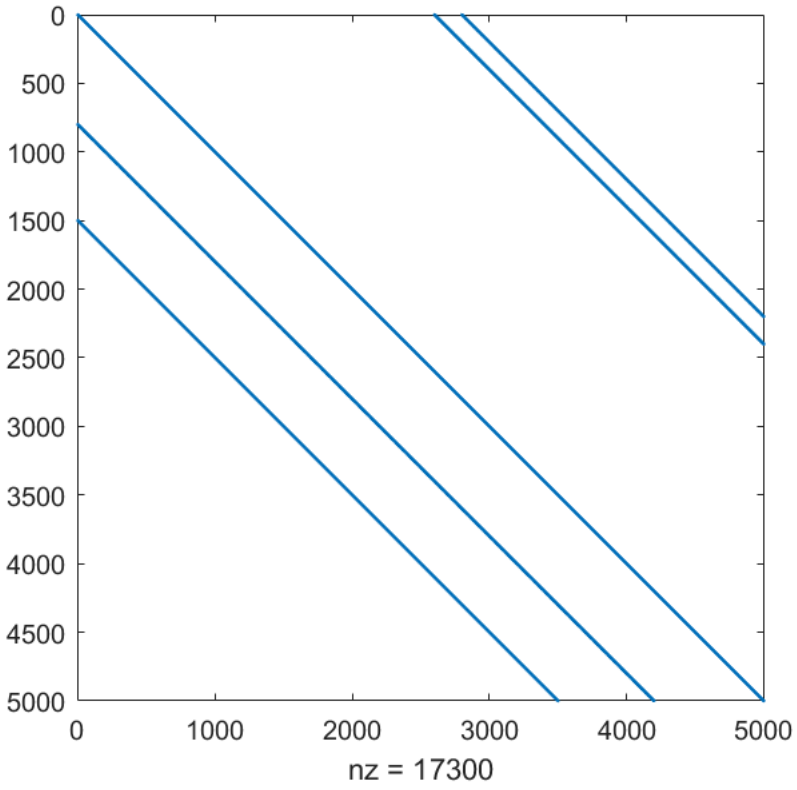
```
[sol, niter] = Jacobi(Z, b)
```

```
sol = 5000×1
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
```

```
    1.0000
    1.0000
    1.0000
     ⋮
niter = 11
```

Esempio d'uso nel caso in cui l'utente specifichi il valore di tolleranza:

```
n = 5000;
e = ones(n,1);
Z = spdiags([e 6*e 10*e -e e], [-1500 -800 0 2600 2800], n, n);
spy(Z)
```




```
b = Z*e;
c = condest(Z)
```

```
c = 6.3495
```

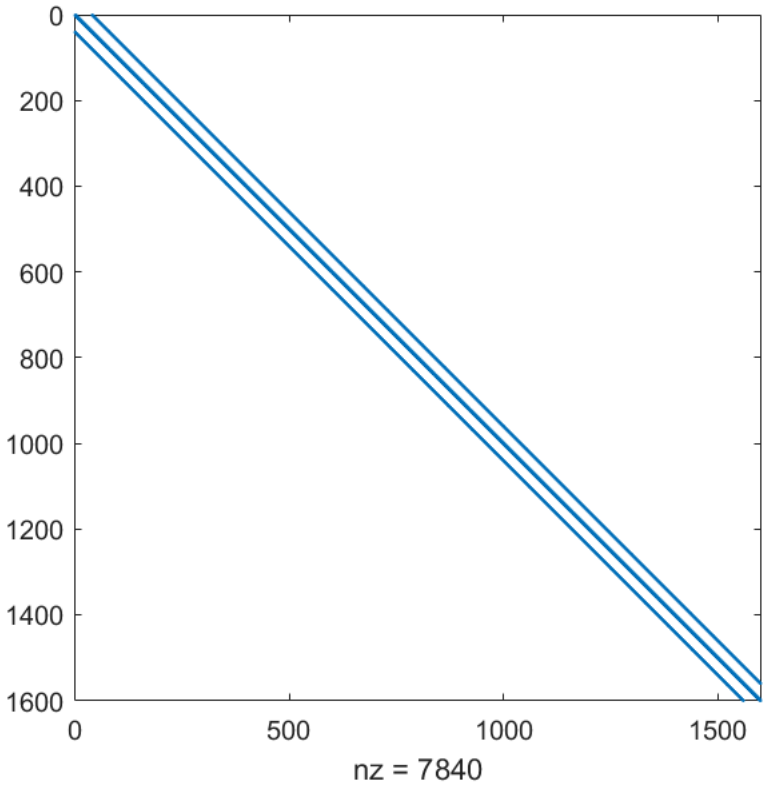
```
[sol, niter] = Jacobi(Z, b, 10^-9)
```

```
sol = 5000×1
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
```

```
    1.0000
      ⋮
niter = 42
```

Esempio d'uso nel caso in cui l'utente specifichi il valore di tolleranza e il numero massimo di iterazioni:

```
X = gallery('poisson',40);
spy(X)
```




```
c = condest(X)
```

```
c = 989.2690
```

```
x = ones(1600,1);
b = X*x;
[sol, niter] = Jacobi(X,b,10^-6,3000)
```

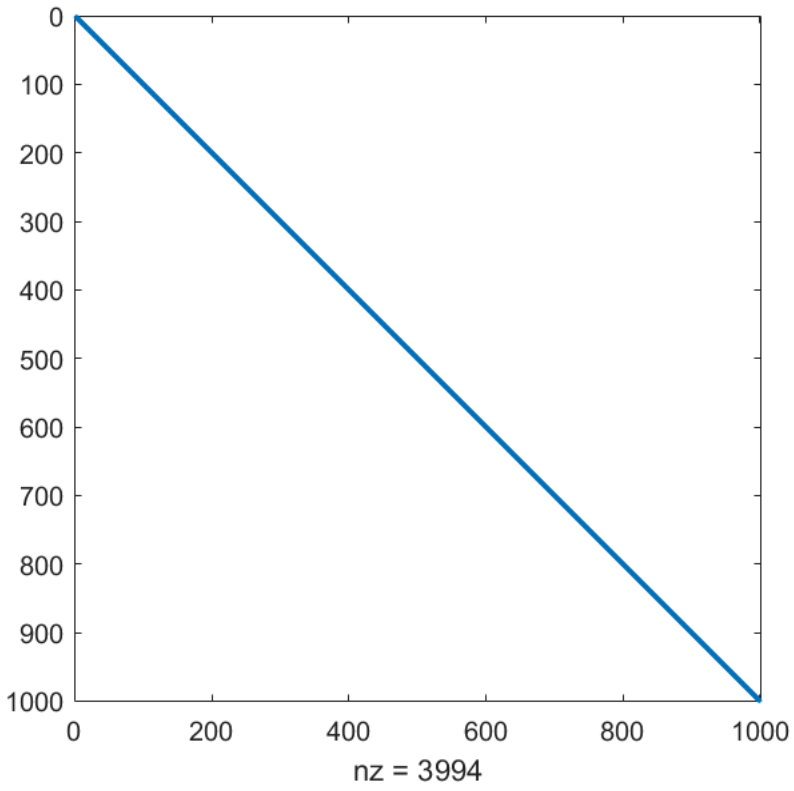
```
sol = 1600×1
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
      ⋮
```

```
niter = 2881
```

Casi di errore

Utilizziamo la matrice sparsa di Toeppen che presenta elementi nulli sulla diagonale. Per questo tipo di matrici, il metodo di Jacobi non è definito.

```
T = gallery('toeppen',1000);
spy(T)
```




```
b = T*ones(1000,1);
[sol, niter] = Jacobi(T,b)
```

```
Error using Jacobi (line 23)
La matrice A ha elementi nulli sulla diagonale. Riordinare la matrice e rieseguire l'algoritmo.
```

Caso in cui la matrice in ingresso è vuota.

```
A = [];
A = sparse(A);
[sol, niter] = Jacobi(A,b);
```

```
Error using Jacobi (line 7)
Matrice A o vettore b vuoti.
```

Caso in cui il vettore b in ingresso è vuoto.

```
A = spdiags([e 3*e 15*e -e e], [-1300 -500 0 400 2500], 50, 50);
b = [];
[sol, niter] = Jacobi(A,b);
```

```
Error using Jacobi (line 7)
Matrice A o vettore b vuoti.
```

Caso in cui la matrice A non è di tipo sparse.

```
A = [0 0 3; 1 0 0; 0 0 8];
[sol, niter] = Jacobi(A,b);
```

```
Error using Jacobi (line 12)
La matrice A non è tipo sparse.
```

Caso in cui A non è una matrice quadrata.

```
A = sparse(10, 12);
[sol, niter] = Jacobi(A,b);
```

```
Error using Jacobi (line 19)
Matrice A non quadrata.
```

Caso in cui b non è un vettore ma un carattere.

```
A = spdiags([e 3*e 15*e -e e], [-1300 -500 0 400 2500], 50, 50);
b = 'c';
[sol, niter] = Jacobi(A,b);
```

```
Error using Jacobi (line 28)
b non è un vettore.
```

Caso in cui b non è un vettore ma una table.

```
A = spdiags([e 3*e 15*e -e e], [-1300 -500 0 400 2500], 50, 50);
b = table(54,75);
[sol, niter] = Jacobi(A,b);
```

```
Error using Jacobi (line 28)
b non è un vettore.
```

Caso in cui b non è un vettore ma una struct.

```
A = spdiags([e 3*e 15*e -e e], [-1300 -500 0 400 2500], 50, 50);
b.Field = 7;
[sol, niter] = Jacobi(A,b);
```

```
Error using Jacobi (line 28)
b non è un vettore.
```

Caso in cui b non è un vettore numerico.

```
A = spdiags([e 3*e 15*e -e e], [-1300 -500 0 400 2500], 3, 3);
b = ['c','d','e'];
[sol, niter] = Jacobi(A,b);
```

```
Error using Jacobi (line 33)
Il vettore b non è vettore numerico.
```

Caso in cui b non rispetta le dimensioni di A.

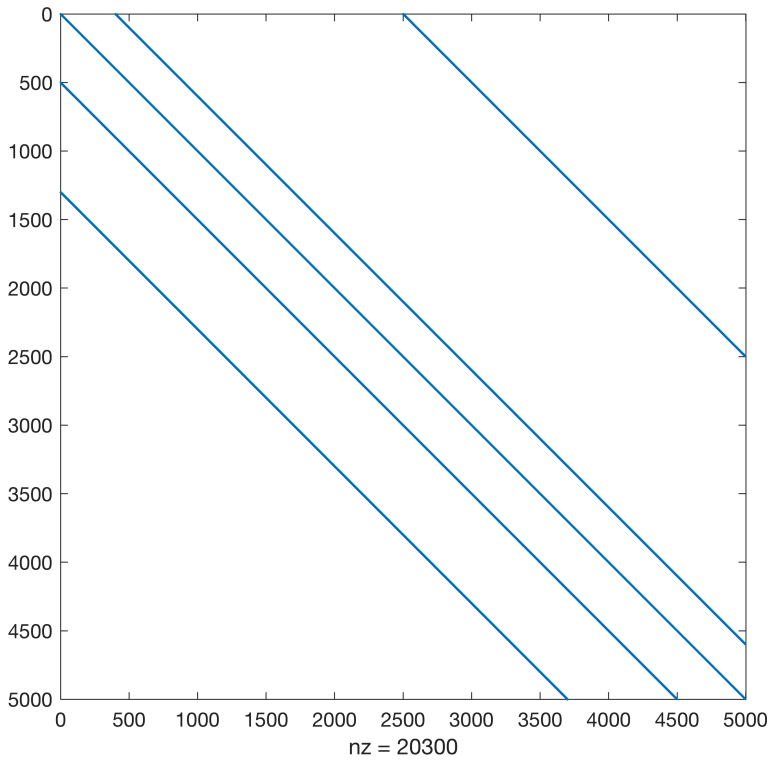
```
A = spdiags([e 3*e 15*e -e e], [-1300 -500 0 400 2500], 50, 50);
b = rand(40,1);
[sol, niter] = Jacobi(A,b);
```

```
Error using Jacobi (line 40)
Il vettore dei termini noti non rispetta le dimensioni di A.
```

Test di accuratezza

Nella seguente sezione si confrontano i risultati ottenuti mediante la funzione di Jacobi con le soluzioni imposte

```
n = 5000;
e = ones(n,1);
Z = spdiags([e 3*e 15*e -e e], [-1300 -500 0 400 2500], n, n);
spy(Z);
```




```
b = Z*e;
```

```
c = condest(Z)
```

```
c =
     2.269252419481775e+00
```

```
[sol, niter, res] = Jacobi(Z, b)
```

```
sol = 5000×1
     1.000000000444483e+00
     1.000000000444483e+00
     1.000000000444483e+00
     1.000000000444483e+00
     1.000000000444483e+00
     1.000000000444483e+00
     1.000000000444483e+00
     1.000000000444483e+00
     1.000000000444483e+00
     1.000000000444483e+00
      ⋮
niter =
    11
res =
     6.623950529603319e-08
```

```
err = norm(e-sol)/norm(sol)
```

```
err =
     2.969125980155711e-08
```

Aumentiamo il numero di cifre significative richieste:

```
[sol, niter, res] = Jacobi(Z, b, 10^-13)
```

```
sol = 5000×1
     9.999999999999959e-01
     9.999999999999959e-01
     9.999999999999959e-01
     9.999999999999959e-01
     9.999999999999959e-01
     9.999999999999959e-01
     9.999999999999959e-01
     9.999999999999959e-01
     9.999999999999959e-01
     9.999999999999959e-01
      ⋮
niter =
    24
res =
     1.234100541057016e-14
```

```
err = norm(e-sol)/norm(sol)
```

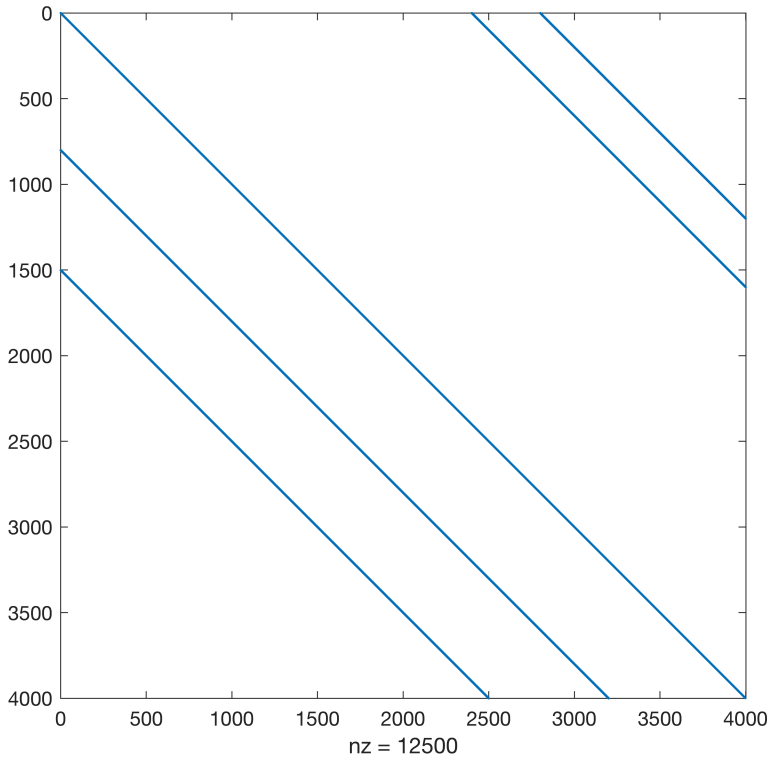
```
err =
     9.608773431926683e-15
```

Velocità di convergenza

Nei paragrafi precedenti è stato già notato come, all'aumentare del condizionamento e del raggio spettrale, la function converga in un numero di iterazioni maggiore. Mostriamo qui alcuni esempi.

Esempio 1

```
e = ones(4000,1);
Z = spdiags([e 6*e 10*e -e e], [-1500 -800 0 2400 2800], 4000, 4000);
spy(Z)
```




```
b = Z*e;
c = condest(Z)
```

```
c =
     4.865382287085668e+00
```

```
[sol, niter] = Jacobi(Z, b, 10^-7)
```

```
sol = 4000×1
     1.000000004034755e+00
     1.000000004034755e+00
     1.000000004034755e+00
     1.000000004034755e+00
     1.000000004034755e+00
     1.000000004034755e+00
     1.000000004034755e+00
     1.000000004034755e+00
     1.000000004034755e+00
     1.000000004034755e+00
     ⋮
niter =
```

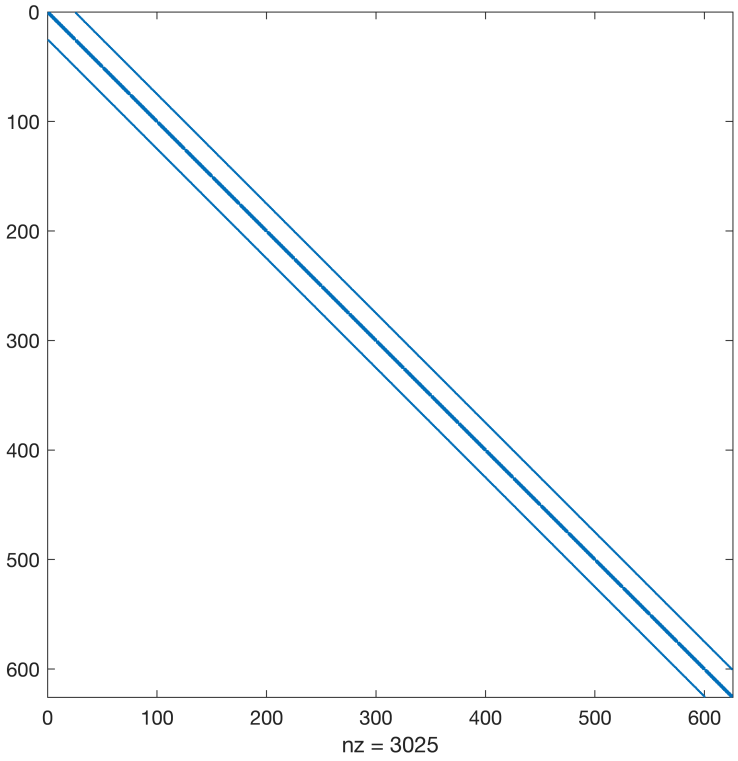
```
27
```

```
B = (-1./diag(Z)')'.*(Z-diag(diag(Z)));
raggio_spettrale = max(abs(eigs(B))) % Raggio spettrale
```

```
raggio_spettrale =
     5.644619182878499e-01
```

Esempio 2

```
A = gallery('poisson',25);
spy(A);
```




```
c = condest(A)
```

```
c =
     3.979511906492402e+02
```

```
x = ones(625,1);
b = A*x;
[sol, niter] = Jacobi(A, b, 10^-7,1800)
```

```
sol = 625×1
     9.999998039306583e-01
     9.999996107308720e-01
     9.999994231868126e-01
     9.999992440846508e-01
     9.999990759652524e-01
     9.999989213694822e-01
     9.999987824452332e-01
     9.999986613403985e-01
     9.999985596850713e-01
```

```
     9.999984791093450e-01
     ⋮
niter =
        1597
```

```
B = (-1./diag(A)')'.*(A-diag(diag(A)));
raggio_spettrale = max(abs(eigs(B))) % Raggio spettrale
```

```
raggio_spettrale =
     9.927088740980539e-01
```

La prima esecuzione converge in 27 iterazioni con precisione a 7 cifre. La seconda esecuzione, a parità di cifre significative richieste, converge in 1597 iterazioni. Ciò trova una spiegazione teorica nell'aumento del raggio spettrale della matrice B calcolata dal metodo di Jacobi.

Casi di warning

Caso in cui il parametro TOL è un carattere.

```
A = spdiags([e 3*e 15*e -e e], [-1300 -500 0 400 2500], 50, 50);
b = rand(50,1);
[sol, niter] = Jacobi(A,b,'c');
```

```
Warning: TOL scorretto. Impostato a 10^(-6)
```

Caso in cui il parametro TOL è minore di eps.

```
A = spdiags([e 3*e 15*e -e e], [-1300 -500 0 400 2500], 50, 50);
b = rand(50,1);
[sol, niter] = Jacobi(A,b,eps/2);
```

```
Warning: TOL scorretto. Impostato a 10^(-6)
```

Caso in cui il parametro TOL non è finito.

```
A = spdiags([e 3*e 15*e -e e], [-1300 -500 0 400 2500], 50, 50);
b = rand(50,1);
[sol, niter] = Jacobi(A,b,inf);
```

```
Warning: TOL scorretto. Impostato a 10^(-6)
```

Caso in cui è specificato anche il quarto parametro ma TOL non è reale.

```
A = spdiags([e 3*e 15*e -e e], [-1300 -500 0 400 2500], 50, 50);
b = rand(50,1);
[sol, niter] = Jacobi(A,b,5i,600);
```

```
Warning: TOL scorretto. Impostato a 10^(-6)
```

Caso in cui il parametro MAXITER non è finito.

```
A = spdiags([e 3*e 15*e -e e], [-1300 -500 0 400 2500], 50, 50);
b = rand(50,1);
[sol, niter] = Jacobi(A,b,10^-9,inf);
```

```
Warning: Numero massimo di iterazioni scorretto. Impostato a 500.
```

Caso in cui il parametro MAXITER non è uno scalare.

```
A = spdiags([e 3*e 15*e -e e], [-1300 -500 0 400 2500], 50, 50);
b = rand(50,1);
[sol, niter] = Jacobi(A,b,10^-9,[1 2 3]);
```

```
Warning: Numero massimo di iterazioni scorretto. Impostato a 500.
```

Caso in cui il parametro MAXITER non è reale.

```
A = spdiags([e 3*e 15*e -e e], [-1300 -500 0 400 2500], 50, 50);
b = rand(50,1);
[sol, niter] = Jacobi(A,b,10^-9,3i);
```

```
Warning: Numero massimo di iterazioni scorretto. Impostato a 500.
```

Potrebbe capitare che il numero di iterazioni massimo non sia sufficiente per ottenere il numero di cifre significative desiderato. Mostriamo di seguito alcuni esempi:

```
X = gallery('poisson',45);
c = condest(X)
x = ones(2025,1);
b = X*x;
[sol, niter] = Jacobi(X,b,10^-6,1000)
[sol, niter] = Jacobi(X,b,10^-6,4000)
err = norm(x-sol)/norm(sol)
```

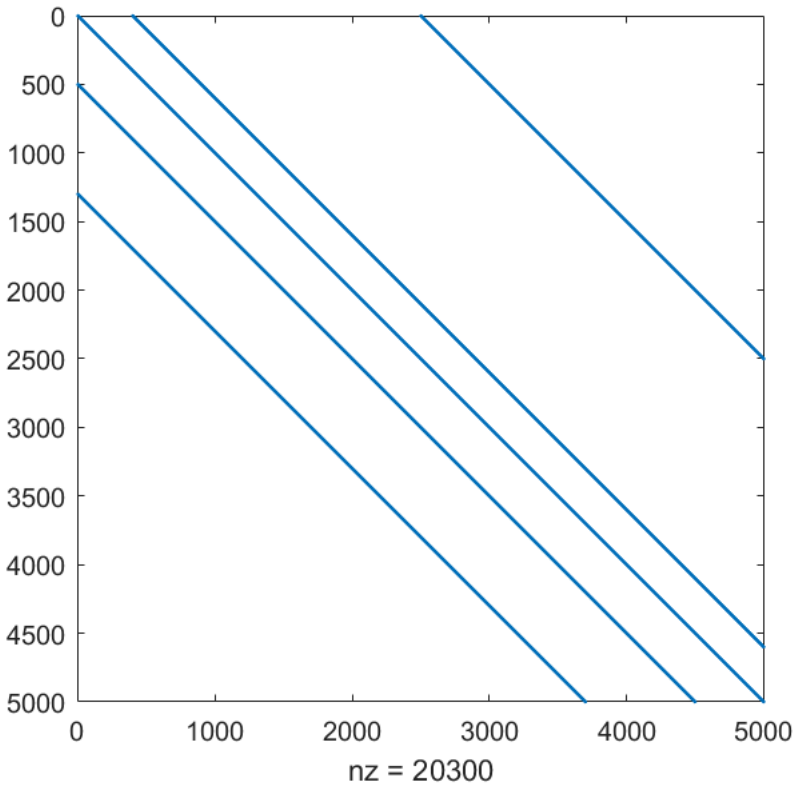
In questo caso, dato che il primo numero massimo di iterazioni è stato impostato a 1000, la function stampa un messaggio di warning specificando che il risultato potrebbe non soddisfare l'accuratezza richiesta. Se si aumenta tale parametro, la function converge al risultato con accuratezza corretta.

Il prossimo esempio, invece, mostra una matrice malcondizionata e la matrice B calcolata dal metodo ha raggio spettrale uguale a 1. In questo caso, il metodo iterativo di Jacobi non converge.

```
D = gallery('dorr',50);
spy(D)
c = condest(D)
e = ones(50,1);
b = D*e;
[sol, niter] = Jacobi(D,b,10^-6,30000)
err = norm(e-sol)/norm(sol)
```

Caso in cui diagonale non è strettamente dominante e buon condizionamento (vale per tutti i tol) NON CONVERGE anche con Tol = 10^-15 e aumentando le iterazioni.

```
W = spdiags([e 5*e 2*e -e e], [-1300 -500 0 400 2500], n, n);
spy(W);
```




```
b = W*e;
[sol4, niter4] = Jacobi(W,b,10^(-15));
```

```
Warning: Attenzione! È stato raggiunto il numero massimo di iterazioni. Il risultato potrebbe non essere
 accurato.
```