SPERIMENTAZIONE NUMERICA

Una precondizionatrice per matrici di Toeplitz rettangolari per il PCG

Mario Correddu

INTRODUZIONE

Data T matrice matrice Toeplitz $\in \mathbb{C}^{m\times n}$ (con $m = kn$ a meno di estendere T in modo Toeplitz) definiamo la matrice di precondizionamento C come:

$$C = F(\sum_{j=1}^{k} \Lambda_j^* \Lambda_j)^{\frac{1}{2}} F^*$$

$$\Lambda_j = F^* C_j F$$

dove F è la trasformata discreta di Fourier e per ogni $j = 1, 2...k$ la matrice C_j è una matrice circolante definita come

$$c_k = \left\{ \begin{array}{cc} \frac{(n-k)t_k + k a_{k-n}}{n} & se \quad 0 \leq k < n \\ c_{n+k} & se \quad 0 < -k < n \end{array} \right\}$$

per t_k l'elemento k-esimo della prima riga se $k \geq 0$ o l'elemento k-esimo della prima colonna se $k < 0$, del j-esimo blocco $n \times n$ della matrice T.

In questo lavoro ci poniamo l'obiettivo di mostrare come si comporta il metodo del gradiente coniugato per la risoluzione ai minimi quadrati del sistema lineare $Tx = b$, se precondizionato con C.

Algoritmo del gradiente coniugato precondizionato per problemi ai minimi quadrati

Dato x_0 una prima approssimazione di x:
$r_0 = b - Tx_0$
$p_0 = s_0 = (C^{-1})^* T^* r_0$
$\gamma_0 = ||s_0||_2^2$

for $k = 0, 1, 2, \ldots$
$q_k = TC^{-1} p_k$
$\alpha_k = \frac{\gamma_k}{||q_k||_2^2}$
$x_{k+l} = x_k + \alpha_k C^{-1} p_k$
$r_{k+l} = r_k - \alpha_k q_k$
$s_{k+1} = (C^{-1})^* T^* r_{k+1}$
$\nu_{k+1} = ||s_{k+1}||_2^2$
$\beta_k = \frac{\nu_{k+1}}{\nu_k}$
$p_{k+l} = s_{k+l} + \beta_k p_k$
return x

Le versioni quadrate delle matrici utilizzate nei test sono state prese da [2] e in tutti è stato posto b come il vettore composto da soli 1 mentre il passo iniziale dell'algoritmo pone x_0 uguale al vettore nullo. Il criterio d'arresto utilizzato è $||s_k||_2 / ||s_0||_2 < 10^{-7}$ e i test sono stati svolti su MATLAB 9.5.

Codici utilizzati

Prodotto matrice circolante vettore

```
function y=circprod(col_C, x)
%calcola il prodotto matrice circolante vettore Cx usando la DFT
% col_C: la prima colonna della matrice C
z=fft(x);
w=fft(col_C);
y=ifft(z.*w);
end
```

Prodotto matrice Toeplitz vettore

```
function y = toepproduct(row_T,col_T,x)

%calcola il prodotto matrice toeplitz vettore Tx
%row_T: prima riga di T
%col_T prima colonna di T
m=length(col_T);
n=length(row_T);

%immerge la matrice T in una matrice circolante C (m+n-1)x(m+n-1),
%di cui T costituisce il primo blocco mxn
col_C=[col_T;row_T(n:-1:2)];

%calcola il prodotto matrice circolante per il vettore modificato
% per rispettare le dimensioni
ylong=circprod(col_C, [x; zeros(m-1,1)]);

y=ylong(1:m);

end
```

Precondizionatrice per matrice quadrata

```
function col_C = sqcircprec(row_T,col_T)
%data una matrice Toeplitz quadrata restituisce la prima colonna dell'
    approssimante circolante
%row_T:prima riga di T
%col_T: prima colonna di T
%col_C: prima colonna di C

n=length(col_T);
%prima riga di C
row_C=zeros(n,1);
row_C(1)= n*row_T(1);
for j=1:n-1
  row_C(j+1)= (n-j)*row_T(j+1)+(j)*col_T(n-j+1);
end
row_C=row_C./n;
col_C=[row_C(1);row_C(n:-1:2)];

end
```

Precondizionatrice per matrice rettangolare

```
function col_C = retcircprec(row_T,col_T)
%data una matrice rettangolare di tipo Toeplitz mxn per m=k*n restituisce la
%matrice precondizionante associata generata dalle singole matrici
%precondizionatrici dei blocchi nxn

m=length(col_T);
n=length(row_T);
k=m/n;

%costruisce per il primo blocco la rispettiva matrice di precondizionamento
col_Ctemp= sqcirc_prec(row_T(1:n),col_T(1:n));
%costruisce il vettore degli autovalori
dtemp=fft(row_Ctemp);
d=(dtemp.*conj(dtemp));
```

```
for j=1:k-1
    %costruisce per ogni blocco la rispettiva matrice di precondizionamento
    col_Ctemp= sqcircprec(col_T(j*n+1:-1:(j-1)*n+2),col_T(j*n+1:(j+1)*n));
    %costruisce il vettore degli autovalori
    dtemp=fft(col_Ctemp);
    d=d+(dtemp.*conj(dtemp));
end

d=sqrt(d);
col_C=ifft(d);
end
```

Gradiente coniugato precondizionato da matrice circolante

```
function x = circprec_cg(row_T, col_T,b,col_C, epsilon)
% per T matrice toeplitz mxn risolve il sistema Tx=b ai minimi
%quadrati tramite il metodo del gradiente coniugato precondizionato
%da una matrice circolante C, con tolleranza epsilon
%row_T: prima riga di T
%col_T: prima colonna di T
%col_C: prima colonna di C
itermax=1000;
m=length(col_T);
n=length(row_T);
%calcolo dell'inversa di C tramite la fft
col_Cinv=  ifft(1./fft(col_C)) ;
%vettore che contiene i residui a ogni passo
iterv=zeros(1,itermax);

%algoritmo del gradiente coniugato precondizionato
%vettore di partenza del metodo
x = zeros(n,1);
r = b;
%calcola C'^(-1))*(T')*r
s= circprod( conj([col_Cinv(1);col_Cinv(n:-1:2)]), toepproduct(conj(col_T),conj
    (row_T),r) );
p=s;
nup=s'*s;
```

```
tol=epsilon *sqrt(nup);
iter=0;

while ( sqrt(nup)>=tol )&&( iter<itermax )
    pp=circprod(col_Cinv,p);
    q=toepproduct(row_T,col_T,pp);
    alpha = nup/(q'*q);
    x = x + alpha*pp;
    r = r-alpha * q;
    s=circprod( conj([col_Cinv(1);col_Cinv(n:-1:2)]), toepproduct( conj(col_T),
        conj(row_T),r ) );
    nu=s'*s;
    beta=nu/nup;
    nup=nu;
    p= s+beta*p;
    iter=iter+1;
    iterv(iter)=nup;
end

%visualizza come si comporta il metodo e stampa le iterazioni fatte
semilogy(iterv(1:iter), "blue"); iter
end
```

Gradiente coniugato per problemi ai minimi quadrati

```
function [iter,x] = minsqgc(row_T,col_T,b, epsilon)
%algoritmo del gradiente coniugato per risoluzione di sistemi ai minimi
    quadrati

itermax=1000;
m=length(col_T);
n=length(row_T);

%vettore che contiene i residui del metodo a ogni passo
iterv(iter)=nup;

%vettore di partenza del metodo
x = zeros(n,1);
```

```
r = b; s=toepproduct(conj(col_T), conj(row_T),r);
p=s;
nup=s'*s;
tol=epsilon *sqrt(nup);

iter=0;
while(sqrt(nup)>=tol)&&(iter<itermax)
    q=toepproduct( row_T, col_T,p);
    alpha = nup/(q'*q);
    x = x + alpha*p;
    r = r-alpha * q;
    s=toepproduct(conj(col_T), conj(row_T),r);
    nu=s'*s;
    beta=nu/nup;
    nup=nu;
    p= s+beta*p;
    iter=iter+1;
    iterv(iter)=nup;

end
%visualizza come si comporta il metodo e stampa le iterazioni fatte
semilogy(iterv(1:iter), "green"); iter
end
```

Gradiente coniugato precondizionato

```
function  [iter,x] = prec_gc(A,b,M, epsilon)
% risolve il sistema Ax=b, dove A e' una matrice nxn,  applicando l'algoritmo
%del gradiente coniugato precondizionato da M

itermax=1000;
dim=size(A);
n=dim(1);
%vettore che contiene i residui del metodo a ogni passo
iterv=zeros(itermax,1);

%vettore di partenza del metodo
x = zeros(n,1);
```

```
r = b; s= M\r;
p=s;
nup=r'*s;
tol= epsilon*norm(s);
iter=0;

while(norm(s)>=tol)&&(iter<itermax) %algoritmo del gradiente coniugato
    q=A*p;
    alpha = nup/(p'*q);
    x = x + alpha*p;
    r = r-alpha * q;
    s= M \ r;
    nu=r'*s;
    beta=nu/nup;
    nup=nu;
    p= s+beta*p;
    iter=iter+1;
    iterv(iter)=norm(s);
end
%visualizza come si comporta il metodo e stampa le iterazioni fatte
semilogy(iterv(1:iter), "red"); iter
end
```

Test numerici

Test 1

In questo primo test consideriamo la matrice definita come :

$$c(i) = \frac{1}{2^i} \text{ per i=1,...,m}$$
$$r(j) = \frac{1}{2^j} \text{ per j=1,...n}$$

dove r e c sono rispettivamente la prima riga e la prima colonna di T.

```
function test1(m,n)
%Test degli autovalori di una matrice Toeplitz T mxn con elementi
%a1j=aj1=1/(2^j): calcola la matrice di precondizionamento C, mostra la
%distribuzione dei valori singolari di T e TC^-1 e restituisce il numero di
%iterazioni necessarie al metodo del gradiente coniugato per raggiungere
%la stessa tolleranza (1e-7) nei due casi

%costruzione matrice T a partire dalla prima riga e colonna e calcolo dei
%valori singolari
c=1./(2.^(0:m-1));
r=c(1:n);
T=toeplitz(c,r);
s=svd(T);

row_T=r';
col_T=c';

%costruzione della matrice di precondizionamento e calcolo dei valori
%singolari della matrice precondizionata
col_C=retcircprec(row_T,col_T);
C= toeplitz(col_C, [col_C(1);col_C(n:-1:2)]);
scond=svd(T/C);

figure (1)
scatter( scond, ones(1,length(scond)),'b*')
hold on
scatter(s, zeros(1,length(s)),'r*')
xlabel('Valori␣singolari')
a=0.05*(max([s;scond])-min([s;scond]));
lb= min([s;scond])-a;
rb=max([s;scond])+a;
```

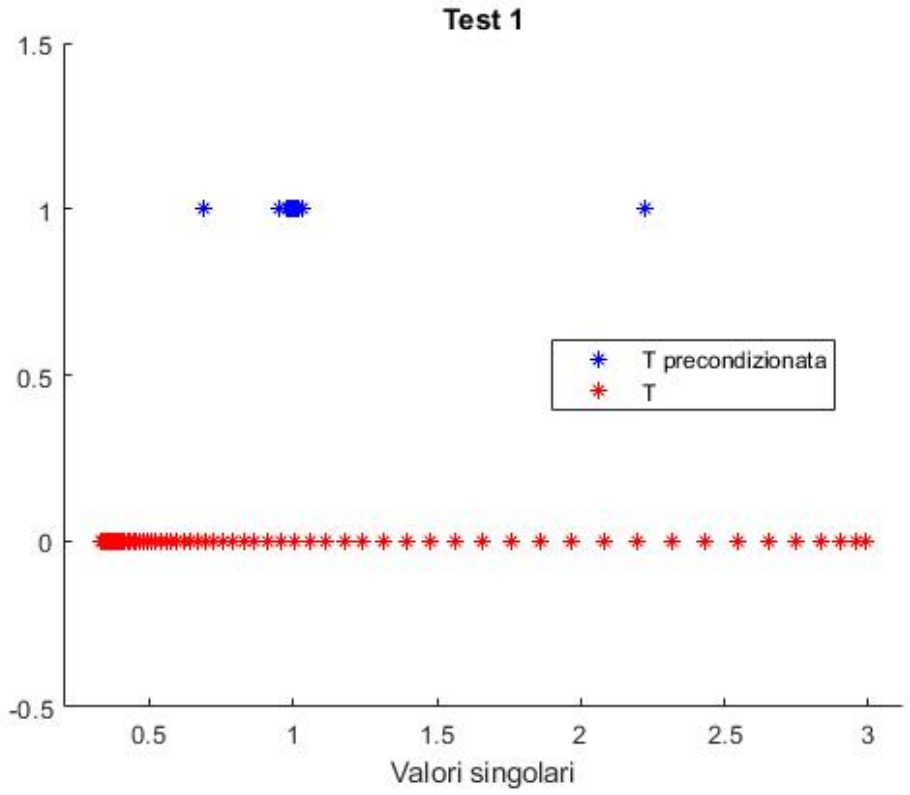
```
axis([lb rb -0.5 1.5])
legend('T precondizionata','T', 'Location', 'best')
title('Test 1')
hold off

%applicazione del gradiente coniugato nei due casi
figure (2)
[iter_C, x_C]=circprec_cg(row_T, col_T,ones(m,1),col_C, 1e-7);
hold on
[iter_C, x_no]=minsqgc(row_T,col_T,ones(m,1), 1e-7);

xlabel('iterazioni')
ylabel('residui')
legend('C-prec','non prec', 'Location', 'best')
hold off
end
```

Osserviamo il grafico dei valori singolari:

Figure 1: Valori singolari di T e TC^{-1} per m=210 e n=70



Vediamo che i valori singolari di T sono distribuiti nell'intervallo $[0,3]$,invece quelli della precondizinata sono concentrati per la maggior parte su 1. Questo per il teorema di Axellson [3] ci suggerisce che potremmo avere una buona velocità di convergenza e infatti:

Table 1: Numero di iterazioni necessarie per soddisfare il criterio di arresto nel caso precondizionato e non precondizionato, per $m = 3n$

n	C-prec	non-prec
40	7	31
50	7	36
60	7	39
70	7	41
80	7	42
100	7	45
120	7	48

Vediamo che confrontando i risultati la convergenza del metodo precondizionato sembra avere numero di passi addirittura costante contro il numero di passi crescenti rispetto a n del metodo senza precondizionamento, il che è corerente con quanto abbiamo visto per la distribuzione dei valori singolari.

Test 2

Per il secondo test confrontiamo le prestazioni della precondizionatrice C con la precondizionatrice M data dalla matrice tridiagonale costruita a partire dalle diagonali principali di $T'T$. Per il test consideriamo 3 matrici, T_1, T_2, T_3:

$$c(k) = k^{-1.1} + i(k^{-1.1}) \text{ per k=1,...,m}$$
$$r(j) = j^{-1.1} + i(j^{-1.1}) \text{ per } j = 1,...n$$

dove r e c sono rispettivamente la prima riga e la prima colonna di T_1, la quale é ben condizionata quando $m = n$,

$$r(1) = c(1) = 0$$
$$c(k) = k^{-1.1} + i(k^{-1.1}) \text{ per k=2,...,m}$$
$$r(j) = j^{-1.1} + i(j^{-1.1}) \text{ per j=2,...n}$$

dove r e c sono rispettivamente la prima riga e la prima colonna di T_2, la quale é mal condizionata per $m = n$, e infine T_3, una matrice sparsa costruita da T_2 ponendo a zero alcuni elementi della prima riga e colonna.

```
function test2_2(m,n)
%poste 3 matrici di Toeplitz mxn, una ben condizionata, una malcondizionata
%risolve il problema ai minimi quadrati con l'algoritmo del gradiente
%coniugato con precondizionatrice C, con precondizionatrice la matrice
%tridiagonale ottenuta dalle 3 diagonali principali di T'T e senza
%precondizianare e  restituisce il numero di iterazioni necessarie al
%metodo per raggiungere la stessa tolleranza (1e-7) nei tre casi

for q=1:3
    switch q
        case {1}
            %ben condizionata
            c=zeros(1,m);
            for j=1:m
                c(j) = j^(-1.1)+1i*j^(-1.1);
            end
            c(1)=norm(c);
            r=c(1:n);

        case {2}
```

```
        %mal condizionata
        v(1)=0;
        v(2:m)=(2:m).^(-1.1);
        c=(v+1i*v);
        r=c(1:n);

    otherwise
        %matrice sparsa costruita artificiosamente
        v(1)=0;
        v(2:m)=(2:m).^(-1.1);
        c1=(v+1i*v);
        r1=c1(1:n);
        r=zeros(1,n);
        c=zeros(1,m);

        for j=1:5:m
            if(j+1<m)
                c(j+1)=c1(j+1);
            end
            if(j+2<n)
                r(j+2)=r1(j+2);
            end
        end
        r(1)=c(1);
end
%costruzione matrice T a partire da c e r
row_T=r.';
col_T=c.';
T=toeplitz(col_T,row_T);
A=T'*T;

%mostra la matrice sparsa
if(q==3)
    figure(4)
    imshow(abs(A))
end

%costruzione della precondizionatrice diagonale
M = (spdiags([[diag(A,-1);0] diag(A) [0;diag(A,1)]],-1:1,n,n));

%applicazoni del metodo del gradiente coniugato
```

```
    figure (q)
    [iter_no, x_no]=minsqgc(row_T,col_T,ones(m,1), 1e-7);
    hold on

    [iter_M, x_M]=prec_gc(T'*T,T'*ones(m,1),M, 1e-7);
    hold on

    col_C=retcircprec(row_T,col_T); %costruzione della matrice di
        precondizionamento
    [iter_C, x_C]=circprec_cg(row_T, col_T,ones(m,1),col_C, 1e-7);

    xlabel('iterazioni')
    ylabel('residui')
    legend('non_prec','M-prec','C-prec', 'Location', 'best')
    hold off

end
```

Table 2: Numero di iterazioni necessarie per raggiungere per soddisfare il criterio di arresto nel caso non precondizionato precondizionato da M e da C, per $m = 2n$

	Matrice ben condizionata			Matrice mal condizionata			Matrice sparsa		
n	non-prec	M-prec	C-prec	non-prec	M-prec	C-prec	non-prec	M-prec	C-prec
100	22	65	7	168	493	15	39	37	14
150	25	84	7	297	1000	15	42	41	14
250	31	132	7	627	1000	16	45	45	15
500	36	246	7	1000	1000	17	49	48	15

Notiamo che in tutti i casi il comportamento del sistema precondizionato da C ha una convergenza nettamente più rapida. Inoltre, la matrice M nei primi due casi non solo non migliora la velocità di convergenza ma addirittura la peggiora.

Figure 2: residui a ogni passo per n=250, per matrice ben condizionata

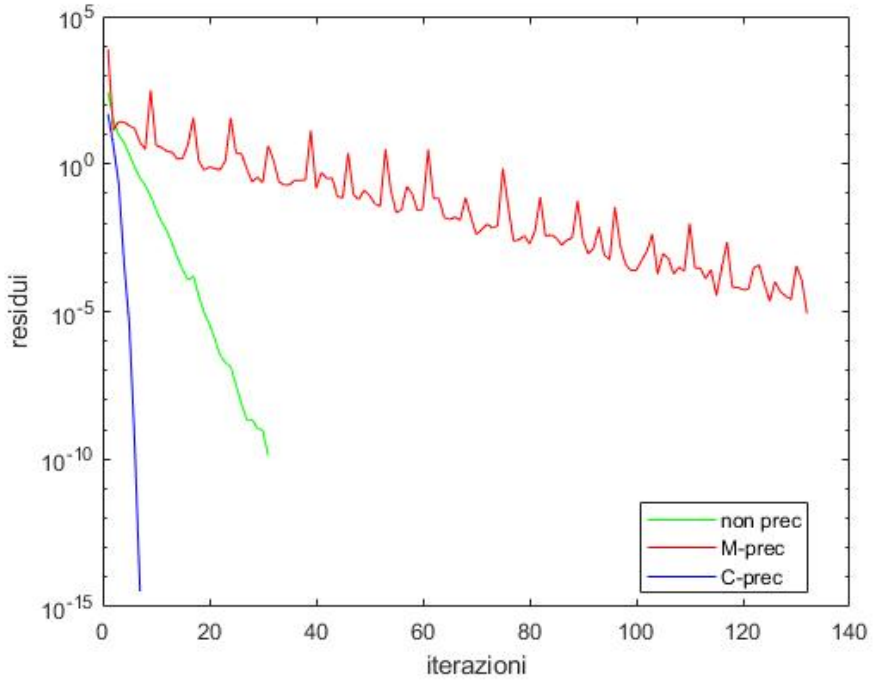
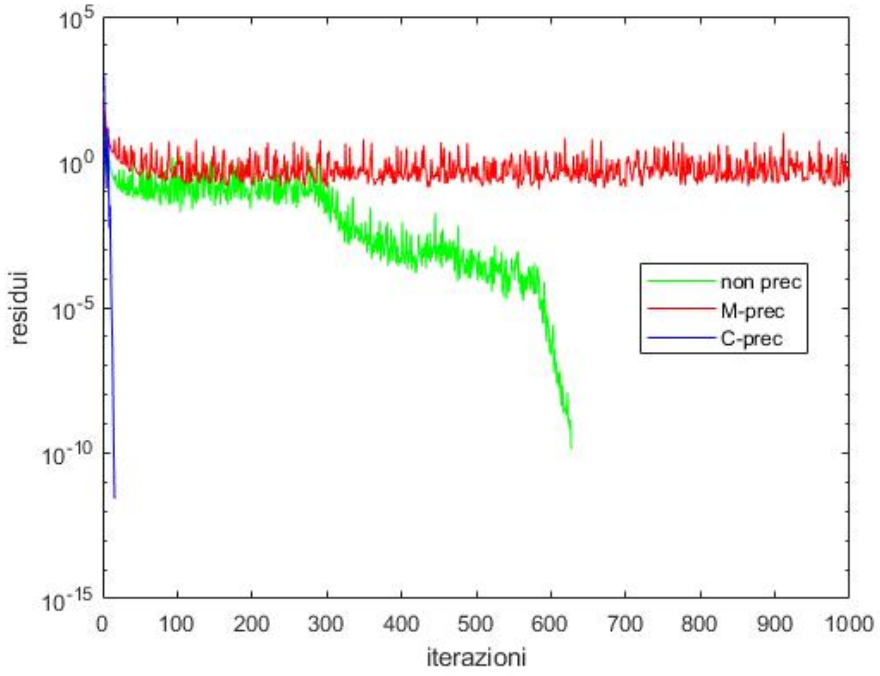
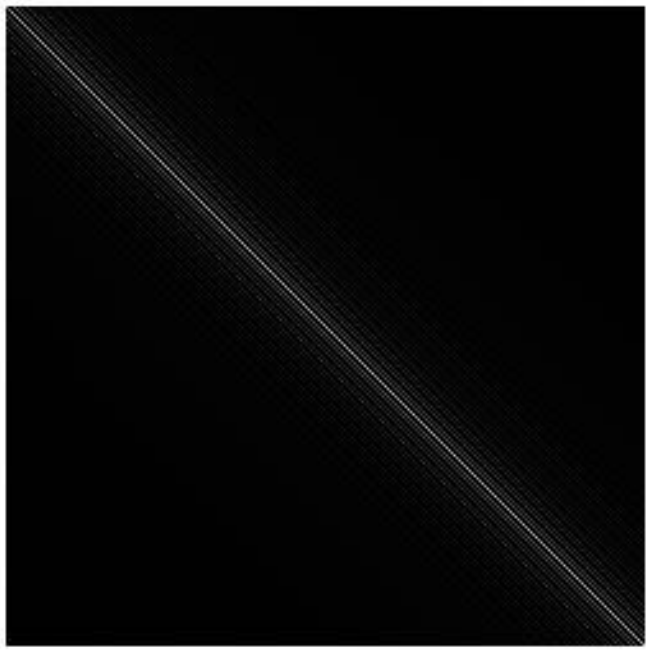


Figure 3: residui a ogni passo per n=250, per matrice mal condizionata



Infatti notiamo un grande numero di oscillazioni nei residui del metodo precondizionato da M, sia nel problema ben condizionato che nel problema mal condizionato dove non riesce a raggiungere la tolleranza se $n \geq 150$. Al contrario precondizionare con C sembra garantire più regolarità in presenza di mal condizionamento, appiattendo le oscillazioni presenti nel sistema non precondizionato.

Figure 4: $T'T$ per $m = 500$ e $n = 250$



Infine si osserva dai risulati per T3 sparsa che nonostante la matrice $T3'T3$ sembri concentrare la maggior parte dell'informazione sulla diagonale, precondizionare con C può essere comunque vantaggioso.

Test 3

Per il terzo test verifichiamo che anche sistemi di grosse dimensioni sono risolvibili utilizzando la precondizionatrice C. A tal scopo consideriamo la matrice T mxn definita come:

$$c(k) = i(k^{-1.1}) \text{ per k=1,...,m}$$
$$r(j) = j^{-1.1} \text{ per j=1,...n}$$

dove r e c sono rispettivamente la prima riga e la prima colonna di T_3,

```
function [x_C, iter]= test3_1(m,n)
%test di grandi dimensioni
col_T=zeros(m,1);

for j=2:m
    col_T(j) = j^(-1.1);
end

row_T=col_T(1:n);
col_T = 1i*col_T;

col_C=retcircprec(row_T,col_T);
[x_C, iter]=circprec_cg(row_T, col_T,ones(m,1),col_C, 1e-7);
xlabel('iterazioni')
ylabel('residui')
end
```

Table 3: Numero di iterazioni e tempi necessari per raggiungere il criterio di arresto nel caso precondizionato da C per m=4n

n	iterazioni	tempo(s)
31250	30	3.97
62500	29	6.43
125000	33	36.30
250000	34	129.58

Nonostante il problema quadrato non sia ben condizionato, vediamo che il fatto che la convergenza richieda pochi passi rende possibile la risoluzione di questi problemi in tempi contenuti nonostante le grandi dimensioni.

Bibliografia

1 Raymond H. Chan, James G. Nagy, and Robert J. Plemmons, Circulant preconditioned Toeplitz least squares iterations, Siam J. matrix anal. appl. Vol. 15, No. 1, pp. 80-97, January 1994

2 R. Chan and M. Yeung, Circulant Preconditioners for Complex Toeplitz Matrices, Research Report 91-6, Dept. of Math., Univ. of Hong Kong, Hong Kong, 1992.

3 O. Axelsson and G. Lindskog, On the rate of convergence of the preconditioned conjugate gradient algorithm, Numer. Math., 48 (1986), pp. 499-523.