Tutoraggio #3

Federico Pichi

8 maggio 2019

Esercizio 1

Implementare il metodo di fattorizzazione di Cholesky per matrici simmetriche e definite positive.

Esercizio 2

Si consideri la matrice

$$A = \begin{bmatrix} 7 & 1 & 3 \\ 1 & 8 & 2 \\ 3 & 2 & 9 \end{bmatrix}$$

ed il vettore $b = Ax_e$, con $x_e = [1, 1, 1]^T$. Utilizzare il metodo LU implementato per fattorizzare la matrice A. Utilizzare il comando **lu** di Matlab per ottenere la matrice di permutazione della pivotazione e commentare il risultato. Fare lo stesso con il metodo di Cholesky implementato al punto precedente e tramite il comando Matlab **chol**. Utilizzare i metodi di risoluzione in avanti e all'indietro per risolvere il sistema lineare $Ax = b$ attraverso le due fattorizzazioni. Ripetere il procedimento per il sistema lineare Bx = d, dove

$$B = \begin{bmatrix} 1 & 1-\epsilon & 3 \\ 2 & 2 & 2 \\ 3 & 6 & 4 \end{bmatrix}, \quad d = \begin{bmatrix} 5-\epsilon \\ 6 \\ 13 \end{bmatrix}, \quad \epsilon \in \mathbb{R}.$$

Commentare i risultati ottenuti per $\epsilon = 0$ ed $\epsilon = 1$.

Esercizio 3

Si consideri la funzione di Runge

$$f(x) = \frac{1}{1 + 25x^2}$$

e la si interpoli su nodi equispaziati in $I = [-1, 1]$ utilizzando un polinomio di grado n. Attraverso i comandi **polyfit** e **polyval** commentare i risultati ottenuti per $n = 2 : 2 : 10$ calcolando l'errore di interpolazione. Ripetere il procedimento utilizzando i nodi di Chebyshev e spiegarne il diverso andamento.

Esercizio 4

Si implementi il metodo di bisezione in Matlab utilizzando una function che prenda in input la funzione, gli estremi dell'intervallo, una tolleranza ed il numero massimo di iterazioni e che restituisca come output lo zero della funzione, il residuo in quel punto e il numero di iterazioni effettuate.

Esercizio 5

Data la funzione $f(x) = coshx + cosx - \gamma$, per $\gamma = 1, 2, 3$ si individui un intervallo contenente uno zero di f e lo si calcoli con il metodo di bisezione con una accuratezza pari a 10^{-10}.

Soluzione 1

Codice 1: Soluzione 1

```
function [A]=chol2(A)
% CHOL2 fattorizzazione di Cholesky di una matrice A di tipo s.d.p..
% A=CHOL2(A) il triangolo superiore H di A è tale che H'*H=A.
[n,m]=size(A);
if n ~= m,  error('Solo sistemi quadrati');  end
A(1,1)=sqrt(A(1,1));
for j=2:n
    for i=1:j-1
        if A(j,j) <= 0,  error('Elemento pivotale nullo o negativo');  end
        A(i,j)=(A(i,j)-(A(1:i-1,i))'*A(1:i-1,j))/A(i,i);
    end
    A(j,j)=sqrt(A(j,j)-(A(1:j-1,j))'*A(1:j-1,j));
end
A = triu(A);
end
```

Soluzione 2

Innanzitutto è possibile effettuare la fattorizzazione LU senza pivotazione in quanto la matrice è a dominanza diagonale stretta. Infatti se controlliamo la matrice di permutazione P dall'output di lu, osserviamo che corrisponde all'identità, ovvero non è stato necessario nessun pivoting. Allo stesso modo la matrice ammette un'unica fattorizzazione di Cholesky in quanto simmetrica e definita positiva.

Codice 2: Soluzione 2

```
A = [7, 1, 3; 1, 8, 2; 3, 2, 9];
x_e = ones(3,1);
b = A*x_e;

A1 = lu2(A);
L1 = tril(A1,-1) + eye(size(A1,1));
U1 = triu(A1);
[L2, U2, P] = lu(A);

y_1=forward(L1,b);
x_1=backward(U1,y_1);

err_lu = norm(x_1-x_e)/norm(x_e);
H1 = chol2(A);
H2 = chol(A);

y_2=forward(H1',b);
x_2=backward(H1,y_2);
err_chol = norm(x_2-x_e)/norm(x_e);

eps = 1;
B = [1, 1-eps, 3; 2, 2, 2; 3, 6, 4];
d = [5-eps; 6; 13];
B1 = lu2(B);
L3 = tril(B1,-1) + eye(size(B1,1));
U3 = triu(B1);

[L4, U4, P4] = lu(B);
```

Osserviamo che la matrice B non è simmetrica definita positiva, dunque non è possibile effettuare la fattorizzazione di Cholesky per nessun ϵ. Per quanto riguarda la fattorizzazione LU, per $\epsilon = 1$, tutto funziona come previsto. Al contrario, per $\epsilon = 0$, anche se la matrice è non singolare è necessaria la tecnica del pivoting per evitare di incorrere in una divisione per zero.

Soluzione 3

Se interpoliamo la funzione $f(x) = 1/(1 + x^2)$ (detta di Runge) su un insieme di nodi equispaziati nell'intervallo $I = [-5, 5]$, l'errore $\max_{x \in I} |E_n f(x)|$ tende all'infinito quando $n \to \infty$. Questo è dovuto al fatto che quando $n \to \infty$ l'ordine di infinito di $\max_{x \in I} |f^{n+1}(x)|$ domina sul resto facendo esplodere la stima. Al contrario, utilizzando i nodi di Chebyshev, stiamo utilizzando una configurazione tale da ottenere una costante di Lebesgue che cresce lentamente al variare di n, ottenendo così la convergenza uniforme del nostro polinomio interpolatore alla funzione data.

Codice 3: Soluzione 3

```
f = @(x) 1./(1 + 25*x.^2);

x_1 = linspace(-1,1,500);
y_1 = f(x_1);
subplot(1,2,1)
plot(x_1,y_1,'r','linewidth',2);
hold on;
a = -1;
b = 1;

for n = 2:2:20

    x = linspace(a,b,n+1);
    y = f(x);
    p = polyfit(x,y,n);
    y_int = polyval(p,x_1);

    err = y_1 - y_int;
    subplot(1,2,1)
    hold on
    plot(x_1,y_int,'--','color',rand(1,3));
    plot(x,f(x),'k.','markersize',30);
    subplot(1,2,2)
    hold on
    plot(x_1,err,'--','color',rand(1,3))
    grid;
end

figure
subplot(1,2,1)
hold on
plot(x_1,y_1,'r','linewidth',2);

for n = 2:2:20

    for i=0:n
        x_cgl = cos(pi*i/n);   % Chebyshev-Gauss-Lobatto
        x_cg = cos(pi*(2*i+1)/(2*(n+1)));   % Chebyshev-Gauss
        x_c(i+1)=((a+b)/2)-((b-a)/2)*x_cgl;
    end

    y = f(x_c);
    p = polyfit(x_c,y,n);
    y_che = polyval(p,x_1);

    err_che = y_1 - y_che;
    subplot(1,2,1)
    hold on
    plot(x_1,y_che,'--','color',rand(1,3));
    plot(x_c,f(x_c),'k.','markersize',30);
    subplot(1,2,2)
    hold on
    plot(x_1, err_che,'--','color',rand(1,3))
    grid;
end
```

Soluzione 4

Codice 4: Soluzione 4

```
function [zero,res,niter]=bisezione(fun,a,b,tol,kmax)
% Questa funzione implementa il metodo di bisezione per la funzione fun
% nell'intervallo [a,b], con tolleranza tol e numero massimo di iterazioni
% kmax.
x = [a, (a+b)*0.5, b];
fx = fun(x);

if fx(1)*fx(3) > 0
    error('Il segno della funzione agli estremi dell'intervallo [a,b] deve essere
        diverso');
elseif fx(1) == 0
    zero = a; res = 0; niter = 0;
    return
elseif fx(3) == 0
    zero = b; res= 0; niter = 0;
    return
end

niter = 0;
I = (b - a)*0.5;
while I >= tol && niter < kmax
    niter = niter + 1;
    if fx(1)*fx(2) < 0
        x(3) = x(2);
        x(2) = x(1)+(x(3)-x(1))*0.5;
        fx = fun(x);
        I = (x(3)-x(1))*0.5;
    elseif fx(2)*fx(3) < 0
        x(1) = x(2);
        x(2) = x(1)+(x(3)-x(1))*0.5;
        fx = fun(x);
        I = (x(3)-x(1))*0.5;
    else
        x(2) = x(find(fx==0));
        I = 0;
    end
end

if niter==kmax && I > tol
fprintf('Il metodo di bisezione si e' arrestato senza soddisfare la tolleranza richiesta
    avendo raggiunto il numero massimo di iterazioni\n');
end
zero = x(2);
x = x(2);
res = fun(x);
end
```

Soluzione 5

Utilizzando il comando fplot studiamo l'andamento della funzione data al variare di γ. Per $\gamma = 1$ essa non ammette zeri reali.

Per $\gamma = 2$, si ha il solo zero $\alpha = 0$ con molteplicità 4 (cioè tale che $f(\alpha) = f'(\alpha) = f''(\alpha) = f'''(\alpha) = 0$ ma $f^{(4)}(\alpha) \neq 0$).

Per $\gamma = 3$, f presenta due zeri semplici, uno nell'intervallo $[-3, -1]$, uno in $[1, 3]$. Nel caso $\gamma = 2$ il metodo di bisezione non può essere impiegato non essendo possibile trovare un intervallo per il quale $f(a)f(b) < 0$. Per $\gamma = 3$ a partire da $[a, b] = [-3, -1]$, il metodo di bisezione converge in 34 iterazioni allo zero $\alpha = -1.8579$ (con $f(\alpha) \approx -3.6 \cdot 10 - 12$).

Analogamente, scegliendo $[a, b] = [1, 3]$, per $\gamma = 3$ si ha convergenza in 34 iterazioni allo zero $\alpha = 1.8579$ (con $f(\alpha) \approx -3.68 \cdot 10 - 12$).

Codice 5: Soluzione 5

```
gamma = [1,2,3];
x = linspace(-3,3);

for g = gamma
    f = @(x) cosh(x)+cos(x)-g;
    plot(x, f(x), 'color', rand(1,3))
    hold on
    grid
end

a = -3;
b = -1;
tol = 1.e-10;
nmax = 200;
[zero_1,res_1,niter_1] = bisezione(f,a,b,tol,nmax);
a = 1;
b = 3;
[zero_2,res_2,niter_2] = bisezione(f,a,b,tol,nmax);
```