Metodi numerici con elementi di Programmazione

A.A. 2013-2014

Esercizi svolti in Laboratorio

Lezione del 19-11-2013

Docente: Vittoria Bruni

Email: vittoria.bruni@sbai.uniroma1.it

Ufficio: Via A. Scarpa,
Pal. B, I piano, Stanza n. 16
Tel. 06 49766648

Ricevimento: Giovedì 14.00-15.00

Il materiale didattico è disponibile sul sito http://ingaero.uniroma1.it/ nella pagina dedicata al corso Metodi Numerici con elementi di Programmazione

Per consultazione: Getting Started with MatLab – The mathworks www.mathworks.com

Esercizio

Si consideri la successione

$$x_{n+1} = \begin{cases} 2x_n + 2 & x_n \; dispari \\ \frac{x_n}{2} + 1 & x_n \; pari \end{cases}$$

Scrivere uno script MatLab (esercizio_lab2.m) che, dato un punto iniziale x0 (x0 deve essere un numero intero), grafichi i primi 500 elementi della successione e, di essi, calcoli e stampi opportunamente il massimo e il minimo.

Richiamare lo script con

x0 = 1 x0=-1.1 x0 = [2 6 9] x0 = [1 .1, .1 2]

x0 = 2 x0 = -2 x0 = 5 x0 = -1002

Soluzione

```
% Lo script esercizio_lab2 legge in input il numero intero x0, grafica
% i primi 500 elementi della successione x(n) e ne stampa il massimo
% e  il minimo.
% La successione x(n) è definita per n>=1 come segue
%       x(n)= 2 x(n-1) + 2    se n è dispari
%       x(n)= x(n-1)/2 + 1    se n è pari
x0 = input('inserisci il punto iniziale (numero intero) ');
if  (x0~=round(x0)) |  (length(x0(:))>1)
    error('il punto iniziale deve essere un numero intero!!!')
end
x(1) = x0;
for n=2:500
     if rem(x(n-1),2)==0
        x(n) = x(n-1)/2 + 1;
     else
        x(n) = 2*x(n-1) + 2;
     end
end
```

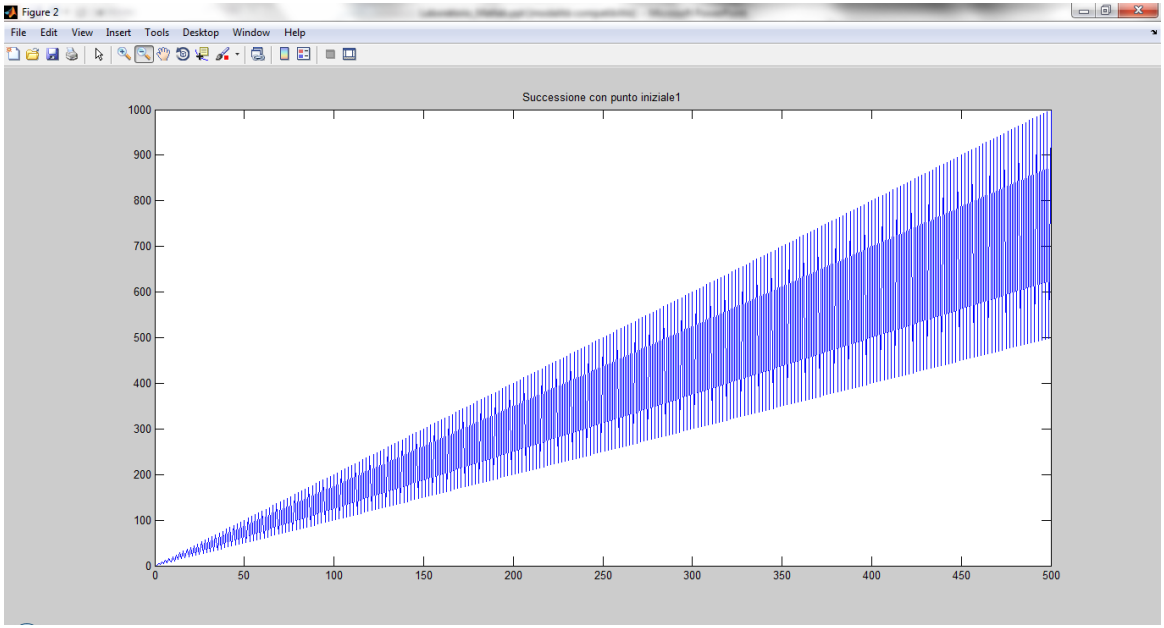
```
figure,
plot(x)
title(['Successione con punto iniziale', int2str(x0)])

M = max(x);
m = min(x);

fprintf('il massimo e il minimo della successione sono %10d \t %10d\n', M,m)
```

Dal Command Window

>> esercizio_lab2
inserisci il punto iniziale (numero intero) 1
il massimo e il minimo della successione sono 1000 1



>> esercizio_lab2
inserisci il punto iniziale (numero intero) -1.1
??? Error using ==> esercizio_lab2 at 8
il punto iniziale deve essere un numero intero!!!

Dal Command Window

>> esercizio_lab2
inserisci il punto iniziale (numero intero) [2 6 9]
??? Error using ==> esercizio_lab2 at 8
il punto iniziale deve essere un numero intero!!!

>> esercizio_lab2
inserisci il punto iniziale (numero intero) [-1 .1; .1 2]
??? Error using ==> esercizio_lab2 at 8
il punto iniziale deve essere un numero intero!!!

Esercizi

- Scrivere lo script Matlab MEG_nopivot.m che implementi il metodo di eliminazione di Gauss senza pivoting per la soluzione di sistemi lineari.
  Risolvere il seguente sistema lineare richiamando lo script MEG_nopivot.m

$$\begin{cases} x_1 + 2x_2 + 3x_3 = 1 \\ 4x_1 + 5x_2 + 6x_3 = 0 \\ 7x_1 + 8x_2 + = 2 \end{cases}$$

Metodo di eliminazione di Gauss

Si eseguono n-1 passi

al passo k si definiscono gli elementi

$$a_{ij}^{(k)} = a_{ij}^{(k-1)} - m_{ik} a_{kj}^{(k-1)}, \quad i = k+1, \ldots, n \quad j = k, k+1, \ldots, n$$

$$b_i^{(k)} = b_i^{(k-1)} - m_{ik} b_k^{(k-1)}, \quad i = k+1, \ldots, n$$

con $m_{ik} = \dfrac{a_{ik}^{(k-1)}}{a_{kk}^{(k-1)}}$

Soluzione

```
% meg_nopivot
% Risolve il sistema lineare Ax=b usando il metodo di
% eliminazione di Gauss senza pivoting

clear all % cancella tutte le variabili in Workspace
%lettura della matrice dei coefficienti e del vettore dei termini noti
A = input('inserisci la matrice del sistema A= ');
b = input('inserisci il vettore dei termini noti b = '');
[m,n] = size(A);

% controllo sulla dimensione di A e b
if (m~=n) | ((m==n) & (n ~=length(b)))
   error('Attenzione alla dimensione di A e b!!!');
end
% controllo dell'esistenza di un'unica soluzione
if det(A) == 0
   error('la matrice dei coefficienti e'' singolare!!!')
end
```

Esercizi

```
for  k =1:n-1 % gestisce il numero di passi del metodo di Gauss
        for  i =k+1:n
             m = A(i,k)/A(k,k); % calcolo dei moltiplicatori
              for j=k:n % modifica  degli elementi sulla i-sima riga
                   A(i,j) = A(i,j)-m*A(k,j);
              end
             b(i)=b(i)-m*b(k); % modifica  della i-sima riga componente di b
        end
end
% metodo di sostituzione all'indietro.
% All'uscita dei cicli annidati precedenti la matrice A è triangolare superiore
x(n) = b(n)/A(n,n);
for  i =n-1:-1:1
    x(i) = (b(i)-sum(A(i,i+1:n).*x(i+1:n)))/A(i,i);
end
x = x'; % la soluzione deve essere un vettore colonna
fprintf('la soluzione del sistema e'' x= \n ')
fprintf('%+6.5f \n',x(:))
```

Osservazione

metodo di sostituzione all'indietro

```
x(n) = b(n)/A(n,n);
for  i =n-1:-1:1
    x(i) = (b(i)-sum(A(i,i+1:n).*x(i+1:n)))/A(i,i);                    (*)
end
```

***Oss*:**

1. l'istruzione precedente è corretta perché x è generato come un vettore riga
2. Il ciclo precedente è equivalente a

```
for  i =n-1:-1:1
        somma = 0;
        for j=i+1:n
            somma = somma + A(i,j)*x(j);
        end
        x(i)=(b(i)-somma)/A(i,i);
end
```

3. **Stabilire se la seguente istruzione è equivalente a (*)**

```
x(i) = (b(i)-A(i,i+1:n)*x(i+1:n)')/A(i,i);
```

Dal Command Window

```
>> MEG_nopivot
inserisci la matrice del sistema A= [1 2 3; 4 5 6; 7 8 0];
inserisci il vettore dei termini noti b = [1 0 2];
la soluzione del sistema e' x=
 -2.00000
+2.00000
-0.33333
```

```
>> MEG_nopivot
inserisci la matrice del sistema A= [1 2 3; 4 5 6; 7 8 0];
inserisci il vettore dei termini noti b = [1 0 2 0];
??? Error using ==> MEG_nopivot at 14
Attenzione alla dimensione di A e b!!!
```

Dal Command Window

```
>> MEG_nopivot
inserisci la matrice del sistema A= [1 2 3; 4 5 6; 0 0 0];
inserisci il vettore dei termini noti b = [1 0 2];
??? Error using ==> MEG_nopivot at 19
la matrice dei coefficienti e' singolare!!!
```

```
>> MEG_nopivot
inserisci la matrice del sistema A= [1 2 3; 4 5 6]
inserisci il vettore dei termini noti b = [1 0 2];
??? Error using ==> MEG_nopivot at 14
Attenzione alla dimensione di A e b!!!
```

Alcune funzioni

D=diag(v) costruisce una matrice diagonale con il vettore v sulla diagonale principale

v=diag(D) estrae la diagonale principale della matrice D

w=diag(D,k) se k > 0 estrae la k-esima diagonale superiore, se k < 0 estrae la k-esima diagonale inferiore

T = triu(A) estrae la parte triangolare superiore di A. T è una matrice triangolare superiore

T = triu(A,k) estrae gli elementi che sono al di sopra della k-esima diagonale di A.

T = tril(A) estrae gli elementi che sono al di sotto della k-esima diagonale di A.

Esercizi

Da svolgere a «casa»

- Scrivere lo script sost_indietro.m che implementi il metodo di sostituzione all'indietro per la soluzione di sistemi lineari con matrice dei coefficienti triangolare superiore (eseguire il controllo sulla matrice)

- Scrivere uno script che implementi il metodo di sostituzione in avanti per la soluzione di sistemi lineari con matrice dei coefficienti triangolare inferiore (eseguire il controllo sulla matrice)

Esercizio

Scrivere lo script Matlab esercizio_lab3.m che, assegnata una matrice reale quadrata T:

a) calcoli l'elemento di modulo massimo t_{max} della prima colonna di T;

b) se t_{max} è diverso dal valore assoluto dell'elemento t_{11}, lo porti, con uno scambio di righe, al posto di t_{11};

c) stampi la nuova matrice.

Usare come input

$$T = \begin{pmatrix} 1 & 5 & 7 \\ -3 & 2 & 1 \\ \frac{1}{2} & 1 & 1 \end{pmatrix} \qquad T = \begin{pmatrix} 2 & \frac{1}{2} & 3 \\ 1 & 4 & 8 \\ 0 & -1 & -1 \end{pmatrix} \qquad T = \begin{pmatrix} 2 & 3 \\ -1 & 8 \\ 0 & 4 \end{pmatrix}$$

Soluzione

```
% Lo script esercizio_lab3 legge in input una matrice quadrata T e
% scambia la prima riga con la k-sima riga se si verifica la  seguente
% condizione
%              abs(T(k,1)) > abs(T(1,1))
% Se è stato effettuato uno scambio di righe, lo script stampa la matrice T
modificata
T = input('inserire una matrice quadrata, T = ');
[m,n] = size(T);
if (m~=n)
   error('La matrice deve essere quadrata!!!');
end
[tmax,ptmax] = max(abs(T(:,1)));
if tmax ~= abs(T(1,1))  % in  modo equivalente if ptmax ~= 1
   appo = T(1,:);
   T(1,:) = T(ptmax,:);
   T(ptmax,:) = appo;
   disp(T)
end
```

Soluzione

Dal Command Window

```
>> esercizio_lab3
inserire una matrice quadrata, T = [1 5 7; -3 2 1; 1/2 1 1]
   -3.0000    2.0000    1.0000
    1.0000    5.0000    7.0000
    0.5000    1.0000    1.0000

>> esercizio_lab3
inserire una matrice quadrata, T = [2 1/2 3; 1 4 8; 0 -1 -1]
>>

>> esercizio_lab3
inserire una matrice quadrata, T = [2 3; -1 8; 0 4]
??? Error using ==> esercizio_lab3 at 9
La matrice deve essere quadrata!!!
```

Esercizio

- Scrivere lo script MEG_pivot.m che implementi il metodo di eliminazione di Gauss con pivoting parziale per la soluzione di sistemi lineari

Suggerimento: si modifichi lo script MEG_nopivot.m adattando lo script esercizio_lab3.m al generico passo *k* del metodo di eliminazione di Gauss

Soluzione

Per usare le istruzioni in esercizio_lab3 al generico passo k del metodo di eliminazione di Gauss (script MEG_nopivot.m) per la soluzione del sistema lineare Ax = b, occorre fare i seguenti cambiamenti, includendo anche lo scambio delle componenti corrispondenti del vettore dei termini noti b

```
[tmax,ptmax] = max(abs(A(k:n,k)));
if tmax ~= abs(A(k,k))
   appo = A(k,:);
   A(k,:) = A(k+ptmax-1,:);
   A(k+ptmax-1,:) = appo;
   disp(A)   % si può omettere
   appo = b(k);
   b(k) = b(k+ptmax-1);
   b(k+ptmax-1) = appo;
end
```

Queste istruzioni vanno inserite nello script MEG_nopivot prima del calcolo del moltiplicatore m (subito dopo il for sull'indice k e prima del for sull'indice i)

Soluzione

Dal Command Window

```
>>MEG_pivot
inserisci la matrice del sistema A= [1 2 3; 4 5 6; 7 8 0];
inserisci il vettore dei termini noti b = [1 0 2];
la soluzione del sistema e' x=
 -2.00000
+2.00000
-0.33333
```