Metodi Numerici con elementi di Programmazione

A.A. 2013-2014

Introduzione al MatLab
V parte

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

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 traingolare 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) ? (usare lo help)

R = rank(A) calcola il numero di righe linearmente indipendenti di A

N = norm(A,n) calcola la norma n di A.

Se n = 1, calcola la norma 1

n = 2 , calcola la norma 2

n = inf, calcola la norma infinito

B=inv(A) calcola la matrice inversa di A (A è una matrice quadrata)

Esercizio

Scrivere uno script Matlab che risolva un sistema lineare con lo algoritmo di Thomas e utilizzarlo per risolvere il problema della passeggiata casuale

Algoritmo di Thomas

$$A = \begin{bmatrix} d_1 & s_1 & 0 & \cdots & 0 \\ a_2 & d_2 & s_2 & \cdots & 0 \\ \cdots & \ddots & \ddots & \ddots & \cdots \\ 0 & 0 & \cdots & d_{n-1} & s_{n-1} \\ 0 & 0 & \cdots & a_n & d_n \end{bmatrix} =$$

$$= \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ \alpha_2 & 1 & 0 & \cdots & 0 \\ 0 & \alpha_3 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & \alpha_n & 1 \end{bmatrix} \begin{bmatrix} u_1 & v_1 & 0 & \cdots & 0 \\ 0 & u_2 & v_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & u_{n-1} & v_{n-1} \\ 0 & 0 & \cdots & 0 & u_n \end{bmatrix} = LU$$

Algoritmo di Thomas

$$
\begin{cases}
u_1 = d_1 & \\
v_i = s_i & i = 1, 2, \ldots, n-1 \\
\alpha_i = a_i / u_{i-1} & i = 2, 3, \ldots, n \\
u_i = d_i - \alpha_i \, v_{i-1} & i = 2, 3, \ldots, n
\end{cases}
$$

Soluzione del sistema lineare $AX = B$

$$
y_1 = b_1 \qquad y_i = b_i - \alpha_i \, y_{i-1} \qquad i = 2, 3, \ldots, n
$$

$$
x_n = y_n / u_n \qquad x_i = (y_i - v_i \, x_{i+1}) / u_i \qquad i = n-1, \ldots, 1
$$

Algoritmo di Thomas

```
% thomas_diag risolve il sistema lineare Ax=b usando
% l'algoritmo di Thomas, con A matrice tridiagonale
% quadrata avente d come diagonale principale, a e s come
% codiagonali inferiore e superiore
%
% INPUT
% d = diagonale principale di A (vettore)
% a = diagonale inferiore di A (vettore)
% s = diagonale superiore di A (vettore)
% b = termine noto
%
% OUTPUT
% x = vettore soluzione
d = input('introduci la diagonale principale di A (vettore) d = ');
a = input('introduci la diagonale inferiore di A (vettore) a = ');
s = input('introduci la diagonale superiore di A (vettore) s = ');
b = input('introduci il vettore dei termini noti b = ');
```

Algoritmo di Thomas

```
% Controllo delle variabili di input: matrice quadrata
la = length(a);
ls = length(s);
ld = length(d);

if  (la~=ls)|(ld~=(la+1))
    disp('LA MATRICE NON è QUADRATA!!!')% alternativa al comando error
    x = [];
else
    n=ld;
    tic
 % fattorizzazione LU di A
    u(1) = d(1);
    v = s;
    for i = 1:n-1
        alfa(i) = a(i)/u(i);
        u(i+1) = d(i+1) - alfa(i)*v(i);
    end
```

Algoritmo di Thomas

```
% calcolo della soluzione L*y = b
    y(1) = b(1);
    for i=2:n
        y(i) = b(i)-alfa(i-1)*y(i-1);
    end

    % U*x = y
    x(n) = y(n)/u(n);
    for i = n-1:-1:1
        x(i) = (y(i)-v(i)*x(i+1))/u(i);
    end
    toc
end

whos
```

A cosa corrispondono i comandi tic e toc ?

Risultati Passeggiata aleatoria

```
>> d=ones(1,N-1);, a=-.5*ones(1,N-2);, s = .5*ones(1,N-2);
>> b = zeros(1,N-1);, b(1) = .5;
>> thomas_diag
introduci la diagonale principale di A (vettore) d = d
introduci la diagonale inferiore di A (vettore) a = a
introduci la diagonale superiore di A (vettore) s = s
introduci il vettore dei termini noti b = b
```

N	||X-xtrue||	Tempo di calcolo	Memoria
11	1.11e-16	0.000061 s	624 bytes
21	1.11e-16	0.000089 s	1376 bytes
51	4.33e-15	0.000179 s	3536 bytes
101	9.10e-15	0.000454 s	7136 bytes
501	4.61e-14	0.002566 s	35936 bytes
5001	3.20e-12	0.093172 s	359936 bytes
10001	6.95e-12	0.364083 s	719936 bytes

Algoritmo di Thomas

```
% thomas.m Risolve il sistema lineare Ax=b con A matrice
% tridiagonale quadrata, usando l'algoritmo di Thomas
% INPUT
% A = matrice tridiagonale
% b = termine noto
%
% OUTPUT
% x = vettore soluzione

A = input('inserisci la matrice del sistema A = ');
b = input('inserisci il vettore dei termini noti b = ');

% Controllo delle variabili di input
% matrice tridiagonale
T = triu(A,-1)-triu(A,2);
B = (A == T);
if  ~all(B(:))
    error('LA MATRICE NON è TRIDIAGONALE!!!')
end
```

Algoritmo di Thomas

```
% matrice non singolare
d = det(A);
if d==0
    error('LA MATRICE è SINGOLARE!!!')
end

% matrice quadrata
[n,m] = size(A)
if n~=m
    error('LA MATRICE DEVE ESSERE QUADRATA')
end

tic
% estraggo la diagonale principale di A e quelle sup e inf
d = diag(A);     % diagonale
a = diag(A,-1); % diagonale inferiore
s = diag(A,1);  % diagonale superiore
```

Algoritmo di Thomas

```
% fattorizzazione LU di A
u(1) = d(1);
v = s;
for i = 1:n-1
      alfa(i) = a(i)/u(i);
      u(i+1) = d(i+1) - alfa(i)*v(i);
end
% calcolo della soluzione
% L*y = b
y(1) = b(1);
for i=2:n
      y(i) = b(i)-alfa(i-1)*y(i-1);
end
% U*x = y
x(n) = y(n)/u(n);
for i = n-1:-1:1
      x(i) = (y(i)-v(i)*x(i+1))/u(i);
end
toc
```

Algoritmo di Thomas

Si genera una matrice tridiagonale

```
>> A = diag(ones(N-1,1))+diag(-.5*ones(N-2,1),1)+...
       diag(-.5*ones(N-2,1),-1);
>> b = zeros(1,N-1);, b(1) = .5;
>> thomas
inserisci la matrice del sistema A = A
inserisci il vettore dei termini noti b = b
```

OSS: lo script non ha comandi di visualizzazione del risultato ma la variabile x è nella memoria!

Esercizio

- Si consideri il sistema Ax = b, con matrice dei coefficienti A = [4 1 0 0 0; 1 4 1 0 0; 0 1 4 1 0; 0 0 1 4 1; 0 0 0 1 4] e vettore dei termini noti b = [10 12 12 12 10]'.
  Lo si risolva usando il metodo di eliminazione di Gauss (usando gli operatori predefiniti di Matlab) e l'algoritmo di Thomas e si confrontino i risultati in termini di precisione della soluzione e tempo di esecuzione

Ciclo while

while condizione (espressione logica)

    blocco di istruzioni

end

Esegue il blocco di istruzioni se la condizione è vera

Esempio: visualizzare i primi 4 numeri naturali

```
>>x=1;
>>while x~=5, disp(x), x=x+1; end
     1
     2
     3
     4
```

Ciclo while

Esempio:

```
a =10;
b = 5;
while a
        a = a-1;
        while b
                b = b-1;
        end
end
```

Esegue il ciclo fino a quando a e b sono entrambe 0

Nota: In un contesto logico una variabile numerica restituisce 1 (vero) se la variabile è diversa da zero, altrimenti 0 (falso).

Ciclo while

Esempio: quante iterazioni si compiono nei due cicli?

```
a =10;
b = 5;
cont_a = 0;
Cont_b_vett = [];
while a
        a = a-1;
        cont_b = 0;
        while b
                b = b-1;
                cont_b = cont_b+1;
        end
        Cont_b_vett = [Cont_b_vett cont_b];
        cont_a = cont_a+1;
end
```

Dal command window

```
>> cont_a
cont_a =
   10
>> cont_b
cont_b =
    0
>> Cont_b_vett
Cont_b_vett =
    5    0    0    0    0    0    0    0    0    0
```

Esercizio

Scrivere uno script Matlab che stampi il quadrato di tutti gli interi positivi x per cui risulta x^3 <2000

```
% script_while
x = 1;
while x^3 < 2000
        disp(x^2)
        x = x+1;
end
```

Dal command window

```
>> script_while
     1
     4
     9
    16
    25
    36
    49
    64
    81
   100
   121
   144
```

Esercizio

```
x =input('introduci il valore in cui calcolare la funzione,x = ');
oldsum = 0;, newsum = 1;
term = 1; % primo termine della serie
n = 0;    % indica il termine della serie

while newsum ~= oldsum, %Itera finché il termine successivo della
                        %serie diventa trascurabile (nella
                        %precisione di macchina)
      n = n + 1;
      term = term * x/n;    % x^n/n! = (x^{n-1}/(n-1)!) * x/n
                            % aggiorna i termini della serie
      oldsum = newsum;
      newsum = newsum+term %aggiorna la somma n-sima della serie
end

approx_value = newsum;
true_value = exp(x);
disp('valore approssimato'),
disp(approx_value)
disp('valore vero')
disp(true_value)
```

Esercizio

Si consideri il seguente codice. Stabilire cosa rappresenta la variabile xnew

```
x = 1;
dif = 100;
while dif > 1e-8
        xnew = cos(x);
        dif = abs(xnew-x);
        x = xnew;
end
xnew
```

Esercizio

Scrivere lo help del seguente script

```
xold = 2;
xnew = 1;
while abs(xnew-xold) > 1e-5
        xold = xnew;
        xnew = xnew/2+3/(2*xnew);
end
```

Algoritmo di bisezione

Con criterio di arresto a posteriori – ciclo while

```
format long;
% funzione di cui si vuole cercare una radice
f = input('inserire la funzione di cui cercare la radice f = ');

% intervallo in cui si vuole cercare la radice
a = input('inserire l''estremo inferiore dell''intervallo a = ');
b = input('inserire l''estremo superiore dell''intervallo b = ');

%inizializzazione dei parametri
iter = 0; err1 = b-a; x0=a;
eps = input('inserire la precisione richiesta per la soluzione, eps =  ')
if isempty(eps)
  % se non viene data la precisione, si fissa eps = .5*10^-5
   eps = .5*10^-5;
end
```

```
while (err1>eps)
   xn = (a+b)/2;
   if ( f(a)*f(xn) < 0 )
      b = xn;
   elseif ( f(xn)*f(b) < 0 )
      a = xn;
   end
   iter = iter+1;
   err1 = abs(xn-x0);
   err2 = abs(f(xn));
   fprintf('%3d\t %15.15f\t %15.15f %15.15f %6.15f\f   %6.15f\n', ...
   [iter a b xn err1 err2])
   x0 = xn;
end
```

Algoritmo di bisezione

Con doppio criterio di arresto a posteriori – ciclo while

```
format long;
% funzione di cui si vuole cercare una radice
f = input('inserire la funzione di cui cercare la radice f = ');
% intervallo in cui si vuole cercare la radice
a = input('inserire l''estremo inferiore dell''intervallo a = ');
b = input('inserire l''estremo superiore dell''intervallo b = ');
%inizializzazione dei parametri
iter = 0; err1 = b-a; err2 = 10^10; x0=a;
eps = input('inserire la precisione richiesta per la soluzione, eps =  ')
if isempty(eps)
  % se non viene data la precisione, si fissa eps = .5*10^-5
   eps = .5*10^-5;
end
```

```
while (err1>eps) & (err2>eps)
   xn = (a+b)/2;
   if ( f(a)*f(xn) < 0 )
      b = xn;
   elseif ( f(xn)*f(b) < 0 )
      a = xn;
   end
   iter = iter+1;
   err1 = abs(xn-x0);
   err2 = abs(f(xn));
   fprintf('%3d\t %15.15f\t %15.15f %15.15f %6.15f\f   %6.15f\n', ...
   [iter a b xn err1 err2])
   x0 = xn;
end
```

Cosa accade se la condizione di entrata nel ciclo while diventa

```
while (err1>eps) | (err2>eps)
   xn = (a+b)/2;
   if ( f(a)*f(xn) < 0 )
      b = xn;
   elseif ( f(xn)*f(b) < 0 )
      a = xn;
   end
   iter = iter+1;
   err1 = abs(xn-x0);
   err2 = abs(f(xn));
   fprintf('%3d\t %15.15f\t %15.15f %15.15f %6.15f\f   %6.15f\n', ...
   [iter a b xn err1 err2])
   x0 = xn;
end
```

Esercizi

- Modificare opportunamente lo script precedente introducendo anche una condizione sul numero massimo di iterazioni consentite e scriverne lo help

- Modificare lo script punto_unito.m usando il ciclo a entrata controllata while-end con criterio di arresto a posteriori e controllo sul numero massimo di iterazioni consentite e scriverne lo help

- Scrivere uno script Matlab che implementi il metodo di Newton-Raphson per la soluzione di equazioni lineari

- Scrivere uno script Matlab che implementi il metodo di delle secanti con estremi variabili per la soluzione di equazioni lineari

Esercizi

Metodo di Newton-Raphson

```
format long;
% funzione di cui si vuole cercare una radice
f = input('introduci la funzione di cui trovare la radice f =  ');
% derivata della funzione f
df = input('introduci la derivata di f = ');

% intervallo in cui si vuole cercare la radice
a = input('inserire l''estremo inferiore dell''intervallo a = ');
b = input('inserire l''estremo superiore dell''intervallo b = ');
x0 = input('inserisci l'approssimazione iniziale x0 = ');

% se non si fornisce il punto iniziale, si sceglie l'estremo superiore
if isempty(x0)
   x0 = b;
end
```

```
%inizializzazione dei parametri
iter = 0; err1 = 10; err2 = 10;
eps = input('inserire la precisione richiesta sulla soluzione, eps = ')
% se non viene data la precisione si fissa eps = .5*10^-5
if isempty(eps)
   eps = .5*10^-5;
end
% si itera usando il criterio di arresto a posteriori |x(n)-x(n-1)|<=eps
while (err1>eps)
        xn = x0-f(x0)/df(x0);
        iter = iter + 1;
        err1 = abs(xn-x0);
        err2 = abs(f(xn));
        x0=xn;
        fprintf('%3d\t %15.15f %6.15f %6.15f\n',[iter xn err1 err2])
end
```

```
%inizializzazione dei parametri
iter = 0; err1 = 10; err2 = 10;
eps = input('inserire la precisione richiesta sulla soluzione, eps = ')
% se non viene data la precisione si fissa eps = .5*10^-5
if isempty(eps)
   eps = .5*10^-5;
end
% si itera usando il criterio di arresto a posteriori |x(n)-x(n-1)|<=eps
while (err1>eps)
        xn = x0-f(x0)/df(x0);
        iter = iter + 1;
        err1 = abs(xn-x0);
        err2 = abs(f(xn));
        x0=xn;
        fprintf('%3d\t %15.15f %6.15f %6.15f\n',[iter xn err1 err2])
end
```

è necessaria questa assegnazione?

Esercizi

Metodo delle secanti

```
format long;
% funzione di cui si vuole cercare una radice
f = input(‘introduci la funzione di cui trovare la radice f =  ’);

% intervallo in cui si vuole cercare la radice
a = input('inserire l''estremo inferiore dell''intervallo a = ');
b = input('inserire l''estremo superiore dell''intervallo b = ');

% come punti iniziali si scelgono gli estremi dell' intervallo
xn1 = a;
xn2 = b;
```

```
%inizializzazione dei parametri
iter = 0; err1 = 10; err2 = 10;
eps = input('inserire la precisione richiesta sulla soluzione, eps = ')
% se non viene data la precisione si fissa eps = .5*10^-5
if isempty(eps)
   eps = .5*10^-5;
end
% si itera usando il criterio di arresto a posteriori |x(n)-x(n-1)|<=eps
while (err1>eps)
        xn = xn2-f(xn2) *(xn2-xn1)/(f(xn2)-f(xn1));
        iter = iter + 1;
        err1 = abs(xn-xn2);
        err2 = abs(f(xn));
        fprintf('%3d\t %15.15f %6.15f %6.15f\n',[iter xn err1 err2])
        xn1 = xn2;
        xn2 = xn;
end
```

```
%inizializzazione dei parametri
iter = 0; err1 = 10; err2 = 10;
eps = input('inserire la precisione richiesta sulla soluzione, eps = ')
% se non viene data la precisione si fissa eps = .5*10^-5
if isempty(eps)
   eps = .5*10^-5;
end
% si itera usando il criterio di arresto a posteriori |x(n)-x(n-1)|<=eps
while (err1>eps)
        xn = xn2-f(xn2) *(xn2-xn1)/(f(xn2)-f(xn1));
        iter = iter + 1;
        err1 = abs(xn-xn2);
        err2 = abs(f(xn));
        fprintf('%3d\t %15.15f %6.15f %6.15f\n',[iter xn err1 err2])
        xn1 = xn2;
        xn2 = xn;
end
```

è necessaria questa assegnazione?

Esercizi

- Modificare gli script precedenti inserendo opportuni controlli e messaggi di errore

- Modificare gli script precedenti in modo da
  i) creare un vettore avente come componenti gli errori commessi ad ogni iterazione
  (Suggerimento: inizializzare fuori dal ciclo while le variabili E1 e E2 e porle uguali al vettore vuoto, cioè **E1=[]; E2=[];**. All'interno del ciclo while aggiungere le assegnazioni **E1 = [E1 err1]);** e **E2 = [E2 err2]);** subito dopo le assegnazioni alle variabili **err1** e **err2**.)
  ii) tracciare il grafico dell'errore
  iii) tracciare il grafico delle approssimazioni successive