Elaborato di

Calcolo Numerico

Anno Accademico 2019/2020

Niccolò *Piazzesi* - 6335623 - *niccolo.piazzesi@stud.unifi.it*
Pietro *Bernabei* - 6291312 - *pietro.bernabei@stud.unifi.it*

Contents

1 Capitolo 1

1.1 Esercizio 1

Sia $f(x)$ una funzione sufficientemente regolare e sia $h > 0$ una quantità abbastanza "piccola". Possiamo sviluppare i termini $f(x-h)$ e $f(x+h)$ mediante il polinomio di Taylor:

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2}f''(x) + \frac{h^3}{6}f'''(x) + O(h^4)$$

$$f(x-h) = f(x) - hf'(x) + \frac{h^2}{2}f''(x) - \frac{h^3}{6}f'''(x) + O(h^4)$$

Sostituiamo i termini nell'espressione iniziale:

$$\frac{f(x-h) - 2f(x) + f(x+h)}{h^2} =$$

$$= \frac{f(x) - hf'(x) + \frac{h^2}{2}f''(x) - \frac{h^3}{6}f'''(x) + O(h^4) - 2f(x) + f(x) + hf'(x) + \frac{h^2}{2}f''(x) + \frac{h^3}{6}f'''(x) + O(h^4)}{h^2} =$$

$$= \frac{h^2 f''(x) + O(h^4)}{h^2} = f''(x) + O(h^2)$$

1.2 Esercizio 2

Eseguendo lo script si ottiene $u = 1.1102e-16 = \frac{\epsilon}{2}$, dove ϵ è la precisione di macchina. ϵ è il più piccolo valore di macchina per il quale $a+\epsilon \neq a$ per un qualsiasi numero a. Quando u assume valore $\frac{\epsilon}{2}$ il controllo interno $1 + u == 1$, che corrisponde alla condizione di uscita, risulta vero, perchè u è minore di ϵ.

1.3 Esercizio 3

Quando si esegue $a - a + b$ il risultato è 100 mentre quando si esegue $a + b - a$ si ottiene 0. La differenza dei risultati è dovuta al fenomeno della cancellazione numerica:

- nel primo caso la sottrazione avviene sullo stesso numero $a = 1e20$. Sottrare un numero da se stesso ha sempre risultato esatto 0.
- nel secondo caso la sottrazione avviente tra i termini $a + b = 1e20 + 100$ e $a = 100$. Poichè 1e20 è molto più grande di 100, a+b è "quasi uguale" ad a. La sottrazione amplifica gli errori di approssimazione causati dalla rappresentazione in aritmetica finita dei numeri coinvolti. A causa di questi errori il calcolatore approssima la differenza con 0.

2 Capitolo 2

2.1 Esercizio 4

```
function x1=radn(x, n)
%
% x1=radn(n.x)
% funzione Matlab che implementa il metodo di newton per il calcolo della
% radice n-esima di un numero positivo x
%
format long e
imax=1000;
tolx=eps;
if x<=0
    error('valore in ingresso errato');
end
x1=x/2;
for i=1:imax
   x0=x1;
   fx=x0^n-x;
   fx1=(n)*x0^(n-1);
   x1=x0-fx/fx1;
   if abs(x1-x0)<=tolx
       break
   end

end
if abs(x1-x0)>tolx
    error('metodo non converge')
end
```

2.2 Esercizio 5

- Metodo di bisezione

```
function [x,i] = bisez(f,a,b,tolx)
    %
    % [x, i] = bisez(f, a, b, tolx)
    % calcola la radice di f(x) utilizzando il metodo di bisezione sull'intervallo [a, b]
    format long e
    fa = feval(f,a);
    fb = feval(f,b);
    if(fa * fb > 0 )
        error('gli estremi  hanno lo stesso segno');
    end
    imax =  ceil(log2(b-a) - log2(tolx));
    for i = 1:imax
        x = (a+b)/2;
        fx = feval(f,x);
        f1x = abs((fb-fa)/(b-a));
        if abs(fx) <= tolx*f1x
            break
        elseif fa*fx<0
            b = x;
            fb = fx;
        else
            a = x;
```

```
            fa = fx;
        end
    end

end
```

- Metodo di Newton

```
function [x,i] = newton( f, f1, x0, tolx, maxit )
       %
       % [x,i] = newton( f, f1, x0, tolx [, maxit] )
       %
       % Metodo di Newton per determinare una approssimazione
       % della radice di f(x)=0 con tolleranza  tolx, a
       % partire da x0, entro maxit iterationi (default = 100).

       format long e
       if nargin<4
              error('numero argomenti insufficienti');
       elseif nargin==4
               maxit = 100;
       end
       if tolx<eps
              error('tolleranza non idonea');
       end
       x = x0;
       for i = 1:maxit
              fx = feval( f, x );
              f1x = feval( f1, x );
              x = x - fx/f1x;
              if abs(x-x0)<=tolx*(1+abs(x0))
                     break;
              else
                     x0 = x;
              end
       end
       if abs(x-x0) > tolx*(1+abs(x0))
              error('metodo non converge');
       end
end
```

- Metodo delle secanti

```
function [x, i]=secanti(f,x0,x1,tolx,maxit)
   %
       % [x,i] = secanti( f,  x0, x1, tolx [, maxit] )
       %
       % Metodo delle secanti per determinare una approssimazione
       % della radice di f(x)=0 con tolleranza  tolx, a
       % partire da x0, entro maxit iterationi (default = 100).
  format long e
  if nargin<4
    error('numero argomenti insufficienti');
  elseif nargin==4
    maxit = 100;
  end
  i=0;
  f0=feval(f,x0);
  for i=1:maxit
```

```
      f1=feval(f,x1);
      df1=(f1-f0)/(x1-x0);
      x=x1-(f1/df1);
      if abs(x1-x0)<=tolx*(1+abs(x0))
        break;
      end
      x0=x1;
      x1=x;
      f0=f1;

  end
  if abs(x-x0) > tolx*(1+abs(x0))
    error('metodo non converge');
  end
end
```

- Metodo delle corde

```
function [x,i] = corde( f, f1, x0, tolx, maxit )
    %
    % [x,i] = corde( f, f1, x0, tolx [, maxit] )
    %
    % Metodo delle corde per determinare una approssimazione
    % della radice di f(x) con tolleranza tolx, a
    % partire da x0, entro maxit iterationi (default = 100).

    format long e
    if nargin<4
           error('numero argomenti insufficienti');
    elseif nargin==4
            maxit = 100;
    end
    if tolx<eps
           error('tolleranza non idonea');
    end
    f1x = feval(f1, x0);
    x = x0;
    for i = 1:maxit
           fx = feval( f, x );
           if fx==0
                  break;
           end
           x = x - fx/f1x;
           if abs(x-x0)<=tolx*(1+abs(x0))
                  break;
           else
                  x0 = x;
           end
    end
    if abs(x-x0) > tolx*(1+abs(x0))
       error('metodo non converge');
     end
end
```

2.3 Esercizio 6

```
f = @(x)(x-cos(x));
```

```
f1 = @(x)(1+sin(x));

x0 = 0;
x1 = 1;
x=zeros(4,4)
y= zeros(4, 4);
for i=3:3:12
   [x(1, i/3), y(1, i/3)] = bisez(f, x0, x1, 10^(-i));
   [x(2, i/3), y(2, i/3)] = newton(f, f1, x0, 10^(-i));
   [x(3, i/3), y(3, i/3)] = corde(f, f1, x0, 10^(-i));
   [x(4,i/3), y(4, i/3)] = secanti(f, x0, x1, 10^(-i), 100);
end
row_names = {'bisezione', 'newton', 'corde', 'secanti'}
colnames = {'10^-3','10^-6','10^-9','10^-12'}
sTable = array2table(x,'RowNames',row_names,'VariableNames',colnames)
l = linspace(1, 4, 4)
plot(l, y')
```

Eseguendo lo script si ottengono i seguenti risultati:

Tolleranza	Metodo	Valore	iterazioni
10^{-3}	Bisezione	0.7382812500000000	8
	Newton	0.7390851333852840	4
	Secanti	0.7390851332150012	5
	Corde	0.7395672022122561	17
10^{-6}	Bisezione	0.7390842437744141	19
	Newton	0.7390851332151607	5
	Secanti	0.7390851332151607	6
	Corde	0.7390845495752126	34
10^{-9}	Bisezione	0.7390851341187954	28
	Newton	0.7390851332151607	5
	Secanti	0.7390851332151607	7
	Corde	0.7390851327392538	52
10^{-12}	Bisezione	0.7390851332147577	39
	Newton	0.7390851332151607	6
	Secanti	0.7390851332151607	7
	Corde	0.7390851332157368	69

2.4 Esercizio 7

Le nuove funzioni utilizzate in questo esercizio sono:

- Metodo di Newton modificato

```
function [x, i] = newtonmod( f, f1,  x0, m, tolx, maxit )
    %
    % [x,i] = newton( f, f1, x0, m, tolx [, maxit] )
    %Metodo di Newton modificato per determinare una approssimazione
    %di una radice di f(x) con molteciplita m,
    % a partire da x0, entro maxit iterationi (default = 100).

    format long e
    if nargin<5
           error('numero argomenti insufficienti');
    elseif nargin==5
            maxit = 100;
    end
    if tolx<eps
           error('tolleranza non idonea');
    end
```

```
    x = x0;
    for i = 1:maxit
            fx = feval( f, x );
            f1x = feval( f1, x );
            if fx==0
                    break;
            end
            x = x - m*(fx/f1x);
            if abs(x-x0)<=tolx*(1+abs(x0))
                    break;
            else
                    x0 = x;
            end
    end

end
```

- Metodo delle accelerazioni di Aitken

```
function [x, i] = aitken( f, f1, x0, tolx, maxit )
        %
        % [x,flag] = aitken( f, f1, x0, tolx [, maxit] )
        %
        % Metodo di accelerazione di Aitken per determinare una approssimazione
        % della radice di f(x)=0 con tolleranza (mista) tolx, a
        % partire da x0, entro maxit iterationi (default = 100).
        1
        % f1 implementa f'(x) mentre in uscita flag vale -1, se
        % la tolleranza non `e soddisfatta entro maxit iterate o
        % la derivata si annulla, altrimenti ritorna il numero
        % di iterazioni richieste.
        %
        format long e
        if nargin<4
                error('numero argomenti insufficienti');
        elseif nargin==4
                 maxit = 100;
        end
        if tolx<eps
                error('tolleranza non idonea');
        end
        x = x0;
        for i = 1:maxit
                x0 = x;
                fx = feval( f, x0 );
                f1x = feval( f1, x0 );
                x1 = x0 - fx/f1x;
                fx = feval( f, x1 );
                f1x = feval( f1, x1 );
                x = x1 -fx/f1x;
                x = (x*x0-x1^2)/(x-2*x1+x0);
                if abs(x-x0)<=tolx*(1+abs(x0))
                        break;
                end
        end
        if abs(x-x0) > tolx*(1+abs(x0))
                error('metodo non converge');
        end
end
```

La radice nulla della funzione $f(x) = x^2 tan(x)$ ha molteciplita m $= 3$, in quanto 0 annulla due volte il termine x^2 e $tan(0) = 0$.

3 Capitolo 3

3.1 Esercizio 8

```
function [LU,p]=palu(A)
% [LU,p]=palu(A)
% funzione Matlab che dato in input matrice A restituisce matrice fattorizzata LU
% e il relativo vettore p di permutazione di LU con pivoting parziale di A

[n,m]=size(A);
if(n~=m)
    error(matrice A non quadrata);
end
LU=A;
p=[1:n];
for i=1:n-1
    [mi,ki]=max(abs(LU(i:n,i)));
    if mi==0
        error('La matrice e'' non singolare')
    end
    ki=ki+i-1;
    if ki>i
        LU([i ki],:);
        p([i ki])=p([ki i]);
    end
    LU(i+1:n,1)=LU(i+1:n,i)/LU(i,i);
    LU(i+1:n,i+1:n)=LU(i+1:n,i+1:n)-LU(i+1:n,i)*LU(i,i+1:n);
end
```

3.2 Esercizio 11

```
function QR = myqr(A)
%QR = myqr(A)
% calcola la fattorizzazione QR di Householder della matrice A

    [m,n] = size(A);
    if  n > m
        error('Dimensioni errate');
    end
    QR = A;
    for i = 1:n
        alfa = norm(QR(i:m,i));
        if alfa == 0
            error('la matrice non ha rango massimo');
        end
        if QR(i,i) >= 0
            alfa = -alfa;
        end
        v1 = QR(i,i) -alfa;
        QR(i,i) = alfa;
        QR(i+1:m,i) = QR(i+1:m,i)/v1;
        beta = -v1/alfa;
        v = [1; QR(i+1:m,i)];
        QR(i:m,i+1:n) = QR(i:m,i+1:n) - (beta * v) * (v' * QR(i:m,i+1:n));
    end
end
```

3.3 Esercizio 12

```
function x = qrsolve(QR, b)
%
%
% x = qrSolve(QR, b)
% risolve il sistema QR*x=b nel senso dei minimi quadrati
%
[m,n] = size(QR);
k = length(b);
if k ~= m
    error('Dati inconsistenti');
end
x=b(:);
for i = 1:n
    v=[1; QR(i+1:m,i)];
    beta = 2/(v'* v);
    x(i:m) = x(i:m) - (beta*(v'*x(i:m))*v);
end
x=x(1:n);
for  j = n:-1:1
    if QR(j,j)==0
        error('Matrice singolare');
    end
    x(j) = x(j) / QR(j,j);
    x(1:j-1) = x(1:j-1) - QR(1:j-1,j)*x(j);
end
return
end
```

4 Capitolo 4

5 Capitolo 5