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 quantita 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

```matlab
function x1=radn(n,x)
%
% x1=radn(n.x)
% funzione Matlab che implementa il metodo di newtown per il calcolo della
% radice n-esima di un numero positivo x
%
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
```

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 ~= n
    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);
x = trisup(QR, x);
return
end
```

```
function x = trisup(A, b)
    %
    % x= trisup(A, b)
    %risolve il sistema triangolare superiore Ax=b memorizzato per colonne
    %
    %
    [m, n] = size(A);
    if m ~= n
        error('Matrice non quadrata');
    end
    x= b(:);
    for  j = n:-1:1
        if A(j,j)==0
            error('Matrice non singolare');
        end
        x(j) = x(j) / A(j,j);
        x(1:j-1) = x(1:j-1) - A(1:j-1,j)*x(j);
    end
    return
end
```

4 Capitolo 4

5 Capitoli 5/6