



Elaborato di  
**Calcolo Numerico**  
Anno Accademico 2019/2020

Niccolò *Piazzesi* - 6335623 - [niccolo.piazzesi@stud.unifi.it](mailto:niccolo.piazzesi@stud.unifi.it)  
Pietro *Bernabei* - 6291312 - [pietro.bernabei@stud.unifi.it](mailto:pietro.bernabei@stud.unifi.it)

# Contents

<b>1</b>	<b>Capitolo 1</b>	<b>3</b>
1.1	Esercizio 1 . . . . .	3
1.2	Esercizio 2 . . . . .	3
1.3	Esercizio 3 . . . . .	3
<b>2</b>	<b>Capitolo 2</b>	<b>4</b>
2.1	Esercizio 4 . . . . .	4
<b>3</b>	<b>Capitolo 3</b>	<b>5</b>
3.1	Esercizio 8 . . . . .	5
3.2	Esercizio 11 . . . . .	5
3.3	Esercizio 12 . . . . .	6
<b>4</b>	<b>Capitolo 4</b>	<b>7</b>
<b>5</b>	<b>Capitoli 5/6</b>	<b>8</b>

# 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:

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

## 1.2 Esercizio 2

Eseguendo lo script si ottiene  $u = 1.1102e - 16 = \frac{\epsilon}{2}$ , dove  $\epsilon$  è la precisione di macchina.  $\epsilon$  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  $\epsilon$ .

## 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  $a = 1e20$ . Sottrarre un numero da se stesso ha sempre risultato esatto 0.
- nel secondo caso la sottrazione avviene 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

```
1 function x1=radn(n,x)
2 %
3 % x1=radn(n,x)
4 % funzione Matlab che implementa il metodo di newtown per il calcolo della
5 % radice n-esima di un numero positivo x
6 %
7 imax=1000;
8 tolx=eps;
9 if x<=0
10     error('valore in ingresso errato');
11 end
12 x1=x/2;
13 for i=1:imax
14     x0=x1;
15     fx=x0^n-x;
16     fx1=(n)*x0^(n-1);
17     x1=x0-fx/fx1;
18     if abs(x1-x0)<=tolx
19         break
20     end
21 end
22 if abs(x1-x0)>tolx
23     error('metodo non converge')
24 end
25
```

## 3 Capitolo 3

### 3.1 Esercizio 8

```
1 function [LU,p]=palu(A)
2 % [LU,p]=palu(A)
3 % funzione Matlab che dato in input matrice A restituisce matrice fattorizzata LU
4 % e il relativo vettore p di permutazione di LU con pivoting parziale di A
5
6 [n,m]=size(A);
7 if(n~=m)
8     error(matrice A non quadrata);
9 end
10 LU=A;
11 p=[1:n];
12 for i=1:n-1
13     [mi,ki]=max(abs(LU(i:n,i)));
14     if mi==0
15         error('La matrice e'' non singolare')
16     end
17     ki=ki+i-1;
18     if ki>i
19         LU([i ki],:);
20         p([i ki])=p([ki i]);
21     end
22     LU(i+1:n,1)=LU(i+1:n,i)/LU(i,i);
23     LU(i+1:n,i+1:n)=LU(i+1:n,i+1:n)-LU(i+1:n,i)*LU(i,i+1:n);
24 end
```

### 3.2 Esercizio 11

```
1 function QR = myqr(A)
2 %QR = myqr(A)
3 % calcola la fattorizzazione QR di Householder della matrice A
4
5 [m,n] = size(A);
6 if n > m
7     error('Dimensioni errate');
8 end
9 QR = A;
10 for i = 1:n
11     alfa = norm(QR(i:m,i));
12     if alfa == 0
13         error('la matrice non ha rango massimo');
14     end
15     if QR(i,i) >= 0
16         alfa = -alfa;
17     end
18     v1 = QR(i,i) - alfa;
19     QR(i,i) = alfa;
20     QR(i+1:m,i) = QR(i+1:m,i)/v1;
21     beta = -v1/alfa;
22     v = [1; QR(i+1:m,i)];
23     QR(i:m,i+1:n) = QR(i:m,i+1:n) - (beta * v) * (v' * QR(i:m,i+1:n));
24 end
25 end
```

### 3.3 Esercizio 12

```
1 function x = qrsolve(QR, b)
2 %
3 %
4 % x = qrSolve(QR, b)
5 % risolve il sistema  $QR \cdot x = b$  nel senso dei minimi quadrati
6 %
7 [m,n] = size(QR);
8 k = length(b);
9 if k ~= m
10     error('Dati inconsistenti');
11 end
12 x=b(:);
13 for i = 1:n
14     v=[1; QR(i+1:m,i)];
15     beta = 2/(v'* v);
16     x(i:m) = x(i:m) - (beta*(v'*x(i:m))*v);
17 end
18 x=x(1:n);
19 for j = n:-1:1
20     if QR(j,j)==0
21         error('Matrice singolare');
22     end
23     x(j) = x(j) / QR(j,j);
24     x(1:j-1) = x(1:j-1) - QR(1:j-1,j)*x(j);
25 end
26 return
27 end
```

## 4 Capitolo 4

## 5 Capitoli 5/6