# Laboratorio di Calcolo Numerico
# Equazioni non lineari.

Ángeles Martínez Calomardo
http://www.math.unipd.it/~acalomar/DIDATTICA/
angeles.martinez@unipd.it

Laurea in Informatica
A.A. 2018–2019

# Metodo Newton

Il metodo di Newton richiede che

- $f$ sia derivabile con continuità su un intervallo $[a, b]$ di $\mathbb{R}$;
- $f'(x) \neq 0$ per ogni $x \in [a, b]$.

Se ben definito, il metodo di Newton genera la successione $\{x_n\}$

$$x_{n+1} := x_n - \frac{f(x_n)}{f'(x_n)} \qquad \text{con } n \geq 0.$$

# Metodo Newton: convergenza locale

Nel caso del metodo di Newton la convergenza non è in generale garantita. Esistono degli esempi in cui il metodo produce una successione non convergente. Ciò nonostante esistono molti teoremi che illustrano quando si è certi che il metodo converga.

Uno zero $\alpha$ si dice semplice se $f(\alpha) = 0$ e $f'(\alpha) \neq 0$.

**Teorema.** Sia $\alpha \in (a, b)$ uno zero semplice di $f : [a, b] \to \mathbb{R}$. Si supponga inoltre $f \in C^2([a, b])$. Allora per $x_0 \in [a, b]$ sufficientemente vicino ad $\alpha$ le iterazioni del metodo di Newton

$$x_{n+1} = x_n - f(x_n)/f'(x_n), \;\; n = 0, 1, 2, \ldots$$

sono ben definite e convergono almeno quadraticamente ad $\alpha$.

# Metodo Newton: alcuni fatti

1. Il metodo di Newton non è sempre convergente.
2. Se converge, non è detto che l'ordine di convergenza sia $p = 2$ (conv. quadratica).
3. Se uno zero $\alpha$ di $f$ non è semplice allora la convergenza non è quadratica.

CRITERIO DI ARRESTO:

Si usa un criterio di arresto basato sul valore dello scarto in quanto fornisce un'ottima approssimazione dell'errore:

$$x_{n+1} - x_n \approx e_n$$

Supponendo di volere una soluzione approssimata con una certa tolleranza `tol`, ci si ferma nel calcolo della successione dei valori, quando $|x_{n+1} - x_n| <$ `tol` per un certo valore di $n$.

# Metodo Newton: implementazione

$$x_{n+1} := x_n - \frac{f(x_n)}{f'(x_n)} \qquad \text{con } n \geq 0.$$

```
while (abs(dif) >= toll) && (n < nmax) % TEST DI ARRESTO

    fx = f(x);            % Calcola il valore f(x_n) (residuo)
    dfx = df(x);          % Calcola il valore f'(x_n)
    dif = -fx/dfx;        % Calcola il valore -f(x_n)/f'(x_n)
    x = x + dif;          % Calcola il valore x_{n+1}
    n = n+1;              % Incrementa il contatore delle iterate

end
```

# Implementazione MATLAB del metodo di Newton–Raphson

`function` newtonfun.m

Scriviamo una `function` MATLAB/OCTAVE che prende come parametri di input:

- la funzione di iterazione `f`,
- la sua derivata prima `df`,
- il punto iniziale $x_0$,
- la tolleranza `toll`,
- il massimo numero di iterazioni `nmax`.

In uscita (parametri di output) la function restituisce:

- $xv$ vettore con le approssimazioni dello zero $\alpha$ ottenute ad ogni passo
- `scarti` vettore con il valore dello scarto (differenza in valore assoluto tra due approssimazioni successive) ad ogni passo
- `iter` il numero di iterazioni a convergenza,
- `flag` variabile di controllo (se 1 la derivata si è annullata)

# Metodo Newton: implementazione

```
while (abs(dif) >= toll) && (n < nmax) && (flag ==0)

    fx = f(x);              % Calcola il valore f(x_n) (residuo)
    dfx = df(x);            % Calcola il valore f'(x_n)
    if dfx == 0
        flag = 1;
    else
        dif = -fx/dfx;      % Calcola il valore -f(x_n)/f'(x_n)
        x = x + dif;        % Calcola il valore x_{n+1}
        xv = [xv; x];       % Aggiunge al vettore la nuova iterata
        scarti = [scarti; dif];  % Aggiunge il nuovo scarto
        n = n+1;            % Incrementa il contatore delle iterate
    end

end
```

# Metodo Newton: implementazione, esempio

Quale esempio, con il file `demonewton.m`, usiamo il metodo di Newton per il calcolo di $\sqrt{2}$, cioè l'unica radice positiva di $f(x) = x^2 - 2$.

```
f=@(x) x.^2-2;
df=@(x) 2*x;
x0=1;
toll=10^(-8);
maxit=100;
[xv,scarti,iter,flag]=newtonfun(f,df,x0,toll,nmax);
```

# Esercizi proposti

Calcolare con i metodi di bisezione e Newton un'approssimazione dello zero $\alpha$ di

$$\exp(-x/4) - x$$
$$x^3 - 2 = 0$$
$$x \log(x) - 1 = 0, \qquad x > 0$$

1. Si rappresentino graficamente le funzioni e si scelga un opportuno intervallo iniziale per il metodo di bisezione, e il punto iniziale per Newton.
2. Quante iterazioni occorrono perché i metodi forniscano una approssimazione della soluzione $\alpha$ supponendo di utilizzare una tolleranza di $10^{-8}$?
3. Per ogni equazione si visualizzino ordinatamente a video i risultati ottenuti con ogni metodo e si realizzi un grafico semilogaritmico con i profili di convergenza dei due metodi, tramite il comando `semilogy`. Si faccia il plot del vettore degli scarti per Newton e del vettore dei residui per la bisezione.
4. Per ogni esercizio si scrivano ordinatamente su file i risultati ottenuti.

**N.B.** Per la terza equazione si scelga opportunamente il punto iniziale $x_0$, e si determini un intervallo $[a, b]$ tale per cui per ogni $x_0 \in [a, b]$ sia garantita la convergenza del metodo di Newton.

# Esercizio (da svolgere in Laboratorio)

Si vuole calcolare la radice `maggiore` della funzione:

$$\boxed{x^2 - 5x + 6}$$

con il metodo di Newton.

Si rappresenti graficamente la funzione e l'asse $x$ nella stessa finestra grafica (si usi il programma `disegna.m`). Dopo aver analizzato il grafico si scelga un opportuno punto iniziale per eseguire la function `newtonfun`.

Si usi lo script `newtonscript` per risolvere il problema dato, usando `toll= 1e-7`.

# Metodo Newton: scrittura a video dei risultati

Si può usare la function seguente per scrivere l'indice di iterata, le approssimazioni successive e il valore corrispondente dello scarto, come colonne di una tabella, usando per ogni valore il formato più opportuno:

```
function [] = risultati_newton(a,b,f,xv,scarti,x0)
%RISULTATI_NEWTON function per visualizzare risultati provenienti
    dal metodo
% di Newton per la ricerca degli zeri di equazioni non lineari
% Uso:
% risultati_newton(a,b,f,xv,scarti,x0)
%
xv=xv(:);
scarti=scarti(:);
n=length(scarti);
fprintf('\nf: %s \tIntervallo: a=%g b=%g Newton x0=%g\n\n', ...
formula(f),a,b,x0);
fprintf('n \t\t x_n \t\t x_n -x_n-1 \n\n');
fprintf('%d\t %20.15f \t %10.2e  \n', ...
[(1:n);xv(2:end)';scarti']);
```

# Metodo di Newton-Raphson per radici multiple

- La convergenza del metodo di Newton è **locale** cioè è garantita solo se $x_0$ è sufficientemente vicino alla soluzione.
- Inoltre, nel caso in cui $\alpha$ sia uno zero **semplice** (cioè $f'(\alpha) \neq 0$) è (almeno) quadratica ovvero:
$$|x_{n+1} - \alpha| \simeq C\,|x_n - \alpha|^2$$
l'errore al passo $n+1$ è asintoticamente proporzionale al quadrato dell'errore al passo $n$ con costante di proporzionalità $C$ (costante asintotica).
- Ma se invece **la molteplicità dello zero è maggiore di uno**, il metodo di Newton converge linearmente.

  Detta $r$ la molteplicità di $\alpha$, la convergenza quadratica può essere ripristinata utilizzando il cosiddetto metodo di **Newton modificato**:

$$\begin{array}{rcll} x_0 & & \text{fissato} & \\ x_{n+1} & = & x_n - r\dfrac{f(x_n)}{f'(x_n)}, & \qquad n \geq 0 \end{array}$$

# Metodo di Newton-Raphson per radici multiple

## Esercizio

*Si scriva la function* `newtonmod.m` *ottenuta a partire dalla function* `newtonfun.m`, *aggiungendo come ultimo parametro di ingresso, oltre a quelli della function* `newtonfun`, *la* **molteplicità della radice**, $r$, *e facendo in modo che implementi il metodo di Newton modificato per la ricerca di radici multiple.*

# Metodo di Newton-Raphson per radici multiple

## Esercizio

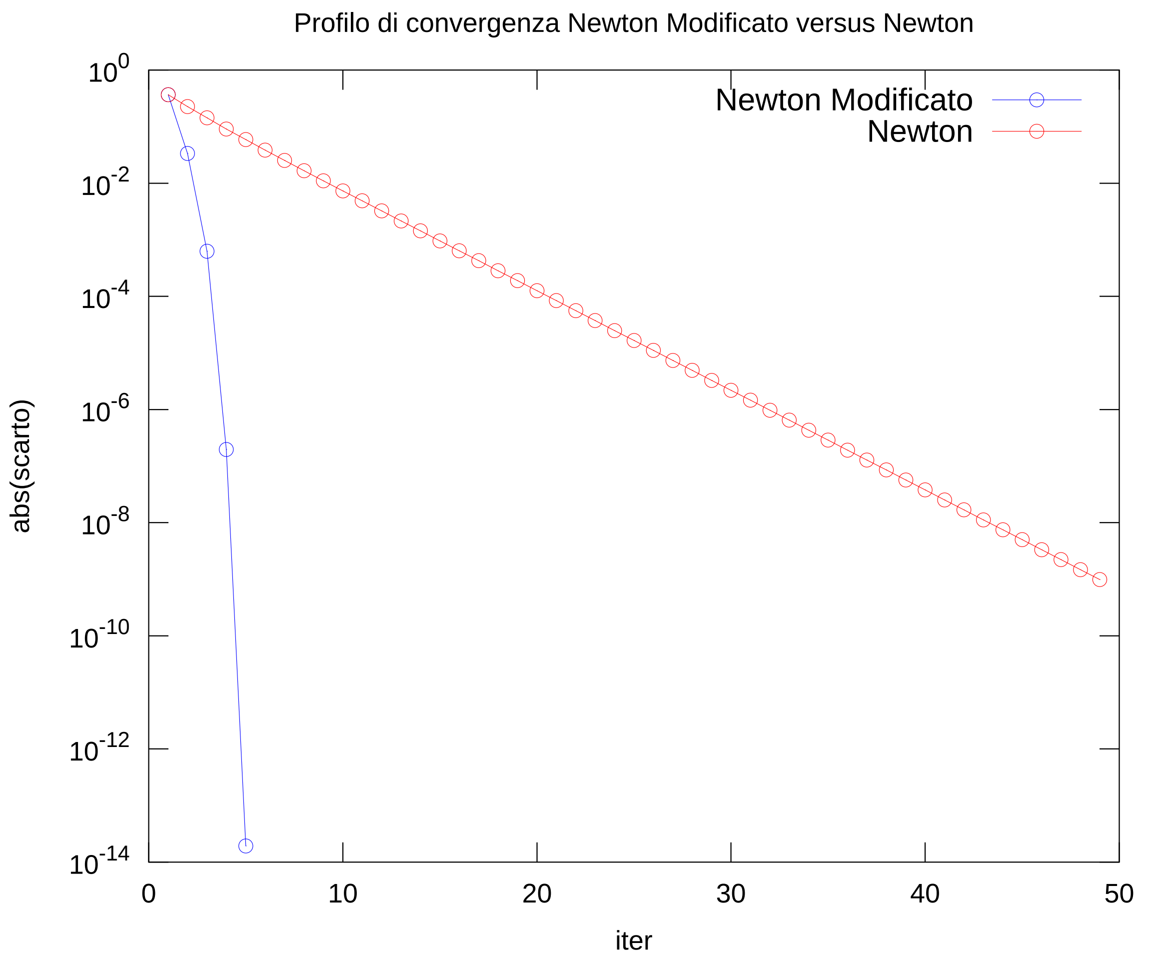
*Si scriva lo script chiamato* `scriptmod.m` *che:*

1. *usi la function* `newtonmod.m` *per approssimare lo zero della funzione* $(x-1)^2\log(x)$ *nell'intervallo* $[1,3]$, *con i valori del parametro* $r=1$ *(Newton) e* $r$ *tale per cui la convergenza è quadratica, con una tolleranza pari a* `tol=` $10^{-7}$.
2. *confronti la convergenza del Metodo di Newton nei due casi mediante un grafico semilogaritmico che rappresenti gli scarti ottenuti in entrambi i casi in funzione del numero di iterazioni. (***Si noti che il grafico deve essere unico con due curve, una per ogni valore di** $r$*).*
3. *si calcoli una stima della costante asintotica* $C$, *ottenuta come rapporto tra scarti consecutivi:*

```
asint= abs(scarti(2:n))./abs(scarti(1:n-1)).^p;
```

# Metodo di Newton-Raphson per radici multiple

Un esempio del grafico richiesto è il seguente:



# Metodo della Secante

## detto anche della secante variabile

Dati $x_0, x_1$ (fissati)

$$x_{n+1} = x_n - \frac{f(x_n)}{h_n}, \ n \geq 1$$

con

$$h_n = \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}} \approx f'(x_n)$$

Ordine e fattore di convergenza

$p = 1.618$ (convergenza superlineare)

$$C = \left| \frac{1}{2} \frac{f''(\alpha)}{f'(\alpha)} \right|^{0.618}$$

# Implementazione del metodo della secante variabile

- In questo schema iterativo si calcola $x_{n+1}$ a partire da $x_{n-1}$ e $x_n$:

$$x_{n+1} = x_n - \frac{f(x_n)}{h_n}, \; n \geq 1 \qquad \text{con} \qquad h_n = \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}$$

- **MOLTO IMPORTANTE**: Nel programma occorre salvare il valore dell'iterata $x_n$ in una variabile ausiliaria prima di sovrascrivere il valore con la nuova approssimazione $x_{n+1}$.

# Esercizio proposto

Scrivere una function che implementi il metodo della secante variabile:

$$x_{n+1} = x_n - \frac{f(x_n)}{h_n}, \quad \text{dove} \quad h_n = \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}, \quad n = 1, 2, \ldots,$$

Più in dettaglio:

Si scriva la function `secvar.m` che abbia come dati di ingresso la funzione $f$, $x_0$, e $x_1$, la tolleranza `toll` e il numero massimo di iterate permesse, `nmax` e restituisca in uscita il numero di iterazioni impiegate `iter`, il vettore `xv` con le iterate $x_n$, $n = 0, \ldots, iter + 1$ e il vettore degli scarti $s_n = x_n - x_{n-1}$, $n = 1, \ldots, iter + 1$.
Si usi il test di arresto sullo scarto.

La sintassi della function deve essere la seguente:

```
function [xv,scarti,iter]=secvar(f,x0,x1,toll,nmax)
```

# Esercizio

## Esercizio

Si scriva uno script che, chiamando la function `secvar.m`, risolva l'equazione $f(x) = 0$ dove

$$f(x) = e^{-x} + \cos(x) - 3,$$

con punti iniziali $x_0 = -1, x_1 = 0$, numero massimo di iterazioni `itmax = ` $100$ e tolleranza `tol` $= 10^{-8}$.

Si confronti l'approssimazione restituita dalla function `secvar.m` con il valore approssimato con il comando MATLAB `fzero(f,-1)`.