Laboratorio di Analisi Numerica
Equazioni non lineari.

Ángeles Martínez Calomardo
amartinez@units.it

Laurea Triennale in Intelligenza Artificiale e Data Analytics
A.A. 2024–2025

Equazioni non lineari

Problema Risolvere $f(x) = 0$ con $x \in [a, b] \subseteq \mathbb{R}$, $f \in C([a, b])$.

- Esempio 1: equazioni polinomiali $p_N(x) = 0$ con p_N polinomio di grado N. Possibili problemi (nessuna soluzione in $\mathbb{R}$, soluzioni multiple).
- Esempio 2: $f(x) = \sin(x) - x$. Soluzione unica poichè f decrescente (vedi derivata).

Soluzione: Metodi iterativi, generano una successione $\{x_k\}_{k \in \mathbb{N}}$ che si desidera convergere ad α tale che $f(\alpha) = 0$.

Metodi iterativi per risolvere $f(x) = 0$

- **Ordine di convergenza** di un metodo iterativo:

  sia $\{x_k\}$ una successione convergente ad α e sia $e_k = x_k - \alpha$ l'errore al passo k. Se esiste un numero $p > 0$ e una costante $C \neq 0$ tale che

$$\lim_{k \to \infty} \frac{|e_{k+1}|}{|e_k|^p} = C$$

  allora p è chiamato *ordine di convergenza* della successione e C è la *costante asintotica di errore*.

  Per $p = 1$ la convergenza di dice **lineare**, per $p = 2$ si dice **quadratica**.

Metodo di bisezione

Sia $f : [a, b] \to \mathbb{R}$ una funzione continua e supponiamo $f(a) \cdot f(b) < 0$.

Il metodo di bisezione genera una successione di intervalli (a_k,b_k) con

- $f(a_k) \cdot f(b_k) < 0$,
- $[a_k, b_k] \subset [a_{k-1}, b_{k-1}]$,
- $|b_k - a_k| = \frac{1}{2}|b_{k-1} - a_{k-1}|$.

Fissata una tolleranza ϵ, si arresta l'algoritmo quando:

- la semilunghezza dell'intervallo corrente è minore della tolleranza

$$\frac{1}{2}\,|b_k - a_k| \leq \epsilon$$

- oppure usando il test sul residuo (non sempre affidabile)

$$\left| f\left(\frac{a_k + b_k}{2}\right)\right| \leq \epsilon.$$

Metodo bisezione: alcuni fatti

- Non ha un ordine di convergenza nel senso della definizione vista prima. Esempio: se applico bisezione per risolvere $\sin(x) - x = 0$, in $[a, b] = [-3, 2]$ la successione $|e_{n+1}/e_n|$ alterna valori 1.5 e $1/6$ e quindi non converge con ordine 1.
- Usa solo il segno della funzione.
- Se $f(a) \cdot f(b) < 0$ e $f \in C([a, b])$ allora converge (sempre!!) a un α tale che $f(\alpha) = 0$.
- Fissata una tolleranza ϵ, e due punti iniziali a, b tali che $f(a) \cdot f(b) < 0$ (con $f \in C([a, b])$) per avere un'errore assoluto $|x_n - \alpha|$ sulla soluzione α inferiore ad ϵ occorrono al più

$$n = \left\lceil \log_2 \left(\frac{b-a}{\epsilon} \right) \right\rceil - 1$$

iterazioni del metodo.

Bisezione: criterio arresto

Sia c un'approssimazione dello zero e si supponga di doverlo determinare a meno di una tolleranza `tol`.

Il criterio del residuo $|f(c)| < \texttt{tol}$ non è spesso accettabile:

- funzione piatta: si pensi a risolvere l'equazione $f(x) = 0$ con $f(x) = 10^{-50} \cdot x$; il residuo $|f(1)| = 10^{-50}$ ma 1 è molto distante da $\alpha = 0$.
- funzione ripida: si pensi a risolvere l'equazione $f(x) = 10^{50} \cdot x = 0$; il residuo $|f(10^{-20})| = 10^{30}$ seppure 10^{-20} sia molto vicino a $\alpha = 0$.

Residuo pesato

Siano $a < b$ e $c = (a+b)/2$. Diciamo residuo pesato $|f(c) \cdot w|$ con

$$w := \frac{1}{f'(c)}.$$

- Il residuo pesato intende fornire **una buona approssimazione dell'errore**.
- Infatti, dalla formula di Taylor centrata in c con passo $e = \alpha - c$:

$$0 = f(\alpha) = f(c+e) = f(c) + f'(c)e + o(e)$$

da cui, trascurando i termini di ordine superiore e prendendo i moduli:

$$|e| \approx \left| \frac{f(c)}{f'(c)} \right|.$$

- Vediamo che l'errore (in modulo) si può approssimare con il residuo pesato con il reciproco della derivata nel punto c.
- Nel caso della bisezione, $f'(c)$ non è noto e si approssima come

$$f'(c) \approx \frac{f(b) - f(a)}{b - a}$$

Bisezione: criterio arresto

Arrestando il metodo in base al valore del residuo pesato $|f(c) \cdot w|$ con

$$w := \left(\frac{f(b) - f(a)}{b - a} \right)^{-1}$$

avremmo:

- funzione piatta: per risolvere l'equazione $f(x) = 0$ con $f(x) = 10^{-50} \cdot x$; w è grande ($w = 10^{50}$) e quindi $|f(1)| = 10^{-50}$ ma $|f(1)w| = 1$;

- funzione ripida: per risolvere l'equazione $f(x) = 0$ con $f(x) = 10^{50} \cdot x$; w è piccolo ($w = 10^{-50}$) e quindi $|f(10^{-20})| = 10^{30}$ ma $|f(10^{-20})w| = 10^{30} \cdot 10^{-50} = 10^{-20}$

Per f più generali, il test del residuo pesato $|f(c)w| < \texttt{tol}$ prova ad adattare situazioni in cui il test del residuo $|f(c)| < \texttt{tol}$ non sia affidabile.

Metodo bisezione: implementazione in MATLAB/OCTAVE

```
function [vc,k,semilunghezza,residuopesato]=bisezione(a,b,
    tolintv,tolres,maxit,f)

if b < a
    s=b; b=a; a=s;
end
fa=feval(f,a); fb=feval(f,b);
if fa == 0
    c=a; k=0; semilunghezza=(b-a)/2; residuopesato=0;
    return;
end
if fb == 0
    c=b; k=0; semilunghezza=(b-a)/2; residuopesato=0;
    return;
end
```

Metodo bisezione: implementazione in MATLAB/OCTAVE

```
for index=1:maxit
%  calcola nuovo punto medio e il valore di f in esso
   c=(a+b)/2; fc=feval(f,c);
%  calcola semilunghezza del nuovo intervallo
   semilun=(b-a)/2; den=(fb-fa);
   if den == 0
       den=eps;
   end
%  calcola residuo pesato
   w=(b-a)/den; wres=abs(fc*w);
%  salva i dati (punto medio, semilunghezza, residuo pesato) nei vettori
     corrispondenti
   vc=[vc;c];
   semilunghezza=[semilunghezza;semilun];
   residuopesato=[residuopesato;wres];
%  effettua TEST DI ARRESTO
   if (wres < tolres) | ...
      (semilun < tolintv) | (fc == 0)
      k=index; fprintf('\n'); return;
   end
% calcola il nuovo intervallo di lunghezza dimezzata
   if sign(fc) == sign(fa)
         a=c; fa=fc;
             else
         b=c; fb=fc;
   end
end
k=maxit; fprintf('\n');
```

Bisezione in Matlab/Octave: demobisezione.m

Salviamo in demobisezione.m il seguente file

```
f=@(x) x.^2-2; a=1; b=2;
tolres=10^(-15);tolintv=10^(-15);maxit=100;
[c,iter,sl,wr]=bisezione(a,b,tolintv,tolres,maxit,f);
for k=1:iter
fprintf('\n [k]:%3.0f [c]: %15.15f [AMP]: %5.2e [WRES]:%5.2e
    ',k,c(k),sl(k),wr(k))
end
fprintf('\n');
```

dove:

- a,b: intervallo iniziale bisezione (input);
- tolres,tolintv,maxit: tolleranze bisezione e massimo numero di iterazioni(input);
- f: passaggio di funzione come variabile (input);
- c: vettore delle approssimazioni successive (punti medi)(output);
- iter: numero di iterazioni di bisezione (numero di punti medi calcolati) (output);
- sl,wr: vettori delle semilunghezze e dei residui pesati (output);

Esercizio

Si scriva uno script MATLAB per risolvere l'equazione:

$$\boxed{5x = \exp(-x)}$$

con il metodo di bisezione. Si usino tolleranze pari a 10^{-8}.

Prima di eseguire la `function bisezione` si rappresenti graficamente la funzione e l'asse x nella stessa finestra grafica, e, dopo aver individuato la posizione della radice dal grafico, si scelga un opportuno intervallo iniziale per il metodo di bisezione.

Si confronti il numero di iterazioni effettivamente eseguite dal metodo con il numero minimo di iterazioni necessarie stimato mediante

$$n = \left\lceil \log_2 \left(\frac{b-a}{\epsilon} \right) \right\rceil - 1$$

Creazione di un profilo di convergenza

Una volta eseguita la function `bisezione.m`:

```
[c,iter,semilun,res]=bisezione(a,b,tolint,tolres,maxit,f);
```

per rappresentare graficamente l'evoluzione dei residui (pesati) si possono usare, ad esempio, le seguenti istruzioni:

```
figure(2)
semilogy(1:iter,abs(res),'m-*');
title('Profilo di convergenza Bisezione ( caso f(x)=5*x - exp
    (-x))');
xlabel('N. iterazioni');
ylabel('residuo pesato');
legend('bisezione');
print -dpdf grafico1.pdf
```

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 $\{x_k\}$

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

Metodo Newton: convergenza locale

Nel caso del metodo di Newton la convergenza non è in generale garantita.

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

Convergenza. 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 α le iterazioni del metodo di Newton

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}, \quad k = 0, 1, 2, \ldots$$

sono ben definite e convergono almeno quadraticamente ad α.

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 sempre $p = 2$ (conv. quadratica).
3. Se uno zero α 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 n quanto fornisce un'ottima approssimazione dell'errore:

$$x_{k+1} - x_k \approx e_k$$

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

Metodo Newton: implementazione

```
dif=tol+1;
iter=0;
xold=x0;
while (abs(dif) > tol) & (iter < itmax)
    fx=feval(f,xold);
    dfx=feval(df,xold);
    iter=iter+1;
    dif=-fx/dfx;
    xnew=xold+dif
    xold=xnew;
end
```

Implementazione MATLAB del metodo di Newton–Raphson

`function` newton.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 `tol`,
- il massimo numero di iterazioni.

In uscita (parametri di output) la function restituisce:

- x vettore con le approssimazioni dello zero α 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,

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;
tol=10^(-8);
maxit=100;
[x,iter,scarti]=newton(f,df,x0,tol,maxit);
```

Esercizio

Si vuole calcolare la radice `maggiore` della funzione:

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

con il metodo di Newton. Si usi tolleranza pari a 10^{-8}.

Si scriva uno script MATLAB che rappresenti graficamente la funzione e l'asse x nella stessa finestra grafica. Dopo aver analizzato il grafico si scelga un opportuno punto iniziale per eseguire la function `newton`.

Si aggiungano allo script le istruzioni necessarie per creare il grafico con il profilo di convergenza del metodo, usando il vettore degli scarti:

```
semilogy(1:iter,abs(scarti),'m-*');
```

Esercizi proposti

Calcolare con i metodi di bisezione e Newton un'approssimazione dello zero α 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 α 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 pesati 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.

Convergenza del Metodo di Newton-Raphson

- La convergenza del metodo di Newton è innanzitutto **locale** cioè è garantita solo se x_0 è sufficientemente vicino alla soluzione.

- Inoltre, nel caso in cui α sia uno zero **semplice** (cioè $f'(\alpha) \neq 0$) è (almeno) quadratica ovvero:

$$|x_{k+1} - \alpha| \simeq C\,|x_k - \alpha|^2$$

  Questa proprietà rende il metodo di Newton assai veloce e, a dispetto della sua età (circa 365 anni), il più usato per equazioni scalari.

- Se invece **la molteplicità dello zero è maggiore di uno**, il metodo di Newton converge **linearmente**.

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

$$\begin{aligned} x_0 \quad &\text{fissato} \\ x_{k+1} \quad = \quad & x_k - r\frac{f(x_k)}{f'(x_k)}, \qquad k \geq 0 \end{aligned}$$

Metodo di Newton-Raphson per radici multiple

Esercizio

Si scriva la function `newtonmod.m` *ottenuta a partire dalla function* `newton.m`, *aggiungendo come ultimo parametro di ingresso, oltre a quelli della function* `newton`, *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

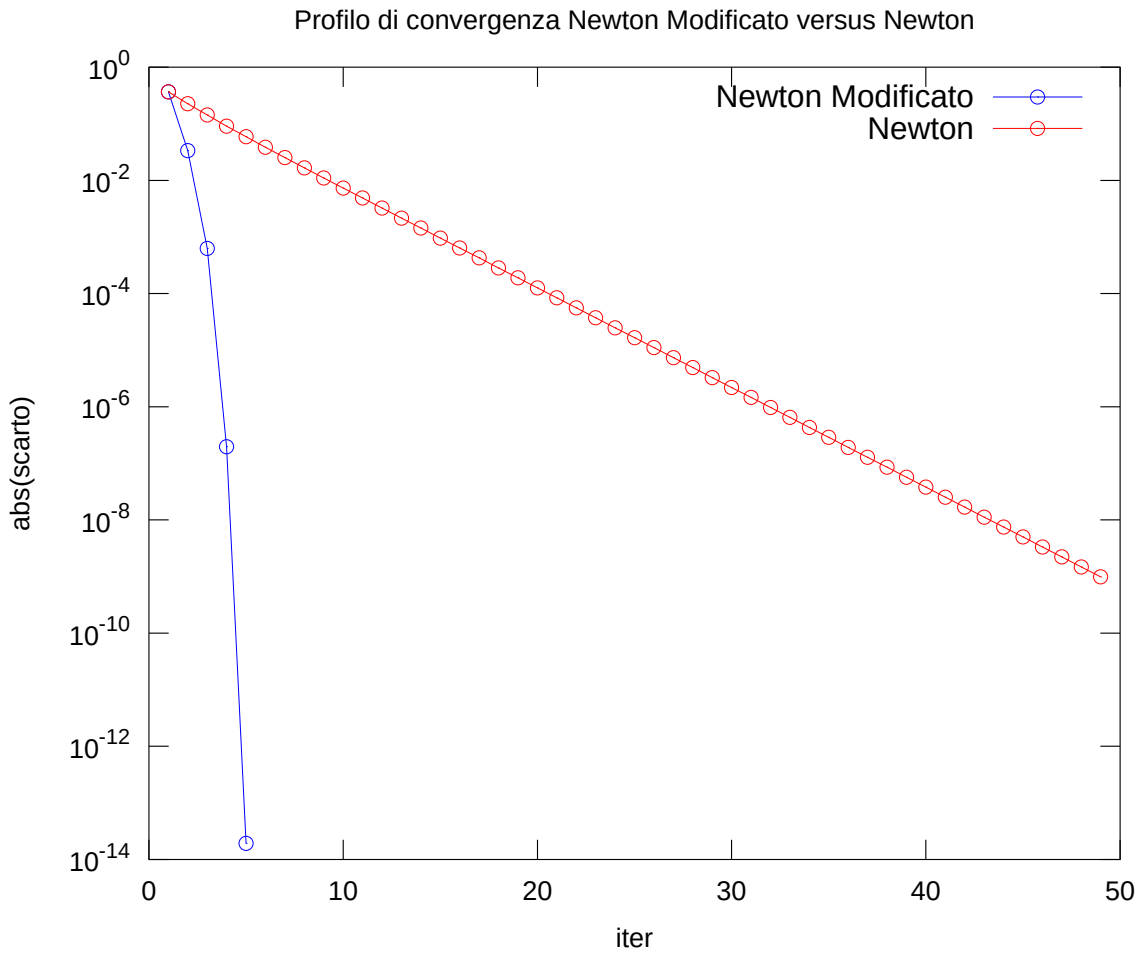
Esercizio

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^{-9}.
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*).*

Metodo di Newton-Raphson per radici multiple

Un esempio del grafico richiesto è il seguente:



Metodo della tangente fissa

Si copi il file contenente la function `newton.m` con il nome `tfissa.m` e si modifichi opportunamente in modo che implementi il metodo della tangente fissa:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_0)}, \qquad k = 0, 1, \ldots, \qquad x_0 \text{ fissato}$$

Tale function deve avere come **dati di ingresso**:

- la funzione f la sua derivata prima df, x_0, `tol` e `itmax`

e restituire come **parametri di output**:

- il numero di iterazioni impiegate `iter`, il **vettore** con le iterate $x_k,\ k = 0, \ldots, iter$ e il **vettore** degli scarti $s_k = x_k - x_{k-1},\ k = 1, \ldots, iter$.

Come test di arresto si usi quello basato sullo scarto.

Metodo della tangente fissa

Esercizio

Si scriva uno script chiamato scripttfissa.m che chiamando le due function (`newton.m` e `tfissa.m`) risolva con i due metodi l'equazione $f(x) = 0$ *dove*

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

e con:

- $x_0 = -1$
- `itmax` $= 100$
- `tol` $= 10^{-9}$.

Creare un (UNICO) grafico semilogaritmico con il profilo di convergenza dei due metodi.
Il grafico può essere salvato tramite il comando `print` nel file `graficotf.pdf`.

```
print -dpdf graficotf.pdf
```

Metodo delle secanti

detto anche della secante variabile

Dati x_0, x_1 (fissati)

$$x_{k+1} = x_k - \frac{f(x_k)}{h_k}, \ k \geq 1$$

con

$$h_k = \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}} \approx f'(x_k)$$

Ordine e fattore di convergenza

$$p = 1.618 \text{ (convergenza superlineare)}$$

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

Implementazione del metodo della secante variabile

- In questo schema iterativo si calcola x_{k+1} a partire da x_{k-1} e x_k:

$$x_{k+1} = x_k - \frac{f(x_k)}{h_k},\ k \geq 1 \qquad \text{con} \qquad h_k = \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}$$

- Nel programma servono quindi tre variabili:
  `xoldold` (x_{k-1}), `xold` (x_k) e `xnew` (x_{k+1}).

- **MOLTO IMPORTANTE**: Occorre aggiornare le variabili `xoldold` e `xold` prima di passare all'iterazione successiva:

```
 iter  xoldold   xold     xnew
                   ↓        ↓
iter+1          xoldold   xold   xnew
                            ↓      ↓
iter+2                   xoldold xold xnew
  ⋮                                 ⋮    ⋮
```

Esercizio proposto

Scrivere una function che implementi il metodo delle secanti:

$$x_{k+1} = x_k - \frac{f(x_k)}{h_k}, \quad \text{dove} \quad h_k = \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}, \quad k = 1, 2, \ldots,$$

Più in dettaglio:

Si scriva la function `secvariabile.m` che abbia come dati di ingresso la funzione $f(x)$, x_0, `itmax` e `tol` e restituisca in uscita il numero di iterazioni impiegate `iter`, il vettore con le iterate $x_k,\ k = 0, \ldots, iter + 1$ e il vettore degli scarti $s_k = x_k - x_{k-1},\ k = 1, \ldots, iter + 1$.
Si usi il test di arresto sullo scarto.

Esercizio

Si scriva uno script che, chiamando tale function risolva l'equazione $f(x) = 0$ dove

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

con punti iniziali $x_0 = -1.5, x_1 = -1$, numero massimo di iterazioni `itmax` = 100 e tolleranza `tol` $= 10^{-10}$.

Metodo di Punto Fisso

Fissato un punto iniziale x_0

$$\begin{aligned} x_1 &= g(x_0) \\ x_2 &= g(x_1) \\ x_3 &= g(x_2) \\ &\vdots \\ x_{k+1} &= g(x_k) \end{aligned}$$

Il criterio di arresto del ciclo iterativo è basato sul valore assoluto dello scarto (differenza tra due iterate succesive):

$$|x_{k+1} - x_k| < tol$$

dove tol è una quantità prefissata, piccola, ma di qualche ordine di grandezza superiore alla precisione di macchina.

Metodo di punto fisso per calcolare uno zero di f

Il metodo iterativo di punto fisso è un metodo per calcolare uno zero di una funzione $f(x)$ in quanto basta porre

$$g(x) = x + f(x)$$

per avere

$$g(x) = x \iff f(x) = 0.$$

Questo non è l'unico modo di determinare un metodo di punto fisso per risolvere $f(x) = 0$. Infatti a partire dall'equazione $f(x) = 0$ si possono trovare diverse funzioni di iterazione non tutte necessariamente convergenti alla radice α.

Ordine di convergenza e costante asintotica dell'errore

Dato uno schema iterativo si ha:

$$\lim_{k\to\infty} \frac{|x_{k+1} - \alpha|}{|x_k - \alpha|^p} = C$$

p (≥ 1)è l'ordine di convergenza del metodo iterativo

C (> 0) è la costante asintotica dell'errore o fattore di convergenza.

Si ha

$$|x_{k+1} - \alpha| \simeq C\,|x_k - \alpha|^p$$

L'errore decresce più rapidamente quanto più grande è p.

La convergenza del metodo iterativo del punto fisso è lineare $(p = 1)$ se

$$C = |g'(\alpha)| < 1 \qquad \text{e} \qquad C \neq 0$$

dove g è la funzione della quale si cerca il punto fisso α.

Implementazione MATLAB del metodo di punto fisso

`function` pfisso.m

Si scriva una `function` MATLAB/OCTAVE che prenda come parametri di input:

- la funzione di iterazione g,
- il punto iniziale x_0,
- la tolleranza `toll`,
- il massimo numero di iterazioni, `maxit`.

In uscita (parametri di output) la function restituisce:

- `xvec` vettore con le approssimazioni calcolate del punto fisso α
- `iter` il numero di iterazioni a convergenza,
- `scarti` vettore degli scarti

Stima della costante asintotica C

Dopo aver eseguito la function `pfisso.m` è possibile ottenere una stima della costante asintotica C, calcolata come rapporto tra scarti consecutivi:

```
asint1= abs(scarti(2:iter)./scarti(1:iter-1));
```

È inoltre possibile ottenere un'approssimazione della costante asintotica usando la formula $C = |g'(\alpha)|$, ovvero come valore assoluto della derivata prima della funzione di iterazione nelle approssimazioni del punto fisso:

```
asint2= abs(feval(dg,xvec));
```

dove `dg` è la derivata prima della funzione di punto fisso, g.

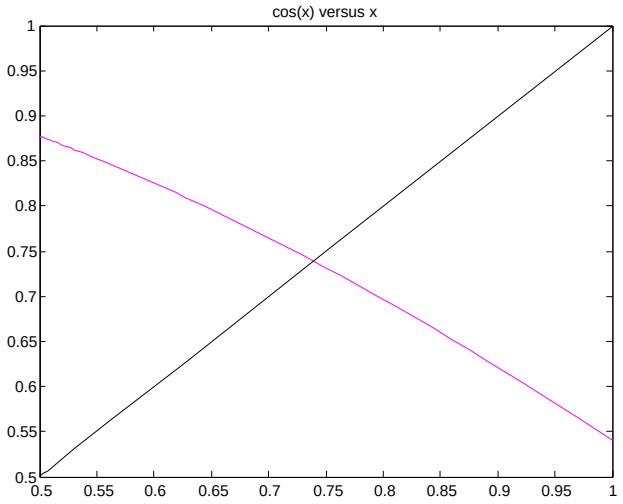
Rappresentazione grafica della funzione di punto fisso

Si consideri l'equazione $x - cos(x) = 0$.

Usando le istruzioni MATLAB/OCTAVE seguenti

```
g=@(x) cos(x);
bis=@(x) x;
fplot(g,[0.5,1],'m-'); hold on;
fplot(bis,[0.5,1],'k-');hold off;
title('cos(x) versus x');
```

otteniamo il grafico mostrato dove si vede que effettivamente l'intersezione tra x ed $\cos(x)$ si trova nell'intervallo $[0.5, 1.0]$ e que il punto fisso è unico.



Esecuzione della function pfisso

Anche se possiamo eseguire la function dalla command window di MATLAB :

```
[xvec,iter,scarti]=pfisso(g,0.1,1e-12,100)
```

è meglio creare uno script con le opportune istruzioni per la definizione della funzione di iterazione g, l'inizializzazione dei parametri di input, e la chiamata alla function `pfisso`.

```
x0=0.1; toll=1.e-12; itmax=100;
[xvec,iter,scarti]=pfisso(g,x0,toll,itmax)
```

Per questi valori dei parametri di input il programma restituisce:

```
xvec(iter)=0.7391
iter=70
abs(scarto(iter))=9.4613e-13
```

E dopo aver definito la derivata prima della funzione di iterazione, ed stimato la costante asintotica nei due modi indicati prima, si ha anche:

```
asint1(end)= 0.6735
asint2(end)= 0.6736
```

Scrittura su file dei dati di output di un programma

Esempio per la function `pfisso.m`

Una volta eseguita la function `pfisso.m`

```
[xvec,iter,scarti]=pfisso(g,x0,toll,itmax)
```

si hanno in memoria i vettori `xvec` (contenente la successione di approssimazioni) e `scarti` (contenente gli scarti calcolati ad ogni iterazione):

Per scrivere sul file *rispfisso.txt* questi vettori visualizzati come colonne di una tabella si deve creare una matrice, A, che abbia **tali vettori come righe**, e poi scriverla a file usando il comando `fprintf`.

```
% scrive dati su file
fid=fopen('rispfisso.txt','w');
fprintf(fid,'   %6s %16s %28s  \n', 'iter', 'xk', 'scarto');
it=[1:1:iter]; %creo il vettore delle iterazioni
A=[it; xvec(2:end)'; abs(scarti(1:end))'];
fprintf(fid,'\n%6d %18.14f %20.4e ', A);
fprintf(fid,'\n \n');
fclose(fid);
```

Esercizio

Esercizio

Si scriva uno script MATLAB *che chiamando le due function* `newton.m` *e* `pfisso.m` *risolva con i due metodi l'equazione*

$$x = cos(x)$$

partendo da un opportuno punto iniziale x_0 *e con* `itmax` $= 100$ *e* `tol` $= 10^{-10}$.

Creare un (UNICO) grafico semilogaritmico con il profilo di convergenza dei due metodi.

Il grafico può essere salvato tramite il comando `print` nel file `confronto.pdf` con l'istruzione:

```
print -dpdf confronto.pdf
```

Esercizio

Esercizio

Si risolva l'equazione $x = 1 + \arctan(x)$ *mediante il metodo iterativo di punto fisso.*

1. *Si determini graficamente l'intervallo contenente la soluzione* α.
2. *Si approssimi* α *usando la function* `pfisso` *con una tolleranza* $tol = 1e-12$, *a partire da un opportuno punto iniziale* x_0.
3. *Si scrivano ordinatamente su file i risultati ottenuti e si riporti in un grafico semilogaritmico il profilo di convergenza del metodo.*

Esercitazione obbligatoria

Esercizio

L'equazione

$$\frac{1}{\sqrt{x}} = -2\log_{10}\left(\frac{e}{3.51d} + \frac{2.52}{N_R\sqrt{x}}\right)$$

è detta di Colebrook-White e si usa per determinare il fattore di frizione x, per flussi turbolenti in tubi lisci o ruvidi. Tale equazione è non lineare e dipende dai parametri:

- *e scabrezza del tubo (in metri)*
- *d diametro del tubo (in metri)*
- *N_R numero di Reynolds*

Si scriva uno script che risolva con il metodo di punto fisso e con il metodo della secante variabile l'equazione data per la seguente combinazione dei parametri:

Caso test $e = 10^{-2}$; $d = 0.20$; $N_R = 10^4$.

Si utilizzi un test di uscita sullo scarto assoluto con tolleranza $TOL = 10^{-8}$.
Lo script deve produrre un grafico **semilogaritmico** *con i profili di convergenza di entrambi i metodi.*