ANALISI NUMERICA
CALCOLO NUMERICO
(A.A. 2012-2013)

Prof. F. Pitolli

Appunti delle lezioni sui metodi per la soluzione di equazioni e sistemi di equazioni non lineari

Problema 1

La **pressione** richiesta per affondare nella sabbia un oggetto largo e pesante può essere **predetta** misurando la pressione richiesta per affondare oggetti più piccoli nello stesso tipo di terreno.
La pressione p richiesta per affondare fino alla profondità d un **piatto circolare** di raggio r può essere approssimata da un'equazione del tipo

$$p(r) = k_1 e^{k_2 r} + k_3 r,$$

dove k_1, $k_2 > 0$ e k_3 sono costanti che dipendono da d e dalla compattezza della sabbia, ma non da r.
Dalle **misure** effettuate è noto che per affondare di **30 cm** nella sabbia bagnata un piatto di raggio **2.5 cm** è richiesta una pressione di **0.78 kg/cm^2**, per affondare un piatto di raggio **5 cm** è richiesta una pressione di **0.93 kg/cm^2** e per affondare un piatto di raggio **7 cm** è richiesta una pressione di **1.16 kg/cm^2**, **determinare** i valori k_1, k_2 e k_3. Si assuma che lo strato sabbioso sia più profondo di 30 cm.
Usare i valori di k_1, k_2 e k_3 trovati per **predire** la dimensione minima richiesta affinché un piatto circolare possa sostenere un peso di **250 kg** in modo da non affondare più di **30 cm**.

Dati del problema:

piatto	raggio (cm)	pressione (kg/cm^2)	profondità (cm)
1	r_1=2.5	p_1=0.78	d=30
2	r_2=5	p_2=0.93	d=30
3	r_3=7	p_3=1.16	d=30

Modello matematico: $p(r) = k_1 e^{k_2 r} + k_3 r$

Primo piatto: $p_1 = k_1 e^{k_2 r_1} + k_3 r_1$
Secondo piatto: $p_2 = k_1 e^{k_2 r_2} + k_3 r_2$
Terzo piatto: $p_3 = k_1 e^{k_2 r_3} + k_3 r_3$

Per trovare k_1, k_2 e k_3 dalle misure effettuate bisogna risolvere il **sistema non lineare**

$$\begin{cases} 0.78 - k_1 e^{2.5\,k_2} - 2.5\,k_3 = 0 \\ 0.93 - k_1 e^{5\,k_2} - 5\,k_3 = 0 \\ 1.16 - k_1 e^{7\,k_2} - 7\,k_3 = 0 \end{cases}$$

Per **predire** il valore del raggio minimo r una volta noti k_1, k_2 e k_3, bisogna risovere l'**equazione non lineare** $\frac{250}{\pi r^2} = k_1 e^{k_2 r} + k_3 r$

Problema 2

Il problema di due specie che competono per la stessa quantità di cibo può essere descritto dal sistema di equazioni differenziali

$$\begin{cases} x'(t) = x(t)[2 - 0.0002\,y(t) - 0.0001\,x(t)] \\ y'(t) = y(t)[4 - 0.0003\,y(t) - 0.0004\,x(t)] \end{cases}$$

dove $x(t)$ e $y(t)$ rappresentano le popolazioni delle due specie al tempo t.

Trovare i valori di **equilibrio** delle due specie.

Soluzione

Si devono trovare i valori di $x(t)$ e $y(t)$ che risolvono **simultaneamente** le equazioni

$$\begin{cases} x'(t) = 0 \\ y'(t) = 0 \end{cases} \Rightarrow \begin{cases} x' = x[2 - 0.0002\,y - 0.0001\,x] = 0 \\ y' = y[4 - 0.0003\,y - 0.0004\,x] = 0 \end{cases}$$

Si tratta quindi di risolvere il **sistema non lineare**

$$\begin{cases} f(x,y) = 2\,x - 0.0002\,x\,y - 0.0001\,x^2 = 0 \\ g(x,y) = 4\,y - 0.0003\,y^2 - 0.0004\,x\,y = 0 \end{cases}$$

Nota: le soluzioni $x = y = 0$, $x = 0$ e $y = 13{\cdot}333$, $x = 20{\cdot}000$ e $y = 0$ sono **soluzioni banali** (una o tutte e due le specie si sono estinte)

Problema 3

La **crescita di una popolazione** può essere modellata, su un periodo di tempo piccolo, assumendo che la popolazione abbia un tasso di crescita proporzionale al numero di individui presenti in ciascun istante. Se $N(t)$ indica il **numero di individui** al tempo t e λ è il **fattore di crescita** della popolazione, allora $N(t)$ soddisfa l'**equazione differenziale**

$$\frac{dN(t)}{dt} = \lambda N(t).$$

La **soluzione analitica** di questa equazione è $N(t) = N_0 e^{\lambda t}$, dove N_0 indica la popolazione iniziale.

Questo modello è valido solo quando la popolazione è isolata e non c'è immigrazione dall'esterno. Se si suppone che ci sia una **immigrazione a un tasso costante** ν, il modello differenziale diventa

$$\frac{dN(t)}{dt} = \lambda N(t) + \nu$$

la cui **soluzione analitica** è $N(t) = N_0 e^{\lambda t} + \frac{\nu}{\lambda}(e^{\lambda t} - 1)$.

Supponendo che la popolazione iniziale sia di **un milione di individui**, che la comunità cresca di $435{\cdot}000$ **immigrati** il primo anno e che $1{\cdot}564{\cdot}000$ **individui** siano presenti alla fine del **primo anno**, **determinare** il tasso di crescita λ della popolazione.

Dati del problema:

individui iniziali:	$N_0 = 1{\cdot}000{\cdot}000$
individui dopo un anno:	$N(1\,anno) = 1{\cdot}564{\cdot}000$
tasso di immigrazione:	$\nu = 435{\cdot}000$

Modello matematico: $N(1\,anno) = N_0\,e^{\lambda} + \frac{\nu}{\lambda}(e^{\lambda} - 1)$

Per trovare λ bisogna risolvere l'**equazione non lineare**

$$e^{\lambda} + \frac{0.435}{\lambda}(e^{\lambda} - 1) - 1.564 = 0$$

Nota: la popolazione è espressa in milioni

Sistemi di equazioni non lineari

Un **sistema di equazioni non lineari** può essere scritto nella forma

$$\begin{cases} f_1(x_1, x_2, ..., x_n) = 0 \\ f_2(x_1, x_2, ..., x_n) = 0 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ f_n(x_1, x_2, ..., x_n) = 0 \end{cases}$$

Supporremo che le **funzioni** $f_i : D \subseteq \mathbf{R}^n \to \mathbf{R}$, $i = 1, \ldots, n$, siano almeno **continue** in D.

Le **soluzioni** del sistema sono i vettori $\Xi = [\xi_1, \xi_2, \ldots, \xi_n]^T$ che **annullano simultaneamente** tutte le n equazioni.

Sistemi non lineari

$F(X) = \mathbf{0}$
$X, F \in \mathbf{R}^n$

dove $X = [x_1, x_2, ..., x_n]^T$
$F = [f_1(X), f_2(X), ..., f_n(X)]^T$
$\mathbf{0} = [0, 0, ..., 0]^T$

$$\begin{cases} f_1(x_1, x_2, ..., x_n) = 0 \\ f_2(x_1, x_2, ..., x_n) = 0 \\ ..... \\ ..... \\ f_n(x_1, x_2, ..., x_n) = 0 \end{cases}$$

Metodo del punto unito $\mathbf{R}^n$

Sistemi non lineari

$$F(X) = \mathbf{0} \Leftrightarrow X = \mathbf{\Phi}(X)$$

con $\mathbf{\Phi} = [\varphi_1(X), \varphi_2(X), ..., \varphi_n(X)]^T$

Se $\mathbf{\Xi} = [\xi_1, \ldots, \xi_n]^T \in \mathbf{R}^n$ è **radice** del sistema non lineare, allora è **punto unito** di $\mathbf{\Phi}$:

$$\boxed{F(\mathbf{\Xi}) = \mathbf{0} \Leftrightarrow \mathbf{\Xi} = \mathbf{\Phi}(\mathbf{\Xi})}$$

Metodo del punto unito in $\mathbf{R}^n$

Problema di punto unito

$$X = \mathbf{\Phi}(X) \Leftrightarrow \begin{cases} x_1 = \varphi_1(x_1, x_2, ..., x_n) \\ x_2 = \varphi_2(x_1, x_2, ..., x_n) \\ \quad ..... \\ x_n = \varphi_n(x_1, x_2, ..., x_n) \end{cases}$$

Per $\mathbf{\Xi} = [\xi_1, \ldots, \xi_n]^T$, **punto unito** della trasformazione $\mathbf{\Phi}$, si ha

$$\mathbf{\Xi} = \mathbf{\Phi}(\mathbf{\Xi}) \Leftrightarrow \begin{cases} \xi_1 = \varphi_1(\xi_1, \xi_2, ..., \xi_n) \\ \xi_2 = \varphi_2(\xi_1, \xi_2, ..., \xi_n) \\ \quad ..... \\ \xi_n = \varphi_n(\xi_1, \xi_2, ..., \xi_n) \end{cases}$$

Metodi iterativi a un punto

Il **punto unito** $\mathbf{\Xi} = \mathbf{\Phi}(\mathbf{\Xi})$, $\mathbf{\Xi} = [\xi_1, \xi_2, ..., \xi_n]^T$, può essere **approssimato** generando la successione

$$\begin{cases} X^{(0)} \text{ dato} \\ X^{(k+1)} = \mathbf{\Phi}(X^{(k)}) \\ \quad k = 0, 1, 2, ... \end{cases} \Leftrightarrow \begin{cases} X^{(0)} = [x_1^{(0)}, x_2^{(0)}, ..., x_n^{(0)}]^T \text{ dato} \\ X^{(k)} = [x_1^{(k)}, x_2^{(k)}, ..., x_n^{(k)}]^T \ k = 1, 2, ... \\ x_1^{(k+1)} = \varphi_1(x_1^{(k)}, x_2^{(k)}, ..., x_n^{(k)}) \\ x_2^{(k+1)} = \varphi_2(x_1^{(k)}, x_2^{(k)}, ..., x_n^{(k)}) \\ \quad ..... \\ x_n^{(k+1)} = \varphi_n(x_1^{(k)}, x_2^{(k)}, ..., x_n^{(k)}) \end{cases}$$

Le funzione φ_i sono chiamate **funzioni di iterazione**.

Convergenza: condizione sufficiente

Definizione. Un'applicazione $\Phi : \mathcal{S} \longrightarrow \mathcal{S}$, dove $\mathcal{S}$ è uno **spazio normato** è detta **contrazione**, se esiste

$\lambda \in (0,1)$ tale che

$$\|\Phi(X) - \Phi(Y)\| \leq \lambda \|X - Y\| < \|X - Y\| \quad \forall X, Y \in \mathcal{S}$$

Teorema Sia $D \subset \mathbb{R}^n$. Se $\Phi : D \to D$ è una **contrazione**

$\Rightarrow$ α) esiste un **unico punto unito** $\Xi \in D$ di Φ

β) la successione $\left\{X^{(k)}\right\}_{k=0,1,\ldots}$, $X^{(k)} = \Phi\left(X^{(k-1)}\right)$, è **convergente** a Ξ per ogni **approssimazione iniziale** $X^{(0)} \in D$

Contrazione: condizione sufficiente

Matrice Jacobiana di Φ

$$J_\Phi(X) = \begin{bmatrix} \frac{\partial \varphi_1(X)}{\partial x_1} & \frac{\partial \varphi_1(X)}{\partial x_2} & \cdots & \frac{\partial \varphi_1(X)}{\partial x_n} \\ \frac{\partial \varphi_2(X)}{\partial x_1} & \frac{\partial \varphi_2(X)}{\partial x_2} & \cdots & \frac{\partial \varphi_2(X)}{\partial x_n} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial \varphi_n(X)}{\partial x_1} & \frac{\partial \varphi_n(X)}{\partial x_2} & \cdots & \frac{\partial \varphi_n(X)}{\partial x_n} \end{bmatrix}$$

Teorema. Se i) le **funzioni di iterazione** φ_1, φ_2, ..., φ_n sono **continue** e **parzialmente derivabili** in D;
ii) esiste $\lambda \in (0,1)$ tale che $\|J_\Phi(X)\| \leq \lambda$ per $X \in D$

$\Rightarrow$ Φ è una **contrazione** in D

Esempio: $n = 2$

La condizione $\|J_\Phi(X)\| \leq \lambda$, $X \in D$, è **sicuramente verificata** se

$$\left|\frac{\partial \varphi_i(X)}{\partial x_k}\right| \leq M_{ik} \qquad i,k = 1,\ldots,n \qquad X \in D$$

con

$$\|M\| \leq \lambda < 1 \quad \text{dove } M = [M_{ik}]_{i,k=1}^n$$

Esempio: $n = 2$

$$\begin{cases} f(x,y) = 0 \\ g(x,y) = 0 \end{cases} \Leftrightarrow \begin{cases} x = \varphi(x,y) \\ y = \psi(x,y) \end{cases} \to \begin{cases} |\varphi_x(X)| \leq M_{11} & |\varphi_y(X)| \leq M_{12} \\ |\psi_x(X)| \leq M_{21} & |\psi_y(X)| \leq M_{22} \end{cases}$$

$$M_{11} + M_{12} \leq \lambda < 1 \quad e \quad M_{21} + M_{22} \leq \lambda < 1$$

$$\|M\| \leq \lambda < 1 \Leftrightarrow \quad oppure \quad M_{11} + M_{21} \leq \lambda < 1 \quad e \quad M_{12} + M_{22} \leq \lambda < 1$$

$$oppure \quad M_{11}^2 + M_{12}^2 + M_{21}^2 + M_{22}^2 \leq \lambda < 1$$

Esercizio

Sia dato il sistema non lineare

$$\begin{cases} f(x,y) = x^2 + y^2 - 3 = 0 \\ g(x,y) = y\sqrt{x} - 1 = 0 \end{cases} \qquad (x,y) \in [a,b] \times [c,d]\,,$$

1.1) posto $a = 0$, $b = \sqrt{3}$, $c = 0$, $d = 2$, individuare in quali dei quattro rettangoli

$Q_1 = (a, a+(b-a)/2) \times (c, c+(d-c)/2)$ $\quad Q_2 = (a, a+(b-a)/2) \times (c+(d-c)/2, d)$

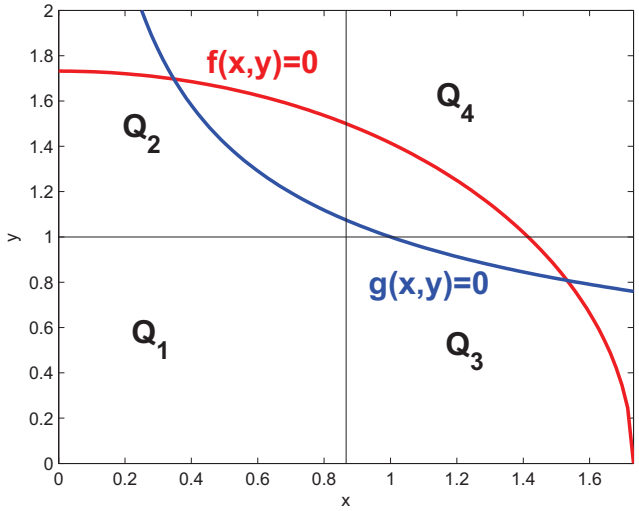
$Q_3 = (a+(b-a)/2, b) \times (c, c+(d-c)/2)$ $\quad Q_4 = (a+(b-a)/2, b) \times (c+(d-c)/2, d)$

sono presenti radici del sistema non lineare dato;

1.2) verificare se le funzioni di iterazione

$$\begin{cases} \phi(x,y) = \sqrt{3 - y^2} \\ \psi(x,y) = \frac{1}{\sqrt{x}} \end{cases}$$

sono adatte ad approssimare le radici individuati al punto precedente con il metodo delle approssimazioni successive. Specificare per ogni radice la scelta dell'approssimazione iniziale.



Caso Lineare

Sistemi non lineari

$F(X) = 0$

$X \in \mathbf{R}^n \quad F(X) : \mathbf{R}^n \to \mathbf{R}^n$

$\to F(X) = AX - B$

Sistemi lineari

$AX = B$

$X, B \in \mathbf{R}^n \quad A \in \mathbf{R}^{n \times n}$

$\downarrow$ $\qquad\qquad$ $\downarrow$

$X = [x_1, x_2, ..., x_n]^T$

$F = [f_1(X), f_2(X), ..., f_n(X)]^T$

$0 = [0, 0, ..., 0]^T$

$$\begin{cases} f_1(x_1, x_2, ..., x_n) = 0 \\ f_2(x_1, x_2, ..., x_n) = 0 \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ f_n(x_1, x_2, ..., x_n) = 0 \end{cases}$$

$X = [x_1, x_2, ..., x_n]^T$

$A = \left[a_{ij}\right]_{i,j=1}^{n}$

$B = [b_1, b_2, ..., b_n]^T$

$$\begin{cases} a_{11}\ x_1 + a_{12}\ x_2 + ... + a_{1n}\ x_n = b_1 \\ a_{21}\ x_1 + a_{22}\ x_2 + ... + a_{2n}\ x_n = b_2 \\ \qquad \ldots\ldots\ldots\ldots\ldots\ldots \\ \qquad \ldots\ldots\ldots\ldots\ldots\ldots \\ a_{n1}\ x_1 + a_{n2}\ x_2 + ... + a_{nn}\ x_{nn} = b_n \end{cases}$$

Metodo del punto unito per sistemi lineari

$A\,X = B \Leftrightarrow X = C\,X + Q$ con $Q = [q_1, q_2, ..., q_n]^T$, $C = \left[c_{ij}\right]_{i,j=1}^{n}$

Se $\overline{X} \in \mathbf{R}^n$ è **soluzione** di $A\,X = B$ allora è **punto unito** di $\Phi = C\,X + Q$:

$$\boxed{A\,\overline{X} = B \Leftrightarrow \overline{X} = C\,\overline{X} + Q}$$

$\Phi(X)$ ammette derivate parziali $\forall X \in \mathbf{R}^n$

$$\Rightarrow \quad J_\Phi(X) = C$$

Condizione sufficiente di convergenza:

$$\|J_\Phi(X)\| = \|C\| \leq \lambda < 1 \text{ per } X \in \mathbf{R}^n$$

Metodo di Newton per sistemi

Sistema non lineare: $\boxed{F(X) = 0 \qquad X = [x_1, x_2, \ldots, x_n]^T}$

Il **metodo di Newton** per la soluzione di sistemi non lineari si basa sulla **linearizzazione** della $F(X) = [f_1(X), \ldots, f_n(X)]^T$

Se le funzioni $f_i(X)$ hanno **derivate parziali limitate**, allora si può sviluppare in **serie di Taylor** la funzione vettoriale $F(X)$ scegliendo come punto iniziale $X^{(k)}$

$$F(X) = F(X^{(k)}) + J_F(X^{(k)})\,(X - X^{(k)}) + \ \ldots$$

dove $J_F(X) = \left[\dfrac{\partial f_i}{\partial x_j}\right]_{i,j=1,\ldots,n}$ è la **matrice jacobiana** della $F(X)$

$$\Rightarrow \quad F(X^{(k+1)}) \approx F(X^{(k)}) + J_F(X^{(k)})\,(X^{(k+1)} - X^{(k)}) = 0$$

$$\Rightarrow \quad \begin{cases} X^{(0)} & \text{dato} \\ X^{(k+1)} = X^{(k)} - \left[J_F(X^{(k)})\right]^{-1} F(X^{(k)}) & k \geq 0 \end{cases}$$

Convergenza del metodo di Newton

Il **metodo di Newton** è un **metodo iterativo** la cui **funzione di iterazione** è $\Phi_N(X) = X - [J_F(X)]^{-1} F(X)$

Teorema. Sia Ξ una soluzione del sistema non lineare

$$F(X) = 0$$

con $F \in C^2(D)$ ($D \in \mathbb{R}^n$ intorno di Ξ).

Sia $\det J_F(X) \neq 0$ per $X \in D$.

$\Rightarrow$ α) $\exists A \subseteq D$ tale che, $\forall X^{(0)} \in A$, la successione $\{X^{(k)}\}_{k=1,2,\ldots}$, $X^{(k)} = \Phi_N(X^{(k-1)})$, **converge** a Ξ;

β) la convergenza è **quadratica**: $\lim_{k\to\infty} \frac{||E^{(k+1)}||}{||E^{(k)}||^2} > 0$.

Osservazioni sul metodo di Newton per sistemi

- La **convergenza** del metodo è legata all'**accuratezza** dell'**approssimazione iniziale**.
- Ad ogni passo bisogna verificare che $\det J_F(X^{(k)}) \neq 0$. Nella pratica, si può avere **instabilità** numerica se $\det J_F(X^{(k)})$ è "*piccolo*" $\rightarrow$ conviene utilizzare una **precisione elevata**.
- Poiché il **costo computazionale** del calcolo di $\det J_F(X^{(k)})$ può essere **elevato**, si preferisce risolvere ad ogni passo il sistema lineare
$$J_F(X^{(k)})Y = -F(X^{(k)}) \Rightarrow X^{(k+1)} = X^{(k)} + Y$$
- **Criterio di arresto**: il procedimento iterativo viene arrestato quando $||X^{(k+1)} - X^{(k)}|| \leq \epsilon$.
- A volte si preferisce ricalcolare $J_F(X^{(k)})$ non ad ogni iterazione ma **dopo 3-4 iterazioni** (metodi di tipo quasi-Newton).

Problema 1: soluzione

Sistema non lineare

$$\begin{cases} f(x,y,z) = 0.78 - x e^{2.5\,y} - 2.5\,z = 0 \\ g(x,y,z) = 0.93 - x e^{5\,y} - 5\,z = 0 \\ h(x,y,z) = 1.16 - x e^{7\,y} - 7\,z = 0 \end{cases}$$

Matrice Jacobiana

$$J_F(x,y,z) = \begin{bmatrix} f_x(x,y,z) & f_y(x,y,z) & f_z(x,y,z) \\ g_x(x,y,z) & g_y(x,y,z) & g_z(x,y,z) \\ h_x(x,y,z) & h_y(x,y,z) & h_z(x,y,z) \end{bmatrix} = \begin{bmatrix} -e^{2.5\,y} & -2.5\,x\,e^{2.5\,y} & -2.5 \\ -e^{5\,y} & -5\,x\,e^{5\,y} & -5 \\ -e^{7\,y} & -7\,x\,e^{7\,y} & -7 \end{bmatrix}$$

Algoritmo

$$\begin{cases} x_0, y_0, z_0 \quad \text{dati} \\ J_F(x_k, y_k, z_k)\, V^{(k)} = F(x_k, y_k, z_k) \qquad k = 0, 1, 2, \ldots \\ X^{(k+1)} = X^{(k)} - V^{(k)} \end{cases}$$

Iterazioni: $x_0 = 1, y_0 = 1, z_0 = 1$

k	x_k	$\lvert x_k - x_{k-1}\rvert$	$\lvert f(x_k,y_k,z_k)\rvert$	y_k	$\lvert y_k - y_{k-1}\rvert$	$\lvert g(x_k,y_k,z_k)\rvert$	z_k	$\lvert z_k - z_{k-1}\rvert$	$\lvert h(x_k,y_k,z_k)\rvert$
0	1.0000	10.0000	13.9025	1.0000	10.0000	15.2483	1.0000	10.0000	1102.5
1.0000	-0.0187	1.0187	0.0772	1.0025	0.0025	1.8816	0.3730	0.6270	19.4658
2.0000	-0.0185	0.0003	0.0120	0.8715	0.1309	0.5102	0.3725	0.0005	6.7979
3.0000	-0.0403	0.0219	0.0652	0.5891	0.2824	0.3450	0.4085	0.0360	0.7933
4.0000	-0.3523	0.3119	0.9685	-0.4247	1.0138	2.7686	0.7482	0.3397	4.0591
5.0000	0.9908	1.3430	0.1603	-0.5831	0.1584	0.0982	0.1556	0.5925	0.0540
6.0000	0.6886	0.3022	0.0407	-0.1912	0.3919	0.1223	0.1575	0.0019	0.1231
7.0000	0.9135	0.2249	0.0119	-0.3781	0.1869	0.0342	0.1653	0.0078	0.0615
8.0000	1.0021	0.0886	0.0002	-0.3758	0.0024	0.0002	0.1554	0.0098	0.0001
9.0000	1.0021	0.0000	0.0000	-0.3760	0.0002	0.0000	0.1554	0.0000	0.0000
10.0000	1.0021	0.0000	0.0000	-0.3760	0.0000	0.0000	0.1554	0.0000	0.0000

La soluzione $x_{10} \approx 1.0021, y_{10} \approx -0.3760, z_{10} \approx 0.1554$ è accettabile?

Esercizio

Scrivere un programma che approssimi la soluzione del sistema non lineare del Problema 1 con il metodo di Newton arrestando le iterazioni quando la differenza tra le approssimazioni successive è inferiore a 10^{-6}.

Eseguire il programma per diverse scelte delle approssimazioni iniziali e verificare se il metodo converge.

Cosa succede quando $x_0 = 0, y_0 = 0, z_0 = 0$?

Individuare un insieme di valori x_0, y_0, z_0 che garantisce la convergenza del metodo a una soluzione compatibile con il problema fisico in esame.

Metodo di Newton $n = 3$: script MATLAB

```
format long;
f  = @(x,y,z)(0.78-x.*exp(2.5*y)-2.5*z);
fx = @(x,y,z)(-exp(2.5*y));
fy = @(x,y,z)(-2.5*x.*exp(2.5*y));
fz = -2.5;
g = @(x,y,z)(0.93-x.*exp(5*y)-5*z);
gx = @(x,y,z)(-exp(5*y));
gy =  @(x,y,z)(-5*x.*exp(5*y));
gz = -5;
h = @(x,y,z)(1.16-x.*exp(7*y)-7*z);
hx = @(x,y,z)(-exp(7*y));
hy = @(x,y,z)(-7*x.*exp(7*y));
hz = -7;
%
xn = 1.; yn = 1.; zn=1.;
iter = 0; errx = 10.; erry = 10.; errz = 10.;
errf = abs(f(xn,yn,zn)); errg = abs(g(xn,yn,zn)); errh =abs(h(xn,yn,zn));
[iter xn errx errf yn erry errg zn errz errh]
```



```
for iter= 1:10
   xv = xn; yv = yn; zv = zn;
   Jn = [fx(xv,yv,zv),fy(xv,yv,zv),fz;
         gx(xv,yv,zv),gy(xv,yv,zv),gz;
         hx(xv,yv,zv),hy(xv,yv,zv),hz];
   [iter det(Jn)]
   Bn = [f(xv,yv,zv);g(xv,yv,zv);h(xv,yv,zv)];
   Vn = Jn\Bn
   xn = xv-Vn(1);
   yn = yv-Vn(2);
   zn = zv-Vn(3);
   errx = abs(xn-xv); erry = abs(yn-yv); errz = abs(zn-zv);
   errf = abs(f(xn,yn,zn)); errg = abs(g(xn,yn,zn)); errh =abs(h(xn,yn,zn));
   [iter xn errx errf yn erry errg zn errz errh]
end
```


Metodo di Newton per sistemi: $n = 2$

Per $n = 2$ si ha: $\begin{cases} f(X) = f(x,y) = 0 \\ g(X) = g(x,y) = 0 \end{cases}$

Formula di Taylor di punto iniziale $X^{(k)} = [x_k, y_k]^T$:

$$\Downarrow$$

$$\begin{cases} f(X) = f(X^{(k)}) + f_x(X^{(k)})(x - x_k) + f_y(X^{(k)})(y - y_k) + R_1 = 0 \\ g(X) = g(X^{(k)}) + g_x(X^{(k)})(x - x_k) + g_y(X^{(k)})(y - y_k) + R_2 = 0 \end{cases}$$

dove $R_1 = R_1(X, X^{(k)})$, $R_2 = R_2(X, X^{(k)})$ rappresentano il **resto**.

La **soluzione approssimata** del sistema non lineare è la soluzione del **sistema lineare** che si ottiene trascurando il resto nello sviluppo precedente.

$$\begin{cases} f_x(X^{(k)})(x_{k+1} - x_k) + f_y(X^{(k)})(y_{k+1} - y_k) = -f(X^{(k)}) \\ g_x(X^{(k)})(x_{k+1} - x_k) + g_y(X^{(k)})(y_{k+1} - y_k) = -g(X^{(k)}) \end{cases}$$

Metodo di Newton per sistemi: $n = 2$

$$\begin{cases} f_x(X^{(k)})(x_{k+1} - x_k) + f_y(X^{(k)})(y_{k+1} - y_k) = -f(X^{(k)}) \\ g_x(X^{(k)})(x_{k+1} - x_k) + g_y(X^{(k)})(y_{k+1} - y_k) = -g(X^{(k)}) \end{cases}$$

$$\Downarrow$$

$$J_F(X^{(k)})\ (X^{(k+1)} - X^{(k)}) = -F(X^{(k)})$$

dove $J_F(X^{(k)}) = \begin{bmatrix} f_x(X^{(k)}) & f_y(X^{(k)}) \\ g_x(X^{(k)}) & g_y(X^{(k)}) \end{bmatrix}$

Il **sistema lineare** ammette soluzione se

$$|J_F^{(k)}| = \det J_F(X^{(k)}) \neq 0$$

La soluzione è

$$\begin{cases} x_{k+1} = x_k - \dfrac{1}{|J_F^{(k)}|}\left[f(X^k)\, g_y(X^{(k)}) - g(X^{(k)})\, f_y(X^{(k)}) \right] \\ y_{k+1} = y_k - \dfrac{1}{|J_F^{(k)}|}\left[g(X^k)\, f_x(X^{(k)}) - f(X^{(k)})\, g_x(X^{(k)}) \right] \end{cases} \qquad k \geq 0$$

Problema 2

Dato il sistema non lineare

$$\begin{cases} f(x,y) = 2\,x - 0.0002\,x\,y - 0.0001\,x^2 = 0 \\ g(x,y) = 4\,y - 0.0003\,y^2 - 0.0004\,x\,y = 0 \end{cases}$$

$i)$ separare le radici;

$ii)$ utilizzare il metodo di Newton per approssimare la radice in $D = [3000, 6000] \times [6000, 9000]$.

Traccia della soluzione

$i)$ Dalle due equazioni si può ricavare y come funzione lineare di x: $\begin{cases} y = 10^4 - 0.5\,x \\ y = \dfrac{4}{3}(10^4 - x) \end{cases}$.

Le due rette si **intersecano** solo quando $\begin{cases} x = 4000 \\ y = 8000 \end{cases}$

$\Rightarrow$ $\overline{X} = [4000, 8000]$ è l'**unica** soluzione non banale del sitema dato.

$ii)$ Le funzioni f, $g \in C^2(D)$.

$$\begin{cases} f_x = 2 - 2 \cdot 10^{-4}\, y - 2 \cdot 10^{-4}\, x & f_y = -2 \cdot 10^{-4}\, x \\ g_y = 4 - 6 \cdot 10^{-4}\, y - 4 \cdot 10^{-4}\, x & g_x = -4 \cdot 10^{-4}\, y \end{cases}$$

Utilizzando come approssimazione iniziale il **punto medio** del dominio D, dopo **30 iterazioni** si ottiene

$x_{30} = 3999.998 \quad y_{30} = 8000.001$

$||E^{(30)}||_\infty = 0.43 \cdot 10^{-2}$

Cosa succede utilizzando il metodo dell'iterazione continua?

Metodo di Newton $n = 2$: programma Fortran

```
      program newton
*
* Programma per la ricerca delle radici di un sistema
* di due equazioni non lineari  11-2012)
*
* Function:
* - double precision f(x,y), g(x,y): funzioni di cui si cercano le radicic
* - double precision derfx(x,y), derfy(x,y): derivate parziali della funzione f
* - double precision dergx(x,y), dergy(x,y): derivate parziali della funzione g
* Input:
* - double precision eps1: errore massimo su max(|x_(i+1) - x_i|,|y_(i+1) - y_i|)
* - double precision eps2: errore massimo su max(|f(x_(i+1),y_(i+1))|,|g(x_(i+1),y_(i+1))|
* - double precision xv:   approssimazione iniziale
* Variabili:
* - double precision err1: errore max(|x_(i+1) - x_i|,|y_(i+1) - y_i|)
* - double precision err2: errore max(|f(x_(i+1),y_(i+1))|,|g(x_(i+1),y_(i+1))
* - integer   iter: numero di iterazione
* - double precision fxy:    f(x_i,y_i)
* - double precision gxy:    g(x_i,y_i)
* - double precision dfx, dfy:    f_x(x_i,y_i), f_y(x_i,y_i)
* - double precision dgx, dgy:    g_x(x_i,y_i), g_y(x_i,y_i)
* - double precision detJi: detJi = f_x(x_i,y_i)*g_y(x_i,y_i)-g_x(x_i,y_i)*f_y(x_i,y_i);
* - double precision xn:   x_(i+1) = x_i- (f(x_i,y_i)*g_y(x_i,y_i)-g(x_i,y_i)*f_y(x_i,y_i))/detJi
* - double precision yn:   y_(i+1) = x_i - (g(x_i,y_i)*f_x(x_i,y_i)-f(x_i,y_i)*g_x(x_i,y_i))/detJi
* Output:
* - double precision xn,yn,err1,fxy,gxy
* - integer n
*
```



```
      implicit none
      double precision eps1, eps2, xn(0:30), yn(0:30), xv, yv
      double precision f, g, x, y
      double precision fxy, gxy, dfx, dfy, dgx, dgy, detJi
      double precision err1, err2
      integer iter, i
      f(x,y)= 2*x-0.0002*x*y-0.0001*x*x
      g(x,y)= 4*y-0.0003*y*y-0.0004*x*y
      dfx(x,y) = 2-2e-4*y -2e-4*x
      dfy(x,y)=-2e-4*x
      dgx(x,y)=-4e-4*y
      dgy(x,y)=4-6e-4*y-4e-4*x
*
* Lettura dati di input
*
      write (*,*) 'Introdurre eps1 e eps2: '
      read (*,*) eps1, eps2
      write (*,*) 'Introdurre approssimazione iniziale: '
      read (*,*) xv, yv
*
* Inizializzazione variabili
*
      fxy = f(xv,yv)
      gxy = g(xv,yv)
      xn(0) = xv
      yn(0) = yv
*
* Inizio ciclo iterativo: il ciclo viene interrotto quando
* gli errori err1 e err2 sono minori delle tolleranze assegnate
* oppure quando viene raggiunto un numero di iterazioni massimo
* (in questo caso e' 10)
*
```



```
      do i = 1, 10
         detJi = dfx(xv,yv)*dgy(xv,yv)-dgx(xv,yv)*dfy(xv,yv)
         if (detJi .ne. 0) then
            xn(i) = xv - (fxy*dgy(xv,yv)-gxy*dfy(xv,yv))/detJi
            yn(i) = yv - (gxy*dfx(xv,yv)-fxy*dgx(xv,yv))/detJi
         else
            write(*,*) 'Jacobiano nullo'
         endif
         fxy = f(xn(i),yn(i))
         gxy = g(xn(i),yn(i))
         err1 = max(abs(xn(i)-xv),abs(yn(i)-yv))
         err2 = max(abs(fxy),abs(gxy))
         write (10,*) 'i= ', i, 'xi=', xn(i), ' yi=',yn(i),
     &                ' err2=',err2, ' err2=',err2
         xv = xn(i)
         yv = yn(i)
         if (err1 .le. eps1 .and. err2 .le. eps2) goto 100
      enddo
      write (*,*) 'Non e'' stata raggiunta la convergenza'
100   continue
*
* Stampa del risultato
*
      write (*,*) 'Approssimazione:', xn(i), yn(i), ' Valore di f e g:',
     &            fxy, gxy, '   Numero di iterazioni: ', i
*
* Fine del programma
*
      stop
      end
```


Caso $n = 1$: equazioni non lineari

Un'**equazione non lineare** è un'equazione del tipo

$$f(x) = 0$$

Le **soluzioni** ξ dell'equazione, cioè quei valori tali che

$$f(\xi) = 0$$

vengono chiamati **radici** dell'equazione non lineare o **zeri** della funzione f.

Ci limiteremo al caso di **radici reali**: $\xi \in \mathbb{R}$.

Separazione delle radici

In genere, le **equazioni non lineari** che nascono nelle applicazioni non possono essere **risolte analiticamente**. Per **approssimare le radici** è necessario ricorrere a un **metodo numerico**.

Prima di utilizzare un metodo numerico bisogna sapere:

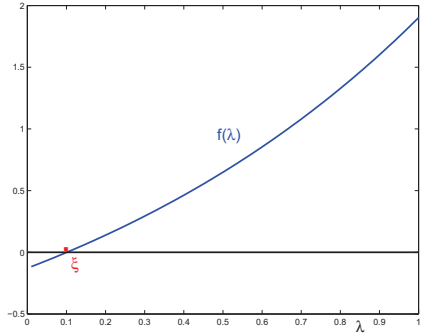
- **quante** sono le radici (reali);
- **dove** si trovano approssimativamente;
- se ci sono delle **simmetrie**.

Per rispondere a queste domande si può ricorrere alla **tabulazione** o al **grafico** della funzione f.

Problema 3: Separazione delle radici

$$f(\lambda) = e^{\lambda} + \frac{0.435}{\lambda}(e^{\lambda} - 1) - 1.564 = 0$$

x	f(x)
0.10000000000000	-0.00133558829528
0.12000000000000	0.02567293855461
0.14000000000000	0.05319595959218
0.16000000000000	0.08124355150079
0.18000000000000	0.10982599066618
0.20000000000000	0.13895375715854



Intervallo di separazione:

$I = [a, b] = [0.10, 0.12]$
$f(a) \approx -0.0013$
$f(b) \approx 0.0257$
$\Rightarrow f(a)f(b) < 0$

$I = [a, b] = [0.05, 0.15]$
$f(a) \approx -0.0667$
$f(b) \approx 0.0672$
$\Rightarrow f(a)f(b) < 0$

Problema 3: Separazione delle radici

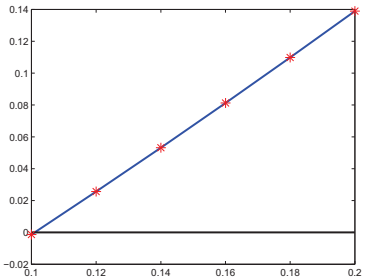
$$f(\lambda) = e^{\lambda} + \frac{0.435}{\lambda}(e^{\lambda} - 1) - 1.564 = 0$$

Tabulazione

```
>> x=linspace(0,1,11)

x =
   0  0.1000  0.2000  0.3000  0.4000  0.5000  0.6000  0.7000  0.8000  0.9000 1.0000

>> f=exp(x)+0.435./x.*(exp(x)-1)-1.564
Warning: Divide by zero.

f =
   NaN  -0.0013  0.1390  0.2932  0.4627  0.6491  0.8542  1.0797  1.3279  1.6011  1.9017

>> x=linspace(0.1,0.2,6)

x =
    0.1000    0.1200    0.1400    0.1600    0.1800    0.2000

>> f=exp(x)+0.435./x.*(exp(x)-1)-1.564

f =
   -0.0013    0.0257    0.0532    0.0812    0.1098    0.1390
```



```
>> format long
>> [x' f']

ans =

   0.10000000000000  -0.00133558829528
   0.12000000000000   0.02567293855461
   0.14000000000000   0.05319595959218
   0.16000000000000   0.08124355150079
   0.18000000000000   0.10982599066618
   0.20000000000000   0.13895375715854
```



```
>> plot(x,f,'b-',[0.1 0.2], [0 0],'k',x,f,'*r')
```

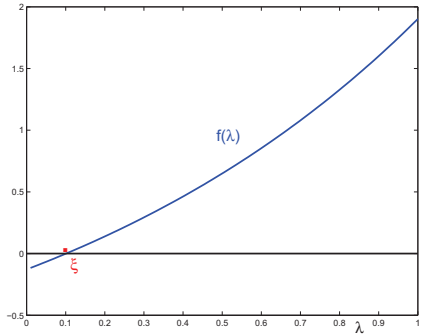
Intervallo di separazione: $I = [a, b] = [0.10, 0.12]$
$\Rightarrow f(a) \approx -0.0013, f(b) \approx 0.0257 \Rightarrow f(a)f(b) < 0$

Problema 3: Separazione delle radici

$$f(\lambda) = e^{\lambda} + \frac{0.435}{\lambda}(e^{\lambda} - 1) - 1.564 = 0$$

Separazione grafica

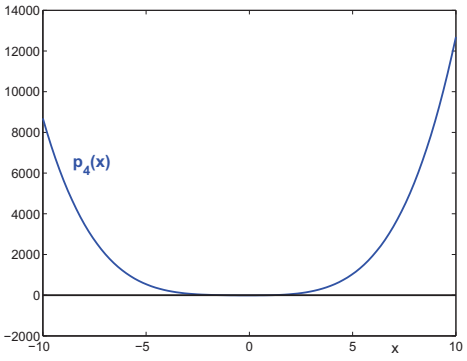
```
>> x=linspace(0,1);
>> f=exp(x)+0.435./x.*(exp(x)-1)-1.564;
Warning: Divide by zero.

>> plot(x,f,x,zeros(1,length(x)))
```

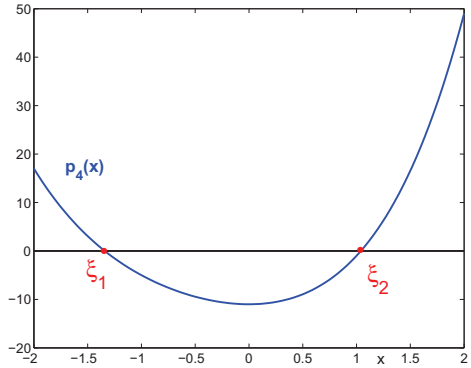


Intervallo di separazione: $I = [a, b] = [0.05, 0.15]$
$\Rightarrow f(a) \approx -0.0667, f(b) \approx 0.0672 \Rightarrow f(a)f(b) < 0$

Separazione delle radici: Esempio 1

Equazioni polinomiali: $p_4(x) = x^4 + 2x^3 + 7x^2 - 11 = 0$





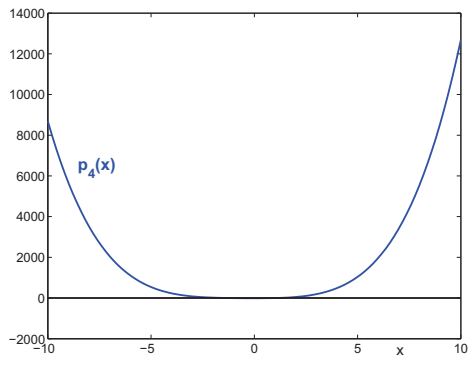

```
>> x=linspace(-10,10);
>> f=x.^4+2*x.^3+7*x.^2-11;
>> plot(x,f,x,zeros(1,length(x)))
```

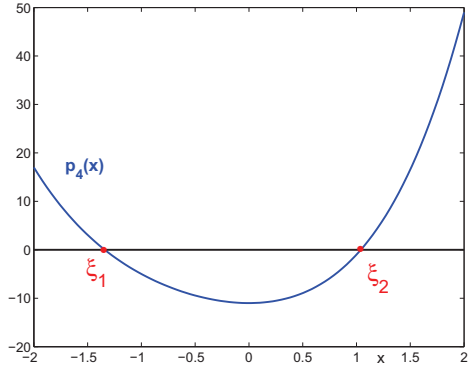
```
>> x=linspace(-2,2);
>> f=x.^4+2*x.^3+7*x.^2-11;
>> plot(x,f,x,zeros(1,length(x)))
```

Delle **4 radici** di $p_4(x)$ **due** sono **reali**, $\xi_1 \in [-1.5, -1]$ e $\xi_2 \in [0.75, 1.25]$, mentre **due** sono **complesse coniugate**.

Separazione delle radici: Esempio 1

Equazioni polinomiali: $p_4(x) = x^4 + 2x^3 + 7x^2 - 11 = 0$






```
gnuplot> plot x**4+2*x**3+7*x**2-11
```

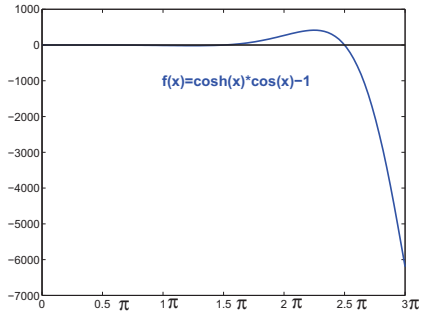
```
gnuplot> set xrange[-2:2]
gnuplot> plot x**4+2*x**3+7*x**2-11
```

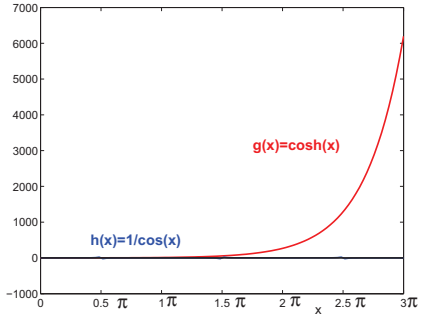
Delle **4 radici** di $p_4(x)$ **due** sono **reali**, $\xi_1 \in [-1.5, -1]$ e $\xi_2 \in [0.75, 1.25]$, mentre **due** sono **complesse coniugate**.

Separazione delle radici: Esempio 2

Equazione trascendente: $f(x) = \cos x \cosh x - 1 = 0$

La funzione $f(x)$ è **simmetrica** rispetto all'origine $\Rightarrow$ se ξ è **radice** lo è anche $-\xi \Rightarrow x > 0$ ($\xi = 0$ è radice banale)



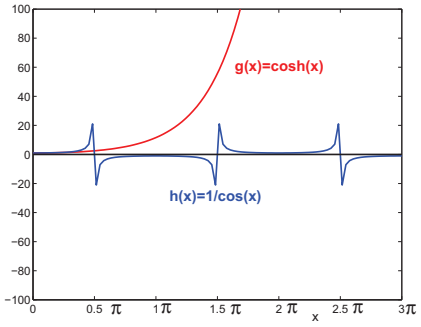



```
gnuplot> set xrange[0:3*pi]
gnuplot> plot cosh(x)*cos(x)-1
```

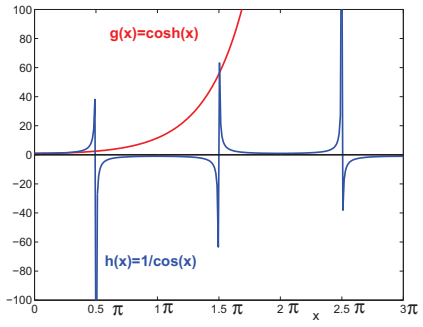
```
gnuplot> set xrange[0:3*pi]
gnuplot> plot cosh(x),1/cos(x)
```

Separazione delle radici: Esempio 2

Equazione trascendente: $f(x) = \cos x \cosh x - 1 = 0$




```
gnuplot> set xrange[0:3*pi]
gnuplot> set yrange[-100:100]
gnuplot> plot cosh(x)*cos(x)-1
```

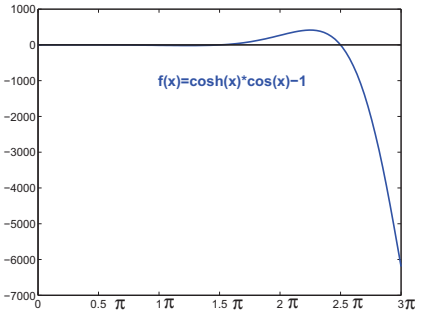



```
gnuplot> set sample 300
gnuplot> replot
```

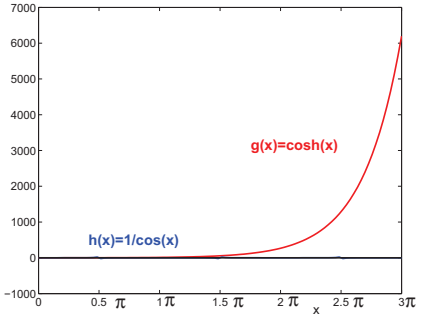

Separazione delle radici: Esempio 2

Equazione trascendente: $f(x) = \cos x \cosh x - 1 = 0$

La funzione $f(x)$ è **simmetrica** rispetto all'origine $\Rightarrow$ se ξ è **radice** lo è anche $-\xi \Rightarrow x > 0$ ($\xi = 0$ è radice banale)



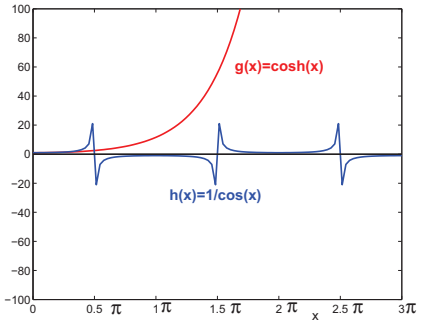

```
>> x=linspace(0,3*pi);
>> f=cosh(x).*cos(x)-1;
>> plot(x,f,x,zeros(1,length(x)))
```



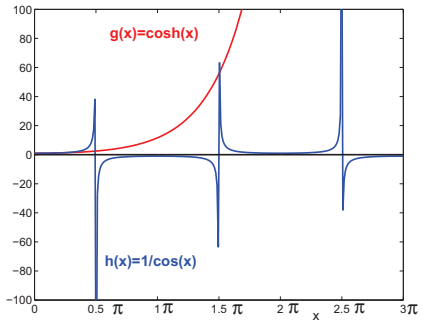

```
>> x=linspace(0,3*pi);
>> g=cosh(x);
>> h=1./cos(x);
>> plot(x,g,'r',x,h,'b',x,zeros(1,length(x)),'k')
```


Separazione delle radici: Esempio 2

Equazione trascendente: $f(x) = \cos x \cosh x - 1 = 0$




```
>> axis([0 3*pi -100 100])
```




```
>> x=linspace(0,3*pi,300);
>> g=cosh(x);
>> h=1./cos(x);
>> plot(x,g,'r',x,h,'b',x,zeros(1,length(x)),'k')
>> axis([0 3*pi -100 100])
```


Separazione delle radici: Esempio 2

Ci sono **infinite radici**, che corrispondono alle **intersezioni** delle due curve h e g:

$$f(x) = 0 \Leftrightarrow h(x) = g(x)$$

Separazione delle radici: in ciascun intervallo

$$\left[\left(2k - \tfrac{1}{2}\right)\tfrac{\pi}{2}, \left(2k + \tfrac{1}{2}\right)\tfrac{\pi}{2}\right], \qquad k = 1, 2, 3, \ldots$$

ci sono **due** radici. Per $k \to \infty$ le due radici si avvicinano ai due estremi dell'intervallo.

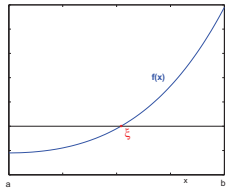
Nota. Molto spesso nelle applicazioni interessa approssimare la radice **più piccola**.

Metodo di bisezione (o metodo dicotomico)

Il **metodo di bisezione** è un metodo molto **semplice**: una volta individuato un intervallo di separazione in cui si trova **una sola** radice, permette di costruire una **successione** $\{x_k\}$ di **approssimazioni** di ξ.

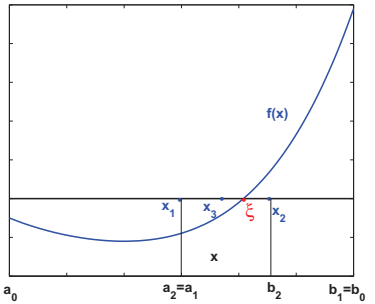
Ipotesi di applicabilità :

- è stato **separato** un intervallo $I = [a, b]$ in cui c'è un'**unica radice** ξ;
- la funzione f è **continua** in I: $f \in C^0[a, b]$;
- $f(a)f(b) < 0$.



Metodo di bisezione: algoritmo

Si genera una **successione** di approssimazioni $\{x_k\}$ con $x_k \in [a_{k-1}, b_{k-1}]$ e $\xi \in [a_{k-1}, b_{k-1}]$.



Algoritmo:

$a_0 = a, \quad b_0 = b$

per $k = 1, 2, 3, \ldots$

$x_k = \frac{a_{k-1}+b_{k-1}}{2}$ (**punto medio** di $[a_{k-1}, b_{k-1}]$)

se $f(x_k) = 0$, allora stop

se $f(a_{k-1})f(x_k) < 0$, allora $[a_k, b_k] = [a_{k-1}, x_k]$

se $f(x_k)f(b_{k-1}) < 0$, allora $[a_k, b_k] = [x_k, b_{k-1}]$

Convergenza del metodo di bisezione

Errore di troncamento: $e_k = \xi - x_k$

Convergenza: $\lim_{k\to\infty} x_k = \xi \Leftrightarrow \lim_{k\to\infty} |e_k| = 0$

Per il **metodo di bisezione** si ha

$$|e_k| < \frac{b_{k-1} - a_{k-1}}{2} = \frac{b_{k-2} - a_{k-2}}{2^2} = \cdots = \frac{b-a}{2^k}$$

$$\Rightarrow \lim_{k\to\infty} |e_k| < \lim_{k\to\infty} \frac{b-a}{2^k} = 0$$

Ordine di convergenza

Sia $\{x_k\}$ una **successione di approssimazioni convergente** a ξ. La successione ha **ordine di convergenza** p e **fattore di convergenza** C, se esistono due reali $p \geq 1$ e $C > 0$ tali che

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

Nota. La convergenza si dice **lineare** se $p = 1$, **quadratica** se $p = 2$.

Metodo di bisezione: ordine di convergenza

Per $k \to \infty$ si ha $\frac{|e_{k+1}|}{|e_k|^1} \simeq \frac{1}{2}$.

$\Rightarrow$ **Ordine di convergenza**: 1 (**lineare**)

Fattore di convergenza: $\frac{1}{2}$

La convergenza è **lenta**, in quanto ad ogni passo l'**errore** viene **dimezzato**, cioè ad ogni passo si guadagna una **cifra binaria**

$\Rightarrow$ poiché $2^{-4} < 10^{-1} < 2^{-3}$, per guadagnare una **cifra decimale** servono 3-4 **iterazioni**.

Metodo di bisezione: criteri di arresto

Nella pratica, a causa degli **errori di arrotondamento** e degli **errori di troncamento** non si verifica **mai** che $f(x_k) = 0$. Quando si arrestano le iterazioni?

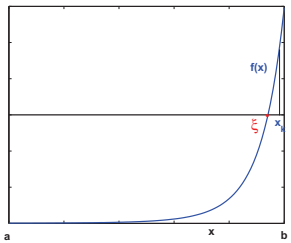
Criteri di arresto a posteriori $\begin{cases} |e_k| \simeq |x_k - x_{k-1}| < \varepsilon \\ |f(x_k)| < \varepsilon \end{cases}$

Criterio di arresto a priori: La **stima a priori** del numero di iterazioni K necessario per ottenere un **errore minore** di ε è

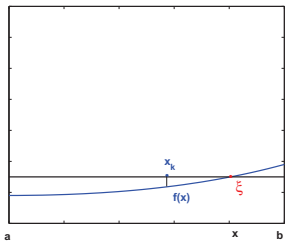
$$|e_k| < \frac{b-a}{2^k} < \varepsilon \quad \Rightarrow \quad K > \frac{\log(b-a) - \log(\varepsilon)}{\log 2}$$

Criteri di arresto: esempi

$|e_k| \simeq |x_k - x_{k-1}| < \epsilon$ oppure $|f(x_k)| < \varepsilon$



$f(x_k)$ è "*grande*" anche se x_k è "*vicino*" a ξ



$f(x_k)$ è "*piccolo*" anche se x_k è "*lontano*" da ξ

Efficienza computazionale

Per valutare l'**efficienza** di un metodo iterativo bisogna tener conto sia dell'**ordine di convergenza** che del **costo computazionale**, cioè della quantità di calcoli richiesta ad ogni passo.

Efficienza computazionale: $E = p^{1/r}$

- p: **ordine** di convergenza del metodo
- r: numero di **valutazioni funzionali** (calcolo di funzioni o derivate) richieste ad ogni passo

Metodo di bisezione: $E = 1$

Metodo di Newton: $E = 2^{1/2}$

Problema 3: soluzione

Si tratta di risolvere nell'incognita λ l'equazione non lineare

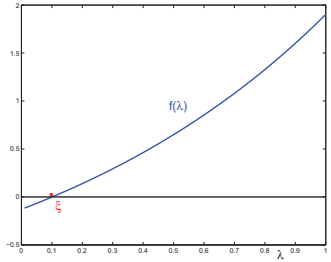
$$N|_{t=1\,\text{anno}} = N_0 e^{\lambda} + \frac{\nu}{\lambda}(e^{\lambda} - 1),$$

dove $N|_{t=1\,\text{anno}} = 1{\cdot}564{\cdot}000$, $N_0 = 1{\cdot}000{\cdot}000$, $\nu = 435{\cdot}000$.

$$\Rightarrow\ f(\lambda) = e^{\lambda} + \frac{0.435}{\lambda}(e^{\lambda} - 1) - 1.564 = 0 \qquad f \in C[0.05, 0.15]$$

Separazione grafica

```
gnuplot> set xrange [0:1]
gnuplot> plot exp(x)+0.435/x*(exp(x)-1)-1.564
```



Intervallo di separazione: $I = [a, b] = [0.05, 0.15]$

$$\Rightarrow f(a) \approx -0.0667, f(b) \approx 0.0672 \Rightarrow f(a)f(b) < 0$$

Problema 3: soluzione

Si tratta di risolvere nell'incognita λ l'equazione non lineare

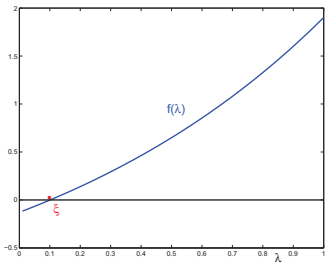
$$N|_{t=1\,\text{anno}} = N_0 e^{\lambda} + \frac{\nu}{\lambda}(e^{\lambda} - 1),$$

dove $N|_{t=1\,\text{anno}} = 1{\cdot}564{\cdot}000$, $N_0 = 1{\cdot}000{\cdot}000$, $\nu = 435{\cdot}000$.

$$\Rightarrow\ f(\lambda) = e^{\lambda} + \frac{0.435}{\lambda}(e^{\lambda} - 1) - 1.564 = 0 \qquad f \in C[0.05, 0.15]$$

Separazione grafica

```
>> x=linspace(0,1);
>> f=exp(x)+0.435./x.*(exp(x)-1)-1.564;
Warning: Divide by zero.


>> plot(x,f,x,zeros(1,length(x)))
```



Intervallo di separazione: $I = [a, b] = [0.05, 0.15]$

$$\Rightarrow f(a) \approx -0.0667, f(b) \approx 0.0672 \Rightarrow f(a)f(b) < 0$$

Iterazioni

k	a_k	b_k	x_{k+1}	$\lvert x_{k+1} - x_k \rvert$	$\lvert f(x_{k+1}) \rvert$
0	0.05000000000000	0.15000000000000	0.10000000000000	10.00000000000000	10.00000000000000
1	0.10000000000000	0.15000000000000	0.12500000000000	0.02500000000000	0.03250506973938
2	0.10000000000000	0.12500000000000	0.11250000000000	0.01250000000000	0.01548498364220
3	0.10000000000000	0.11250000000000	0.10625000000000	0.00625000000000	0.00704990930651
4	0.10000000000000	0.10625000000000	0.10312500000000	0.00312500000000	0.00285098217571
5	0.10000000000000	0.10312500000000	0.10156250000000	0.00156250000000	0.00075615469668
6	0.10000000000000	0.10156250000000	0.10078125000000	0.00078125000000	0.00029010206821
7	0.10078125000000	0.10156250000000	0.10117187500000	0.00039062500000	0.00023292996053
8	0.10078125000000	0.10117187500000	0.10097656250000	0.00019531250000	0.00002861013771
9	0.10097656250000	0.10117187500000	0.10107421875000	0.00009765625000	0.00010215388987
10	0.10097656250000	0.10107421875000	0.10102539062500	0.00004882812500	0.00003677037077

Metodo delle bisezioni: script MATLAB

```
format long;
f = @(x)exp(x)+0.435/x*(exp(x)-1)-1.564
a = 0.05; b = 0.15; xn = (a+b)/2;
iter = 0; err1 = 10; err2 = 10;
[iter a b xn err1 err2]
for i= 1:10
   if ( f(a)*f(xn) < 0 )
      b = xn;
   elseif ( f(xn)*f(b) < 0 )
      a = xn;
   end
   xv = xn;
   xn = (a+b)/2;
   iter = iter+1;
   err1 = abs(xn-xv);
   err2 = abs(f(xn));
   [iter a b xn err1 err2]
end
```

Metodo delle bisezioni: programma Fortran

```
      program bisezioni
*
* Programma per la ricerca degli zeri di una funzione
* con il metodo di bisezione.
*
* Function:
* - real f(x): funzione di cui si cercano gli zeri
* Input:
* - real  eps1: errore massimo su |x_(i+1) - x_i|
* - real  eps2: errore massimo su |f(x_(i+1))|
* - real  a, b: intervallo di separazione
* Variabili:
* - real  err1: errore |x_(i+1) - x_i|
* - real  err2: errore |f(x_(i+1))|
* - integer  iter: numero di iterazione
* - real  y:    f(x_i)
* - real  xn:   x_(i+1)
* - real  xv:   x_i
*
      real eps1, eps2, a, b, xn, xv, y
      real err1, err2
      integer  iter
```



```
*
* Lettura dati di input
*
      write (*,*) 'Introdurre eps1 e eps2:'
      read (*,*) eps1, eps2
      write (*,*) 'Introdurre intervallo iniziale'
      read (*,*) a, b
*
* Verifica delle ipotesi del metodo
*
      if ( f(a)*f(b) .gt. 0 ) stop 'a e b non soddisfano f(a)f(b)<0'
*
* Inizializzazione variabili
*
      err1 = 100
      err2 = 100
*
* Calcolo del punto medio
*
      xn = ( a + b ) * .5
*
* Inizio ciclo iterativo: il ciclo viene interrotto quando
* gli errori err1 e err2 sono minori delle tolleranze assegnate
* oppure quando viene raggiunto un numero di iterazioni massimo
* (in questo caso e' 50)
*
```



```
      iter = 0
      do while ((err1.gt.eps1 .or. err2.gt.eps2) .and. iter.lt.50)
        iter = iter+1
        if ( f(a)*f(xn) .lt. 0 ) then
          b = xn
        else if ( f(xn)*f(b) .lt. 0 ) then
          a = xn
        endif
        xv = xn
        xn = ( a + b ) * .5
        err1 = abs(xn-xv)
        y = f(xn)
        err2 = abs(y)
        write (*,*) 'x,err1,y,iter:', xn,  err1, y, iter
        write (20,*) 'x,err1,y,iter:', xn,  err1, y, iter
      enddo
*
* Stampa del risultato
*
      write (*,1000) xn, y, iter
1000  format (/, ' Approssimazione:', f8.4, ' Valore di f:', e12.6,
     &          ' Num. di iterazioni:', i4)
*
      stop
      end
```



```
*
* Funzione di cui si vogliono calcolare gli zeri (da modificare ogni volta)
*
      real function f(x)
      real  x
*
      f = exp(x)+0.435/x*(exp(x)-1)-1.564
*
      return
      end
```

Punto unito di una trasformazione

Un **punto unito di una trasformazione** è una soluzione dell'equazione non lineare

$$x = \varphi(x) \quad \Leftrightarrow \quad \varphi(x) - x = 0$$

Se ξ è **punto unito** di φ, allora

$$\xi = \varphi(\xi) \quad \Leftrightarrow \quad \varphi(\xi) - \xi = 0$$

Trovare il **punto unito** di φ significa trovare l'ascissa del punto di **intersezione** tra la retta $y = x$ e la curva $y = \varphi(x)$.

Punto unito: Esempio 1

Trovare i **punti uniti** della **funzione di iterazione**

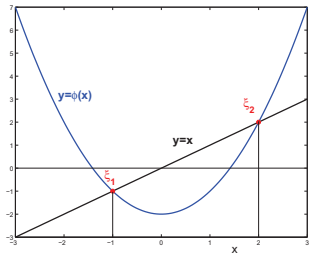
$$\varphi(x) = x^2 - 2 \qquad \text{per} \qquad -2 \le x \le 3.$$

Si tratta di trovare i valori di x per i quali

$$\varphi(x) = x^2 - 2 = x \quad \Rightarrow \quad \varphi(x) - x = x^2 - x - 2 = 0$$

Ci sono **due** punti uniti:

$$\xi_1 = -1 \quad \Rightarrow \quad \varphi(-1) = (-1)^2 - 2 = -1$$
$$\xi_2 = 2 \quad \Rightarrow \quad \varphi(2) = (2)^2 - 2 = 2$$



Metodo delle approssimazioni successive

Per **approssimare** un punto unito si può utilizzare il **metodo delle approssimazioni successive**:

$$\begin{cases} x_0 & \text{approssimazione iniziale} \\ x_k = \varphi(x_{k-1}) & k = 1, 2, \ldots \end{cases}$$

La funzione φ è detta **funzione di iterazione**.

Convergenza

Il metodo è **convergente** se per la **successione delle approssimazioni** $\{x_k = \varphi(x_{k-1})\}_{k \ge 1}$ vale

$$\boxed{\lim_{k \to \infty} |\xi - x_k| = 0} \quad \Leftrightarrow \quad \boxed{\lim_{k \to \infty} x_k = \xi}$$

Criterio di arresto

Se il metodo è convergente, una **buona approssimazione** di ξ è data dal valore x_K per il quale $\boxed{|x_K - x_{K-1}| \le \epsilon}$

Convergenza: condizione necessaria

Teorema. Se la successione

$$\begin{cases} x_0 & \text{approssimazione iniziale} \\ x_k = \varphi(x_{k-1}) & k = 1, 2, \ldots \end{cases}$$

è **convergente** a un valore τ e φ è **continua** in $\tau \Rightarrow \tau$ è **punto unito** di φ, cioè $\tau = \varphi(\tau)$.

Dimostrazione.

$$\tau = \lim_{n \to \infty} x_k = \lim_{n \to \infty} \varphi(x_{k-1}) = \varphi\left(\lim_{n \to \infty} x_{k-1}\right) = \varphi(\tau)$$

Metodo del punto unito

Il **metodo delle approssimazioni successive** può essere utilizzato per approssimare le **radici** di un'equazione non lineare.

Il **metodo del punto unito** consiste nel riscrivere l'equazione non lineare di partenza in un problema equivalente di punto unito:

$$f(x) = 0 \qquad \underset{\Longleftrightarrow}{f(x) = \varphi(x) - x} \qquad x = \varphi(x)$$

Se ξ è **radice** di f allora è **punto unito** di φ:

$$f(\xi) = 0 \quad \Leftrightarrow \quad \xi = \varphi(\xi)$$

Metodo del punto unito: Esempio 1

Trovare le **radici** dell'**equazione non lineare** $f(x) = x^2 - x - 2 = 0$

Possiamo trasformare l'equazione non lineare in un **problema equivalente di punto unito**:

$$f(x) = x^2 - x - 2 = 0 \qquad \Rightarrow \quad x = x^2 - 2$$

Trovare gli **zeri** di f equivale a trovare i **punti uniti** della **funzione di iterazione**

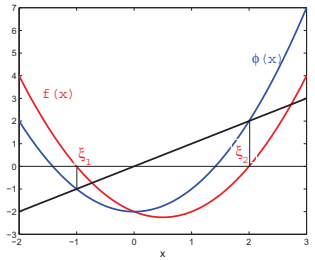
$$\varphi(x) = x^2 - 2$$

Ci sono **due** punti uniti:

$\xi_1 = -1 \quad \Rightarrow \quad \varphi(-1) = (-1)^2 - 2 = -1$
$\xi_2 = 2 \quad \Rightarrow \quad \varphi(2) = (2)^2 - 2 = 2$

ξ_1 e ξ_2 sono anche gli zeri di f:

$$f(\xi_1) = 0 \quad f(\xi_2) = 0$$



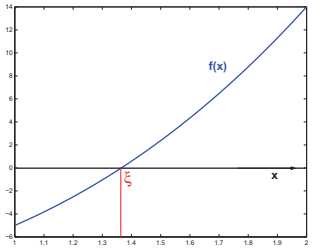
Metodo del punto unito: Esercizio

Verificare che l'equazione $f(x) = x^3 + 4x^2 - 10 = 0$ ha un'**unica** radice in $I = [1, 2]$ e trovare un'opportuna **funzione di iterazione** per approssimarla.

Soluzione. $f'(x) = 3x^2 + 8x > 0$ per $x \in I$

$\Rightarrow$ $f(x)$ è **monotona crescente** $\Rightarrow$ $\begin{cases} \min\limits_{1 \le x \le 2} f(x) = f(1) = -5 < 0 \\ \max\limits_{1 \le x \le 2} f(x) = f(2) = 14 > 0 \end{cases}$

$\Rightarrow$ **esiste** una radice in $[1, 2]$ e, per la **monotonia** di f in I, è **unica**



Esercizio: funzioni di iterazione

Per trovare una funzione di iterazione bisogna operare sull'equazione $x^3 + 4x^2 - 10 = 0$

1) $x - \dfrac{f(x)}{f'(x)} = x \qquad x = x - \dfrac{x^3 + 4x^2 - 10}{3x^2 + 8x} = \varphi_1(x)$

2) $x^3 + 4x^2 - 10 + x = x \quad \Rightarrow \quad x = x - x^3 - 4x^2 + 10 = \varphi_2(x)$

3) $x^2 = \dfrac{1}{4}(10 - x^3) \quad \Rightarrow \quad x = +\dfrac{1}{2}(10 - x^3)^{1/2} = \varphi_3(x)$

4) $x^3 = 10 - 4x^2 \quad \Rightarrow \quad x = (10 - 4x^2)^{1/3} = \varphi_4(x)$

5) $\dfrac{x^3 + 4x^2 - 10}{x} = 0 \quad \Rightarrow \quad x = \left(\dfrac{10}{x} - 4x\right)^{1/2} = \varphi_5(x)$

Metodo delle approssimazioni successive

$$\begin{cases} x_0 = 1.5 \\ x_k = \varphi(x_k - 1) \quad k = 1, 2, 3, \ldots \end{cases}$$

Iter	$x_k = \varphi_1(x_{k-1})$	$x_k = \varphi_2(x_{k-1})$
1	1.50000000000000	1.500000000000
2	1.37333333333333	-0.875000000000
3	1.36526201487463	6.732421875000
4	1.36523001391615	-469.720012001693
5	1.36523001341410	1.03×10^8
6	1.36523001341410	

Iter	$x_k = \varphi_3(x_{k-1})$	$x_k = \varphi_4(x_{k-1})$	$x_k = \varphi_5(x_{k-1})$
1	1.500000000	1.500000000	1.500000000
2	1.286953768	1.000000000	0.816496581
3	1.402540803	1.817120593	2.996908806
4	1.345458374	0.737397496 − 1.277209928i	0.000000000 − 2.941235061i
5	1.375170253	2.497908940 + 0.406090203i	2.753622388 + 2.753622388i
6	1.360094193	1.629663175 − 1.951899706i	1.814991519 − 3.534528790i
7	1.367846968	2.897699627 + 1.057120250i	2.384265848 + 3.434388064i
8	1.363887003	2.312403297 − 2.130274605i	2.182771900 − 3.596879228i
9	1.365916733	3.053449960 + 1.539322326i	2.296997587 + 3.574104462i
10	1.364878217	2.713757970 − 2.154809526i	2.256510286 − 3.606561220i
11	1.365410061	3.109236945 + 1.821357227i	2.279179049 + 3.601936572i
12	1.365137821	2.927459008 − 2.147135127i	2.271142587 − 3.608371470i
13	1.365277208	3.130026434 + 1.971686530i	2.275631311 + 3.607451621i
14	1.365205850	3.036651865 − 2.137943594i	2.274039927 − 3.608725567i
15	1.365242384	3.138432311 + 2.048621871i	2.274928362 + 3.608543344i
16	1.365223680	3.091432208 − 2.132017331i	2.274613384 − 3.608795481i
17	1.365233256	3.142102509 + 2.087260835i	2.274789213 + 3.608759411i
18	1.365228353	3.118681230 − 2.128745110i	2.274726876 − 3.608809311i
19	1.365230863	3.143794018 + 2.106492439i	2.2747616734 + 3.60880217i
20	1.365229578	3.132180323 − 2.127044558i	2.274749337 − 3.608812048i
...			
25	1.365230029	3.145180034 + 2.123063434i	2.274754931 + 3.608812641i
...			
30	1.365230013	3.144978450 − 2.125383768i	2.274754877 − 3.608812723i

Metodo delle approssimazioni successive: script MATLAB

```
format long;
phi  = @(x)(10-4*x.^2)^(1/3)
xn = 1.5;
iter = 0; err1 = 10; err2 = xn-phi(xn);
[iter xn err1 err2]
for i= 1:30
   xv = xn;
   xn = phi(xv);
   iter = iter+1;
   err1 = xn-xv;
   err2 = xn-phi(xn);
   [iter xn err1 err2]
end
```


Convergenza: condizione sufficiente

Teorema. Se φ è **derivabile** in $I = [a, b]$ e

$i)$ $\varphi : I \to I \Leftrightarrow a \leq \min_{x \in I} \varphi(x) \leq \max_{x \in I} \varphi(x) \leq b$

$ii)$ $\exists\ \lambda \in (0, 1)$ tale che $|\varphi'(x)| \leq \lambda,\ x \in I$

$\Rightarrow \alpha)$ esiste un **unico punto unito** $\xi \in I$ di $\varphi(\xi)$

$\beta)$ la successione $x_k = \varphi(x_{k-1})$ è **convergente** a ξ per ogni
approssimazione iniziale $x_0 \in I$

Dimostrazione dell'esistenza

$$\varphi : I \to I \quad \overset{i)}{\Rightarrow} \quad \varphi(a) \geq a \qquad \varphi(b) \leq b$$

Se vale una delle uguaglianze

$\Rightarrow \exists$ almeno **un punto unito**

Altrimenti $F(x) := x - \varphi(x)$ con $F \in C(I)$

$\Rightarrow F(a) = a - \varphi(a) < 0$ e $F(b) = b - \varphi(b) > 0$

$\Rightarrow \exists$ almeno uno **zero** di $F(x)$

$\Leftrightarrow \exists$ almeno un **punto unito** di φ

Dimostrazione dell'unicità

Supponiamo per **assurdo** che esistano **due punti uniti**

$\xi_1 \neq \xi_2$

$$\Downarrow$$

$$0 < |\xi_1 - \xi_2| = \underbrace{|\varphi(\xi_1) - \varphi(\xi_2)|}_{\textbf{punto unito}} =$$

$$= \underbrace{|\varphi'(\sigma)(\xi_1 - \xi_2)|}_{\textbf{Th. di Lagrange}} = |\varphi'(\sigma)|\,|(\xi_1 - \xi_2)| \leq$$

$$\leq \underbrace{k}_{ii)} |\xi_1 - \xi_2| < |\xi_1 - \xi_2|$$

Dimostrazione della convergenza

$$\forall x_0 \in I \Rightarrow \quad 0 < |e_k| = |\xi - x_k| = |\varphi(\xi) - \varphi(x_{k-1})| \leq$$

$$\leq \lambda\, |\xi - x_{k-1}| = \lambda |e_{k-1}|$$

$$\Rightarrow \quad |e_k| \leq \lambda\, |e_{k-1}| \leq \lambda^2\, |e_{k-2}| \leq \ldots \leq$$

$$\leq \lambda^k\, |e_0| \leq \lambda^k (b - a)$$

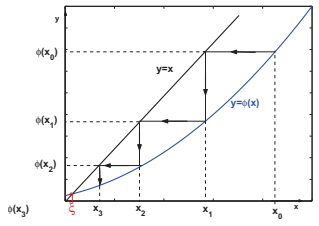
$$\Rightarrow \lim_{n\to\infty} |e_k| \leq \lim_{n\to\infty} \lambda^k (b-a) \underbrace{=}_{ii)} 0$$

Proprietà della successione delle approssimazioni

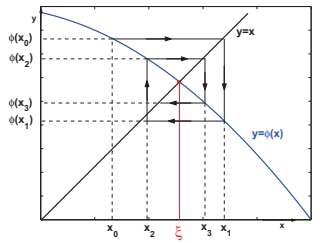
Dal **Teorema di Lagrange** si ha

$$e_k = \xi - x_k = \varphi(\xi) - \varphi(x_{k-1}) =$$

$$= \varphi'(t_k)(\xi - x_{k-1}) = \varphi'(t_k) e_{k-1} \qquad t_k \in [x_{k-1}, \xi]$$



$0 \leq \varphi'(x) < 1$



$-1 < \varphi'(x) \leq 0$

Proprietà della successione delle approssimazioni

- **Se** $0 \le \varphi'(x) < 1$ per $x \in I$ la successione $\{x_k = \varphi(x_{k-1})\}$, $n = 1, 2, ...$, è **monotona crescente** (se $e_0 > 0$) o **decrescente** (se $e_0 < 0$) $\Rightarrow$ le approssimazioni sono per **difetto** (se $\xi > x_0$) o per **eccesso** (se $\xi < x_0$)
- **Se** $-1 < \varphi'(x) \le 0$ per $x \in I$ la successione $\{x_k = \varphi(x_{k-1})\}$, $n = 1, 2, ...$, **non è monotona** $\Rightarrow$ le approssimazioni sono **alternativamente** per **difetto** e per **eccesso**

Ordine di convergenza

Teorema. Se

i) $\varphi \in C^{\mathbf{p}}(I)$ con I intorno di un punto unito ξ di φ

ii) la successione delle approssimazioni $\{x_k\}$ è **convergente**

iii) $\varphi(\xi) = \xi, \quad \varphi^{(\nu)}(\xi) = 0 \qquad \nu = 1, ..., \mathbf{p}-1$
$\varphi^{(\mathbf{p})}(\xi) \neq 0$

$\Rightarrow$ il metodo ha **ordine di convergenza** p

Dimostrazione

Sviluppo in serie di Taylor:

$$\varphi(x_k) = \varphi(\xi) + \varphi'(\xi)(x_k - \xi) + \frac{1}{2}\varphi''(\xi)(x_k - \xi)^2 + ... + \frac{1}{(p-1)!}\varphi^{(p-1)}(\xi)(x_k - \xi)^{p-1} + \underbrace{\frac{1}{p!}\varphi^{(p)}(t_k)(x_k - \xi)^p}_{\text{errore nella forma di Lagrange}} =$$

$$\underbrace{=}_{iii)} \varphi(\xi) + \frac{1}{p!}\varphi^{(p)}(t_k)(-e_k)^p \qquad t_k \in [x_k, \xi]$$

$$\Downarrow$$

$$e_{k+1} = \xi - x_{n+1} = \varphi(\xi) - \varphi(x_k) = -\frac{1}{p!}\varphi^{(p)}(t_k)(-1)^p \ (e_k)^p$$

$$\Rightarrow \boxed{\lim_{n\to\infty} \frac{|e_{k+1}|}{|e_k|^p} = \underbrace{\frac{|\varphi^{(p)}(\xi)|}{p!}}_{\substack{\text{fattore di} \\ \text{convergenza}}} = C > 0}$$

ordine di convergenza

Esempi

- **Se** $\varphi'(\xi) \neq 0 \Rightarrow \ p = 1$ la convergenza è **lineare**:

$$C = |\varphi'(\xi)| \le k := \max_{x\in[a,b]} |\varphi'(x)|$$

coefficiente di contrazione

- **Metodo delle tangenti**: $\boxed{\varphi_T(x) = x - \frac{f(x)}{f'(x)}}$

$$\varphi_T'(\xi) = \frac{f(\xi)f''(\xi)}{(f'(\xi))^2} = 0$$

$$\varphi_T''(\xi) = \frac{f''(\xi)}{f'(\xi)} \begin{cases} \neq 0 \ \textbf{se} \ f''(\xi) \neq 0 & \Rightarrow p = 2 \\ = 0 \ \textbf{se} \ f''(\xi) = 0 & \Rightarrow p > 2 \end{cases}$$

Se $p = 2$ la convergenza è **quadratica**

Criteri di arresto

Se k è il **coefficiente di contrazione** allora

- $|e_k| = |\xi - x_k| = |\varphi(\xi) - \varphi(x_{k-1})| \leq k\,|\xi - x_{k-1}|$
- $$\begin{aligned} |\xi - x_{k-1}| &= |\xi - x_k + x_k - x_{k-1}| \leq \\ &\leq |\xi - x_k| + |x_k - x_{k-1}| = \\ &= |\varphi(\xi) - \varphi(x_{k-1})| + |x_k - x_{k-1}| \leq \\ &\leq k\,|\xi - x_{k-1}| + |x_k - x_{k-1}| \end{aligned}$$

$$\Rightarrow |\xi - x_{k-1}| \leq \frac{1}{1-k}|x_k - x_{k-1}|$$

$$\boxed{|e_k| \leq \frac{k}{1-k}|x_k - x_{k-1}|}$$ **Stima a posteriori**

$$\boxed{|e_k| \leq \frac{k^n}{1-k}|x_1 - x_0| \leq \frac{k^n}{1-k}(b-a)}$$ **Stima a priori**

Esercizio

Data la funzione non lineare:

$$f(x;\alpha) = \sqrt{x-1} - e^{\alpha x}$$

dipendente dal parametro *reale* α:

1.1) determinare per quali valori di α la funzione ammette radici reali;

1.2) posto $\alpha = -2$, per ciascuna radice individuare un intervallo di separazione e una funzione di iterazione adatte ad approssimare la radice con il metodo delle approssimazioni successive;

1.3) caratterizzare la succesione delle approssimazioni (monotonia, ordine di convergenza, velocità di convergenza).

Traccia della soluzione

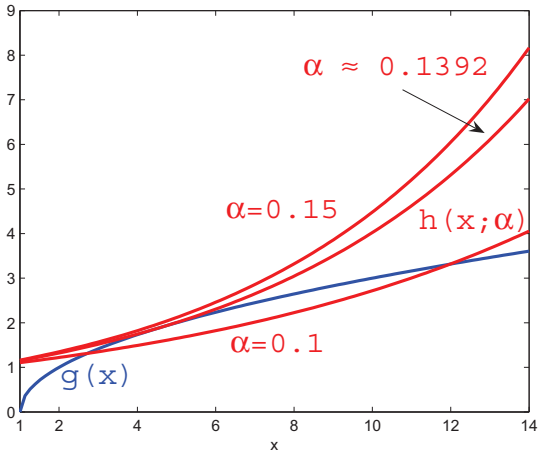
1.1) Per $\alpha < 0$ Le funzioni $g(x) = \sqrt{x-1}$, $x \geq 1$, e $h(x) = e^{\alpha x}$ hanno un unico punto di intersezione: la radice appartiene all'intervallo $(1,2)$ e si avvicina sempre più al valore 1 al diminuire di α.

Per $\alpha = 0$ si ricava facilmente la radice $\xi = 2$.

Per $\alpha > 0$ sufficientemente "grande", le funzioni $g(x)$ e $h(x)$ non hanno nessun punto di intersezione, mentre per $\tilde{\alpha} > \alpha > 0$ hanno due punti di intersezione. Il valore di $\tilde{\alpha}$ si ottiene risolvendo il sistema non lineare

$$\begin{cases} g(\tilde{x}) = h(\tilde{x}, \tilde{\alpha}) \\ g'(\tilde{x}) = h'(\tilde{x}, \tilde{\alpha}) \end{cases}$$

nelle incognite $\tilde{x}$ e $\tilde{\alpha}$. Una buona approssimazione è $\tilde{\alpha} \approx 0.1392025$; per questo valore di α le funzioni $g(x)$ e $h(x)$ hanno un unico punto di intersezione in $\tilde{x} \approx 4.5085164$ (vedi figura).



1.2) Per $\alpha = -2$ esiste un'unica radice nell'intervallo di separazione $I = [1, 2]$. La funzione $\phi(x) = 1 + e^{-4x}$ verifica le ipotesi del teorema di punto unito in quanto $\phi(x)$ e $|\phi'(x)| = 4e^{-4x}$ sono funzioni monotone descrescenti con

$$1 \leq \phi(2) = \min_{x \in I} \phi(x) \approx 1.0003 \leq \phi(x) \leq \phi(1) = \max_{x \in I} \phi(x) \approx 1.0183 \leq 2$$

$$\tau = \max_{x \in I} |\phi'(x)| = |\phi'(1)| \approx 0.0733 < 1$$

Quindi il procedimento iterativo

$$\begin{cases} x_0 \in I \\ x_{k+1} = \phi(x_k) \qquad k \geq 0 \end{cases}$$

converge all'unica radice in I.

1.3) Poiché $\phi'(x) < 0$ per $x \in I$, la succesione delle approssimazioni è alternativamente per eccesso e per difetto e l'ordine di convergenza è pari ad uno. Il valore piccolo del fattore di contrazione τ suggerisce che la velocità di convergenza sia elevata. Infatti dalla stima a priori

$$K > \log \frac{(1-\tau)\varepsilon}{b-a} \frac{1}{\log \tau}$$

si deduce che il numero di iterazioni necessarie affinché l'approssimazione abbia 6 decimali esatti è $K \geq 6$.

Iterazioni: metodo delle approssimazioni successive

k	x_k	$\lvert x_k - x_{k-1}\rvert$	$\lvert f(x_k)\rvert$
0	1.50000000000000	10.00000000000000	0.49752124782333
1	1.00247875217667	-0.49752124782333	-0.01565618430634
2	1.01813493648301	0.01565618430634	0.00110086517289
3	1.01703407131012	-0.00110086517289	-0.00007517425540
4	1.01710924556551	0.00007517425540	0.00000514392576
5	1.01710410163975	-0.00000514392576	-0.00000035193254
6	1.01710445357229	0.00000035193254	0.00000002407844
7	1.01710442949385	-0.00000002407844	-0.00000000164739
8	1.01710443114124	0.00000000164739	0.00000000011271
9	1.01710443102853	-0.00000000011271	-0.00000000000771
10	1.01710443103624	0.00000000000771	0.00000000000053

Iterazioni: metodo delle tangenti

k	x_k	$\lvert x_k - x_{k-1}\rvert$	$\lvert f(x_k)\rvert$
0	1.50000000000000	10.00000000000000	0.65731971281868
1	0.68515524761425	0.81484475238575	0.61593426123022
2	1.28302878384136 - 0.05580270020285i	0.60047206985058	0.46220904153119
3	0.86820404468583 + 0.04184080139294i	0.42616172707007	0.40017745640448
4	1.13975410004429 + 0.01781783113849i	0.27261059345710	0.27370105922316
5	0.96392214546911 - 0.01057413637556i	0.17810946091954	0.22786630197331

Iterazioni: metodo delle tangenti

```
                0   1.00000000000000  10.00000000000000  -0.13533528323661

Warning: Divide by zero.
> In @(x)(1./(2*sqrt(x-1))+2*exp(-2*x))
  In Script_tangenti at 12

ans =

   1.00000000000000   1.00000000000000                  0   0.13533528323661

Warning: Divide by zero.
> In @(x)(1./(2*sqrt(x-1))+2*exp(-2*x))
  In Script_tangenti at 12

ans =

   2.00000000000000   1.00000000000000                  0   0.13533528323661

Warning: Divide by zero.
> In @(x)(1./(2*sqrt(x-1))+2*exp(-2*x))
  In Script_tangenti at 12
```

Come si può giustificare il comportamento del metodo delle tangenti?

Esercizio

Individuare una funzione di iterazione adatta ad approssimare la radice dell'equazione non lineare

$$f(\lambda) = e^{\lambda} + \frac{0.435}{\lambda}(e^{\lambda} - 1) - 1.564 = 0$$

nell'intervallo $[0.05, 0.15]$ con il metodo delle approssimazioni successive.

Soluzione

Dall'equazione non lineare si può ricavare la **funzione di iterazione**

$$\varphi_1(x) = \frac{1}{1.564}\left(\lambda e^{\lambda} + 0.435(e^{\lambda} - 1)\right)$$

φ_1 è **monotona crescente** in $I = [0.05, 0.15]$ con

$$0.0649 \approx \varphi_1(0.05) \leq \varphi_1(x) \leq \varphi_1(0.15) \approx 0.2120$$

inoltre $|\varphi_1'(x)| = \frac{e^{\lambda}}{1.564}(1.435 + \lambda) \leq |\varphi_1'(0.15)| \approx 1.178$

Consideriamo un'altra **funzione di iterazione**

$$\varphi_2(x) = \log\left(1.564 - \frac{0.435}{\lambda}(e^{\lambda} - 1)\right)$$

i) $\varphi_2(x)$ è **monotona decrescente** in $I = [0.05, 0.15]$ con

$$0.05 < 0.0905 \approx \varphi_2(0.15) \leq \varphi_2(x) \leq \varphi_2(0.05) \approx 0.1115 < 0.15$$

ii) $|\varphi_2'(x)|$ è **monotona crescente** in $I = [0.05, 0.15]$ con

$$|\varphi_2'(x)| \leq \varphi_2(0.15) \approx 0.50 < 1$$

$\Rightarrow$ la successione $\{x_k = \varphi_2(x_{k-1})_{k\geq 1}$ **converge** alla radice per ogni $x_0 \in I$

Iterazioni

k	$x_k = \varphi_1(x_{k-1})$	$\lvert x_k - x_{k-1}\rvert$	$\lvert f(x_k)\rvert$
0	0.10000000000000	10.00000000000000	-0.03541286063299
1	0.13541286063299	0.03541286063299	-0.05360628710046
2	0.18901914773345	0.05360628710046	-0.08728511576238
3	0.27630426349583	0.08728511576238	-0.15929024388636
4	0.43559450738219	0.15929024388636	-0.35369164947571
5	0.78928615685791	0.35369164947571	-1.16969371781202
6	1.95897987466993	1.16969371781202	-12.37665034099773
7	0.00000143356302*1.0e+007	0.00000123766503*1.0e+007	-2.15316849866277*1.0e+007
8	2.15316993222579*1.0e+007	2.15316849866277*1.0e+007	-Inf
9	Inf	Inf	NaN
10	Inf	NaN	NaN

Iterazioni

k	$x_k = \varphi_2(x_{k-1})$	$\lvert x_k - x_{k-1}\rvert$	$\lvert f(x_k)\rvert$
0	0.10000000000000	10.00000000000000	-0.00120776062808
1	0.10120776062808	0.00120776062808	0.00025397482838
2	0.10095378579970	-0.00025397482838	-0.00005342976283
3	0.10100721556253	0.00005342976283	0.00001123925127
4	0.10099597631126	-0.00001123925127	-0.00000236428377
5	0.10099834059503	0.00000236428377	0.00000049734771
6	0.10099784324732	-0.00000049734771	-0.00000010462151
7	0.10099794786884	0.00000010462151	0.00000002200806
8	0.10099792586078	-0.00000002200806	-0.00000000462959
9	0.10099793049037	0.00000000462959	0.00000000097388
10	0.10099792951649	-0.00000000097388	-0.00000000020486

Metodi di linearizzazione

Si approssima la funzione $f(x)$ in un intorno I di ξ con la sua **tangente** o con la sua **secante**, calcolate tramite un opportuno **sviluppo in serie di Taylor**.

- **Metodo di Newton-Raphson o metodo delle tangenti**
- **Metodo delle secanti**

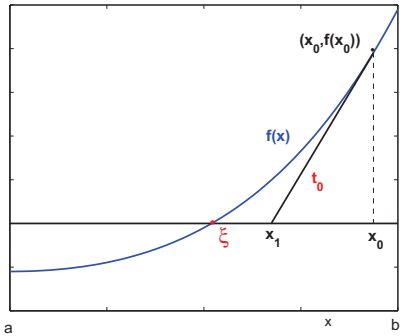
Metodo di Newton-Raphson

Approssimazione iniziale: x_0

Prima iterazione: t_0 è la **retta tangente** a $f(x)$ nel punto $(x_0, f(x_0))$:

$t_0 \to y = f(x_0) + f'(x_0)(x - x_0)$

Nuova approssimazione x_1**: intersezione** tra t_0 e $y = 0$.



$$f(x_0) + f'(x_0)(x_1 - x_0) = 0 \;\to\; \boxed{x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}}$$

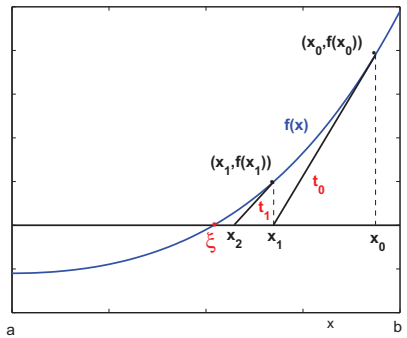
Metodo di Newton-Raphson

Nuova approssimazione: x_1

Seconda iterazione: t_1 è la **retta tangente** a $f(x)$ nel punto $(x_1, f(x_1))$:

$t_1 \to y = f(x_1) + f'(x_1)(x - x_1)$

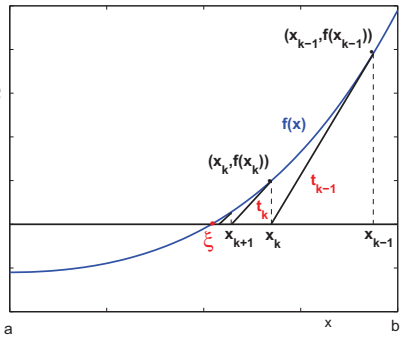
Nuova approssimazione x_2**: intersezione** tra t_1 e $y = 0$.



$$f(x_1) + f'(x_1)(x_2 - x_1) = 0 \;\to\; \boxed{x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}}$$

Metodo di Newton-Raphson: algoritmo

Ad ogni **iterazione** $k = 1, 2, \ldots$ la **nuova approssimazione** x_k è data dall'**intersezione** tra la **retta** t_{k-1}, **tangente** a $f(x)$ nel punto $(x_{k-1}, f(x_{k-1}))$, e $y = 0$.



$t_{k-1} \to y = f(x_{k-1}) + f'(x_{k-1})(x - x_{k-1})$

$$f(x_{k-1}) + f'(x_{k-1})(x_k - x_{k-1}) = 0$$

Algoritmo:

$$\begin{cases} x_0 \quad \text{dato} \\ x_k = x_{k-1} - \dfrac{f(x_{k-1})}{f'(x_{k-1})}, \quad k = 1, 2, \ldots \end{cases}$$

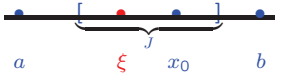
Metodo di Newton-Raphson: convergenza

$$\begin{cases} x_0 \quad \text{dato} \\ x_k = x_{k-1} - \dfrac{f(x_{k-1})}{f'(x_{k-1})}, \quad k = 1, 2, \ldots \end{cases}$$

Ipotesi di applicabilità :

- è stato **separato** un intervallo $I = [a, b]$ in cui c'è un'**unica radice** ξ;
- f, f', f'' sono **continue** in I: $f \in C^2[a, b]$;
- $f'(x) \neq 0$ per $x \in [a, b]$.

$\Rightarrow$ esiste un **intorno** $J \subseteq I$ di ξ tale che, se $x_0 \in J$, la **successione delle approssimazioni** $\{x_k\}$ **converge** a ξ.



Metodo di Newton-Raphson: ordine di convergenza

Ordine di convergenza: $\lim_{k\to\infty} \dfrac{|e_{k+1}|}{|e_k|^p} = C$

$$e_{k+1} = \xi - x_{k+1} = \left(\xi - \frac{f(\xi)}{f'(\xi)}\right) - \left(x_k - \frac{f(x_k)}{f'(x_k)}\right) =$$
$$= (\xi - x_k) - \left(\frac{f(\xi)}{f'(\xi)} - \frac{f(x_k)}{f'(x_k)}\right)$$

Sviluppo in serie di Taylor:

$$f(x_k) = f(\xi) + f'(\xi)\underbrace{(x_k - \xi)}_{-e_k} + \tfrac{1}{2}f''(\xi)(x_k - \xi)^2 + \ldots$$
$$f'(x_k) \simeq f'(\xi)$$

$$|e_{k+1}| \simeq \left| e_k - \frac{f(\xi) - f(\xi) + f'(\xi)e_k - \frac{1}{2}f''(\xi)e_k^2}{f'(\xi)} \right| = \left| \frac{\frac{1}{2}f''(\xi)e_k^2}{f'(\xi)} \right|$$

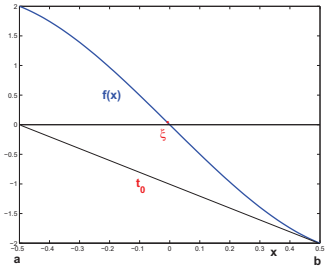
$\lim_{k\to\infty} \dfrac{|e_{k+1}|}{|e_k|^2} = \dfrac{1}{2}\left|\dfrac{f''(\xi)}{f'(\xi)}\right| \Rightarrow \boxed{p \geq 2}$ se $f(x) \in C^3[a, b]$ la convergenza è almeno **quadratica**

Metodo di Newton-Raphson: esempio

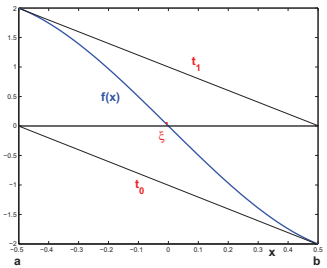
Approssimare la radice $\xi = 0$ dell'equazione

$$f(x) = 4x^3 - 5x = 0$$

con il **metodo di Newton-Raphson** scegliendo come approssimazione iniziale una volta $x_0 = 0.5$ e una volta $x_0 = 0.4$.



$x_{2k} = 0.5$



$x_{2k+1} = -0.5$

Estremo di Fourier

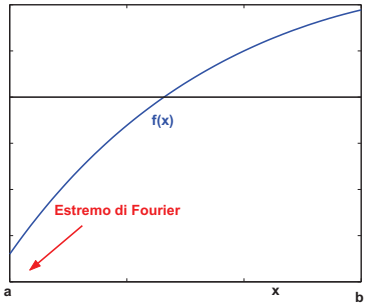
Nel caso particolare in cui $f(x)$ ha **concavità fissa** in un intervallo I si può garantire la **convergenza** anche senza procedere a una separazione preliminare della radice.

Estremo di Fourier:

Data una funzione f **continua** e **convessa** in $I = [a, b]$ con $f(a)f(b) < 0$, si dice **estremo di Fourier** di I l'estremo verso cui f rivolge la convessità .

Se **esiste** f'', allora l'estremo di Fourier è a o b a seconda che $f(a)f''(a) > 0$ oppure $f(b)f''(b) > 0$.

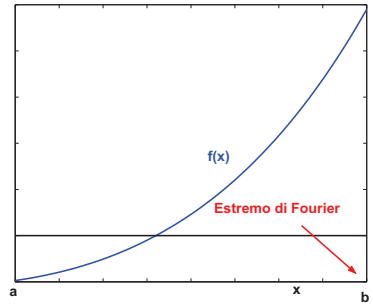
Estremo di Fourier: esempi



$f''(x)<0$ per $x \in [a, b]$

$\begin{cases} f(a)f''(a) > 0 \\ f(b)f''(b) < 0 \end{cases}$

$\Rightarrow$ a è **estremo di Fourier**



$f''(x)>0$ per $x \in [a, b]$

$\begin{cases} f(a)f''(a) < 0 \\ f(b)f''(b) > 0 \end{cases}$

$\Rightarrow$ b è **estremo di Fourier**

Metodo di Newton-Raphson: convergenza

Ipotesi di applicabilità :

- $f(a)f(b) < 0$
- f, f', f'' sono **continue** in I: $f \in C^2[a, b]$;
- $f'(x) \neq 0$ per $x \in [a, b]$;
- $f''(x) \neq 0$ per $x \in [a, b]$ e x_0 è l'**estremo di Fourier** di $[a, b]$.

$\Rightarrow$

1) esiste un'**unica radice** $\xi \in [a, b]$;

2) la **successione delle approssimazioni**

$$\left\{ x_k = x_{k-1} - \frac{f(x_{k-1})}{f'(x_{k-1})} \right\} \qquad k = 1, 2, \ldots$$

è **monotona** e **converge** a ξ;

3) se $f \in C^3[a, b]$, la convergenza è **quadratica**.

Problema 3: soluzione

$$f(\lambda) = e^{\lambda} + \frac{0.435}{\lambda}(e^{\lambda} - 1) - 1.564 = 0 \qquad \lambda \in I = [0.05, 0.15]$$

- Nell'intervallo I è stato separato un **unico zero**
- f, f', f'' sono **continue** in I;
- $f'(x) = e^{\lambda} - \frac{0.435}{\lambda^2}(e^{\lambda} - 1) + \frac{0.435}{\lambda}e^{\lambda} \neq 0$ per $x \in I$;

$\Rightarrow$ Lo zero può essere approssimato con il **metodo delle tangenti**

Inoltre $f''(x) = e^{\lambda} + \frac{0.87}{\lambda^3}(e^{\lambda} - 1) - \frac{0.87}{\lambda^2}e^{\lambda} + \frac{0.435}{\lambda}e^{\lambda} > 0$ in I

$\Rightarrow$ esiste l'**estremo di Fourier** di I:

$f(0.15)f''(0.15) > 0 \quad \Rightarrow \quad x_0 = 0.15$

Iterazioni

k	x_k	$\lvert x_k - x_{k-1}\rvert$	$\lvert f(x_k)\rvert$
0	0.15000000000000	10.00000000000000	0.06715354664030
1	0.10211384134812	0.04788615865188	0.00149497988981
2	0.10099851665312	0.00111532469500	0.00000078594305
3	0.10099792968591	0.00000058696721	0.00000000000022
4	0.10099792968575	0.00000000000016	0.00000000000000
5	0.10099792968575	0.00000000000000	0.00000000000000

Cosa succede se si sceglie come **approssimazione iniziale** $x_0 = 0.05$?

Metodo di Newton: script MATLAB

```
format long;
f  = @(x)exp(x)+0.435/x*(exp(x)-1)-1.564
df = @(x)exp(x)+0.435/x*exp(x)-0.435/x^2*(exp(x)-1)
xn = 0.15;
iter = 0; err1 = 10; err2 = f(xn);
[iter xn err1 err2]
for i= 1:5
   xv = xn;
   xn = xv-f(xv)/df(xv);
   iter = iter+1;
   err1 = abs(xn-xv);
   err2 = abs(f(xn));
   [iter xn err1 err2]
end
```


Metodo di Newton: programma FORTRAN

```
      program newton
*
* Programma per la ricerca degli zeri di una funzione
* con il metodo di Newton.
*
* Function:
* - double precision f(x): funzione di cui si cercano gli zeri
* - double precision fder(x): derivata della funzione f
* Input:
* - double precision eps1: errore massimo su |x_(i+1) - x_i|
* - double precision eps2: errore massimo su |f(x_(i+1))|
* - double precision xv:   approssimazione iniziale
* Variabili:
* - double precision err1: errore |x_(i+1) - x_i|
* - double precision err2: errore |f(x_(i+1))|
* - integer   iter: numero di iterazione
* - double precision xn:   x_(i+1)=x_i-f(x_i)/f'(x_i)
* - double precision xv:   x_i
* Output:
* - double precision xn,err1,y
* - integer iter
*
```

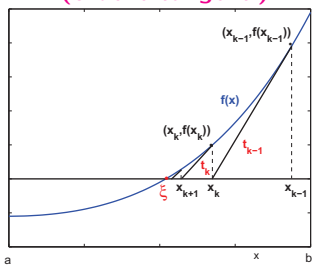


```
      implicit none
      real*8 eps1, eps2, xn, xv, y
      real*8 err1, err2
      real*8 f, fder
      integer iter
*
* Lettura dati di input
*
      write (*,*) 'Introdurre eps1 e eps2: '
      read (*,*) eps1, eps2
      write (*,*) 'Introdurre approssimazione iniziale: '
      read (*,*) xv
*
* Inizializzazione variabili
*
      iter = 0
      err1 = 1000
      err2 = 1000
*
* Inizio ciclo iterativo: il ciclo viene interrotto quando
* gli errori err1 e err2 sono minori delle tolleranze assegnate
* oppure quando viene raggiunto un numero di iterazioni massimo
* (in questo caso e' 10)
*
```

```
      do while ((err1.gt.eps1.or.err2.gt.eps2).and.iter.lt.50)
        iter = iter + 1
        xn = xv - f(xv) / fder(xv)
        y = f(xn)
        err1 = abs(xn-xv)
        err2 = abs(y)
        write (*,*) 'x=', xn,' err1=',err1,' f(x)=', y,' i=',iter
        xv = xn
      enddo
*
* Stampa del risultato
*
      write (*,*) 'Approssimazione:', xn, ' Valore di f:', y,
     &            '   Numero di iterazioni: ', iter
      if (err1.gt.eps1.or.err2.gt.eps2) then
         write (*,*) 'Non e'' stata raggiunta la convergenza'
      endif
*
* Fine del programma
*
      stop
      end
```


Metodi di linearizzazione

Metodo di Newton-Raphson (o delle tangenti)

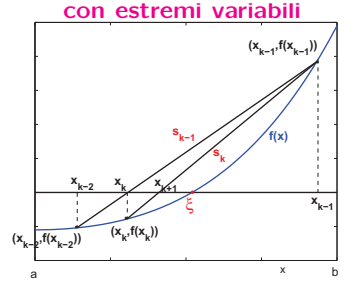


Ad ogni **iterazione** $k = 1, 2, \ldots$ la **nuova approssimazione** x_k è data dall'**intersezione** tra la **retta** t_{k-1}, **tangente** a $f(x)$ nel punto $(x_{k-1}, f(x_{k-1}))$, e $y = 0$.

Algoritmo:

$$\begin{cases} x_0 \quad \text{dato} \\ x_k = x_{k-1} - \dfrac{f(x_{k-1})}{f'(x_{k-1})}, \quad k = 1, 2, \ldots \end{cases}$$

Metodo delle secanti con estremi variabili



Ad ogni **iterazione** $k = 2, 3, \ldots$ la **nuova approssimazione** x_k è data dall'**intersezione** tra la **retta** s_{k-1}, **secante** $f(x)$ nei punti $(x_{k-2}, f(x_{k-2}))$ e $(x_{k-1}, f(x_{k-1}))$, e $y = 0$.

Algoritmo:

$$\begin{cases} x_0, x_1 \quad \text{dati} \\ x_k = x_{k-1} - f(x_{k-1}) \dfrac{x_{k-1} - x_{k-2}}{f(x_{k-1}) - f(x_{k-2})}, \quad k = 2, \ldots \end{cases}$$

Metodo delle secanti

$$\begin{cases} x_0, x_1 \quad \text{dati} \\ x_k = x_{k-1} - f(x_{k-1}) \dfrac{x_{k-1} - x_{k-2}}{f(x_{k-1}) - f(x_{k-2})}, \quad k = 2, \ldots \end{cases}$$

Vantaggi:

- si può usare quando **non si conosce** la derivata di $f(x)$ o quando $f(x)$ è **nota per punti**
- ad ogni passo richiede **una sola** valutazione funzionale

Svantaggi:

- servono **due approssimazioni iniziali** x_0 e x_1
- la scelta di x_0 e x_1 deve essere "**accurata**"

Convergenza del metodo delle secanti

Se • è stato **separato** un intervallo $I = [a, b]$ **simmetrico** intorno alla **radice** ξ,
- f, f', f'' sono **continue** in I: $f \in C^2[a, b]$,
- $f'(x) \neq 0$ per $x \in [a, b]$,

$\Rightarrow$ esiste un **intorno** $J \subseteq I$ di ξ tale che, se $x_0, x_1 \in J$, la **successione delle approssimazioni** $\{x_k\}$ **converge** a ξ con convergenza **superlineare**, cioè $2 > p > 1$.

Se $f''(x) \neq 0$ in I, l'**ordine di convergenza** è

$$p = \frac{1+\sqrt{5}}{2} \Rightarrow E = p \simeq 1.62$$

Esercizio

Scrivere un programma per risolvere il problema 3 con metodo delle secanti.

Esempio: $x_0 = 0.05 \qquad x_1 = 0.15$

k	x_{k-1}	x_k	x_{k+1}	$\lvert x_{k+1}-x_k\rvert$	$\lvert f(x_{k+1})\rvert$
1	0.05000000000000	0.15000000000000	0.09981947116877	0.05018052883123	0.00157706647874
2	0.15000000000000	0.09981947116877	0.10097089447657	0.00115142330781	0.00003619938533
3	0.09981947116877	0.10097089447657	0.10099794471093	0.00002705023436	0.00000002011856
4	0.10097089447657	0.10099794471093	0.10099792968556	0.00000001502538	0.00000000000026

Esercizio

Data l'equazione non lineare

$$f(x) = x^3 + \alpha - \cos x = 0:$$

1) individuare per quali valori di $\alpha \in \mathbf{R}$ l'equazione non ammette radici positive

2) per $\alpha = 1/3$ separare la radice più piccola $\tilde{\xi}$

3) fornire una stima a priori del numero di iterazioni necessarie per approssimare $\tilde{\xi}$ con un errore inferiore a $\varepsilon = 10^{-6}$ tramite il metodo di bisezioni;

4) tramite un programma Matlab, determinare quante iterazioni sono necessarie per approssimare con la stessa tolleranza ε la radice $\tilde{\xi}$ con i metodi di Newton e delle secanti?

Traccia della soluzione

1) Tracciando **qualitativamente** i grafici delle funzioni

$$y = g(x) = x^3 + \alpha \qquad y = h(x) = \cos x,$$

si deduce che $\alpha \geq 1$.

2) L'intervallo $[0, 1]$ contiene l'**unica** radice dell'equazione

$$f(x) = x^3 + 1/3 - \cos x = 0$$

Infatti $\quad f(0)f(1) < 0$

$$f'(x) = 3x^2 + \sin(x) > 0 \qquad \text{per } x \in [0, 1]$$

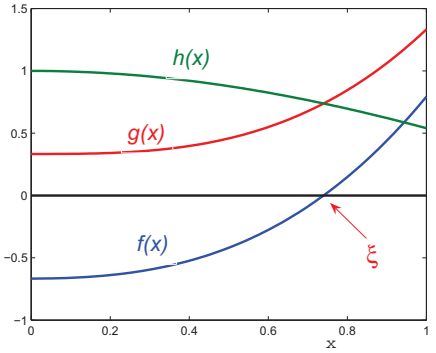
Grafici

$f(x) = x^3 + 1/3 - \cos x$

$g(x) = x^3 + 1/3$

$h(x) = \cos(x)$

La radice è l'**intersezione** tra il grafico di $f(x)$ e l'asse delle x (e coincide, ovviamente, con l'**intersezione** tra i grafici di $g(x)$ e $h(x)$).



3) Nell'intervallo $[0,1]$ sono verificate le **ipotesi di applicabilità** del metodo di bisezioni.

Quindi il numero di iterazioni K per cui $|e_K| \leq \varepsilon$ si ricava dalla relazione

$$|e_K| = |\xi - x_K| \leq \frac{b-a}{2^K} \leq \varepsilon$$

$$\Rightarrow \quad K > \frac{\log(b-a) - \log\varepsilon}{\log 2}.$$

In questo caso $a = 0$, $b = 1$, $\varepsilon = 10^{-6}$ $\Rightarrow K > 20$.

4) Si possono verificare analiticamente le **ipotesi di applicabilità** dei metodi delle tangenti e delle secanti nell'intervallo $I = [0,1]$:

- Nell'intervallo I è stato separato un **unico zero**
- f, f', f'' sono **continue** in I
- $f'(x) = 3x^2 + \sin(x) > 0$ per $x \in I$
- $f''(x) = 6x + \cos(x) > 0$ per $x \in I$

$\Rightarrow$ l'estremo $b = 1$ è l'**estremo di Fourier** dell'intervallo.

Il numero di iterazioni si può calcolare eseguendo le iterate con un programma Matlab.

k	x_k (bisez.)	$\lvert x_k - x_{k-1}\rvert$	x_k (Newton)	$\lvert x_k - x_{k-1}\rvert$	x_k (secanti)	$\lvert x_k - x_{k-1}\rvert$
1	0.500000000		1.		0., 1.	
2	0.750000000	0.25e+0	0.793560583	0.21e+0	0.456715571	0.54e+0
3	0.625000000	0.12e+0	0.742925006	0.51e-1	0.658587384	0.20e+0
4	0.687500000	0.62e-1	0.739971453	0.29e-2	0.775393863	0.12e+0
5	0.718750000	0.31e-1	0.739961685	0.98e-5	0.736625691	0.39e-1
6	0.734375000	0.16e-1	0.739961685	0.11e-9	0.739832822	0.32e-2
7	0.742187500	0.78e-2			0.739962167	0.13e-3
8	0.738281250	0.39e-2			0.739961685	0.48e-6
9	0.740234375	0.19e-2				
10	0.739257812	0.97e-3				
11	0.739746097	0.49e-3				
12	0.739990234	0.24e-3				
13	0.739868164	0.12e-3				
14	0.739929199	0.61e-4				
15	0.739959717	0.30e-4				
16	0.739974976	0.15e-4				
17	0.739967346	0.76e-5				
18	0.739963531	0.38e-5				
19	0.739961624	0.19e-5				
20	0.739962578	0.95e-6				

Tempo di calcolo

Bisezione: $\simeq$ 4.87 secondi, **Newton**: $\simeq$ 8.47 secondi, **Secanti**: $\simeq$ 4.46 secondi

Riferimenti bibliografici

L. Gori, *Calcolo Numerico*: Cap. 3, §§ 3.1, 3.2, 3.3, 3.4 (escluso metodo di falsa posizione), 3.5 3.6 (escluso metodo delle secanti con estremo fisso), 3.7, 3.9, 3.10

F. Pitolli, *Metodi alle differenze finite per problemi ai limiti*: §3

L. Gori, M.L. Lo Cascio, F. Pitolli *Esercizi di Calcolo Numerico*
Cap. 1: 1.1-1.11, 1.13, 1.14, 1.18-1.21, 1.25-1.26, 1.27-1.30
Cap. 7: 7.11-7.15, 7.18, 7.20, 7.22, 7.29, 7.36, 7.40, 7.43, 7.53-7.56