

Equazioni e sistemi non lineari

Lucia Gastaldi

DICATAM - Sezione di Matematica,
<http://lucia-gastaldi.unibs.it>



UNIVERSITÀ
DEGLI STUDI
DI BRESCIA

- 1 Ricerca degli zeri di una funzione
 - Problema e definizioni
 - Metodo di Newton-Raphson
 - Test d'arresto
 - Algoritmo ed esercizi
 - Metodo delle secanti
 - Function di Matlab

- 2 Soluzione di sistemi non lineari
 - Il metodo di Newton-Raphson per sistemi

Zeri di funzione

Problema

Data $f : [a, b] \rightarrow \mathbb{R}$ si cerca $x \in [a, b]$ tale che $f(x) = 0$.

Indichiamo con α uno zero di f .

Teorema

Supponiamo che la funzione $f : [a, b] \rightarrow \mathbb{R}$ sia **continua in** $[a, b]$ e che $f(a) \cdot f(b) < 0$; allora esiste $\alpha \in (a, b)$ tale che $f(\alpha) = 0$.

Ordine di convergenza di un metodo iterativo

Definizione

Si dice che un **metodo iterativo** è **convergente di ordine** $p > 1$ se vale

$$\lim_{k \rightarrow \infty} \frac{|\alpha - x_{k+1}|}{|\alpha - x_k|^p} = C \neq 0.$$

Si dice che un **metodo iterativo** **converge linearmente** se esiste un numero positivo $0 < C < 1$ tale che

$$\lim_{k \rightarrow \infty} \frac{|\alpha - x_{k+1}|}{|\alpha - x_k|} = C.$$

Si dice che un **metodo iterativo** **converge superlinearmente** se vale

$$\lim_{k \rightarrow \infty} \frac{|\alpha - x_{k+1}|}{|\alpha - x_k|} = 0.$$

Metodo di Newton-Raphson

Supponiamo di avere calcolato il valore x_k .

La migliore approssimazione lineare della funzione f nel punto x_k è data dalla retta tangente

$$t_k(x) = f(x_k) + f'(x_k)(x - x_k).$$

Ponendo $t_k(x) = 0$, si ricava il nuovo punto della successione x_{k+1} .

Iterata di Newton-Raphson

Dato x_0 ,

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

Teorema di convergenza locale quadratica

Teorema

Sia $f : [a, b] \rightarrow \mathbb{R}$ una funzione di classe \mathbf{C}^2 . Sia α tale che

$$f(\alpha) = 0, \quad f'(\alpha) \neq 0.$$

Allora esiste $\eta > 0$ tale che se il punto iniziale x_0 soddisfa

$$|\alpha - x_0| \leq \eta$$

allora si ha:

- ▶ Per ogni $k \in \mathbb{N}$, $|\alpha - x_k| \leq \eta$;
- ▶ $\lim_{k \rightarrow \infty} x_k = \alpha$;
- ▶ $\lim_{k \rightarrow \infty} \frac{x_{k+1} - \alpha}{(x_k - \alpha)^2} = \frac{f''(\alpha)}{2f'(\alpha)}$.

Nota Bene: se $f'(\alpha) = 0$, il metodo converge ma la convergenza diventa di tipo lineare.

Test d'arresto

Si deve trovare un modo per imporre che l'errore sia inferiore ad una tolleranza prestabilita, ossia tale che

$$\frac{|\alpha - x_k|}{|\alpha|} \leq \text{toll}, \quad \text{oppure } |\alpha - x_k| \leq \text{toll se } \alpha = 0.$$

Due possibilità:

$$|f(x_k)| \leq \text{toll}$$

non efficiente se $|f'(\alpha)| \approx 0$ oppure $|f'(\alpha)| \gg 1$.

$$|x_{k+1} - x_k| \leq \text{toll}$$

efficiente se il metodo converge **superlinearmente**.

Algoritmo di Newton-Raphson

- ▶ Dati f , f' , x_0 , $toll$ e $nmax$;
- ▶ valuta $y = f(x_0)$ e la derivata $dy = f'(x_0)$;
- ▶ inizializza $\delta = 1$ e $iter = 0$;
- ▶ Se $|\delta| \leq toll x_0$ il test d'arresto è verificato, x_0 è la soluzione cercata; **stop**.
- ▶ Se $iter > nmax$, è stato raggiunto il numero massimo di iterazioni senza arrivare a convergenza; **stop**.
- ▶ Altrimenti:
 - ▶ calcola $\delta = -y/dy$;
 - ▶ aggiorna $x_0 = x_0 + \delta$.
 - ▶ valuta $y = f(x_0)$ e la derivata $dy = f'(x_0)$;
 - ▶ incrementa l'indice di iterazione $iter = iter + 1$.
- ▶ Ripeti da 4.

Function `newton`

Scrivere la function `newton` che calcolare uno zero della funzione f .

```
[zero,fz,iter,xk]=newton(f,df,x0,toll,Nmax)
```

Input

<code>f</code>	function che contiene l'espressione della funzione f ;
<code>df</code>	function che contiene l'espressione della derivata f' ;
<code>x0</code>	punto iniziale per l'iterazione;
<code>toll</code>	tolleranza desiderata;
<code>Nmax</code>	numero massimo di iterazioni da eseguire;

Output

<code>zero</code>	soluzione cercata;
<code>fz</code>	valore di f nello zero calcolato;
<code>iter</code>	numero di iterazioni utilizzate;
<code>xk</code>	vettore delle approssimazioni successive.

Uso del calcolo simbolico

Voglio calcolare la derivata della funzione $f(x) = \sin(x) - \cos(2x)$.

```
syms x
f=sin(x)-cos(2*x)
f1=diff(f)
f=matlabFunction(f)

df=matlabFunction(f1)
```

dichiara che x è una variabile simbolica;
definisce la funzione simbolica f ;
calcola l'espressione della derivata f' ;
trasforma la funzione simbolica f
in una function-handle di nome f
trasforma la funzione simbolica $f1$
in una function-handle di nome df

Esercizi

Esercizio 1

Testare la function `newton` per determinare gli zeri delle seguenti funzioni:

$$f(x) = x^2 - 2, \quad x \in [0, 2]$$

$$f(x) = 3x - 1, \quad x \in [0, 1]$$

$$f(x) = \arctan(x), \quad x \in [-2, 2]$$

$$f(x) = \sin(x) - \cos(2x), \quad x \in [-\pi, \pi].$$

Per ciascuna funzione

- ▶ fare il grafico,
- ▶ scegliere un valore del dato iniziale nell'intervallo assegnato,
- ▶ calcolare la soluzione con la function `newton`,
- ▶ marcare lo zero trovato sul grafico della funzione.

Esercizio 2

La funzione

$$f(x) = e^x - 2x^2$$

ha tre zeri, $\alpha_1 < 0$, α_2 e α_3 positivi.

- ▶ Fare il grafico della funzione.
- ▶ Per $i = 1, 2, 3$ trovare un valore di x_0 in modo che il metodo di Newton converga a α_i .
- ▶ Marcare gli zeri trovati sul grafico della funzione.

Esercizio 3

Si consideri la funzione

$$f(x) = x - \frac{3\sin(3x)}{x} \quad x \in [0, 6].$$

La funzione e la sua derivata sono implementate nelle function `sin3` e `dsin3` rispettivamente.

Usare la function `newton` per calcolare lo zero di f e rappresentare il procedimento iterativo con le seguenti scelte del dato iniziale $x_0 = 0.1$, $x_0 = 2$, $x_0 = 2.1$, $x_0 = 2.2$, $x_0 = 2.4$, $x_0 = 2.5$ e $x_0 = 3$.

Convergenza del metodo di Newton

Modificare la function `newton` che implementa l'algoritmo di Newton-Raphson con la seguente riga di dichiarazione:

```
[zero,fz,iter,xk]=newton1(f,df,x0,tol,Niter)
```

Input

<code>f</code>	function che contiene l'espressione della funzione f ;
<code>df</code>	function che contiene l'espressione della derivata f' ;
<code>x0</code>	punto iniziale per l'iterazione;
<code>tol</code>	tolleranza desiderata;
<code>Niter</code>	numero massimo di iterazioni da eseguire.

Output

<code>zero</code>	soluzione cercata;
<code>fz</code>	valore di f nello zero calcolato;
<code>iter</code>	numero di iterazioni utilizzate;
<code>xk</code>	vettore delle approssimazioni successive.

Verifica della convergenza

Avendo a disposizione il vettore \mathbf{x}_k si possono vedere le seguenti informazioni:

- ▶ *Comportamento della successione*

Il comando `plot(xk)` riporta in un grafico i valori di x_k in funzione del valore k .

- ▶ *Convergenza* Per controllare la convergenza del metodo si può calcolare la differenza `diff` fra due iterate successive e plottarle su un grafico in scala semilogaritmica in funzione di k .

Esercizio 4

La funzione $f(x) = \sin(x) - \cos(2x)$ nell'intervallo $[-\pi, \pi]$ si annulla in tre punti $-\pi/2$, $\pi/6$ e $5\pi/6$.

Fare il grafico della funzione insieme all'asse delle ascisse.

In corrispondenza dei seguenti dati iniziali $x_0 = -1$, $x_0 = 1$, $x_0 = 2$ eseguire la seguente procedura:

- ▶ calcolare lo zero della funzione e riportarlo sul grafico della funzione;
- ▶ riportare la successione delle approssimanti in una figura con il comando `plot(xk)`;
- ▶ a partire dal vettore `xk`, calcolare la differenza fra due iterate successive nel vettore `diff` e plottarla in scala semilogaritmica `semilogy(diff)`.

Metodo delle secanti

Supponiamo di avere calcolato il valore x_k .

Nel caso in cui non si disponga della derivata di f oppure il costo del calcolo sia eccessivo si può considerare la secante che passa per $(x_k, f(x_k))$ e $(x_{k-1}, f(x_{k-1}))$:

$$s_k(x) = f(x_k) + \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}(x - x_k).$$

Ponendo $s_k(x) = 0$, si ricava il nuovo punto della successione x_{k+1} .

Iterata delle secanti

Dati x_0 e x_1

$$x_{k+1} = x_k - \frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})} f(x_k).$$

Teorema di convergenza locale superlineare

Teorema

Sia $f : [a, b] \rightarrow \mathbb{R}$ una funzione di classe \mathbf{C}^2 . Sia α tale che

$$f(\alpha) = 0, \quad f'(\alpha) \neq 0.$$

Allora esiste $\eta > 0$ tale che se il punto iniziale x_0 soddisfa $|\alpha - x_0| \leq \eta$ allora si ha:

- ▶ Per ogni $k \in \mathbb{N}$, $|\alpha - x_k| \leq \eta$;
- ▶ $\lim_{k \rightarrow \infty} x_k = \alpha$;
- ▶ $\lim_{k \rightarrow \infty} \frac{\alpha - x_{k+1}}{(\alpha - x_k)(\alpha - x_{k-1})} = -\frac{f''(\alpha)}{2f'(\alpha)}$
- ▶ $\lim_{k \rightarrow \infty} \frac{|\alpha - x_{k+1}|}{|\alpha - x_k|^p} = M^{-(1+1/p)}$

con $p = (1 + \sqrt{5})/2$ e $M = |f''(\alpha)|/(2|f'(\alpha)|)$.

Algoritmo delle secanti

- ▶ Dato x_0 e x_1 ;
- ▶ valuta $f_0 = f(x_0)$ e $f_1 = f(x_1)$.
- ▶ Se il test d'arresto è verificato, x_1 è la soluzione cercata; **stop**.
- ▶ Altrimenti:
 - ▶ calcola $dy = (f_1 - f_0)/(x_1 - x_0)$;
 - ▶ calcola $\delta = -f_1/dy$;
 - ▶ aggiorna $x_0 = x_1$, $x_1 = x_1 + \delta$;
 - ▶ aggiorna $f_0 = f_1$;
 - ▶ valuta $f_1 = f(x_1)$.
- ▶ Ripeti da 3.

Function `secanti`

Modificare la function `newton` in modo da ottenere la **function `secanti.m`** che implementa l'algoritmo delle secanti con la seguente riga di dichiarazione:

```
function [zero,fz,iter,xk]=secanti (f,x0,x1,tol,Nit)
```

Input

<code>f</code>	function che contiene l'espressione della funzione f ;
<code>x0,x1</code>	dati iniziali per l'iterazione;
<code>tol</code>	tolleranza desiderata;
<code>Nit</code>	numero massimo di iterazioni da eseguire;

Output

<code>zero</code>	soluzione cercata;
<code>fz</code>	valore di f nello zero calcolato;
<code>iter</code>	numero di iterazioni utilizzate;
<code>xk</code>	successione dei valori x_k .

Test

Esercizio 5

Usare le function **newton** e **secanti** per calcolare gli zeri delle seguenti funzioni, con x_0 assegnato e x_1 scelto opportunamente:

$$\begin{aligned} f(x) &= \sin x - \cos 2x & x_0 &= 1 \\ f(x) &= x^3 - 7x^2 + 11x - 5 & x_0 &= 2 \text{ e } x_1 = 7 \\ f(x) &= x^4 - 12x^3 + 47x^2 - 60x + 24 & x_0 &= 0 \text{ e } x_1 = 2 \end{aligned}$$

Fare un grafico di ciascuna funzione e riportare sul grafico lo zero ottenuto con i due metodi.

Riportare in scala semilogaritmica la differenza fra due iterate successive ottenute con i due metodi.

Verificare l'ordine di convergenza calcolando il rapporto

$$\frac{x_{k+1} - x_k}{(x_k - x_{k-1})^2} \quad \text{per Newton}$$

$$\frac{x_{k+1} - x_k}{(x_k - x_{k-1})^p}, \quad \text{per secanti } p = (1 + \sqrt{5})/2$$

fzero

Calcola gli zeri di una funzione reale di variabile reale con la seguente sintassi

```
[x,fval]=fzero(fun,x0)
```

```
[x,fval]=fzero(@fun,x0) se fun è una function
```

Input

<code>fun</code>	function che contiene la funzione f
<code>x0</code>	dato iniziale

Output

<code>x</code>	approssimazione dello zero calcolato
<code>fval</code>	valore di f in x .

Si possono ottenere delle informazioni complete sulle iterazioni usando il comando

```
[x,fval]=fzero(@fun,x0,optimset('disp','iter'))
```

fzero

Algoritmo di Dekker-Brent

- ▶ Cerca un intervallo $[a, b]$ in modo che $f(a)f(b) < 0$.
- ▶ Usa un passo delle secanti per trovare c .
- ▶ Ripete i passi seguenti finché $|b - a| < \varepsilon|b|$ o $f(b) = 0$.
 - ▶ Ordina a , b e c in modo tale che:

$$f(a)f(b) < 0, \quad |f(b)| < |f(a)|,$$

c è il valore precedente di b .

- ▶ Se $c \neq a$, usa un passo IQI (Inverse Quadratic Interpolation).
- ▶ Se $c = a$, usa il passo delle secanti.
- ▶ Se IQI o le secanti forniscono un valore interno a $[a, b]$, lo accetta.
- ▶ Altrimenti, usa il metodo delle bisezioni.

Utilizzo di `fzero`

`[x,fval]=fzero(fun,x0)` fornisce il valore della funzione `fun` nello zero `x`.

`[x,fval,exitflag]=fzero(fun,x0)` fornisce un valore `exitflag` che indica l'esito di `fzero`

Valore	Esito
1	convergenza verso la soluzione <code>x</code>
-1	l'algoritmo è interrotto da un <code>output function</code>
-3	sono stati trovati valori NaN o Inf
-4	sono stati trovati valori complessi durante la ricerca di un intervallo con cambio di segno
-5	<code>fzero</code> potrebbe essere arrivata a convergenza in un punto singolare
-6	<code>fzero</code> non trova un intervallo con cambio di segno.

Esercizio 6

Usare la function `fzero` per risolvere gli esercizi 1, 2 e 3, scegliendo gli stessi valori iniziali per x_0 e confrontare i risultati ottenuti con il metodo di Newton.

Il metodo di Newton-Raphson per sistemi

Consideriamo una funzione a valori vettoriali $F : A \rightarrow \mathbb{R}^n$ con $A \subseteq \mathbb{R}^n$:

$$F(\mathbf{x}) = \begin{cases} f_1(x_1, x_2, \dots, x_n) \\ f_2(x_1, x_2, \dots, x_n) \\ \dots\dots\dots \\ f_n(x_1, x_2, \dots, x_n) \end{cases}$$

Problema

Trovare $\mathbf{x} = (x_1, x_2, \dots, x_n) \in A$ tale che $F(\mathbf{x}) = 0$.

Linearizzazione

Per semplicità consideriamo $n = 2$ quindi abbiamo il sistema:

$$\begin{cases} f(x, y) = 0 \\ g(x, y) = 0 \end{cases}$$

Supponiamo di essere arrivati a calcolare un'approssimazione (x_k, y_k) e consideriamo l'approssimazione di f e g con i piani tangenti nel punto (x_k, y_k) :

$$\begin{cases} f(x, y) \approx f(x_k, y_k) + f_x(x_k, y_k)(x - x_k) + f_y(x_k, y_k)(y - y_k) \\ g(x, y) \approx g(x_k, y_k) + g_x(x_k, y_k)(x - x_k) + g_y(x_k, y_k)(y - y_k) \end{cases}$$

Iterazione del metodo di Newton-Raphson

La nuova approssimazione si ottiene come

$$x_{k+1} = x_k + \delta_x \quad y_{k+1} = y_k + \delta_y$$

dove il vettore $\delta = (\delta_x, \delta_y)^T$ è la soluzione del sistema

$$J(x_k, y_k)\delta = -F(x_k, y_k)$$

e

$$J(x_k, y_k) = \begin{pmatrix} f_x(x_k, y_k) & f_y(x_k, y_k) \\ g_x(x_k, y_k) & g_y(x_k, y_k) \end{pmatrix}$$
$$F(x_k, y_k) = \begin{pmatrix} f(x_k, y_k) \\ g(x_k, y_k) \end{pmatrix}$$

Algoritmo di Newton-Raphson

Newton_sist.m

- ▶ Dato \mathbf{x}_0 .
- ▶ Se il test d'arresto è verificato, \mathbf{x}_0 è la soluzione cercata; **stop**.
- ▶ Altrimenti:
 - ▶ valuta $Y = F(\mathbf{x}_0)$ e lo Jacobiano $A = J(\mathbf{x}_0)$;
 - ▶ risolvi $A\delta = -Y$;
 - ▶ aggiorna $\mathbf{x}_0 = \mathbf{x}_0 + \delta$.
- ▶ Ripeti da 2.

Nota bene

F è il nome di una function che fornisce il valore di F in un vettore colonna di dimensione n .

J è il nome di una function che fornisce il valore dello Jacobiano come array $n \times n$.

Function newtonsys

Modificare la function `newton` in modo da risolvere un sistema non lineare mediante il seguente comando:

```
[z,fz,iter]=newtonsys(f,fd,x0,tol,Nit)
```

Input

<code>f</code>	function_handle per la funzione f (vettore colonna);
<code>df</code>	function_handle per lo Jacobiano J (matrice);
<code>x0</code>	punto iniziale per l'iterazione (vettore colonna);
<code>tol</code>	tolleranza desiderata;
<code>Nit</code>	numero massimo di iterazioni da eseguire;

Output

<code>zero</code>	soluzione cercata;
<code>fz</code>	valore di f nello zero calcolato;
<code>iter</code>	numero di iterazioni utilizzate;

fsolve

Risolve i sistemi di equazioni non lineari in più variabili.

Appartiene al toolbox `optim`.

```
[x,fval]=fsolve(@fun,x0)
```

Input

`fun` nome della function che contiene la funzione f
`fun` accetta in input un vettore x e dà in output
il vettore dei valori di f valutata in x .

`x0` dato iniziale

Output

`x` approssimazione dello zero calcolato

`fval` valore di `fun` in `x`.

Esercizi

Esercizio 7

Applicare il metodo di Newton-Raphson per trovare gli zeri delle seguenti funzioni:

$$F_1(x, y) = \begin{pmatrix} x + y - 3 \\ x^2 + y^2 - 9 \end{pmatrix} \quad \begin{array}{l} \text{radici: } (0, 3) \ (3, 0) \\ x_0 = (1, 5), \ x_0 = (2, 3) \end{array}$$

$$F_2(x, y) = \begin{pmatrix} x^2 + y^2 - 2 \\ e^{x-1} + y^3 - 2 \end{pmatrix} \quad \begin{array}{l} \text{radice: } (1, 1) \\ x_0 = (1.5, 2), \ x_0 = (2, 3) \end{array}$$

Trovare la soluzione usando il metodo di Newton (`newtonsys`) e la function di Matlab `fsolve`.