Equazioni e sistemi non lineari

Lucia Gastaldi

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

Indice

Zeri di funzione

Problema

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

Indichiamo con α uno zero di f.

Teorema

Supponiamo che la funzione $f : [a, b] \to \mathbb{R}$ 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

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

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

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

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

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

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

$$\mathbf{t}_k(x) = f(x_k) + f'(x_k)(x - x_k).$$

Ponendo $\mathbf{t}_k(x) = 0$, 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 $f : [a, b] \to \mathbb{R}$ una funzione di classe $\mathbf{C}^2$. Sia α tale che

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

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

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

allora si ha:

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

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

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

Due possibilità:

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

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

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

efficiente se il metodo converge superlinearmente.

Algoritmo di Newton-Raphson

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

Function newton

La function `newton` calcola uno zero di una funzione f.

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

Input	
`f`	function che contiene l'espressione della funzione f;
`df`	function che contiene l'espressione della derivata f';
`x0`	punto iniziale per l'iterazione;
`toll`	tolleranza desiderata;
`Nmax`	numero massimo di ietrazioni da eseguire;
Output	
`zero`	soluzione cercata;
`fz`	valore di f nello zero calcolato;
`iter`	numero di iterazioni utilizzate;
`xk`	vettore delle approssimazioni successive.

NB Se non interessa conoscere i valori di `xk` il comando

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

fornisce soltanto le informazioni desiderate.

Esercizi

Esercizio 1

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

$$f(x) = x^2 - 2, \quad x \in [0, 2]$$
$$f(x) = 3x - 1, \quad x \in [0, 1]$$
$$f(x) = \arctan(x), \quad x \in [-2, 2]$$
$$f(x) = \sin(x) - \cos(2x), \; x \in [-\pi, \pi].$$

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.

Verifica della convergenza

Avendo a disposizione il vettore `xk` 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 3

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 $x0 = -1$, $x0 = 1$, $x0 = 2$ 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)`.

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	
`fun`	function che contiene la funzione f
`x0`	dato iniziale
Output	
`x`	approssimazione dello zero calcolato
`fval`	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 `x`
-1	l'algoritmo è interrotto da un output function
-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	`fzero` potrebbe essere arrivata a convergenza in un punto singolare
-6	`fzero` non trova un intervallo con cambio di segno.

Esercizio 4

Usare la function `fzero` per risolvere gli esercizi 1, 2 e 3, scegliendo gli stessi valori iniziali per $x0$ e confrontare i risultati ottenuti con il metodo di Newton.

Il metodo di Newton-Raphson per sistemi

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

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

Problema

Trovare $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in A$ 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

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

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

1. Dato $\mathbf{x}_0$.
2. Se il test d'arresto è verificato, $\mathbf{x}_0$ è la soluzione cercata; `stop`.
3. Altrimenti:
   - 3.1 valuta $Y = \mathtt{F}(\mathbf{x}_0)$ e lo Jacobiano $A = \mathtt{J}(\mathbf{x}_0)$;
   - 3.2 risolvi $A\delta = -Y$;
   - 3.3 aggiorna $\mathbf{x}_0 = \mathbf{x}_0 + \delta$.
4. 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 `nxn`.

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	
`f`	function_handle per la funzione f (vettore colonna);
`df`	function_handle per lo Jacobiano J (matrice);
`x0`	punto iniziale per l'iterazione (vettore colonna);
`tol`	tolleranza desiderata;
`Nit`	numero massimo di ietrazioni da eseguire;
Output	
`zero`	soluzione cercata;
`fz`	valore di f nello zero calcolato;
`iter`	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 5

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

$$F_1(x,y) = \begin{pmatrix} x+y-3 \\ x^2+y^2-9 \end{pmatrix}$$

radici: $(0,3)$ $(3,0)$
$x_0 = (1,5),\ x_0 = (2,3)$

$$F_2(x,y) = \begin{pmatrix} x^2+y^2-2 \\ e^{x-1}+y^3-2 \end{pmatrix}$$

radice: $(1,1)$
$x_0 = (1.5,2),\ x_0 = (2,3)$

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