1 Approssimazione di funzioni e dati

Il problema dell'*approssimazione* di una funzione f consiste generalmente nel trovare una funzione $\bar{f}$ più semplice di f che sia *sufficientemente rappresentativa*, che verrà usata al suo posto allo scopo di facilitare i calcoli.

Questa strategia governa l'**integrazione numerica** e l'**interpolazione** di funzioni. Nel caso dell'interpolazione, si assume che la funzione sia nota solamente *parzialmente* in un insieme di punti (detti **nodi d'interpolazione**) e si cerca una **funzione continua approssimante** (eg., **polinomiale**, **trigonometrica**, oppure **razionale**) i cui valori nei nodi coincidano con i valori della funzione data.

1.1 Interpolazione polinomiale

Supponiamo che venga dato un insieme di $n+1$ coppie $\{x_i, y_i\}$, $i = 0, \ldots, n$, dove i punti x_i sono tutti *distinti* e vengono chiamati **nodi**. Si può dimostrare che *esiste* un *unico* polinomio $\Pi_n \in \mathbb{P}_n$ di grado minore uguale ad n, tale che

$$\Pi_n(x_i) = y_i, \quad i = 0, \ldots, n. \tag{1}$$

Π_n viene detto **polinomio interpolatore** dei valori y_i nei nodi x_i. **Polinomio di interpolazione**, **polinomio interpolante**, **polinomio interpolatore** oppure **interpolante polinomiale** sono tutte terminologie frequenti che vengono usate in maniera equivalente tra di loro.

Quando $y_i = f(x_i)$, dove f è una data funzione continua, allora Π_n è detto polinomio interpolatore di f e viene indicato con $\Pi_n f$.

Si può dimostrare che l'equazione esplicita di Π_n è

$$\Pi_n(x) = \sum_{k=0}^{n} y_k \ell_k(x), \tag{2}$$

laddove ℓ_k sono detti **polinomi caratteristici di Lagrange** e sono definiti come

$$\ell_k(x) = \prod_{j=0, j\neq k}^{n} \frac{x - x_j}{x_k - x_j}, \quad k = 0, \ldots, n. \tag{3}$$

La (2) è chiamata **forma di Lagrange** del polinomio interpolatore.

1.2 Comandi `polyfit` e `polyval`

Supponiamo che vengano dati due vettori `x=[x1,...,xn+1]` e `y=[y1,...,yn+1]` di uguale lunghezza $n+1$. L'istruzione `c=polyfit(x,y,n)` di MATLAB genera i *coefficienti* del **polinomio interpolatore di Lagrange** di grado n, ordinati in maniera descrescente rispetto alle potenze.

Più in generale, dati due vettori `x` ed `y` di lunghezza $n+1$, l'istruzione `c=polyfit(x,y,m)` calcola *di default* i coefficienti del polinomio di grado m che meglio approssima le $n+1$ coppie di dati `(x(i),y(i))` **nel senso dei minimi quadrati**, ovvero il polinomio $\tilde{p}_m \in \mathbb{P}_m$ che soddisfa

$$\sum_{i=0}^{n}[y_i - \tilde{p}_m(x_i)]^2 \leq \sum_{i=0}^{n}[y_i - p_m(x_i)]^2. \tag{4}$$

Generalmente si avrà $m \leq n$. Si può dimostrare che per m uguale ad n, il polinomio dei minimi quadrati definito dalla (4) coincide con il polinomio interpolatore di Lagrange (2). Di conseguenza, il comando `polyfit` in questo caso calcola il polinomio interpolatore.

Dato il vettore dei coefficienti `c=[c1,...,cn+1]`, l'istruzione `p=polyval(c,z)` permette di calcolare i valori del polinomio interpolatore in punti arbitrari `z(j)`, per $j = 1, \ldots, k$.

Esercizio 1.1 (Polinomio di interpolazione di un insieme di dati)**.** *Dati i vettori* `x=[-55 -25 5 35 65]` *e* `y=[-3.25 -3.2 -3.02 -3.32 -3.1]`*, calcolare il polinomio interpolante di grado* $n = 4$ *dell'insieme di dati* $(x(i), y(i))$ *con l'utilizzo del comando* `polyfit` *e lo si rappresenti graficamente. Calcolare infine i polinomi di migliore approssimazione nel senso dei minimi quadrati di grado* $m < n$*.*

Soluzione:

```
%insieme di dati
x=[-55 -25 5 35 65];
y=[-3.25 -3.2 -3.02 -3.32 -3.1];

%polinomio interpolatore di ordine 4
c=polyfit(x,y,4)

%vettore di 100 punti equispaziati nell'intervallo
%tra la prima e l'ultima componente del vettore x
%che serve per generare il plot del polinomio
z=linspace(x(1),x(end),100);

p=polyval(c,z);
plot(z,p,x,y,'r*');
grid on;
legend('polinomio interpolatore di grado 4'...
        ,'insieme di dati','FontSize',10)
```

Infine, per costruire il polinomio di interpolazione di Lagrange $\Pi_n f$ di grado n di una funzione `f` su $n+1$ punti, è sufficiente memorizzare le ascisse in un vettore `x`, e successivamente costruire il vettore `y` valutando la funzione `f` in `y`, e procedere seguendo le istruzioni precedenti (cf. gli Esercizi 1.2 e 1.4).

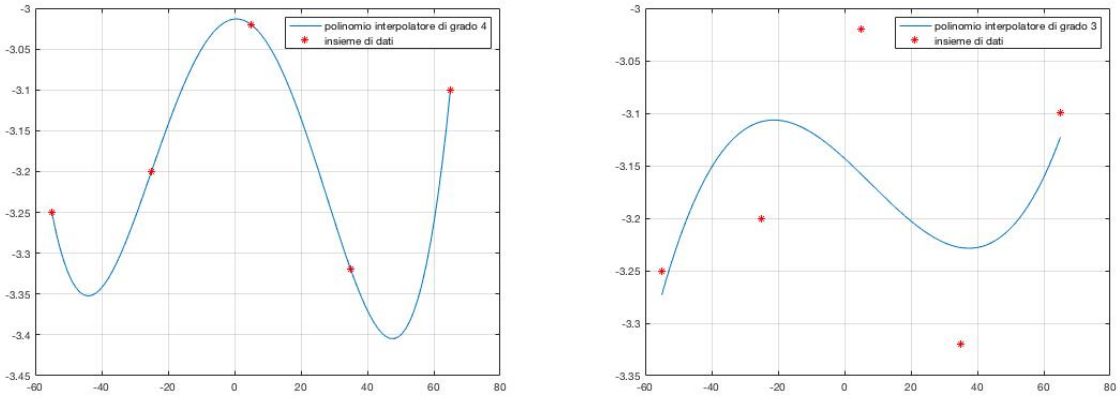

Figure 1: Polinomio interpolatore di grado 4 (sinistra) e polinomio approssimante nel senso dei minimo quadrati di grado 3 (a destra) dei dati dell'Esempio 1.1.

1.3 L'errore di interpolazione

L'oggetto di questa sezione riguarda la domanda: *quanto è accurato il polinomio interpolante di una data funzione*?

Per definizione, il polinomio interpolante coincide con la funzione nei nodi, ma quanto possiamo aspettarci che esso sia una buona approssimazione **globale** della funzione data? L'Esercizio 1.2 riporta un esempio famoso in cui le cose vanno male.

Esercizio 1.2 (Fenomeno di Runge)**.** *Interpolare la* ***funzione di Runge*** $f(x) = \frac{1}{1+x^2}$ *su un insieme di* $n+1$ *nodi equispaziati nell'intervallo* $I = [-5, 5]$ *e commentare i risultati ottenuti dalla rappresentazione grafica al crescere di* n*.*

Soluzione:

```
%troviamo il polinomio interpolatore della funzione di Runge
%f(x)=1/(1+x^2) su un insieme di 13 punti equispaziati tra -5 e 5

n=13
x=linspace(-5,5,n);
f=@(x) 1./(1+x.^2);
y=f(x);
c=polyfit(x,y,n-1);

%rappresentiamo graficamente il polinomio e la funzione
z=linspace(-5,5,1000);
p=polyval(c,z);
plot(x,y,'o--r',z,p);
grid on;
legend('Funzione di Runge 1/(1+x^2)','polinomio interpolatore ...
 di grado 12','FontSize',11,'Location','SouthEast')
```

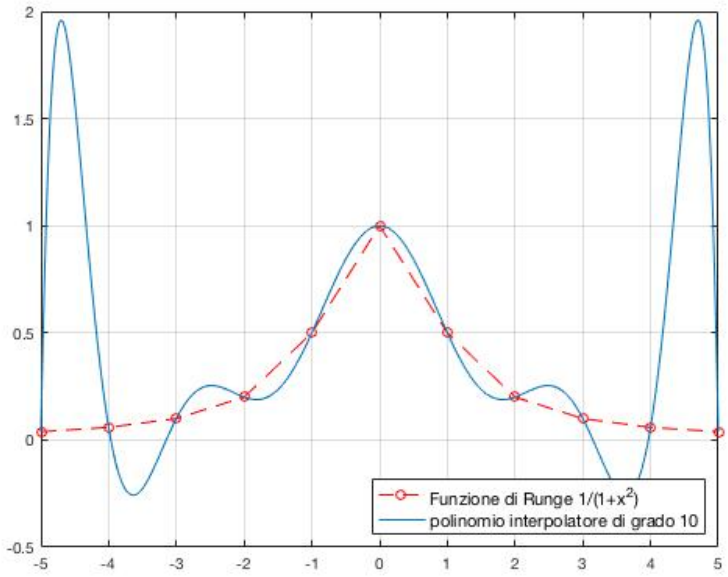

Figure 2: Polinomio interpolatore di grado 10 della funzione di Runge $f(x) = \frac{1}{1+x^2}$ su un insieme di 11 nodi equispaziati nell'intervallo [-5,5].

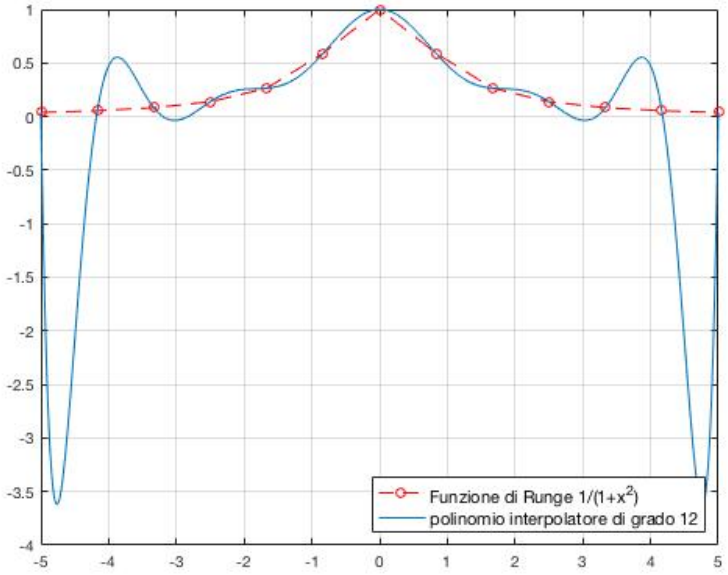

Figure 3: Polinomio interpolatore di grado 12 della funzione di Runge $f(x) = \frac{1}{1+x^2}$ su un insieme di 13 nodi equispaziati nell'intervallo [-5,5].

Cosa va male nell'interpolazione della funzione di Runge? Per rispondere alla domanda, consideriamo un intervallo limitato I che contiene $n+1$ nodi d'interpolazione *distinti* x_i, $i = 0, \ldots, n$ su cui vogliamo interpolare una data funzione continua f. Indichiamo con $E_n f$ **l'errore d'interpolazione** che si commette sostituendo ad f il suo polinomio interpolatore $\Pi_n f$, ossia

$$E_n f(x) = f(x) - \Pi_n f(x). \tag{5}$$

Se f è derivabile con continuità fino all'ordine $n+1$, è valido il seguente risultato che ci permette di quantificare $E_n f(x)$: per ogni $x \in I$, esiste $\xi_x \in I$ tale che

$$E_n f(x) = \frac{f^{(n+1)}(\xi_x)}{(n+1)!} \prod_{i=0}^{n} (x - x_i). \tag{6}$$

Nel caso di una distribuzione uniforme di nodi di passo $h > 0$, si può facilmente dedurre dalla (6) che

$$\max_{x \in I} |E_n f(x)| \leq \frac{\max_{x \in I} |f^{(n+1)}(x)|}{4(n+1)} h^{n+1}. \tag{7}$$

Idealmente, si vorrebbe che la quantità a destra di (7) tendesse a zero per $n \to \infty$, così anche l'errore massimo $\max_{x \in I} |E_n f(x)|$ (quantità a sinistra) tenderebbe a zero quando $n \to \infty$. Questo è valido **se e soltanto se** per $n \to \infty$ l'ordine di infinito di

$$\max_{x \in I} |f^{(n+1)}(x)| \tag{8}$$

non supera quello di infinitesimo di

$$\frac{h^{n+1}}{4(n+1)}. \tag{9}$$

Quindi, purtroppo, il solo fatto che $h^{n+1}/(4(n+1))$ tenda a zero per $n \to \infty$, non è sufficiente per garantire convergenza dell'errore. Può capitare nei casi più svantaggiati che

$$\lim_{n \to \infty} \max_{x \in I} |E_n f(x)| = \infty \tag{10}$$

Di fatto, quando interpoliamo la funzione di Runge su *nodi equispaziati*, si verifica la (10). Questo succede perché per $n \to \infty$ l'ordine di infinito di $|\max_{x \in I} f^{(n+1)}(x)|$ supera quello di infinitesimo di $h^{n+1}/(4(n+1))$. L'Esercizio 1.3 permette di verificare numericamente questa conclusione. La mancanza di convergenza si manifesta in forti oscillazioni del polinomio interpolatore rispetto a quello di f, che si amplificano in prossimità degli estremi dell'intervallo (cf. le Figure 2 e 3). Questo comportamento è noto come **fenomeno di Runge**.

Esercizio 1.3 (Fenomeno di Runge: errore di interpolazione)**.** *Costruire uno script che permetta di calcolare il massimo dei valori assoluti delle derivate della funzione di Runge fino all'ordine 21. Confrontare con il massimo dei valori assoluti di*

$$\frac{\prod_{i=0}^{n}(x - x_i)}{(n+1)!} \tag{11}$$

e commentare sul fenomeno di Runge. Si faccia riferimento alla formula (6) *sull'errore d'interpolazione. Si calcoli infine, con i risultati ottenuti* $\max_{x \in I} |E_n f(x)|$.

Soluzione:

```
%Questo script calcola il massimo dei valori assoluti di f^{(n)}
%per n=1,...,21
syms x;
n=20;
f=1/(1+x^2);

%calcoliamo la derivata prima di f
df=diff(f,1);

%convertiamo la variabile simbolica df in un function handle
cdf=matlabFunction(df);
%in Octave: cdf=function_handle(df)

for i=1:n+1
  df=diff(df,1);
  cdfn=matlabFunction(df);
  x=fzero(cdfn,0);

  %memorizziamo i massimi dei valori assoluti in un vettore M
  M(i)=abs(cdf(x));
  cdf=cdfn;
end

%Questo script calcola il massimo dei valori assoluti
%dell'espressione (11)

%iniziamo concentrandoci sul numeratore di (11), che e' la
%decomposizione di un polinomio che ha come radici

%i valori x0,..xn
z=linspace(-5,5,10000);
for n=0:20
    h=10/(n+1);
    x=[-5:h:5];
    c=poly(x);
    r(n+1)=max(polyval(c,z));

    %aggiorniamo il denominatore di (11)
    r(n+1)=r(n+1)/prod([1:n+1]);
end

%confrontiamo gli ordini di infinito dei valori memorizzati in M
% con gli ordini di infinitesimo dei valori memorizzati in r,
```

```
%ad esempio per n=3,9,15,21
M([3,9,15,21])
r([3,9,15,21])

%per ottenere il massimo del valore assoluto dell'errore
%di interpolazione secondo la (6) e la (7) basta calcolare
M([3,9,15,21]).*r([3,9,15,21])
```

In `format short e`, l'array `M([3,9,15,21])` restituisce i valori

`4.6686e+00, 3.2426e+05, 1.2160e+12, 4.8421e+19,`

mentre l'array `r([3,9,15,21])` restituisce i valori

`1.1574e+01, 5.1814e-02, 1.3739e-05, 4.7247e-10,`

da cui si deduce immediatamente che l'ordine del massimo del valore assoluto dell'errore di interpolazione cresce ad crescere di n. Di fatto, il prodotto `M([3,9,15,21]).*r([3,9,15,21])` restituisce i valori

`5.4034e+01, 1.6801e+04, 1.6706e+07, 2.2877e+10.`

1.4 Interpolazione rispetto ai nodi di Chebyshev

Dopo queste osservazioni, è spontaneo chiedersi: *si può fare qualcosa per garantire convergenza di* $\Pi_n f$ *ad* f*?*

Riscriviamo l'equazione (6) che ci permette di quantificare l'errore d'interpolazione

$$E_n f(x) = \frac{f^{(n+1)}(\xi_x)}{(n+1)!} \prod_{i=0}^{n} (x - x_i). \tag{12}$$

Possiamo osservare che abbiamo **controllo** solo sulla **scelta** dei nodi $\{x_i\}$. Di fatto, il fenomeno di Runge può essere evitato utilizzando opportune distribuzioni di nodi, quali i **nodi di Chebyshev-Gauss-Lobatto** oppure i **nodi di Chebyshev-Gauss**.

Esempio 1.1 (Nodi di Chebyshev-Gauss-Lobatto)**.** *Dato un arbitrario intervallo* $[a, b]$*, i nodi di Chebyshev-Gauss-Lobatto sono definiti come*

$$x_i = \frac{a+b}{2} + \frac{b-a}{x} \hat{x}_i, \quad \hat{x}_i = -cos(\pi i/n), \ i = 0, \ldots, n. \tag{13}$$

Si può fare vedere che $\Pi_n f$ associato a questa distribuzione di nodi tende a f per $n \to \infty$, per ogni $x \in [a, b]$, sotto le ipotesi di continuità e derivabilità con continuità di f in $[a, b]$.

Per definizione, questi nodi sono le ascisse di punti equispaziati sulla semicirconferenza di raggio uno. Utilizzando questa distribuzione di nodi che , e si addensa vicino agli estremi dell'intervallo, al crescere del grado, le oscillazioni che erano presenti nel polinomio interpolatore (cf. le Figure 2 e 3) si smorzano e l'approssimazione diventa sempre più accurata.

La stessa proprietà di convergenza viene garantita quando utilizziamo i nodi di Chebyshev-Gauss.

Esempio 1.2 (Nodi di Chebyshev-Gauss)**.** *Dato un arbitrario intervallo* $[a, b]$*, i nodi di Chebyshev-Gauss sono definiti come*

$$x_i = \frac{a+b}{2} - \frac{b-a}{x} \cos\left(\frac{2i+1}{n+1}\frac{\pi}{2}\right), \ i = 0, \ldots, n. \tag{14}$$

Esercizio 1.4 (Interpolazione della funzione di Runge sui nodi di Chebyshev-Gauss-Lobatto)**.** *Riconsiderare la funzione di Runge e calcolare la sua interpolazione sui nodi di Chebyshev-Gauss-Lobatto* (13)*. Calcolare infine il massimo del valore valore assoluto delle differenze fra* f *ed il suo polinomio interpolatore in* 1000 *punti equispaziati nell'intervallo* $[-5, 5]$*.*

Soluzione:

```
%Questo script considera l'interpolazione della funzione di Runge
%rispetto ai nodi di Chebyshev-Gauss-Lobatto e calcola l'errore
%di interpolazione

%generiamo i nodi
a=-5;
b=5;
n=10;
xc=-cos(pi*[0:n]/n);
x=(a+b)*0.5+(b-a)*xc*0.5;

%costruiamo il polinomio interpolatore
f=@(x) 1./(1+x.^2);
y=f(x);
c=polyfit(x,y,n);

%valutiamo l'errore
z=linspace(-5,5,1000);
p=polyval(c,z);
fz=f(z);
err=max(abs(p-fz))
```

1.5 Altri tipi di interpolazioni

2 Ricerca degli zeri di funzioni non lineari

Data una funzione reale di variabile reale $f : [a, b] \to \mathbb{R}$, siamo interessati a trovare le **radici** $\alpha \in [a, b]$ tale che $f(\alpha) = 0$. Il valore α viene detto uno **zero** di f. Richiamiamo il

noto *Teorema degli zeri di una funzione continua* che afferma che se f è continua in $[a,b]$ e $f(a)f(b)<0$, allora esiste almeno uno $\alpha\in(a,b)$ tale che $f(\alpha)=0$.

Abbiamo visto nelle Lezioni 3-4 il comando `fzero` per il calcolo delle radici di una generica funzione.

In generale è impossibile avere metodi numerici che calcolino gli zeri di una generica funzione in un numero finito di passi. Di conseguenza, essi sono necessariamente **iterativi**: partendo da uno o più dati iniziali scelti in maniera conveniente, si genera una successione di valori che convergono ad uno zero della funzione data, sotto opportune ipotesi.

2.1 Il metodo di bisezione

Supponiamo di considerare una funzione continua su $[a,b]$ che soddisfa le ipotesi del teorema degli zeri e che cambi di segno agli estremi, cosìcché siamo certi che esiste almeno uno zero α in $[a,b]$. Supponiamo inoltre per semplicità questo zero sia unico; in caso contrario, tramite uno studio grafico, ci si può ricondurre ad un sottointervallo di $[a,b]$ che contenga soltanto uno zero.

Esempio 2.1 (Il metodo di bisezione)**.** *Data $f:[a,b]\to\mathbb{R}$ continua tale che $f(a)f(b)<0$, si ponga $a^{(0)}=a$, $b^{(0)}=b$, $I(0)=(a^{(0)},b^{(0)})$, $x^{(0)}=(a^{(0)}+b^{(0)})/2$. Si calcoli per $k\geq 1$, il semi-intervallo $I^{(k)}=(a^{(k)},b^{(k)})$ di $I^{(k-1)}=(a^{(k-1)},b^{(k-1)})$ tramite i seguenti passi*

1. *calcolare $x^{(k-1)}=(a^{(k-1)}+b^{(k-1)})/2$;*
2. *se $f(x^{(k-1)})=0$, allora $\alpha=x^{(k-1)}$ ed il metodo si arresta;*
3. *altrimenti*
   - *(a) se $f(a^{(k-1)})f(a^{(k-1)})<0$, si pone $a^{(k)}=a^{(k-1)}$ e $b^{(k)}=x^{(k-1)}$;*
   - *(b) se $f(x^{(k-1)})f(b^{(k-1)})<0$, si pone $a^{(k)}=x^{(k-1)}$ e $b^{(k)}=b^{(k-1)}$;*
4. *definire $x^{(k)}=(a^{(k)}+b^{(k)})/2$ e incrementare k di uno.*

Prima di scrivere un `M`-file per il metodo di bisezione, facciamo alcune osservazioni

- ⋆ la strategia del metodo consiste nel selezionare ad ogni passo il semi-intervallo in cui f cambia di segno agli estremi, così si è certi che ogni intervallo selezionato conterrà lo zero α;
- ⋆ siccome la lunghezza dei sotto-intervalli $|I^{(k)}|=b^{(k)}-a^{(k)}$ tende a 0 per $k\to\infty$ (dimenzzandosi ogni volta), si è certi anche che la successione dei punti medi $\{x^{(k)}\}$ dei sotto-intervalli, **convergerà** ad α;
- ⋆ l'**errore** al passo k soddisfa

$$e^{(k)}=|x^{(k)}-\alpha|<\frac{1}{2}|I^{(k)}|=\left(\frac{1}{2}\right)^{k+1}(b-a), \tag{15}$$

quindi, fissata una tolleranza ε si può facilmente dimostrare che come **criterio d'arresto** è sufficiente fermarsi dopo $k_{\min}$ iterazioni, essendo $k_{\min}$ il primo intero che soddisfa la disuguaglianza

$$k_{\min} > \log_2\left(\frac{\varepsilon}{b-a}\right). \tag{16}$$

```
%Questo M-file approssima uno zero della funzione f
%nell'intervallo [a,b] con il metodo di bisezione.
%La funzione f deve essere definita su variabile di tipo array
e puo' essere una anonymous function oppure una funzione definita
%in un M-file

%input

%output

%verifiche iniziali
function [zero,res,niter]=bisezione(f,a,b,tol,kmax)
x = [a, (a+b )*0.5, b];
fx = f(x);
if fx(1)*fx(3) > 0
     error('Il segno di f agli estremi di [a,b] deve essere
       ...diverso');
elseif fx(1) == 0
     zero = a; res = 0; niter = 0;
return
     elseif fx(3) == 0
zero = b; res = 0; niter = 0; return
end

niter = 0;
I = (b - a)*0.5;
     while I >= tol && niter < kmax
          niter = niter + 1;
          if fx(1)*fx(2) < 0
               x(3) = x(2);
               x(2) = x(1)+(x(3)-x(1))*0.5;
               fx = f(x);
               I = (x(3)-x(1))*0.5;
          elseif fx(2)*fx(3) < 0
               x(1) = x(2);
               x(2) = x(1)+(x(3)-x(1))*0.5;
               fx = f(x);
```

```
            I = (x(3)-x(1))*0.5;
        else
            x(2) = x(find(fx==0)); I = 0;
        end
    end
if (niter==kmax && I > tol)
    fprintf('Il metodo di bisezione si e'' arrestato',...
    'senza soddisfare la tolleranza richiesta,...
    'avendo raggiunto il numero massimo di iterazioni');
end
zero = x(2);
x = x(2);
res = fun(x);
```

Link alle lezioni precedenti:

Lezioni 13-15
Lezioni 11-12
Lezioni 9-10
Lezioni 7-8
Lezioni 5-6
Lezioni 3-4
Lezioni 1-2

Referenze

1. MATLAB® The language of technical computing: computation, visualization, programming, The MathWorks Inc
2. MATLAB: An introduction with applications, A. Gilat, Wiley
3. Scientific Computing with MATLAB and Octave, A. Quarteroni, , F. Saleri, P. Gervasio, Springer
4. MATLAB Guide1 D. J. Higham and N. J. Higham, SIAM
5. Numerical Computing with MATLAB, C. Moler, SIAM