Laboratorio Didattico di Matematica Computazionale

a.a. 2018-2019

Esercitazione 7

Interpolazione e fit di dati

Le tecniche di interpolazione o *data fitting* si usano per risolvere problemi come il seguente. Supponiamo di avere un insieme di *nodi di interpolazione*, cioè punti $\{x_i\}_{i=0,\dots,n}$, con $x_i < x_{i+1}$, in un intervallo $[a, b]$ della retta reale, e di conoscere i valori $y_i = f(x_i)$ di una certa funzione f (non nota) in corrispondenza dei nodi. Vogliamo approssimare i valori di f in altri punti di $[a, b]$.

L'idea è determinare una funzione g che imiti f in modo "plausibile" e in particolare assuma i valori prescritti sugli x_i, in modo esatto (interpolazione) o approssimato (fit). Una volta determinata g, la si valuta nei nuovi punti.

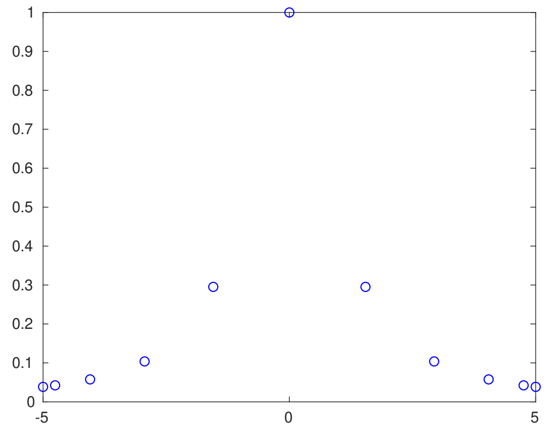
1. Interpolazione

Per eseguire l'interpolazione in Matlab useremo il comando `interp1`. Vediamo alcuni casi interessanti.

La tecnica di interpolazione più semplice è l'*interpolazione lineare*: prendiamo g come la funzione lineare a tratti che collega i punti (x_i, y_i).

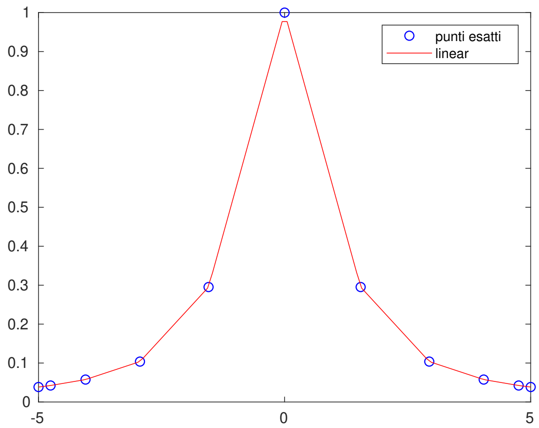
Per esempio, supponiamo di avere $f(x) = 1/(1 + x^2)$ e di conoscere i valori di f nei nodi $x_i = 5\cos(i/\pi)$ per $i = 0, \dots, 11$. Costruiamo l'esempio in Matlab e disegnamo i punti da interpolare:

```
n=11;
x=5*cos(linspace(0, pi, n));
f=@(t) 1./(1+t.^2);
y=feval(f,x);
plot(x,y,'bo')
```



Ora cerchiamo un'approssimazione di f in 100 punti equispaziati sull'intervallo $[-5, 5]$, usando l'interpolazione lineare:

```
m=100;
interpx=linspace(-5,5,m);
interpy=interp1(x,y,interpx,'linear');
hold on
plot(interpx,interpy,'r')
legend('punti esatti', 'linear')
```

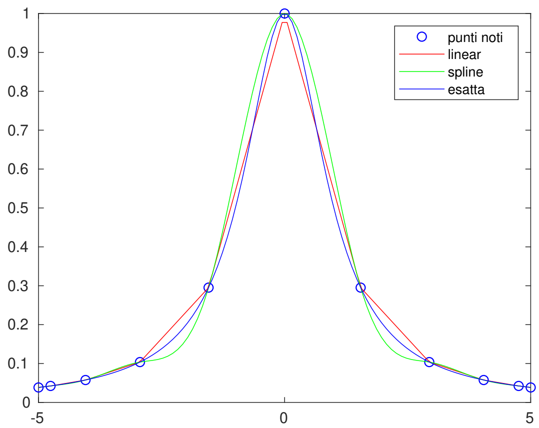


Un tipo di interpolazione molto diffuso è l'interpolazione con *spline cubiche*. In questo caso g viene scelta come una funzione polinomiale a tratti, con polinomi di grado al più 3, e raccordi C^2 nei nodi. Proviamo ad applicare questo tipo di interpolazione al nostro esempio:

```
interpysp=interp1(x,y,interpx,'spline');
plot(interpx, interpysp, 'g')
legend('punti noti', 'linear', 'spline')
```

Possiamo aggiungere al grafico anche i valori esatti della funzione:

```
yesatta=feval(f,interpx);
plot(interpx,yesatta,'b')
legend('punti noti', 'linear', 'spline', 'esatta')
```



Provate anche ad usare `interp1` con l'opzione `'cubic'` o `'pchip'`, che produce un'interpolazione C^1 cubica a tratti di tipo Hermite che rispetta la monotonia. Riuscite a costruire un esempio in cui le opzioni `'spline'` e `'pchip'` diano risultati significativamente diversi?

Esercizio 1 *Scrivere uno script Matlab che calcoli e tracci in scala semilogaritmica l'errore di approssimazione in ciascun punto di* `interpx`*, per ciascuna delle tre tecniche di interpolazione viste (lineare, spline cubica, Hermite cubica). L'errore di approssimazione in un punto è il valore assoluto della differenza tra il valore della funzione esatta e il valore della funzione di interpolazione in quel punto. Che cosa osservate?*

Un'altra tecnica per interpolare $n+1$ punti dati (x_i, y_i) consiste nella scelta di g come l'unico polinomio di grado n che assume i valori prescritti y_i nei nodi x_i (*interpolazione polinomiale*). Un modo conveniente per esprimere il polinomio interpolatore è dato dai *polinomi di Lagrange.*

Ricordiamo che i polinomi di Lagrange rispetto ai nodi $x_0, \dots, x_n$ sono definiti come

$$L_i(x) = \prod_{j=0,\dots,n;\, j\neq i} \frac{x - x_j}{x_i - x_j}, \qquad i = 0, \dots, n$$

e formano una base dello spazio dei polinomi di grado al più n.

Si dimostra che il polinomio di interpolazione $p(x)$ si può scrivere nella base di Lagrange come

$$p(x) = \sum_{i=0}^{n} y_i L_i(x).$$

Esercizio 2 *Scrivere una function* `v=lagrange(x,y,interpx)` *che prenda in ingresso*

- *il vettore* `x` *di lunghezza n contenente i nodi di interpolazione,*
- *il vettore* `y` *di lunghezza n contenente i valori da interpolare in corrispondenza dei nodi,*
- *il vettore* `interpx` *di lunghezza m contenente i punti sui quali vogliamo valutare il polinomio interpolatore,*

e restituisca in output il vettore `v` *di lunghezza m contenente i valori del polinomio interpolatore sui punti di* `interpx`*. Suggerimento: valutate direttamente i polinomi di Lagrange su* `interpx`*, senza determinare il polinomio interpolatore in base monomiale.*

Esercizio 3 *Scrivere uno script che usi la function* `lagrange` *appena definita per risolvere il problema dell'interpolazione sull'esempio visto in questa sezione, e disegni l'errore di approssimazione come nell'Esercizio 1.*

Esercizio 4 *(facoltativo) Sia $f(x) = 1/(1+25x^2)$ definita su $[-1,1]$. Eseguite e disegnate delle interpolazioni polinomiali di grado crescente, su nodi equispaziati. Che cosa osservate? L'approssimazione migliora al crescere del grado?*

2. Fit di dati

Dati i punti (x_i, y_i), $i = 1, \dots, m$ e un intero positivo $n < m$, il problema del fit polinomiale consiste nel determinare un polinomio $p(x)$ di grado n tale che $p(x_i) \approx y_i$ nel senso dei minimi quadrati, cioè in modo da minimizzare la quantità

$$(p(x_1) - y_1)^2 + (p(x_2) - y_2)^2 + \cdots + (p(x_m) - y_m)^2.$$

In Matlab il calcolo dei coefficienti di $p(x)$ si può effettuare con il comando

`p=polyfit(x,y,n)`

dove `x` è il vettore degli x_i e `y` è il vettore degli y_i.

Esercizio 5 *Si consideri la funzione* $f(x) = 1/(x+(1-x)^2)$ *sull'intervallo* $[-2, 2]$. *Si valuti* $f(x)$ *in 20 punti equispaziati* x_i, $i = 1, \ldots, 20$ *sull'intervallo* $[-2, 2]$ *e si calcolino i coefficienti del polinomio* $p(x)$ *di grado 3 tale che* $p(x_i) \approx f(x_i)$ *nel senso dei minimi quadrati. Si tracci nella stessa figura il grafico di* $f(x)$ *e quello di* $p(x)$. *Si ripeta poi l'operazione per* $n = 5, 8, 10$, *sempre nella stessa figura. È vero che all'aumentare del grado del polinomio migliora l'approssimazione della funzione?*

Esercizio 6 *Come nell'esercizio precedente, sia* $f(x) = 1/(x+(1-x)^2)$ *definita sull'intervallo* $[-2, 2]$, *e siano gli* x_i *definiti come sopra. Consideriamo 100 punti* $z_1, \ldots, z_{100}$ *equidistanziati nell'intervallo. Per le approssimazioni seguenti, valutare gli errori di approssimazione sugli* z_i *e tracciarne il grafico:*

- `polyfit` *con* $n = 5$,
- `interp1` *con opzioni lineare e spline,*
- *interpolazione polinomiale con la function* `lagrange` *dell'Esercizio 2.*