Capitolo 4

Approssimazione

Disponendo di un set di dati del tipo (x, y) discreti, come ad esempio delle misurazioni, voglio indovinare i punti corrispondenti a dati che non sono stati direttamente misurati.
Ci sono due metodi:

- interpolazione: faccio passare una curva esattamente per tutti i punti misurati;
- approssimazione: ipotizzando che i dati siano affetti da errore (come minimo dovuti alla rappresentazione nel calcolatore), cerco una curva che si avvicini quanto più possibile ai dati.

4.1 Approssimazione (o regressione)

Si cerca una curva il più possibile semplice $y = g(x)$ che si avvicini maggiormente ai dati (x_i, y_i) $i = 1...m$. Questa curva è detta *modello*: è una $g(x)$ che assume ad esempio una forma come $c_0 + c_1 x \left(+c_2 x^2\right)$ oppure $c_0 + c_1 \cos x + c_2 \sin x$, le incognite sono i coefficienti. Quando il modello dipende linearmente dai coefficienti (ovvero i coefficienti non vengono ad esempio moltiplicati fra loro) si dice che è *lineare*.

4.1.1 Approssimazione ai minimi quadrati

Assumendo che $y_i \neq g(x_i)$, considero i quadrati degli scarti tra dati e modello, la funzione scelta sarà tale che

$$\min_{c_0, c_1 \ldots} \sum_{i=1}^{m} \left[y_i - g(x_i)\right]^2$$

Date le *funzioni di base* $g_1(x), g_2(x), ..., g_n(x)$ vado a cercare la curva di regressione nel sottospazio vettoriale generato da tali funzioni:

$$g(x) \in \langle g_1, ..., g_n \rangle = \{c_1 g_1(x) + c_2 g_2(x) + ... + c_n g_n(x)\}$$

E' comodo trattare il problema con oggetti di algebra lineare.

Definisco $\underline{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_m \end{pmatrix}$ la colonna dei dati, $\underline{c} = \begin{pmatrix} c_1 \\ \vdots \\ c_n \end{pmatrix}$ la colonna dei coefficienti incogniti e la matrice dei minimi quadrati $A = \begin{pmatrix} g_1(x_1) & g_2(x_1) & \cdots & g_m(x_1) \\ g_1(x_2) & g_2(x_2) & & \\ \vdots & & & \\ g_1(x_n) & & & g_m(x_n) \end{pmatrix}$; l'ordine con cui si fissano i coefficienti in c deve essere rispettato anche nella matrice A.

Nel caso della retta $A = \begin{pmatrix} 1 & x_1 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix}$, nel caso della parabola $A = \begin{pmatrix} 1 & x_1 & x_1^2 \\ \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 \end{pmatrix}$, nel caso trigonometrico $A = \begin{pmatrix} 1 & \cos x_1 & \sin x_1 \\ \vdots & \vdots & \vdots \\ 1 & \cos x_n & \sin x_n \end{pmatrix}$.

$g(x_i) = (i - \text{esima riga di } A) \cdot \underline{c}$

Quindi gli scarti sono $\underline{y} - A\underline{c} =: \underline{r}$ dove $\underline{r}$ viene chiamato *residuo*.

Il problema diventa trovare il vettore $\underline{c} \in \mathbb{R}^n$ tale che

$$\min_{\underline{c} \in \mathbb{R}^n} \left\| \underline{y} - A\underline{c} \right\|_2^2$$

Osservazione: cercare $\underline{r} = 0$ riconduce a trovare le soluzioni del sistema rettangolare $m \times n$ $A\underline{c} = \underline{y}$ che è sovradeterminato (non è pertanto sempre risolvibile); quindi ci si accontenta di trovare il minimo residuo possibile.

4.1.1.1 Algoritmo di calcolo via fattorizzazione QR

Supponiamo di essere in grado di calcolare la fattorizzazione QR della matrice A ($A = Q \begin{pmatrix} R \\ 0 \end{pmatrix}$) con costo $n^2 \left(m - \frac{n}{3} \right)$.

$\underline{r} = \underline{y} - Q \begin{pmatrix} R \\ 0 \end{pmatrix} \underline{c} = QQ^T \underline{y} - Q \begin{pmatrix} R\underline{c} \\ 0 \end{pmatrix} = Q \left\{ Q^T y - \begin{pmatrix} R\underline{c} \\ 0 \end{pmatrix} \right\}$ avendo moltiplicato Q per la trasposta essendo ortogonale.

Considero nel calcolo successivo $Q^T y =: \begin{pmatrix} h_1 \\ h_2 \end{pmatrix}$, ambedue i componenti vettori ($h_1 \in \mathbb{R}^n$, $h_2 \in \mathbb{R}^{m-n}$).

Se Q è una isometria allora $\|\underline{r}\|_2^2 = \left\| Q \left\{ Q^T y - \begin{pmatrix} R\underline{c} \\ 0 \end{pmatrix} \right\} \right\|_2^2 = \left\| Q^T y - \begin{pmatrix} R\underline{c} \\ 0 \end{pmatrix} \right\|_2^2 = \left\| \begin{pmatrix} h_1 - R\underline{c} \\ h_2 \end{pmatrix} \right\|_2^2 = \|h_1 - R\underline{c}\|_2^2 + \|h_2\|_2^2$. Quindi

$$\min_{\underline{c} \in \mathbb{R}^n} \left\| \underline{y} - A\underline{c} \right\|_2^2 = \min_{\underline{c} \in \mathbb{R}^n} \left\{ \|h_1 - R\underline{c}\|_2^2 + \|h_2\|_2^2 \right\}$$

Facciamo l'ipotesi che A sia a rango pieno ($\operatorname{rk}(A) = n \Rightarrow \det R \neq 0$): se $\underline{c}$ risolve il sistema $n \times n$ $R\underline{c} = h_1 \Rightarrow \left\| \underline{y} - A\underline{c} \right\|_2^2 = \|h_2\|_2^2$, e non può scendere sotto questo valore. Abbiamo quindi trovato il minimo.

Algoritmo 4.1 Algoritmo di approssimazione ai minimi quadrati

1) calcolo della fattorizzazione QR (costo $n^2\left(m-\frac{n}{3}\right)$, è il passo che costa di più)
2) calcolo di $Q^T y = \begin{pmatrix} h_1 \\ h_2 \end{pmatrix}$ (costo mn)
3) se $\mathrm{rk}(A) = n$, risolvo $R\underline{c} = h_1$ che restituisce $\underline{c}$; questo passo, essendo R triangolare superiore, richiede solo la sostituzione all'indietro, quindi costo $\frac{n^2}{2}$
4) $\|\underline{r}\|_2 = \|h_2\|_2$ (costo $m-n$)

4.1.1.2 Algoritmo di calcolo "geometrico"

Si nota che il vettore $A\underline{c}$ appartiene all'immagine di A, e al variare di $\underline{c}$ esso percorre tutto il sottospazio:

$$\{A\underline{c} : c \in \mathbb{R}^n\} \equiv \mathfrak{R}(A) =< Ae_1, ..., Ae_n >$$

Il vettore che ci interessa sarà quello corrispondente alla proiezione sul sottospazio del vettore degli input.

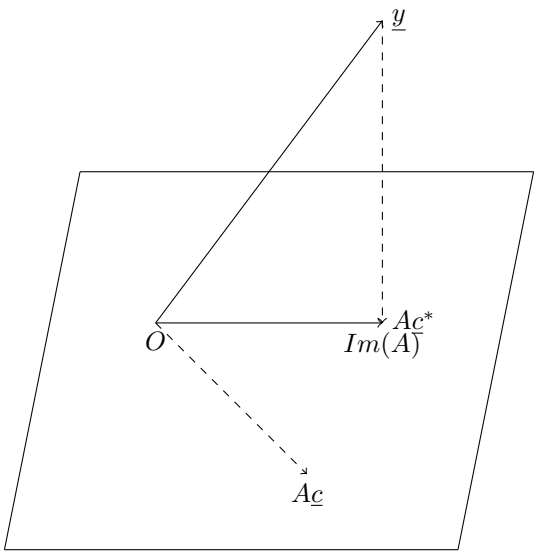

Figura 4.1: Interpretazione geometrica

Il vettore $\underline{r} = \underline{y} - A\underline{c}^*$ è quindi il residuo.
Si ha anche che $\underline{r}$ è perpendicolare a tutte le colonne di A: $(Ae_i)^T \underline{r} = 0$ ovvero

$$\begin{pmatrix} a_{11} & a_{21} & \cdots & a_{m1} \\ a_{12} & & & \\ \vdots & & \ddots & \\ a_{1n} & & & a_{mn} \end{pmatrix} \underline{r} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \text{ da cui } A^T(y - Ac^*) = 0 \Rightarrow A^T y - A^T A c^* = 0 \Rightarrow$$

$$\left(A^T A\right) c^* = \left(A^T y\right)$$

dette *equazioni normali* (o di *Eulero*)

Algoritmo 4.2 Algoritmo "geometrico"

1) calcolo A^TA (costo $\frac{1}{2}mn^2$)
2) calcolo A^Ty (costo mn)
3) risolvo il sistema lineare $n \times n$ $\left(A^TA\right)c = \left(A^Ty\right)$ (costo $\frac{n^3}{3}$ perchè si applica anche Gauss)
Non dà il valore del residuo minimo

L'algoritmo funziona se il sistema è risolubile, si dimostra che se $\mathrm{rk}\,(A) = n \Rightarrow \det\left(A^TA\right) \neq 0$, in totale costa la metà dell'algoritmo tramite fattorizzazione QR.

4.1.1.3 Esempi

Retta di regressione $g\,(x) = \alpha x + \beta$

$$A = \begin{pmatrix} x_1 & 1 \\ \vdots & \vdots \\ x_m & 1 \end{pmatrix} \underline{c} = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

$$y = \begin{pmatrix} y_1 \\ \vdots \\ y_m \end{pmatrix}$$

Dato l'input, calcolare i coefficienti.

x_i	0	1	2	3
y_i	1	2.1	2.9	3.2

$$A = \begin{pmatrix} 0 & 1 \\ 1 & 1 \\ 2 & 1 \\ 3 & 1 \end{pmatrix} y = \begin{pmatrix} 1 \\ 2.1 \\ 2.9 \\ 3.2 \end{pmatrix} A^T = \begin{pmatrix} 0 & 1 & 2 & 3 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

$$A^TA = \begin{pmatrix} 14 & 6 \\ 6 & 4 \end{pmatrix} A^Ty = \begin{pmatrix} 17.5 \\ 9.2 \end{pmatrix}$$

$$\begin{cases} 14\alpha + 6\beta = 17.5 \\ 6\alpha + 4\beta = 9.2 \end{cases} \quad \begin{cases} \alpha = 0.74 \\ \beta = 1.19 \end{cases}$$

Si vede una proprietà generale:

$$A^TA = \begin{pmatrix} \sum x_i^2 & \sum x_i \\ \sum x_i & m \end{pmatrix} A^Ty = \begin{pmatrix} \sum x_iy_i \\ \sum y_i \end{pmatrix}$$

Queste fromula, divise per m, assumono un significato statistico (medie, varianze, covarianze...).

Parabola di regressione $g\,(x) = \alpha x^2 + \beta x + \gamma$

x_i	0	1	2	3
y_i	0	0	1	2

$$\underline{c} = \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix}$$

$$A = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \\ 4 & 2 & 1 \\ 9 & 3 & 1 \end{pmatrix} \quad y = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 2 \end{pmatrix}$$

$$A^T A = \begin{pmatrix} 0 & 1 & 4 & 9 \\ 0 & 1 & 2 & 3 \\ 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \\ 4 & 2 & 1 \\ 9 & 3 & 1 \end{pmatrix} = \begin{pmatrix} 98 & 36 & 14 \\ 36 & 14 & 6 \\ 14 & 6 & 4 \end{pmatrix}$$

$$A^T y = \begin{pmatrix} 22 \\ 8 \\ 3 \end{pmatrix}$$

Curva trigonometrica di regressione $g(x) = \alpha + \beta \cos x + \gamma \sin x$

x_i	$-\frac{\pi}{2}$	0	$\frac{\pi}{2}$	π
y_i	1	0	$\frac{1}{2}$	1

$$\underline{c} = \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix}$$

$$A = \begin{pmatrix} 1 & 0 & -1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & -1 & 0 \end{pmatrix}$$

$$A^T A = \begin{pmatrix} 4 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{pmatrix} \quad A^T y = \begin{pmatrix} \frac{5}{2} \\ -1 \\ -\frac{1}{2} \end{pmatrix}$$

$$\begin{cases} \alpha = \frac{5}{2} \\ \beta = -\frac{1}{2} \\ \gamma = -\frac{1}{4} \end{cases} \rightarrow g(x) = \frac{5}{2} - \frac{1}{2}\cos x - \frac{1}{4}\sin x$$

Esercizi Trovare la retta e la parabola di regressione per

x_i	-1	0	1	2
y_i	0	1	2	4

Trovare i parametri per $g(x) = \alpha x + \beta x^2 + \gamma x^3$ dati

x_i	1	2	3	4
y_i	0.5	2	4	8

4.2 Interpolazione

Diversamente dal caso precedente, dobbiamo fidarci ciecamente dei dati e dobbiamo cercare una curva esatta. Ogni punto in input costituisce un vincolo preciso. Questo metodo si usa specialmente in applicazioni grafiche.

I dati sono i punti (x_i, y_i), il modello è $g(x) : g(x_i) = y_i \; \forall i = 0, ..., n$: ci sono $n+1$ equazioni quindi il modello deve avere almeno $n+1$ gradi di libertà.

4.2.1 Polinomio interpolatore

Cerchiamo i coefficienti di $g(x) = c_0 + c_1x + c_2x^2 + ... + c_nx^n$ in modo tale che la curva passi per i punti dati. Si dimostra che questo polinomio interpolatore si trova sempre. Ma non è granchè adatto perchè già a partire dal settimo-ottavo grado il polinomio genera un grafico con troppe oscillazioni, mentre con un grado più basso funziona abbastanza bene.

Il trucco è quindi cercare tanti polinomi di grado basso raccordati tra loro, per esempio posso unire i punti con segmenti (polinomi di grado uno) però in quel caso ci sarebbero spigoli ($g'(x)$ non continua). Per evitarlo dovremmo garantire la continuità delle derivate e mantenere un grado non superiore a tre.

4.2.2 Interpolazione tramite funzioni spline

$S(x)$ è una *spline* in $[a,b]$ (con $a = x_0 < x_1 < ... < x_{n-1} < x_n = b$) se:

1. $S\backslash_{[x_{i-1},x_i]}$ (S ristretta all'intervallo $[x_{i-1}, x_i]$ è un polinomio di grado non superiore a 3
2. $S \in \mathcal{C}^2[a,b]$ (S è continua con le sue derivate S' e S'')
3. S è una *spline interpolante* se $S(x_i) = y_i$

Esempio $S(x) = \begin{cases} x^3 & x \geqslant 0 \\ 0 & x \leqslant 0 \end{cases}$ è una spline sulle ascisse $x_0 = -1$, $x_1 = 0$, $x_2 = 1$

- L'espressione è un polinomio di grado zero o di grado tre.
- Le due funzioni sono $\mathcal{C}^\infty$ nel loro sottointervallo; bisogna verificare la continuità (anche per le derivate) nei punti di connessione usando i limiti sinistro e destro.

S continua in $0 \iff \lim_{x\to 0^-} S(x) = \lim_{x\to 0^+} S(x)$

$$S'(x) = \begin{cases} 3x^2 & x \geqslant 0 \\ 0 & x \leqslant 0 \end{cases}$$

$$S''(x) = \begin{cases} 6x & x \geqslant 0 \\ 0 & x \leqslant 0 \end{cases}$$

4.2.2.1 Spline parametrizzate

Definisco per brevità $S_i := S\backslash_{[x_{i-1},x_i]}$.

Si può decidere di parametrizzare la spline così:

$$S_i(x) = \alpha_i x^3 + \beta_i x^2 + \gamma_i x + \delta_i$$

I gradi di libertà sono $4n$, dove n è il numero di tratti in cui è scomposta la spline. $n+1$ vincoli sono posti dall'essere interpolante.

I punti critici sono i punti intermedi: si può riscrivere la condizione 2) come

$\forall i = 1...n-1 \ \lim_{x\to x_i^-} S(x) = \lim_{x\to x_i^+} S(x) \iff S_i(x_i) = S_{i+1}(x_i)$

Equazioni simili vanno a garantire la continuità della prima e seconda derivata: $S_i'(x_i) = S_{i+1}'(x_i)$ e $S_i''(x_i) = S_{i+1}''(x_i)$, in totale sono $3(n-1)$ equazioni.

Riassumendo ci sono $4n-2$ equazioni in $4n$ incognite. Si deduce che la soluzione non è unica qindi $\exists\infty$ spline interpolanti.

4.2.2.2 Categorie di spline date le condizioni al contorno

L'unicità della spline è garantita da due condizioni aggiuntive, dette *condizioni al contorno*, che portano a tre tipi di spline:

1. Spline completa: $S'(a) = y_a^{'}$, $S'(b) = y_b^{'}$ dove $y_a^{'}$ e $y_b^{'}$ sono dati (altrimenti non conviene)
2. Spline periodica: $S'(a) = S'(b)$ e $S''(a) = S''(b)$ se vale che $y_0 = y_n$ (i valori iniziale e finale sono alla stessa altezza, quindi è replicabile)
3. Spline naturale: $S''(a) = S''(b) = 0$ definibile in tutti casi ma meno accurata.

4.2.2.3 Spline parametrizzate tramite momenti

Il modo più efficiente di calcolare la spline è usare altri parametri detti *momenti* che costituiscono le incognite: $M_i = S''(x_i)\ \ i = 0...n$

Il grafico di S'' è una spezzata continua.

Scrivo l'equazione della retta passante per due punti (x_{i-1}, M_{i-1}) e (x_i, M_i):

$$S_i''(x) = M_{i-1}\frac{x_i - x}{x_i - x_{i-1}} + M_i\frac{x - x_{i-1}}{x_i - x_{i-1}}$$

Pongo $h_i := x_i - x_{i-1}$e integro:

$S_i'(x) = \int S_i''(x)\,dx = \frac{-(x_i-x)^2 M_{i-1} + (x-x_{i-1})^2 M_i}{2h_i} + C_i$ (costante arbitraria)

$S_i(x) = \int S_i'(x)\,dx = \frac{(x_i-x)^3 M_{i-1} + (x-x_{i-1})^3 M_i}{6h_i} + C_i(x - x_{i-1}) + D_i$ (altra costante arbitraria)

Adesso si è ottenuto S_i rispetto ai momenti. Imponendo i vincoli d'interpolazione posso ricavare in funzione dei momenti il valore delle costanti C_i e D_i. Conoscendo i momenti ho la descrizione di S.

Consideriamo $S_i'(x_i) = S_{i+1}'(x_i)$ per $i = 1...n-1$. Variando i si ritrovano $n-1$ equazioni ognuna delle quali coinvolge solo tre incognite consecutive ($M_{i-1}M_i$ per il primo membro, M_iM_{i+1} per il secondo). Ne viene fuori un sistema lineare di $(n-1)$ equazioni in $(n+1)$ incognite, a cui vanno aggiunte le due condizioni al contorno; la condizione di spline naturale diminuisce invece il sistema di due incognite (perchè si scrive $M_0 = M_n = 0$).

Algoritmo 4.3 Calcolo della spline con i momenti

1. Risolvo il sistema ed ottengo $M_0, ..., M_n$
2. Calcolo $C_i, D_i\ i = 1, \ldots, n$
3. Valuto $S_i(x)$ in x scegliendo i in modo tale che $x \in [x_{i-1}, x_i]$

Il costo dell'algoritmo è lineare in n.

Al passo 1 la matrice che moltiplica il vettore dei momenti è infatti una matrice tridiagonale (e Gauss è più economico).

4.2.2.4 Proprietà delle spline

Proprietà computazionali Il costo per la costruzione è $O(n)$, l'algoritmo è stabile e il problema è ben condizionato (perturbando i dati, il valore approssimato col metodo polinomiale varia di molto, mentre con la spline mantiene solo un piccolo errore).

Errore analitico Considero una $f(x) = \cos x$ e un intervallo fisso $[a, b]$ diviso in n intervalli di ampiezza $h = \frac{b-a}{n}$, genera un insieme $\{x_i\}_{i=0}^{n}$ da cui si ricava $y_i = f(x_i)$.

Vado a definire la spline interpolante $S(x_i) = y_i$, invece di valutare $f(x)$ valuto $S(x)$: quanto vale l'errore analitico generato dalla sostituzione?

$$|f(x) - S(x)|$$

La soluzione per diminuire questo errore sta nell'aumentare il numero di intervalli. La domanda è quindi: come dipende l'errore analitico da n (e quindi da h)?

Studio la convergenza dell'approssimazione: $\lim_{h\to 0} |f(x) - S(x)|$

Teorema: se $f \in \mathcal{C}^4[a, b] \Rightarrow \forall p = 0, 1, 2, 3\ \exists c_p > 0 : \left|f^{(p)}(x) - S^{(p)}(x)\right| \leqslant c_p h^{4-p}\ \forall x \in [a, b]$

Se f ha tre derivate continue, allora vale la disuguaglianza che maggiora l'errore analitico, che tende ad essere piccolo per h piccoli. S è nel teorema una spline completa.

Proprietà di minima curvatura Data $g(x) \in \mathcal{C}^2[a, b]$ la *curvatura media* è $\int_a^b [g''(x)]^2 dx$: ad esempio per una retta è 0.

Teorema di minima curvatura: sia $X = \left\{g \in \mathcal{C}^2[a, b] : g(x_i) = y_i\right\}$ allora la spline naturale $S(x)$ risolve il problema $\min_{g\in X} \int_a^b [g''(x)]^2 dx$

La spline completa ha massima precisione, la spline naturale ha minime oscillazioni.

4.3 Esercitazione con Matlab

Algoritmo 4.4 Operazioni su matrici

$B = ones(3)$ Matrice di tutti uno
$null(B)$ Nucleo di B
$orth(B)$Immagine di B
$rank(B)$Rango di B
clc Clear

Algoritmo 4.5 Esercizio sui minimi quadrati per una retta ed una parabola

$t = (1900 : 10 : 1990)$ Vettore spaziato di 10 sulle ascisse
$p = ()$ Dati del censimento della popolazione americana
$uno = ones\,(size\,(t))$ Matrice di tutti uno con la dimensione di t
$A = [uno\ t]$ Matrice composta da colonna di uno e da t
$AtA = A' * A$ A trasposto per A
$Atb = A' * p$
$format\ short\ e$ passaggio alla notazione esponenziale
$c = AtA \backslash Atb$ Risoluzione del sistema lineare c contiene i coeff della retta di regressione
$retta = A * c$
$format\ short$ Torna alla notazione fissa
$[p\ retta]$ Colonna esatta e colonna approssimata (per confronto)
$plot\,(t, p, 'ko', t, retta, 'r-')$ La prima viene disegnata con pallini neri, l'altra continua rossa.
$res1 = norm\,(p - retta)$ Calcolo il residuo
Per la parabola
$A = \left[uno\ t\ t^2\right]$
Ricalcolo AtA, Atb, c
$parab = A * c$
$[p\ retta\ parab]$
$plot\,(t, p, 'ko', t, retta, 'r-', t, parab, 'b:')$
$res2 = norm\,(p - parab)$

Algoritmo 4.6 Fattorizzazione QR per la parabola

$[Q, R] = qr\,(A)$ Fattorizzazione QR di A
$Q' * Q$ Dovrebbe dare l'identità, al posto degli zeri ci sono numeri circa equivalenti alla precisione di macchina
$A - Q * R$ Idem come sopra
$R = R\,(1 : 3, :)$ Estrae le prime 3 righe
$h = Q' * p$
$h1 = h\,(1 : 3)$
$h2 = h\,(4 : 10)$
$cqr = R \backslash h1$ Coefficienti della parabola
$res = norm\,(h2)$
$[c\ cqr]$
$err = norm\,(c - cqr)$ L'errore algoritmico è sopra la precisione di macchina, sappiamo che il metodo QR è quello più preciso

Algoritmo 4.7 Polinomi di approssimazione

clear
$x = -pi : pi/5 : pi$
$y = abs\,(x)$
$plot\,(x, y, 'ko')$
$p2 = polyfit\,(x, y, 2)$ Restituisce i coefficienti del polinomio approssimante di grado 2
$xx = -pi : pi/50 : pi;$
$y2 = polyval\,(p2, xx)$; Valuta il polinomio dati i coefficienti e gli input.
$plot\,(x, y, 'ko', xx, y2, 'r-')$
$p4 = polyfit\,(x, y, 4)$
$y4 = polyval\,(p4, xx)$;
$plot\,(x, y, 'ko', xx, p4, 'r-')$
Applichiamo lo stesso procedimento con il grado 6 e 10, quest'ultimo è il polinomio interpolatore (grado = dati), è troppo oscillante
$ye = y + rand\,(size\,(y))\,/10$ Perturbazione casuale
$figure\,(2)$ Cambia la finestra in cui si disegna
$pe4 = polyfit\,(x, ye, 4)$;
$ye4 = polyval\,(pe4, xx)$;
$plot\,(x, y, 'ko', xx, y4, 'r-', xx, ye4, 'b:')$ Non si perturba di molto, col grado 10 vediamo invece che è mal condizionato.

Algoritmo 4.8 Spline

$ys = spline\,(x, y, xx)$;
$plot\,(x, y, 'ko', xx, ys, 'r-', xx, y10, 'b:')$ Evita le oscillazioni
$yes = spline\,(x, ye, xx)$;
$plot\,(x, y, 'ko', xx, ys, 'r-', xx, yes, 'b:')$ Varia di poco con i dati perturbati
$x = -pi : pi/5 : pi;$
$y = cos\,(x)$;
$figure\,(1)\,; plot\,(x, y, 'ko')$;
$c3 = cos\,(3)$
$s3 = spline\,(x, y, 3)$
Programma per migliorare l'approssimazione di $c3$ aumentando n
$for\ n = 5 : 50$
$x = -pi : pi/n : pi;$
$y = cos\,(x)$;
$s3 = spline\,(x, y, 3)$;
$err\,(n) = abs\,(c3 - s3)$;
end
$plot\,(5 : 50, err\,(5 : 50))$
$semilogy\,(5 : 50, err\,(5 : 50))$ In scala logaritmica si visualizza meglio.