Tesina Calcolo Numeri 2

UNIVERSITA' DEGLI STUDI DI CAGLIARI

Facoltà di Ingegneria Elettronica

INTERPOLAZIONE DI FUNZIONI

Studente: Michela Ragusa

Docente: Prof. Giuseppe Rodriguez

A.A. 2007/08

Indice

1 Introduzione all'interpolazione 2
1.1 Differenze divise ……..…….... 2

2 Interpolazione Polinomiale 3
2.1 Funzione interpol.m ……... …….. 4
2.2 Polinomio Interpolante di Newton .....……..... 6
2.3 Funzione intnew.m ……..... 6
2.4 Polinomio Interpolante di Lagrange…….. …… 9
2.5 Funzione intlag.m ……. 10
2.6 Polinomio Interpolante di Neville …….. 11
2.7 Funzione intnev.m …… 13

3 Errore di Interpolazione
3.1 Nodi di Chebyshev……. 16

4 Interpolazione a Tratti 19
4.1 Polinomi a Tratti…….. 19
4.2 Implementazione in Matlab……... 20
4.3 Funzioni Spline…… 22

5 Approssimazione ai Minimi Quadrati
5.1 Polinomio di Approssimazione……. 25
5.2 Implementazione in Matlab…… 26

Capitolo 1

Introduzione all'interpolazione

Se si ha a che fare con funzioni $f(x)$di forma non elementare o sconosciuta, di cui si possiede una tabulazione di un numero finito di punti, in cui si conosca il valore che la $f(x)$ assume, si può effettuare una stima del valore della funzione $f(x)$ in un punto diverso, utilizzando l'operazione di interpolazione.

Questa consiste nel sostituire la funzione originale $f(x)$con una più semplice, per esempio un polinomio, che si discosti da essa il meno possibile.

1.2 Differenze divise

Introduciamo, prima di tutto, la definizione di Differenze Finite, che servirà in

seguito.

Definizione 1.2.1:

Data $f(x)$: $I \subseteq \mathrm{R} \to \mathrm{R}$ e siano $x_0, x_1, \dots, x_{k-1} \in I$ con $x_i \neq x_j$, se $i \neq j$; la funzione

$$f[x_0, x_1, \dots, x_{k-1}, x] = \frac{f[x_0, x_1, \dots, x_{k-2}, x] - f[x_0, x_1, \dots, x_{k-1},]}{x - x_{k-1}} \qquad (1.1)$$

ove, per k = 1, $f[x_0, x] = \frac{f(x) - f(x_0)}{x - x_0}$, è definita $\forall\, x \in I,\ x \neq x_i,\ i = 0, 1, \ldots, k\text{-}1$, e si chiama **differenza divisa di ordine k.**

Se $f(x)$ è divisibile su *I,* la (1.1) è valida su tutto *I.*

Vale il seguente teorema di espansione, dimostrabile per induzione.

Teorema 1.2.1:

Sia $f(x)$: $I \subseteq \mathrm{R} \to \mathrm{R}$ e siano $x_0, x_1, \ldots, x_{k-1} \in I$, con $x_i \neq x_j$, se $i \neq j$; vale l'identità

$$f(x) = f(x_0) + (x - x_0) f[x_0, x_1] + (x - x_0)(x - x_1) f[x_0, x_1, x_2] + \cdots$$
$$+ (x - x_0)(x - x_1) \ldots (x - x_k) f[x_0, x_1, \ldots, x_k, x] \qquad (1.2)$$

Capitolo 2

Interpolazione Polinomiale

(Parabolica)

Siano dati $k+1$ punti reali $x_0, x_1, \dots, x_{k-1} \in I$, due a due distinti, in corrispondenza dei quali siano noti i $k+1$ valori reali $f(x_0), f(x_1), \dots\dots\dots, f(x_k)$. L'*interpolazione parabolica* consiste nel determinare un polinomio di grado al più k

$$P_k(x) = a_k x^k + a_{k-1}x^{k-1} + \dots\dots\dots + a_1 x + a_0 \qquad (2.1)$$

tale che $P_k(x_i) = f(x_i)$ per $i = 0,1, \dots\dots\dots, k$; (2.2)

Il polinomio $P_k(x_i)$ si chiama *polinomio di interpolazione*.

Nell'insieme dei polinomi del tipo (2.1) ne esiste uno de uno solo che verifica la (2.2). Infatti imponendo che il polinomio (2.1) verifichi le (2.2) si ottiene il sistema lineare di $k+1$ equazioni nelle $k+1$ incognite a_i, $i = 0,1, \dots\dots\dots, k$,

$$\begin{aligned} &a_0 + a_1 x_0 + \cdots + a_{k-1}x_0^{k-1} + a_k x_0^k = f(x_0) \\ &\dots\dots \quad \dots \quad \dots \quad \dots\dots \quad \dots\dots \quad \dots \\ &a_0 + a_1 x_k + \cdots + a_{k-1}x_k^{k-1} + a_k x_k^k = f(x_k) \end{aligned} \qquad (2.3)$$

Il sistema (2.3) ha la seguente matrice dei coefficienti

$$\begin{pmatrix} 1 & x_0 & \dots & \dots & \dots & x_0^{k-1} & x_0^k \\ 1 & x_1 & \dots & \dots & \dots & x_1^{k-1} & x_1^k \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 1 & x_k & \dots & \dots & \dots & x_k^{k-1} & x_k^k \end{pmatrix}$$

il cui determinante è $\prod_{0\le j<i<k}(x_i - x_j)$ e risulta diverso da zero, essendo i punti x_i due a due distinti. Tale matrice e'detta *matrice di Vandermonde* e il determinante è detto *determinante di Haar.*

Teorema 2.1 (condizione di unisolvenza):

il polinomio interpolante esiste, ed è unico, se il determinante della matrice dei coefficienti di Vandermonde è diverso da zero.

Osservazione 2.1:

il polinomio di interpolazione è di grado minore di k se, nella soluzione del sistema (2.3), risulta $a_k = 0$.

Un insieme di funzioni che verifica la condizione di unisolvenza viene detto *sistema di Chebychev.*

2.1 Funzione Interpol.m

La matrice di Vandermonde quadrata può essere costruita con la function vander(v), dove v è un vettore che contiene i nodi di griglia. I vettori x_i e y_i contengono i punti di interpolazione della funzione, mentre il vettore x i punti di valutazione:

```
function p=interpol(xi,yi,x)
% funzione che calcola il polinomio
% interpolante utilizzando la matrice di Vandermonde
```

```
% Sintassi p=interpol(xi,yi,x)
% calcolo della matrice di Vandermonde
V = vander(xi);
% calcolo dei coefficienti
c = V\yi(:);
% calcolo del polinomio di interpolazione
p = polyval(c,x);
```

Ad esempio per la funzione sen(πx) ho utilizzato 4 punti di interpolazione e 18 punti di valutazione, grafico in Figura 2.1

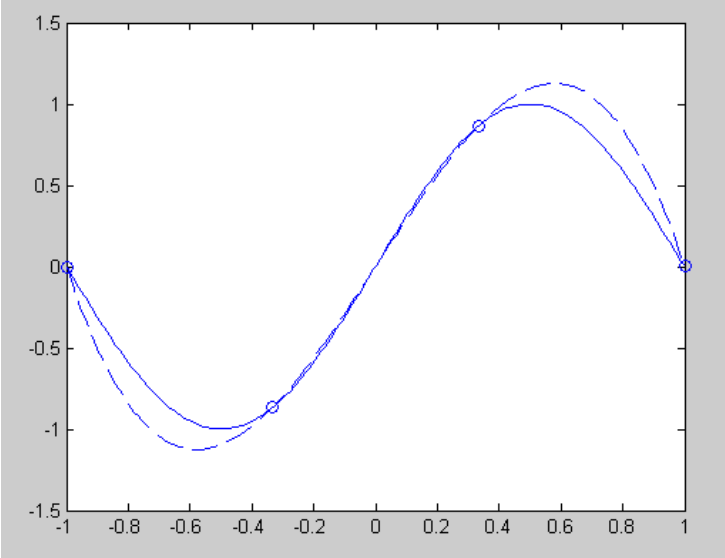

Figura 2.1: interpolazione della funzione sen(πx) con 4 nodi.

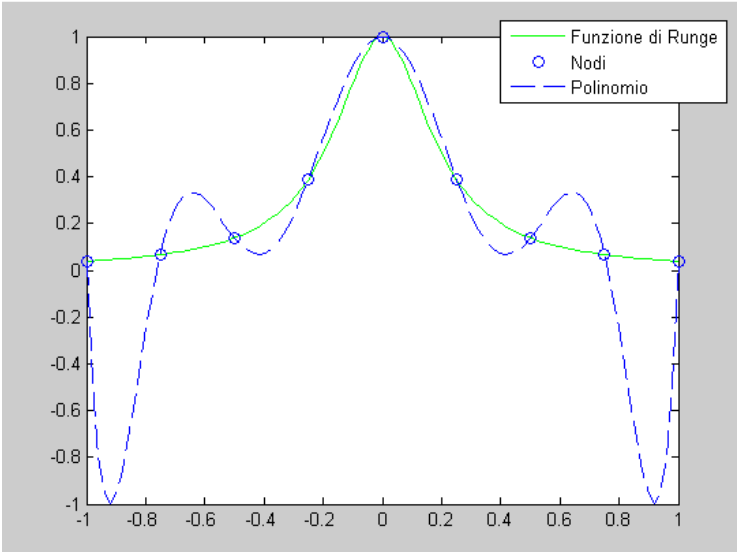

Figura 2.2: interpolazione della funzione di Runge con 9 nodi.

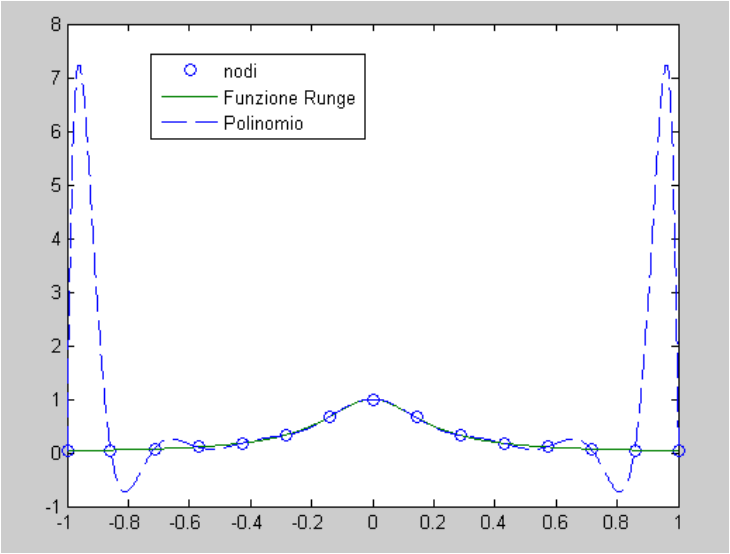

Figura 2.3: interpolazione della funzione di Runge con 15 nodi.

Come si può notare, nella funzione di Runge l'andamento del polinomio di interpolazione non segue esattamente l'andamento della funzione reale, come invece accade con la funzione sen(πx).

Il polinomio di interpolazione passa sempre per i punti di campionamento, ma ha delle oscillazioni rispetto all'andamento della funzione.

Le figure 2.2 e 2.3 mostrano come non è detto che all'aumentare dei nodi l'errore diminuisca.

Proviamo ad utilizzare altri polinomi di interpolazione.

2.2 Polinomio Interpolante di Newton

Per la costruzione di $P_k(x)$ esistono procedimenti più pratici che non la risoluzione del sistema 2.3. Si possono, ad esempio, utilizzare le differenze divise, in base al seguente teorema

Teorema 2.2 (polinomio di Newton):

il polinomio

$$P_k(x) = f(x_0) + (x - x_0)f[x_0, x_1] + (x - x_0)(x - x_1)f[x_0, x_1, x_2] + \cdots \\ + (x - x_0)(x - x_1) \dots (x - x_k)f[x_0, x_1, \dots, x_k, x] \qquad (2.4)$$

Verifica la condizione (2.2).

Il polinomio (2.4) è detto *polinomio interpolante di Newton.*

2.3 Funzione Intnew.m

La funzione intnew.m calcola il polinomio interpolante di Newton partendo dal vettore x_i con i nodi di interpolazione, y_i con il valore dei nodi e x con i punti in cui si vuole calcolare il polinomio

function y = intnew(xi,yi, x)

```
% Funzione che determina in un insieme di punti
% Sintassi y = intnew(xi,yi, x)
% y   vettore contenente i valori assunti
%  dal polinomio interpolante
xi = xi(:);
yi = yi(:);
x = x(:);
n = length(xi);
a = yi;
% calcolo delle differenze divise
for i = 2:n
  a(i:n) = (a(i:n)-a(i-1:n-1))/ (xi(i:n)-xi(1:n-i+1));
end
% calcolo del polinomio
y = a(n) * ones(size(x));
for i = n-1:-1:1
  y = y .* (x-xi(i)) + a(i);
end
```

Di seguito la Figura 2.2 e la Figura 2.3, mostrano l'interpolazione della funzione $sen(\pi x)$ con il polinomio di Newton, utilizzando prima 4 poi 8 punti di interpolazione.

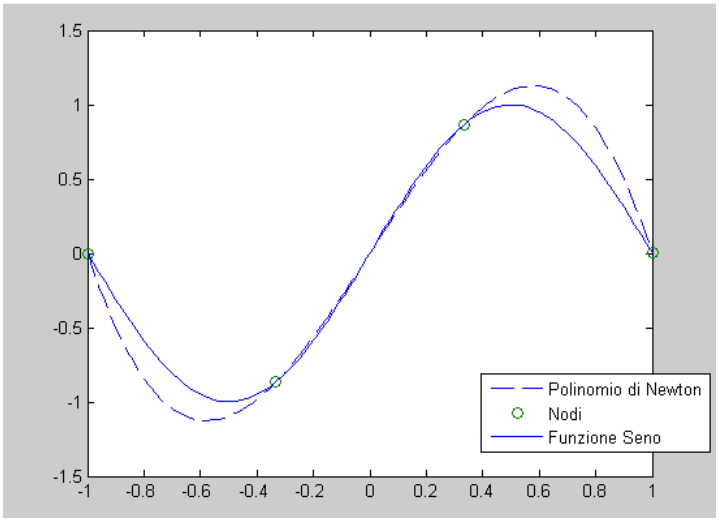

Figura 2.4: interpolazione della funzione sen(πx) con 4 nodi.

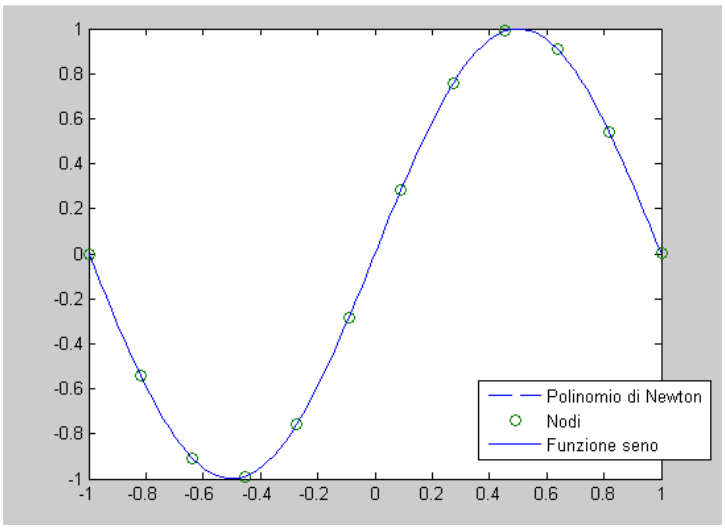

Figura 2.5: interpolazione della funzione sen(πx) con 12 nodi.

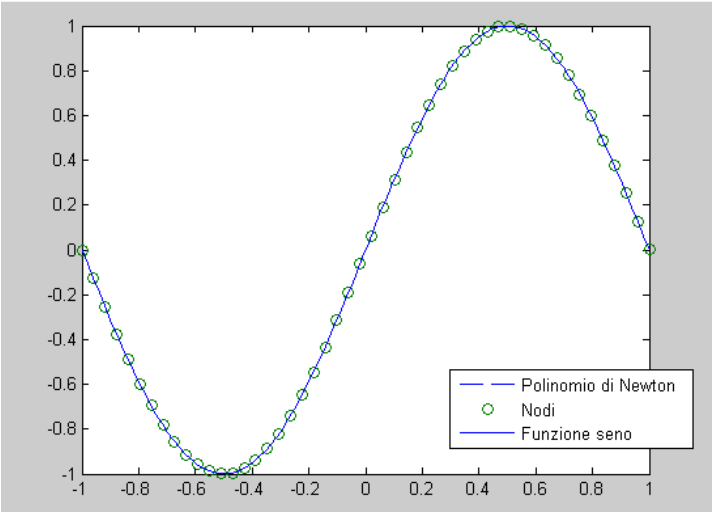

Figura 2.6: interpolazione della funzione sen(πx) con 50 nodi.

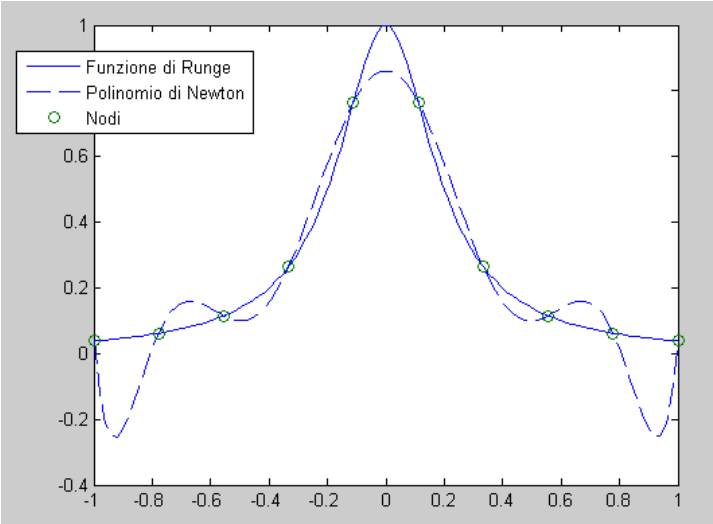

Figura 2.7: interpolazione della funzione di Runge con 10 nodi.

Per la funzione sen(πx), nelle figure 2.4 2.5 e 2.6, l'aumentare dei nodi aumenta la precisione del polinomio di interpolazione, indipendentemente dal polinomio utilizzato, ma, al crescere di nodi, figura 2.8, anche per il seno si presenta lo stesso fenomeno.

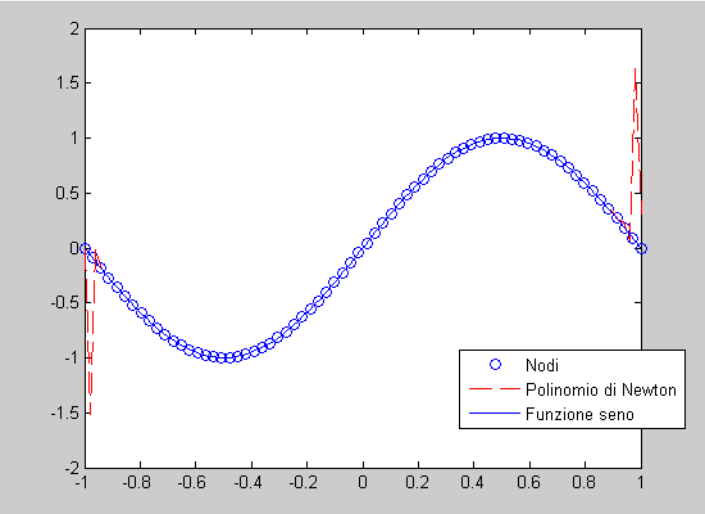

Figura 2.8: interpolazione della funzione sen(πx) con 70 nodi.

Si può anche vedere che, con un polinomio differente, non si hanno miglioramenti.

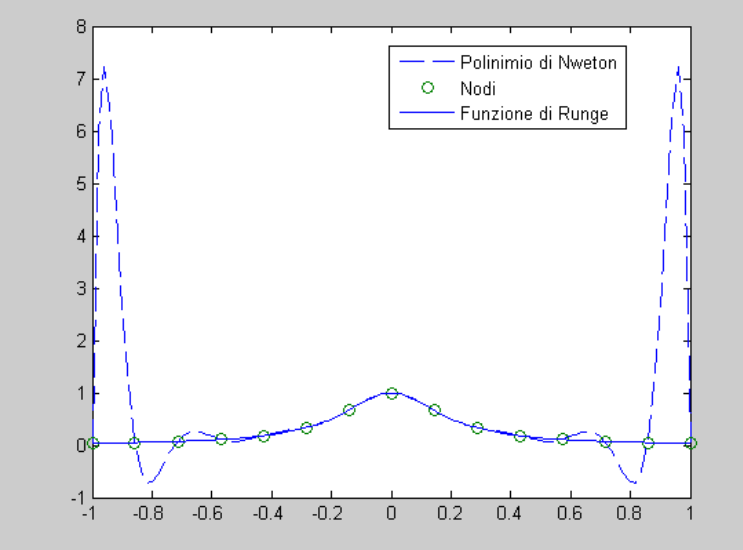

Figura 2.9: interpolazione della funzione di Runge con 15 nodi.

2.4 Polinomio Interpolante di Lagrange

Una seconda forma di polinomio di interpolazione si può ottenere per mezzo della funzione polinomiale di grado k:

$$l_r = \frac{(x - x_0) \dots (x - x_{r-1})(x - x_{r+1}) \dots (x - x_k)}{(x_r - x_0) \dots (x_r - x_{r-1})(x_r - x_{r+1}) \dots (x_r - x_k)} \tag{2.5}$$

$r = 0,1, \dots\dots, k$. I polinomi (2.5) godono delle proprietà

$$l_r(x_s) = \delta_{r,s} \quad r, s = 0,1, \dots\dots, k$$

di conseguenza il polinomio

$$L_k(x) = \sum_{r=0}^{k} l_r(x)\, f(x_r)$$

verifica la condizione (2.2). $L_k(x)$ si chiama *polinomio interpolante di Lagrange* e i polinomi (2.5) sono detti *polinomi fondamentali dell'interpolazione di Lagrange.*

2.5 Funzione Intlag.m

La funzione Intlag.m calcola il polinomio di Lagrange.

```
function p = intlag(xi,yi,x)
%  Funzione che calcola il polinomio di Lagrange
%  Sintassi p = intlag(xi,yi, x)
%  xi e yi vettori dei punti di interpolazione
%  x vettore dei punti di valutazione
n =length(xi);
m = length(x);
```

```
p=zeros(m,1);
for i=1:n
   ind = [1:i-1,i+1:n];
   den(i) = prod(xi(i)-xi(ind));
end
for k=1:m
   p(k) = 0;
   for i=1:n
    ind = [1:i-1,i+1:n];
    phi =  prod( x(k) - xi(ind) )/den(i);
    p(k) = p(k) + yi(i) * phi;
   end
end
```

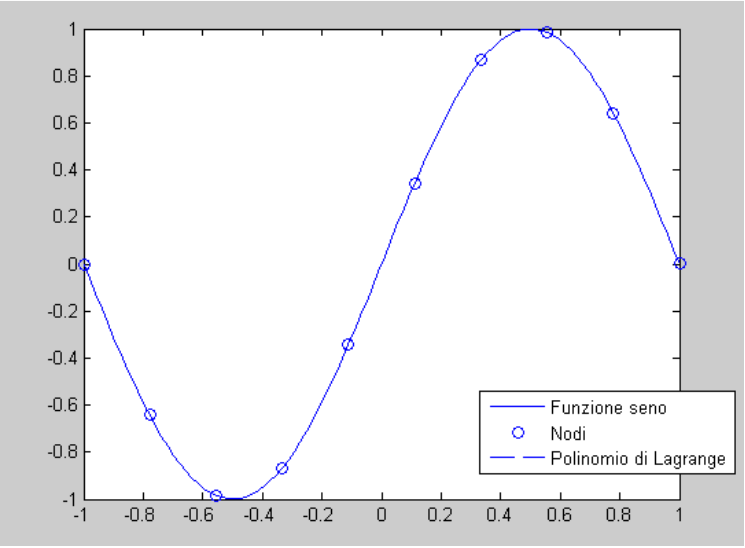

Figura 2.3: interpolazione della funzione sen(πx) con 10 nodi.

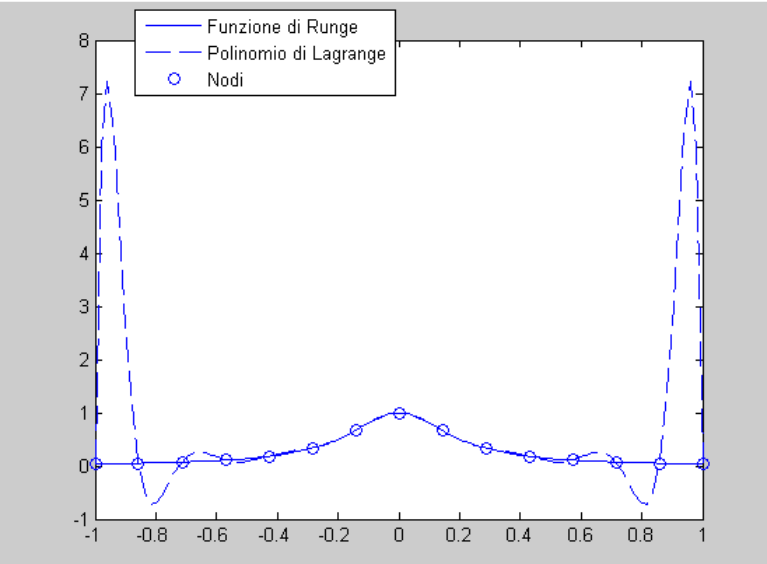

Figura 2.3: interpolazione della funzione di Runge con 15 nodi.

2.5 Formula di Neville

Spesso quello di cui si ha bisogno e' soltanto il valore del polinomio in un nuovo punto x e non tutta la definizione del polinomio stesso. In questo caso e' possibile evitare di ricalcolare tutto il polinomio utilizzando l'algoritmo di Neville.

Definiamo ora questi polinomi:

$$P_i(x) = f_i, \quad \forall\, i = 0, \ldots, n$$

dove

$$P_{i,i+1,\dots j}(x) = \frac{P_{i,i+1,\dots j-1}(x)(x - x_j) - P_{i,i+1,\dots j}(x - x_i)}{x_i - x_j} \qquad (2.6)$$

con $P_{i,i+1,\dots j}(x) \in \prod(j - i)$

Per questo polinomio vale

$$P_{i,i+1,\dots j-1}(x_k) = f_k \quad k = i, \dots, j-1$$
$$P_{i+1,\dots j}(x_k) = f_k \quad k = i+1, \dots, j$$

che implica $P_{i,i+1,\dots j}(x_k) = f_k$

Teorema 2.3 (polinomio di Neville):

Vale

$$P_{0,1,\dots n}(x) = P_n(x) \qquad (2.7)$$

Cioè, il polinomio n-esimo calcolato con il metodo di Neville è esattamente il polinomio interpolante

Sia

$$T_{i,k} = P_{k,i}(x_{i-k}, \dots, x_i)$$

dove i è il grado e k è l'ascissa. Riscrivendo la (2.6)

$$T_{ik}(x) = \frac{T_{i,k-1}(x)(x - x_{i-k}) + T_{i-1,k-1}(x_i - x)}{x_i - x_{i-k}} \qquad (2.8)$$

Riportando il risultato dell'algoritmo nella Figura (2.4), posti al primo passo $T_i = f_k$, con i quali si può trovare il polinomio $P_{0,1,\dots n}(x)$

$x_0 \quad y_0 = T_0$

$\qquad\qquad\quad T_{01}$

$x_1 \quad y_1 = T_1 \qquad\qquad T_{012}$

$\qquad\qquad\quad T_{12}$

$x_2 \quad y_2 = T_2 \qquad\qquad \ldots\ldots \qquad\qquad T_{01..n} = P_n(x)$

$. \qquad . \qquad\quad T_{23}$

$. \qquad . \qquad\qquad\quad T_{n-2,n-1,n}$

$. \qquad . \qquad T_{n-1,n}$

$x_n \quad y_n = T_n$

Figura 2.3: algoritmo di Neville

Se si deve aggiungere un'ascissa si deve solamente aggiungere una riga nella struttura triangolare della tabella della Figura (2.3).

2.6 Funzione Intnev.m

La funzione intnev.m implementa l'algoritmo di Neville

function p=intnev(xi, yi, x)

% Funzione per l'implementazione dell'algoritmo di Neville

% Sintassi p = intlag(xi,yi, x)

% xi e yi vettori dei punti di interpolazione

% x vettore dei punti di valutazione

```
 n=length(xi);
 di=linspace(xi(1),xi(n),x);

 for k=1:x
   T=yi;
  for i=1:n-1
    for j=n:-1:i+1

        % La ricorrenza dell'algoritmo

        T(j)=T(j)+((di(k)-xi(j))/(xi(j)-xi(j-i)))*(T(j)-T(j-1));

    end
       x(k)=T(n);
  end
end
```

Tutti i risultati ottenuti sin d'ora sono dati dal fatto che si utilizzano, come nodi di campionamento, dei punti equispaziati. Non si ottengono risultati migliori cambiando la distanza tra i nodi. Tali risultati rimangono invariati.

Per analizzare meglio i dati definiamo prima l'errore di interpolazione polinomiale.

Capitolo 3

Errore di Interpolazione

L'errore di interpolazione valuta puntualmente il discostamento tra la mia funzione $f(x)$ e il polinomio interpolante $P(x)$.

Esso e' definito dalla seguente formula:

$$E_n(x) = \frac{f^{(n+1)}(\xi_x)}{(n+1)!} W_n(x)$$

dove $W_n(x) = \prod_0^n (x - x_i)$ ed n indica il grado del polinomio.

Naturalmente i punti di valutazione dell'errore devono essere differenti dai nodi, perché, come si notare, $W_n(x)$ si annulla nei nodi.

Come si e' visto nei capitoli precedenti, utilizzando nodi equispaziati, il discostamento tra la $f(x)$ e il polinomio interpolante $P(x)$ diminuisce all'aumentare dei nodi sino a quando l'errore di interpolazione risulta maggiore ai bordi dell'intervallo rispetto ai valori assunti al centro. Tale comportamento, detto fenomeno di Runge, si accentua sempre di più all'aumentare del numero di nodi.

Si verifica che esistono dei punti x interni all'intervallo dei nodi, tali per cui:

$$\lim_{n\to\infty} |f(x) - \prod\nolimits_n f(x)| = 0$$

Non si ha convergenza uniforme dell'errore.

3.1 Nodi di Chebyshev

L'andamento dell'errore all'aumentare dei nodi, che si manifesta per errori equispaziati, non si riscontra invece se vengono utilizzati i nodi di Chebyshev.

Tali nodi si ottengono suddividendo uniformemente una circonferenza unitaria, quindi facendo la proiezione delle corde che uniscono i punti di suddivisione sul diametro a loro ortogonale ed infine scegliendo le intersezione tra le corde ed il diametro. I nodi non saranno più equispaziati sul diametro, pur essendo stati costruiti a partire da punti equispaziati sulla circonferenza.

Tali nodi vengono anche definiti come zeri dei polinomi di Chebyshev, polinomi che sono le componenti di una successione polinomiale che inizia con i polinomi:

$T_0(x) = 1$

$T_1(x) = x$

$T_2(x) = 2x^2 - 1$

$T_3(x) = 4x^3 - 3x$

$T_4(x) = 8x^4 - 8x^2 + 1$

…….

In formula, questi nodi sono calcolabili come:

$$x_k = cos\left(\frac{2k+1}{2n+2}\pi\right) \qquad\qquad k = 0, \dots, n$$

In Matlab tali nodi possono essere calcolati con la funzione

nodicheb.m

```
%Calcolo dei nodi di Chebyshev

function xi = nodicheb(n,a,b)

xi = linspace(a,b,n);

for i=0:n
   xc = cos((2*i+1)/(n+1)*(pi/2));
   x = (a + b)*0.5 + (a - b)*xc*0.5;
   xi(i+1)=x;
 end
```

Analizzo nuovamente l'interpolazione della funzione di Runge, utilizzando i nodi di Chebyshev, con ad esempio, il polinomio di Newton:

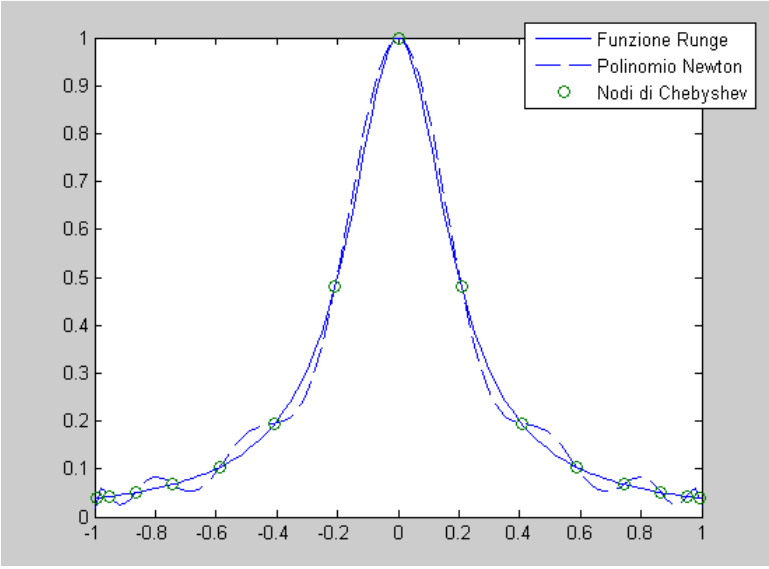

Figura 3.1: Funzione di Runge con 15 nodi di Chebyshev.

Mentre con i nodi equispaziati, già con n=15, si presentava il fenomeno di Runge, questo non accade con i nodi di Chebyshev.

Una altra analisi si può fare plottando la differenza tra la funzione $f(x)$ e il polinomio interpolante $P(x)$, dove $P(x)$ è stato calcolato prima con i nodi equispaziati poi con quelli di Chebyshev:

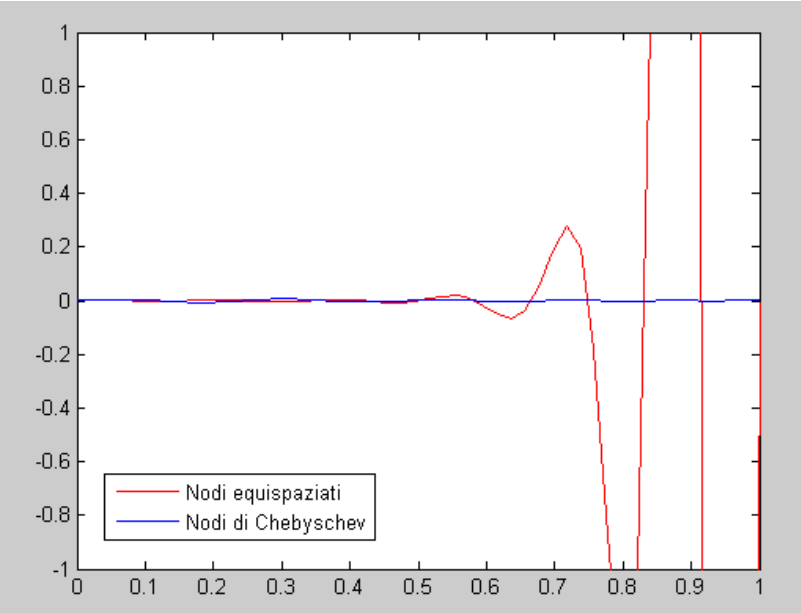

Figura 3.2: Errore di interpolazione con nodi equispaziati e di Chebyshev.

Anche da questo grafico si nota come la differenza tra la $f(x)$ ed il polinomio interpolante, al centro del grafico, sia simile sia con nodi equispaziati che con quelli di chebyshev, mentre ai bordi, tale differenza aumenta velocemente per i nodi equispaziati e risulta constante per quelli di Chebyshev.

Capitolo 4

Funzioni Polinomiale a Tratti

L'interpolazione polinomiale ha diversi vantaggi, primo tra tutti la facilità di calcolo. Tuttavia presenta alcuni difetti, il principale dei quali è rappresentato dal fatto che all'aumentare del grado n del polinomio interpolante si presentano solitamente delle forti oscillazioni. Questo rende alcune volte l'interpolazione polinomiale inacettabile.

Per ovviare al problema accennato si puo' pensare di interpolare la funzione $f(x)$ utilizzando dei polinomi di grado basso, pertanto poco oscillanti, ma solo in opportuni sottointervalli, cioè utilizzando le funzioni cosiddette polinomiali a tratti, tra le quali le piu' comunemente utilizzate sono le *funzioni spline*.

4.1 Polinomi a Tratti

Per interpolazione polinomiale a tratti si intende l'interpolazione di un set di dati con più polinomi, ciascuno dei quali definito in un sottoinsieme dell'intervallo dato.

Sia $[a, b]$ un intervallo limitato e chiuso e sia Δ la partizione dell'intervallo $[a, b]$ data da:

$$\Delta = \{x_i\}_{i=0,\dots,m}$$

su un insieme di punti, detti nodi, tali che:

$$a = x_0 < x_1 < \cdots < x_m = b$$

La partizione di $[a, b]$ indotta dall'insieme Δ risulta:

$I_0 = [x_0, x_1]$

$I_i = [x_i, x_{i+1}]$

$I_m = [x_{m-1}, x_m]$

Definiamo quindi lo spazio del polinomi a tratti come:

$$T\ P_n = \{\varphi | \exists p_0, p_1, \dots, p_n; tale\ che\ \varphi(x) = p_i(x), \forall x\ \in I_i, i = 0,1, \dots, m\}$$

4.2 Implementazione in Matlab

La routine di interpolazione polinomiale a tratti in matlab si chiama *interp1.m*. La sintassi è la seguente:

f = interp1(xi,yi,x,TIPO)

dove con TIPO si intende il tipo di interpolazione richiesta.

- *'nearest'* : interpolazione con polinomio costante a tratti
- *'linear'* : (default) polinomio lineare a tratti
- *'cubic'* : polinomio cubico a tratti (con derivate continue)
- *'spline'* : interpolazione con spline cubica

Ad esempio, interpoliamo la funzione $f(x) = e^x * \cos(4x)$ nell'intervallo [0,3].

Nel primo grafico considelo l'interpolazione costante a tratti con 10 nodi di interpolazione:

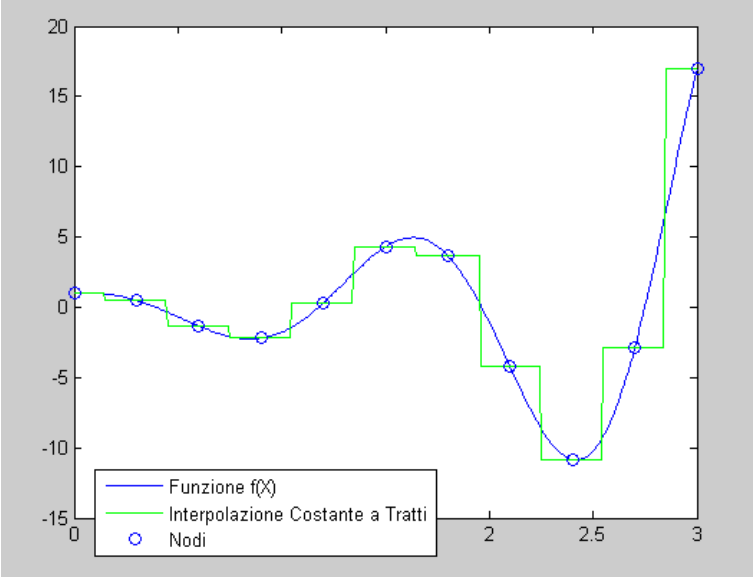

Figura 4.1: interpolazione costate a tratti di $f(x) = e^x * \cos(4x)$.

Nel secondo grafico considero invece l'interpolazione lineare a tratti, sempre con 10 nodi di interpolazione:

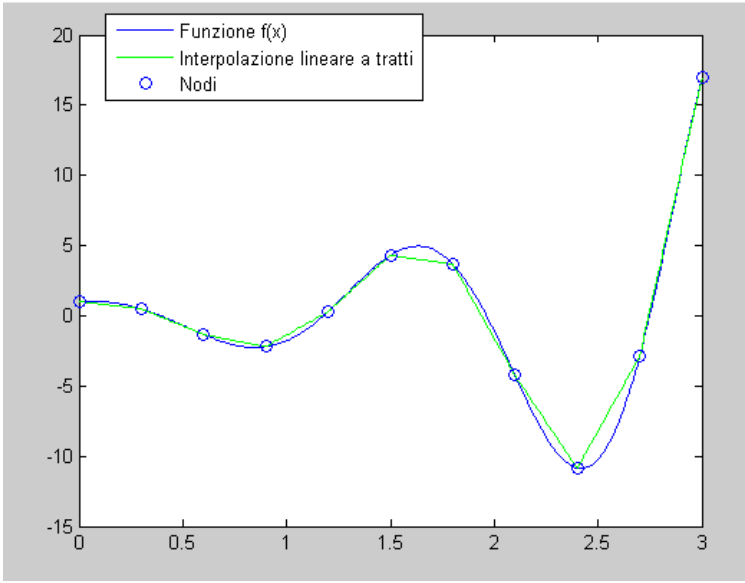

Figura 4.2: interpolazione lineare a tratti di $f(x) = e^x * \cos(4x)$.

Si può anche effettuare tale interpolazione per la funzione di Runge:

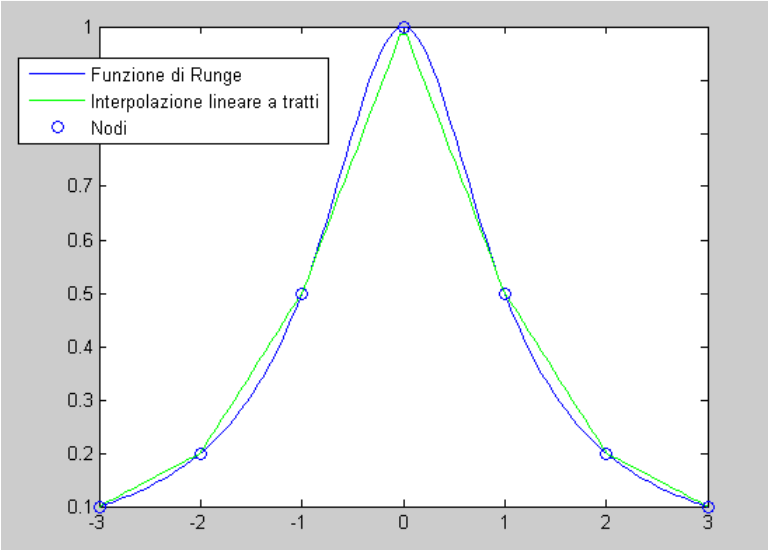

Figura 4.3: interpolazione lineare a tratti per la funzione di Runge con 7 nodi.

Con l'interpolazione a tratti si guadagna in flessibilità ma si perde la regolarità della funzione, soprattutto nell'interpolazione continua a tratti, dove si creano dei gradini tra un tratto e l'altro. I polinomi derivanti dall'interpolazione polinomiale a tratti non hanno più la derivata prima continua, mentre alcune volte ciò è richiesto.

4.3 Funzioni Spline

Il problema legato ala derivata non continua viene risolto tramite le *funzioni spline*. Per esse infatti si considera la seguente definizione:

Definizione 4.3.1: dicesi funzione spline di grado *m*, relativo ai punti $x_0, x_1, \dots, x_k$ una funzione $S_m(x)$: $[x_0, x_k] \to R$ tale che:

1. $S_m(x)$ è un polinomio di grado non superiore ad *m* in ogni intervallo $[x_{i-1}, x_i]$ con *i=1,2,…,k;*
2. $S_m(x_i) = y_i$ con *i=0,1,…,k;*

3. $S_m(x) \in C^{m-1}([x_o, x_k])$.

Consideriamo il caso di *m*=3, che da luogo alle cosidette *spline cubiche*. Tale funzione è composta da *k* polinomi $p_i(x), i = 1, \dots, k$, di grado al più 3.

Ciascun polinomio $p_i(x): [x_{i-1}, x_i] \to R$ è definito da quattro coefficienti. $S_3(x)$ risulterà quindi definita dai *4k* coefficienti dei polinomi che la compongono. Imponendo che sia verificata la proprietà 2 e 3, si ottengono le *4k-2* condizioni

$p_i(x_{i-1}) = y_{i-1}$ con *i=1,2,…,k*

$p_i(x_i) = y_i$ con *i=1,2,…,k*

$p_i'(x_i) = p_{i+1}'(x_i)$ con *i=1,2,…,k-1*

$p_i''(x_i) = p_{i+1}''(x_i)$ con *i=1,2,…,k-1*

Le due ulteriori condizioni si scelgono fra le seguenti:

$p_1''(x_0) = p_k''(x_k) = 0$ *Spline Naturale*

$p_1'(x_0) = p_k'(x_k), p_1''(x_0) = p_k''(x_k)$ *Spline Periodica*

$p_1'(x_0) = y_0', p_k'(x_k) = y_k'$ *Spline Vincolata*

Prendendo nuovamente in considerazione la funzione $f(x) = e^x * \cos(4x)$, l'interpolazione con la funzione spline risulta dalla Figura 4.4:

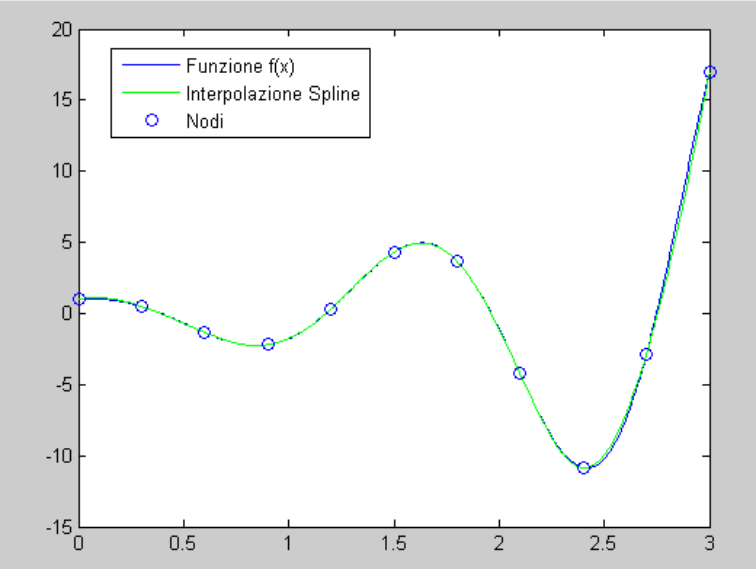

Figura 4.4: Interpolazione della $f(x) = e^x * \cos(4x)$ con funzione spline e 10 nodi.

Per verificare che con la funzione spline non si presentasse in fenomeno di Runge, con considerato l'interpolazione con la funzione di Runge utilizzando 25 nodi, ricordando che già con 15 nodi tale fenomeno era presente con una interpolazione polinomiale. In Figura 4.5 è mostrato il risultato:

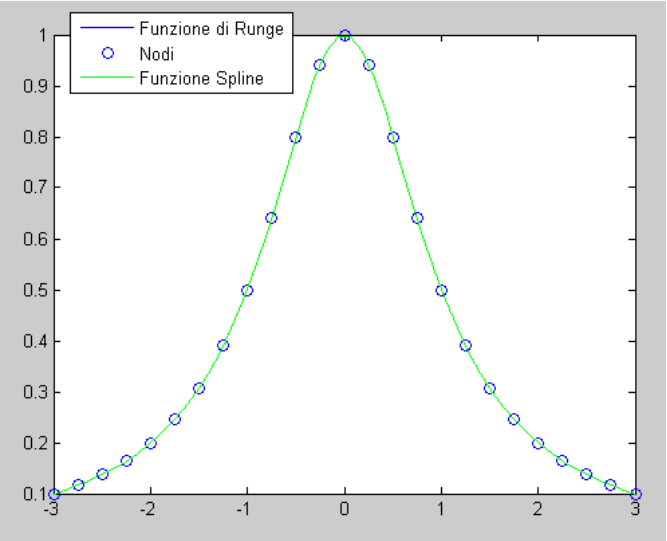

Figura 4.5: Funzione di Runge interpolata con Funzione Spline con 25 nodi.

Capitolo 5

Approssimazione ai Minimi Quadrati

Finora si è sempre considerato il problema di costruire un polinomio che passa per punti fissati. Se i valori assegnati sono affetti da errori, come in genere capita se tali valori sono ottenuti da osservazioni sperimentali, allora non ha più molto senso "costringere" il polinomio ad assumere tali valori.

In questo caso è più significativo richiedere che $p(x)$ sia vicino ai valori f_i, senza che $p(x_i) = f_i$ nei punti x_i.

5.3 Polinomio di Approssimazione

Sia $\varphi_0(x), \varphi_1(x), \dots, \varphi_n(x)$ una base per $P_n(x)$ e siano $(x_i, f_i), i = 0, \dots, m+1 > n$ coppie dei punti assegnati. Si cerca il polinomio $p(x)$ tale che la quantità

$$F = \sum_{i=0}^{m} (p(x_i) - f_i)^2$$

risulti minima. Il polinomio $p(x)$ così ottenuto approssima i dati nel senso dei minimi quadrati.

Per valutare qualche esempio si può utilizzare la funzione di Matlab polyfit.m, che calcola i coefficienti del polinomio $P_n(x)$ di grado n che meglio approssima (in norma 2 discreta, da cui il nome di minimi quadrati discreti) la funzione $f(x)$, avente nel vettore di nodi X i valori Y.

5.4 Implementazione in Matlab

Per prima cosa si deve simulare una situazione di errore nella funzione, sovrapponendo una perturbazione causale alla funzione originale.

Prendiamo ad esempio la funzione:

*f=inline('x.^3 -3.*x +2');*

la funzione affetta da errore sarà:

*fx=f(x)+0.2*rand(1,100);*

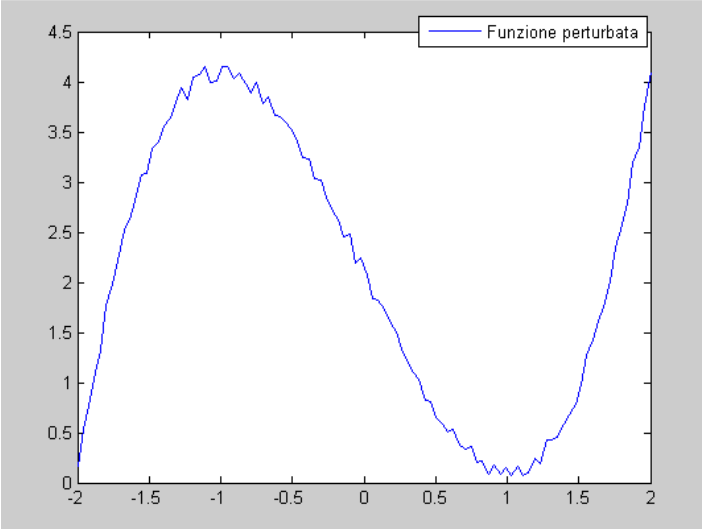

Figura 5.1: funzione affetta da errore

Calcolo quindi i coefficienti del polinomio interpolante, ad esempio di grado 3:

a = polyfit(x,fx,3);

a questo punto costruisco il polinomio con la funzione di Matlab polyval.m:

p = (a,x);

Plottando il risultato ottengo la Figura 5.2, con 100 nodi di valutazione e il polinomio di grado 3

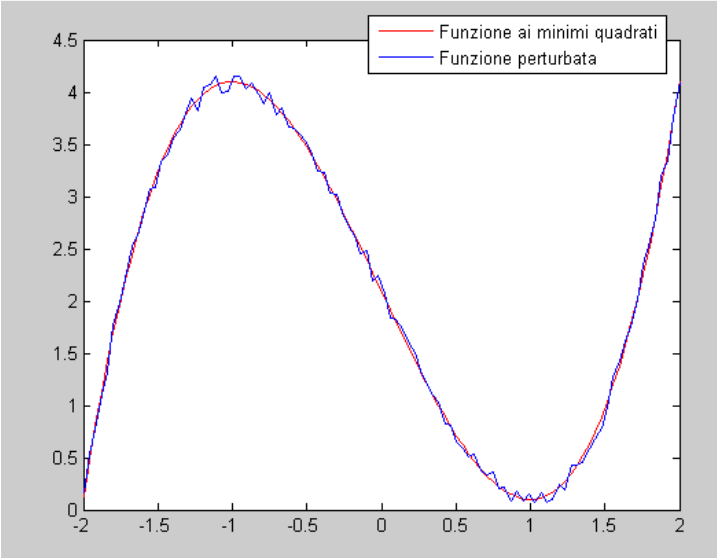

Figura 5.2: Interpolazione con polinomio di grado 3

Si può per assurdo notare che alcune funzioni, se presente una perturbazione, hanno un polinomio di interpolazione del senso dei minimi quadrati, migliore che se interpolate esattamente.

Ad esempio la funzione *f(x)=abs(x-1)* la considero prima interpolata esattamente con un polinomio di grado 10. Le oscillazioni presenti alla fine del grafico di Figura 5.3 sono dovute alla funzione stessa.

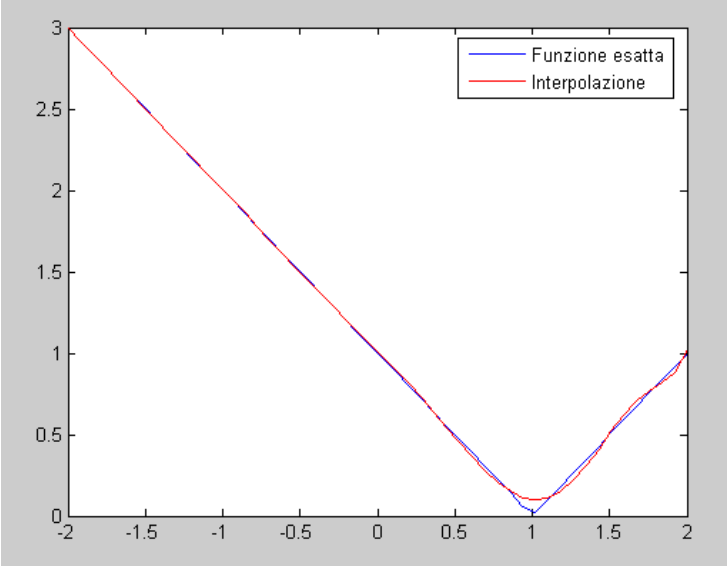

Figura 5.3: Interpolazione partendo da dati esatti

Mentre, in Figura 4.4, si nota come la stessa funzione perturbata, ha un polinomio nel senso dei minimi quadrati, più regolare:

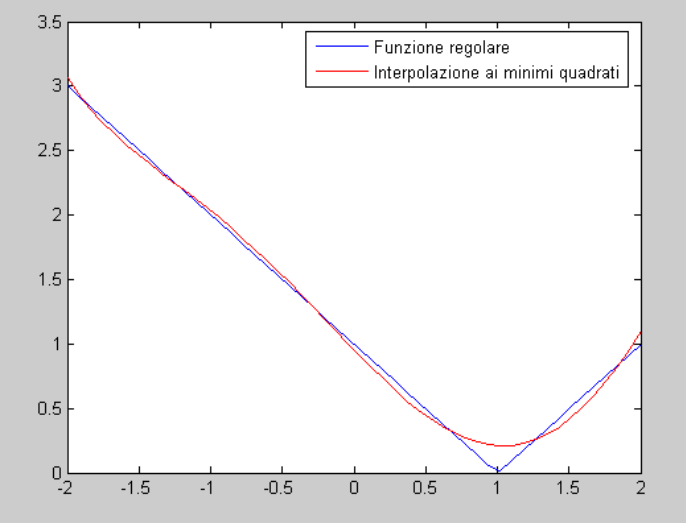

Figura 5.4: Interpolazione partendo da dati perturbati. Si è utilizzato un polinomio di grado 5.