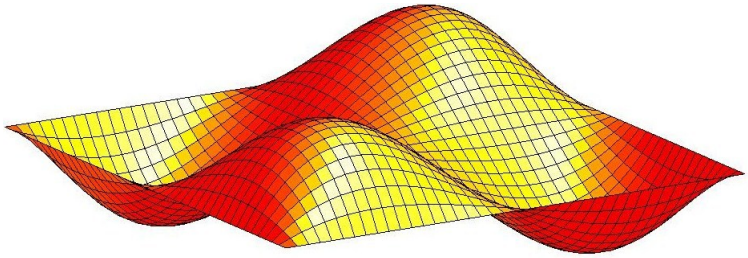
A.A. 2010/2011

TESINA CALCOLO NUMERICO 2

Analisi dell'errore dei Polinomi interpolanti e applicazioni in 3D

Atzei Alessio

Littarru Gaia



Indice

Capitolo 1

Introduzione

Si presenta frequentemente la necessità di dover approssimare una funzione $f(x)$con un'altra funzione $\Phi(x)$. Per esempio, supponiamo di avere una legge fisica che dipende da un parametro x. Siamo interessati a conoscere l'andamento della legge fisica dopo aver misurato i valori in laboratorio. Risulta utile trovare un'altra funzione passante per quei valori (x_i, y_i) i=0,...,n tali che si abbia $y_i = f(x_i)$ che approssimi la funzione di partenza: si tratta di un problema di interpolazione. Accade invece, che si conosca bene la funzione $f(x)$ ma si vuole comunque approssimarla poichè si vuole risolvere un problema di difficile risoluzione come per esempio il calcolo di un integrale o la risoluzione di un'equazione differenziale. Si vuole pertanto sostituire la funzione con una sua approssimazione che agevoli la risoluzione del problema. Un esempio è il calcolo di un integrale nel quale non si riesce a trovare la primitiva della funzione:

$$\int_a^b f(x)dx = F(b) - F(a) \approx \int_a^b \varphi(x)dx$$

Capitolo 2

Approssimazione di funzioni

2.1 Interpolazione

Consideriamo le coppie di numeri sul piano reale

$$(x_i, y_i)\, i = 0, 1..., n \tag{2.1}$$

ossia si conosce una successione di $n+1$ ascisse e $n+1$ ordinate

x	$x_0, x_1, ..., x_n$
y	$y_0, y_1, ..., y_n$

Tabella 2.1: Coppie valori

Si cerca una funzione $\Phi = \Phi(x)$ dipendente da $x_o, x_1, ..., x_n$tale che

$$\Phi(x_i) = y_i \quad \text{per cui i=0,1,...,n} \tag{2.2}$$

essendo y_i dei valori assegnati. La funzione Φ interpola i valori $\{y_i\}$ nei nodi $\{x_i\}$.

A seconda del tipo di funzione interpolante si parla di *approssimazione polinomiale* (Φ è un polinomio algebrico), *approssimazione trigonometrica*(Φ è un polinomio trigonometrico), *approssimazione composita* o *mediantesplines* (in questo caso Φ è localmente un polinomio).

Si consideri il caso di dati provenienti dal campionamento di una funzione incognita $f(x)$. Si approssima $f(x)$ tramite la combinazione lineare di $n+1$ funzioni $\varphi_j(x)$, $j = 0, ..., n$

$$\Phi(x) = \sum_{j=0}^{n} a_j \varphi_j(x) \tag{2.3}$$

Si devono trovare i coefficienti a_j imponendo le condizioni di interpolazione (2.2) ottendo così il sistema di equazioni lineari

$$\sum_{j=0}^{n} \varphi_j(x_i) \cdot a_j = y_i,\ i = 0, ..., n,$$

che in termini matriciali può essere visto così $\phi\mathbf{a} = \mathbf{y}$ dove

$$\phi = \begin{bmatrix} \varphi_0(x_0) & \varphi_1(x_0) & ... & \varphi_n(x_0) \\ \varphi_0(x_1) & \varphi_1(x_1) & ... & \varphi_n(x_0) \\ . & . & & . \\ . & . & & . \\ . & . & & . \\ \varphi_0(x_n) & \varphi_1(x_n) & ... & \varphi_n(x_0) \end{bmatrix}$$

$$\mathbf{a} = (a_0, a_1, ..., a_n)^T$$

$$\mathbf{y} = (y_0, y_1, ...y_n)^T.$$

La funzione che interpola i dati (2.1) esiste ed è unica se e solo $det(\phi) \neq 0$ chiamato **determinante di Haar**. L'insieme di funzioni che verifica tale condizione è detto **sistema di Chebychev**.

2.2 Interpolazione polinomiale

I polinomi sono utilizzati frequentemente per l'approssimazione di funzioni; a tal scopo si usa il polinomio di Taylor, ossia una serie di Taylor troncata, che ben approssima la funzione in prossimità di un valore iniziale x_0. Però il **Teorema di Weierstraiss** permette una migliore capacità di approssimazione dei polinomi ed afferma:
Sia $f \epsilon\ C[a, b]$. Per ogni $\epsilon > 0$ esiste un intero n ed un polinomio P_n tale che

$$\|f - P_n\|_\infty < \epsilon.$$

Considerando che la norma infinito di una funzione è definita come

$$\|f\|_\infty = \max_{x\epsilon[a,b]} |f(x)|\,.$$

Si ottiene la seguente espressione

$$\|f - P_n\|_\infty = \max_{x\epsilon[a,b]} |f(x) - p_n(x)| < \epsilon.$$

Il polinomio oscilla intorno alla funzione f; ϵ è la massima distanza tra la funzione e il polinomio approssimante.

Si può dimostrare che tale polinomio di grado n che meglio approssima la funzione f rispetto alla norma-∞ la interpola inoltre su $n+1$ punti e che il posizionamento di tali punti è difficile da determinare.

Il problema dell'interpolazione polinomiale è così formulato: siano dati i punti (x_i, y_i) $i = 0, ..., n$, si determina

$$p_n(x_i) = y_i \qquad \text{i=0,...,n.}$$

Teorema. Il polinomio interpolante esiste ed è unico se e solo se $x_i \neq x_j$ per $i \neq j$. Tale teorema nasce come conseguenza della condizione $det(\phi) \neq 0$ che determina l'unicità dei polinomi interpolanti.

Senza dare dimostrazione del teorema si analizza al paragrafo 2.2.1 che tratta il problema dell'interpolazione usando la base canonica.

2.2.1 Polinomio base canonica

Data la base canonica $\{1, x, x^2, ..., x^n\}$ si esprime il polinomio interpolante come

$$p_n(x) = \sum_{j=0}^{n} a_j x^j. \tag{2.4}$$

Si impongono le condizioni di interpolazione $p_n(x_i) = y_i$ ottenendo il sistema lineare $X\mathbf{a} = \mathbf{y}$ dato da

$$\sum_{j=0}^{n} a_j x_i^j = y_i, \quad i = 0, ..., n \tag{2.5}$$

con matrice dei coefficienti

$$X = \begin{bmatrix} 1 & x_0 & x_0^2 & ... & x_0^n \\ 1 & x_1 & x_1^2 & ... & x_1^n \\ . & . & . & & . \\ . & . & . & & . \\ . & . & . & & . \\ 1 & x_n & x_n^2 & ... & x_n^n \end{bmatrix}. \tag{2.6}$$

Tale matrice è detta di **Vandermonde** e si può dimostrare che il suo determinate ha la seguente forma:

$$det(X) = \prod_{i,j=0, i>j}^{n} (x_i - x_j).$$

É immediato osservare che la tesi è verificata e che per $x_i \neq x_j$ con $i \neq j$ il $det(X) \neq 0$ e il polinomio interpolante esiste ed è unico.

Questo approccio presenta diversi svantaggi:

1. la matrice X risulta malcondizionata, $k(X) >> 1$, se alcuni dei nodi x_i sono molto vicini;
2. costo computazionale elevato $O(n^3)$;
3. la rappresentazione in forma canonica è molto instabile in quanto piccoli errori sui coefficienti a_j implicano grandi variazioni sui valori di $p_n(x)$.

2.2.2 Polinomio interpolante di Lagrange

Il polinomio interpolante è rappresentato utilizzando una base diversa da quella canonica. La rappresentazione canonica (2.4) è la più semplice da utilizzare ma da un lato richiede un alto costo computazionale per determinare i coefficienti. Siano dati

$$(x_i, y_i), \quad i = 0, ..., n,$$

i **polinomi caratteristici di Lagrange** $\{L_j(x)\}_{j=0}^n$ espressi come

$$L_j(x) = \prod_{k=0, k\neq j}^{n} \frac{x - x_k}{x_j - x_0} = \frac{(x - x_0)...(x - x_{j-1})(x - x_{j+1})...(x - x_n)}{(x_j - x_0)...(x_j - x_{j-1})(x_j - x_{j+1})...(x_j - x_n)}. \tag{2.7}$$

$L_j(x)$ sono tutti polinomi di grado n, invece nella base canonica i polinomi sono di grado crescente. Il polinomio è ben definito in quanto il denominatore non si annulla poiché tutti i nodi sono distinti infatti deve verificare la seguente proprietà di interpolazione

$$L_j(x_i) = \delta_{ij} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}.$$

Significa che $\Phi_{ij} = \varphi_j(x_i) = \delta_{ij}$ ossia la matrice $\Phi_{ij} = I$. Di conseguenza i polinomi di Lagrange costituiscono un sistema di Chebychev.

É possibile osservare la proprietà appena enunciata dalla figura 2.1 la quale riporta il grafico delle funzioni $L_j(x)$ corrispondenti alle ascisse di interpolazione $\{1, 2, 3, 4, 5\}$ $(n = 4)$. Tali ascisse sono state ottenute campionando la funzione $\sin(2\pi x)$ nei punti $\{x_1, x_2, x_3, x_4, x_5\}$.

Il **polinomio interpolante di Lagrange** assume la forma

$$p_n(x) = \sum_{j=0}^{n} y_j L_j(x). \tag{2.8}$$

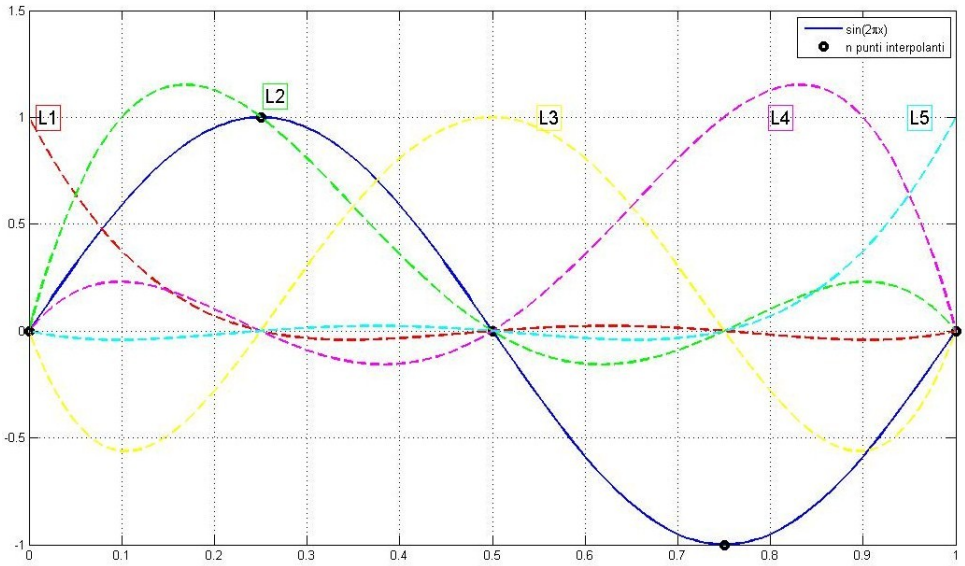


Figura 2.1: Polinomi caratteristici di Lagrange

Interpola i dati assegnati poiché

$$p_n(x_i) = \sum_{j=0}^{n} y_j \delta_{ij} = y_i, \ i = 0, ...n.$$

La valutazione in un punto del polinomio di Lagrange richiede una complessità computazionale pari a $O(n^2)$ mentre la base canonica $O(n^3)$ per il calcolo dei coefficienti e $O(n)$ per ogni valutazione. La base di Lagrange è meno instabile di quella canonica però se i nodi si avvicinano $x_i \approx x_j$ tende a destabilizzarsi.

Si vuole ora effettuare una sperimentazione numerica sull'interpolazione utilizzando il polinomio di Lagrange. A tal fine è stata creata una funzione in Matlab che costruisce il polinomio interpolante di Lagrange secondo la definizione 2.8. Tale funzione crea inoltre una matrice che memorizza in ogni colonna i punti necessari per la costruzione dei polinomi caratteristici. L'applicazione del listato con i parametri indicati più avanti ha prodotto la figura 2.2

Listing 2.1: Funzione new_lagint. Polinomio interpolante di Lagrange

```
%FUNZIONE PER LA COSTRUZIONE DEL POLINOMIO INTERPOLANTE DI LAGRANGE
    VALUTATO IN m PUNTI

function [p,L]=new_lagint(x,t,f)
%La funzione fornisce la valutazione del polinomio interpolante
%ed una matrice che contiene gli n polinomi caratteristici
```

```
m = length(x); %numero dei punti scelti per valutare la funzione
n= length (t); %numero dei punti di interpolazione
L= zeros(m,n); % inizializzazione della matrice mxn
for j=1:m
    p(j)=0;
    for k=1:n
        k_lr=[1:k-1 , k+1:n];%ulteriore indice per evitare divisioni
            per 0
        lr=prod(x(j)-t(k_lr))/prod(t(k)-t(k_lr));%polinomio di
          lagrange calcolato in un punto O(n^2)
        L(j,k)=lr;
        p(j)=p(j)+f(k)*lr; %combinazione lineare delle funzioni lr
    end
end
```

L'algoritmo appena descritto nel listato è stato utilizzato impostando i seguenti parametri:

- $m = 201$;
- $n = 4$;
- *i nodi sono stati presi equispaziati*
- *listato di riferimento* : *Programma* 01.

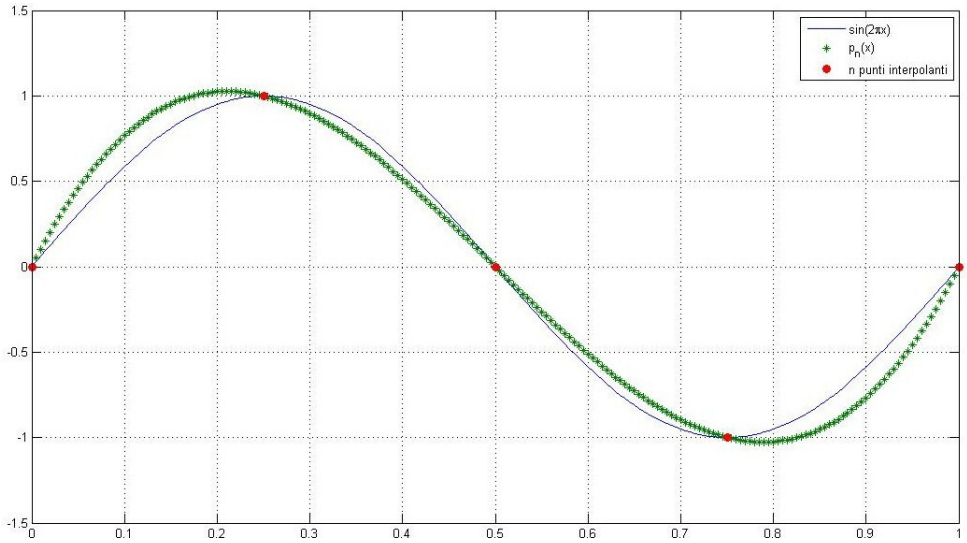


Figura 2.2: Polinomio interpolante di grado $n = 4$

2.2.3 Polinomio interpolante di Newton

Il polinomio $p_n(x)$ che interpola tutti i punti (x_i, y_i) può essere scritto nella seguente forma

$$p_n(x) = p_{n-1}(x) + g_n(x) \tag{2.9}$$

dove $g_n(x)$ è un polinomio di grado n, mentre $p_{n-1}(x_i) = y_i$. Poichè si vuole che $g_n(x)$ interpoli gli stessi punti interpolati da p_{n-1} allora si deve verificare la condizione

$$g_n(x_i) = 0, \quad i = 0, ..., n-1.$$

Pertanto

$$g_n(x) = a_n \cdot \omega_{n-1}(x) = a_n \prod_{i=0}^{n-1}(x - x_i).$$

Dove a_n è il coefficiente del termine di grado massimo del polinimio g_n, quindi di p_n. Si determina a_n imponendo che p_n interpoli anche l'ultimo punto (x_n, y_n) ottenendo così

$$a_n = \frac{y_n - p_{n-1}(x_n)}{\omega_{n-1}(x_n)}.$$

Risulta più utile rappresentare il coefficiente di grado massimo di un polinomio che interpola una funzione $f(x)$ nei punti di ascisse $x_0, ..., x_n$ in questo modo

$$a_n = f[x_0, ..., x_n],$$

simbolo che indica le **differenze divise** di ordine $n + 1$.
Si utilizza la formula ricorsiva di Neville ottenendo la seguente relazione

$$f[x_0, ..., x_n] = \frac{f[x_1, ..., x_n] - f[x_0, ..., x_{n-1}]}{x_n - x_0}, \tag{2.10}$$

tramite la quale si riesce a calcolare qualsiasi differenza divisa

$$f[x_i] = y_i, \quad i = 0, ..., n.$$

Se si deve interpolare in un solo punto allora il polinomio interpolante è una costante, se si vuole interpolare in due punti si ha

$$f[x_i, x_{i+1}] = \frac{f[x_{i+1}] - f[x_i]}{x_{i+1} - x_i}.$$

Si vuole trovare la relazione per interpolare in n punti e tal proposito si sfrutta lo **schema delle differenze divise** indicato sotto.

$$
\begin{array}{llllllll}
x_0 & y_0 = f[x_0] & & & & & & \\
 & & f[x_0,x_1] & & & & & \\
x_1 & y_1 = f[x_1] & & f[x_0,x_1,x_2] & & & & \\
 & & f[x_1,x_2] & & \ddots & & & \\
x_2 & y_2 = f[x_2] & & f[x_1,x_2,x_3] & & \ddots & & \\
 & & f[x_2,x_3] & & & & f[x_0,...,x_n] & \\
\vdots & \vdots & & \vdots & & \iddots & & \\
 & & \vdots & & \iddots & & & \\
\vdots & \vdots & & f[x_{n-2},x_{n-1},x_n] & & & & \\
 & & f[x_{n-1},x_n] & & & & & \\
x_n & y_n = f[x_n] & & & & & &
\end{array}
$$

Si tiene in memoria la tabella e supponendo in seguito di voler inserire un nuovo punto di interpolazione (x_{n+1}, y_{n+1}) è sufficiente inserire una nuova riga risparmiando così in calcoli aggiuntivi. Tramite le differenze divise si può riscrivere il polinomio

$$
\begin{aligned}
p_n(x) &= p_{n-1}(x) + f[x_0,...,x_n]\omega_{n-1}(x) \\
&= p_{n-2}(x) + f[x_0,...,x_{n-1}]\omega_{n-2}(x) + f[x_0,...,x_n]\omega_{n-1}(x) \\
&= \dots \\
&= f[x_0] + f[x_0,x_1]\omega_0(x) + \cdots + f[x_0,...,x_n]\omega_{n-1}(x),
\end{aligned}
$$

trovando la **forma di Newton** del polinomio interpolante

$$
\begin{aligned}
p_n(x) &= f[x_0] + f[x_0,x_1](x-x_0) + f[x_0,x_1,x_2](x-x_0)(x-x_1) + \\
&+ f[x_0,...,x_{n-1}](x-x_0)\cdots(x-x_{n-2}) \\
&+ f[x_0,...,x_n](x-x_0)\cdots(x-x_{n-1}).
\end{aligned} \tag{2.11}
$$

Dove è stata utilizzata la base di potenze traslate

$$\omega_j(x) = (x-x_0)(x-x_1)\cdots(x-x_j),$$

che risulta simile alla base canonica. I coefficienti hanno grado crescente come nella base canonica ma sono più semplici da calcolare.

Si illustra la funzione in Matlab che costruisce il polinomio interpolante di Newton secondo la definizione 2.11. Tale funzione crea inoltre una matrice triangolare inferiore che memorizza le differenze divise risultanti dall'espressione 2.10.

Listing 2.2: Funzione new_newton. Polinomio interpolante di Newton

```
%COSTRUZIONE DEL POLINOMIO DI NEWTON VALUTATO IN m PUNTI

function [p,N]=new_newton(x,y,t)
%output: p: polinomio newton valutato in m punti
%        N:matrice con calcolo delle differenze finite
m = length(x); %numero dei punti scelti per valutare la funzione
n= length (t); %numero dei punti di interpolazione
N=zeros(n,n);  %inizializzo la matrice quadrata
N(:,1)=y;      %la prima colonna sono le ordinate di interpolazione
for j=2:n
    for i=2:n

        if(j>i) N(i,j)=0;
            else
             N(i,j)= (N(i,j-1)-N(i-1,j-1))/(t(i)-t(i-j+1)) ;
        end
    end
end
p=zeros(m,1); %polinomio interpolante newton
pr=zeros(n,1);%coefficienti della combinazione lineare
pr(1)=1;
for l=1:m
    for k_r=2:n
    pr(k_r)= pr(k_r-1)*(x(l)-t(k_r-1)); %costruzione base potenze
      traslate
    end
    for k=1:n
    p(l)=p(l)+(N(k,k)*pr(k)); %valutazione polinomio Newton in un
      punto
    end
end
```

2.2.4 Errore di interpolazione

Con l'utilizzo di un polinomio interpolante si compie un errore e risulta utile calcolarlo. Se si usano dei dati sperimentali (x_i,x_j) $i = 0, ..., n$ si può ipotizzare che esiste una funzione (o una legge fisica) che li abbia prodotti pertanto $y_i = f(x_i)$. Se calcolo il polinomio $p_n(x)$ nei dati forniti allora questo risulterà pari alla funzione. É utile calcolare lo scarto tra i valori di $f(x)$e $p_n(x)$ per $x \neq x_i$.
Sia $f \epsilon C^{n+1}[a,b]$ e sia p_n il polinomio di grado n che interpola f sulle ascisse $\{x_0, ..., x_n\}$, cioè

$$p_n(x_i) = f(x_i),\ i = 0, ..., n.$$

Per ogni x esiste un punto ξ_x appartenente al più piccolo intervallo che contiene i punti $\{x, x_0, ..., x_n\}$ tale che la funzione errore definita da

$$E_n(x) = f(x) - p_n(x) = \frac{f^{(n+1)}(\xi_x)}{(n+1)!}\omega_n(x) \tag{2.12}$$

dove $\omega_n(x) = (x - x_0)(x - x_1)...(x - x_n)$ è un polinomio di grado $n + 1$.
Nella figura 2.3 è rappresentato l'errore calcolato come differenza $f(x) - p(x)$ in tutti gli m punti usati per la rappresentazione, dove:

- $m = 201$
- *listato di riferimento* : *Programma* 01

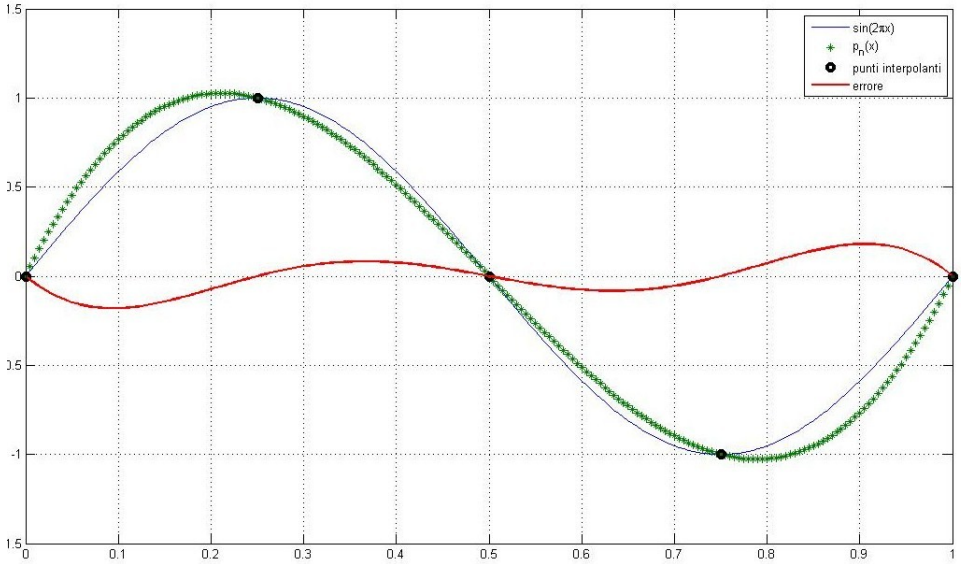


Figura 2.3: Errore polinomio interpolante di Lagrange con n=4

Errore di interpolazione per il polinomio di Newton

Il formalismo di Newton consente di dare una rappresentazione alternativa per l'errore. Sia $p(x)$ il polinomio che interpola la funzione $f(x)$ in $(n + 1)$ punti. Chiamiamo queste ascisse $\{x_0, x_1, ..., x_n\}$. Supponiamo ora di voler aggiungere un punto di interpolazione $x \in \mathbb{R}$ di coordinate $(x, f(x))$. Come noto dalla teoria vista precedentemente non è necessario ricalcolare un nuovo polinomio, ma esprimiamo $p_{n+1}(z)$ semplicemente come

$$p_{n+1}(z) = p_n(z) + f[x_0, x_1, ..., x_n, x]\, \omega(z)$$

Imponendo la condizione di interpolazione in $(x,\ f(x))$ si ha

$$p_n(x) + f[x_0, x_1, ..., x_n, x]\omega_n(x) = f(x).$$

É possibile ottenere la valutazione dell'errore utilizzando l'espressione appena vista

$$E_n(x) = f(x) - p_n(x) = f[x_0, x_1, ..., x_n, x]\omega_n(x).$$

Questa espressione è analoga alla 2.12 ma risulta più generale in quanto non richiede la differenziabilità e nemmeno la continuità della funzione $f(x)$. Fornisce dunque una espressione effettivamente calcolabile per l'errore in un punto x poichè essa non dipende da un punto incognito ζ_x.

É inoltre evidente il legame tra le differenze divise e la derivata di una funzione, per ogni x, esiste un punto $\zeta_x \in I(x_0 .... x_n,\ x)$ tale che

$$f[x_0, x_1, ..., x_n, x] = \frac{f^{(n+1)}(\xi_x)}{(n+1)!}$$

se la funzione è derivabile $(n+1)$ volte le due valutazioni dell'errore dunque coincidono.

2.2.5 Osservazioni sull'errore e i tre teoremi fondamentali

Il teorema 2.12 non pone alcuna limitazione riguardo al posizionamento del punto x; pertanto tale teorema ben si presta anche nei problemi di estrapolazione, ossia quando servono delle valutazioni del polinomio interpolante al di fuori dell'intervallo che contiene i nodi.

Risulta importante tenere conto delle seguenti informazioni fornite dal teorema:

1. il fattore $\left|f^{(n+1)}(\xi_x)\right|$ può essere maggiorato da una costante, infatti come ipotesi iniziale del teorema si è supposta la funzione $f(x)$ derivabile $(n+1)$ volte. Per un n sufficientemente grande, il rapporto può essere dunque reso piccolo a piacere;

2. il fattore $\omega_n(x)$ è un polinomio di grado $(n+1)$ e dipende dalla disposizione dei nodi $\{x_i\}$. Per cui il posizionamento delle ascisse di interpolazione influenza l'andamento dell'errore che al crescere di n ha oscillazioni sempre più ampie.

La conseguenza fondamentale è che l'errore di interpolazione non tende necessariamente a zero quando n tende all'infinito ma il polinomio interpolante potrebbe anche non essere più una buona approssimazione della funzione continua.

Teorema 1. Teorema di Faber. Per ogni distribuzione dei nodi esiste almeno una funzione $f(x) \in C_{[a,b]}$ tale che l'errore di interpolazione $\|E_n(f)\|_\infty$ non converga a zero per $n \to \infty$.

Teorema 2. Per ogni funzione continua esiste comunque almeno una distribuzione dei nodi tale che l'errore di interpolazione converga a zero per $n \to \infty$.

Teorema 3. Teorema di Berstein. Se $f(x) \in C^1_{[a,b]}$ e se le ascisse di interpolazione $\{x_i\}_{i=0}^n$ sono gli zeri del polinomio di *Chebychev* di grado $(n+1)$ allora l'errore di interpolazione tende a zero per $n \to \infty$.

2.2.6 Polinomio di Chebychev

Fino ad ora sono state utilizzate ascisse di interpolazione equispaziate, siccome tale posizionamento dei nodi non garantisce un errore di interpolazione tendente a zero al crescere del numero delle ascisse e dunque del grado del polinomio interpolante, si è deciso di utilizzare anche i nodi di Chebychev e di proporre alcuni confronti con i nodi equispaziati.

Il terzo dei tre teoremi presentati garantisce la convergenza dell'errore a zero se si utilizzano nodi di Chebychev. Ci saranno ovviamente dei limiti numerici anche a questo approccio che verranno trattati più avanti con degli esempi pratici; per semplicità si è partiti consideranto l'intervallo di riferimento $[-1, 1]$.

Si dimostra che i nodi cercati saranno gli zeri del *polinomio di Chebychev* di grado (n+1), $T_{n+1}(x)$.

Tale polinomio può essere definito in vari modi, utilizziamo quello più conveniente:

$$T_{n+1}(x) = \cos[(n+1)\vartheta]$$

con $x = \cos(\vartheta)$ dove ϑ è una variabile di comodo definita nell'intervallo $[0, \pi]$.

Si verifica semplicemente come

$T_0(x) = \cos(0) = 1$

$T_0(x) = \cos(\vartheta) = x$

$T_0(x) = \cos(2\vartheta) = 2\cos^2\vartheta - 1 = 2x^2 - 1.$

Risulta fondamentale che tale costruzione del polinomio $T_{n+1}(x)$ rende semplice la valutazione degli zeri, infatti

$$\cos((n+1)\vartheta) = 0$$

$$(n+1)\vartheta = \frac{\pi}{2} + k\pi = \frac{(2k+1)\pi}{2}$$

da si ottiene $\vartheta = \frac{2k+1}{2n+2}\pi$ *per* $n = 0, \pm1, \pm2, ...$

Infine si sono trovati i valori delle ascisse di interpolazione

$$x_k = \cos\vartheta_k = \cos(\frac{2k+1}{2n+2}\pi) \quad con \quad k = 0, 1, 2.....n.$$

É stato già precisato che l'intervallo di riferimento è $[-1, 1]$, quindi se si vogliono ottenere i nodi di Chebychev in un intervallo qualsiasi $[a, b]$ occorrerà applicare la trasformazione

$$t = \frac{b-a}{2}\, x + \frac{b+a}{2}. \tag{2.13}$$

2.2.7 Funzione di Runge

Tale funzione viene definita nell'intervallo $[-1, 1]$ ed ha la seguente espressione

$$f(x) = \frac{1}{1+25x^2}.$$

Essa è una funzione infinitamente differenziabile. L'errore di interpolazione corrispondente ai nodi equispaziati tende all'infinito, mentre l'errore di interpolazione con i nodi di Chebychev tende a zero. La figura 2.4 rappresenta la funzione di Runge e i polinomi interpolanti di Lagrange valutati sia coi nodi equispaziati sia che con quelli di Chebychev.

- *listato di riferimento* : *Programma* 02

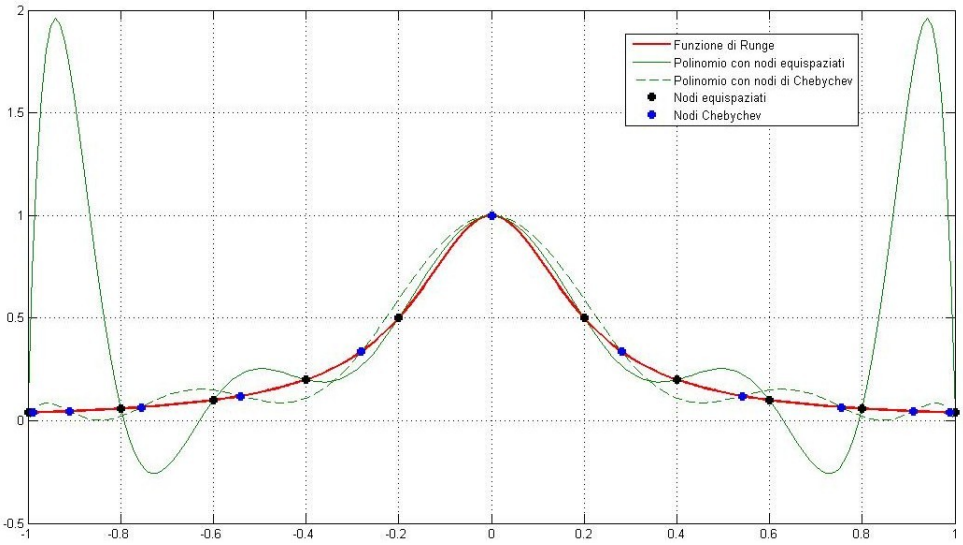


Figura 2.4: Interpolazione con Lagrange della funzione di Runge

La funzione di Runge mette in evidenza che l'utilizzo di nodi equispaziati non garantisce la convergenza dell'errore a zero e il polinomio interpolante non risulta più una buona approssimazione della funzione di partenza.

2.3 Applicazioni in Matlab con i polinomi interpolanti

Applicazione 1 : Polinomi interpolanti di Lagrange

La prima applicazione mostra per una funzione $f(x)$ i polinomi interpolanti ottenuti variando il grado e la scelta dei nodi (equispaziati o di Chebychev).

Una volta fissata la disposizione dei nodi, scegliamo il grado minimo a cui siamo interessati e il grado massimo, per semplicità rappresentiamo otto polinomi per otto gradi differenti; i parametri scelti inizialmente sono:

- $f(x) = sin(2\pi x)$
- $n = 3 : 10$
- 201 *punti per la visualizzazione e valutazione dell' errore*
- *listato di riferimento* : *Programma* 04.

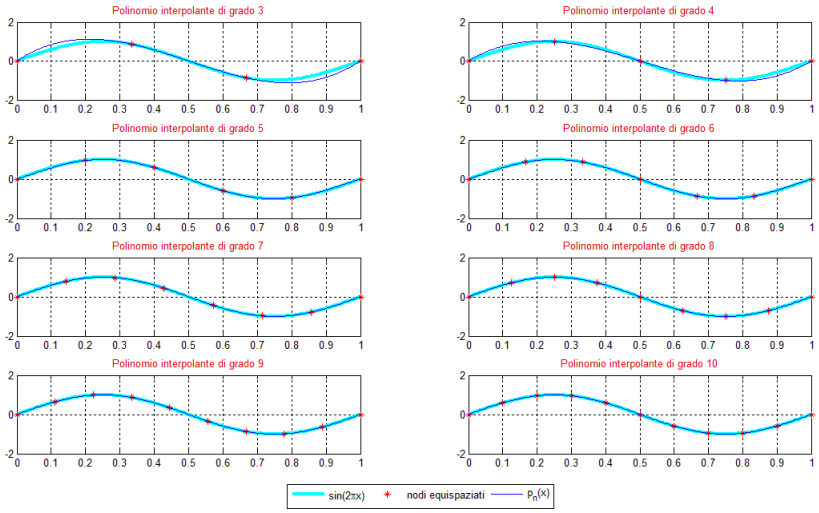


Figura 2.5: Nodi Equispaziati: polinomi interpolanti Lagrange con grado variabile

L'intervallo di riferimento per la funzione in esame è $[0, 1]$, quando si scelgono i nodi di Chebychev è necesaria la trasformazione imponendo $a = 0$ *e* $b = 1$ e applicando dunque la 2.13.

Il grafico dell'errore 2.7 è ottenuto calcolando la norma 2 del vettore errore ottenuto per ognuno dei 16 polinomi rappresentati nei grafici 2.6e2.7. Si potrà dunque apprezzare come al variare del grado varia l'errore nel caso di nodi equispaziati e di Chebychev.

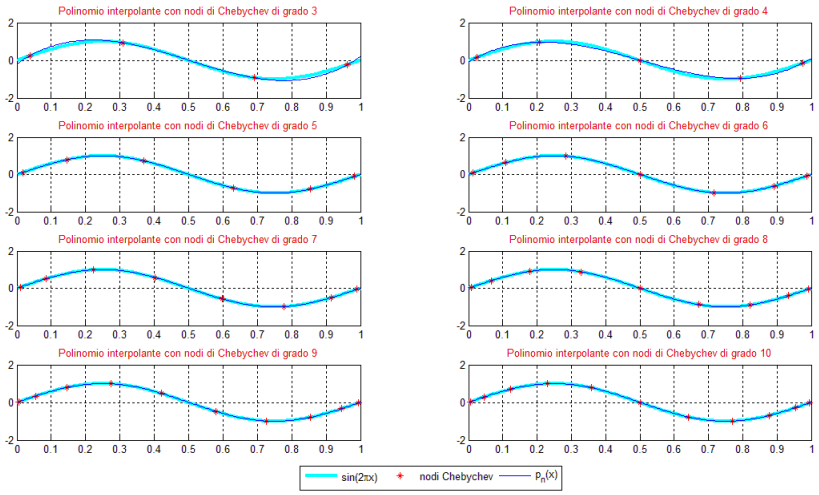


Figura 2.6: Nodi Chebychev:polinomi interpolanti Lagrange con grado variabile

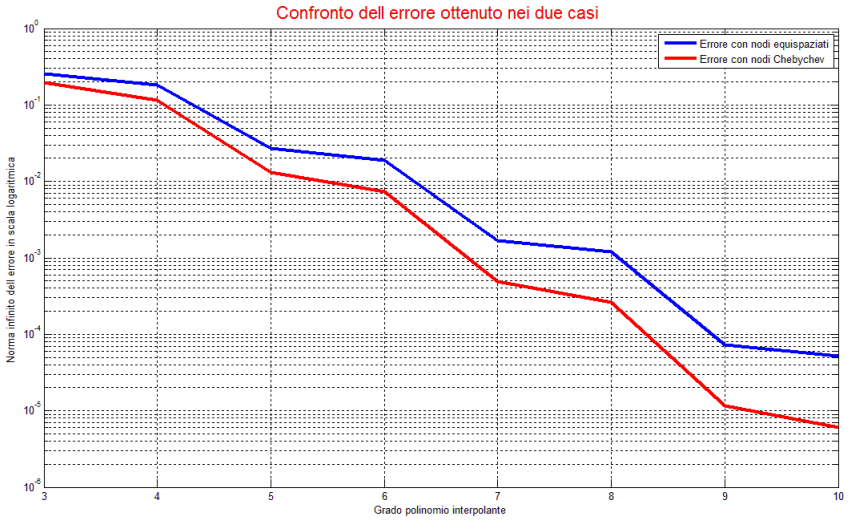


Figura 2.7: Confronto errore al variare del grado del polinomio interpolante di Lagrange e dei nodi

Applicazione 2 :Polinomi interpolanti di Newton

La seconda applicazione realizzata mette a confronto per la funzione $f(x)$ i polinomi interpolanti di Newton ottenuti variando il grado e la scelta dei nodi (equispaziati o di Chebychev).

Una volta fissata la disposizione dei nodi, si scelgono il grado minimo e il grado massimo; per semplicità sono stati rappresentati ancora una volta otto polinomi per otto gradi differenti; i parametri scelti inizialmente sono:

- $f(x) = sin(2\pi x)$
- $n = 3 : 10$
- 201 *punti per la visualizzazione e valutazione dell' errore*
- *listato di riferimento* : *Programma* 06.

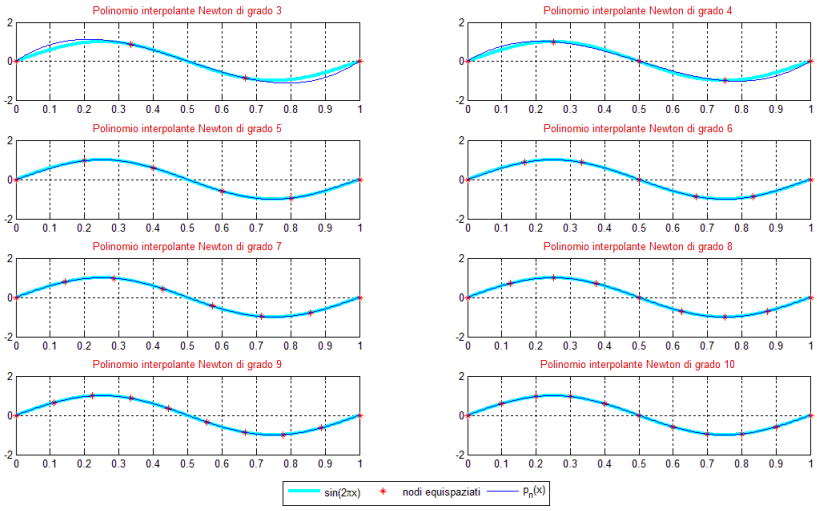


Figura 2.8: Nodi Equispaziati: polinomi interpolanti Newton con grado variabile

Utilizzando la funzione *ylim* è stato fissato l'intervallo di riferimento nell'asse y. L'intervallo scelto è stato $[-2, 2]$.

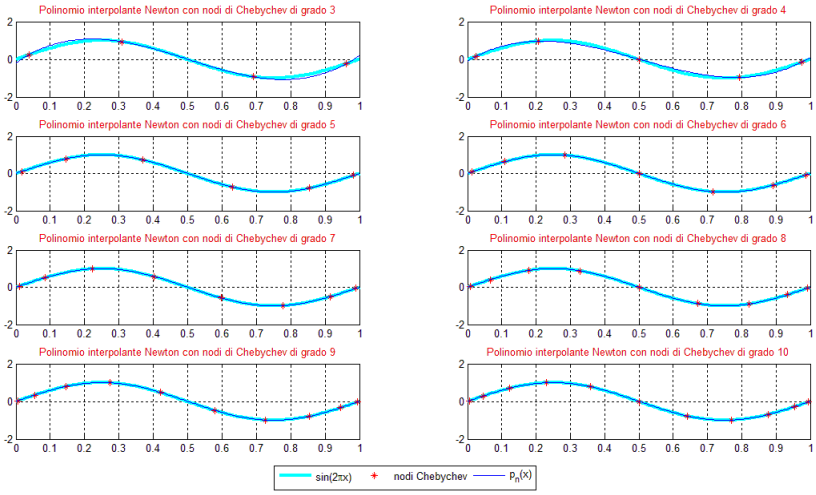


Figura 2.9: Nodi Chebychev:polinomi interpolanti Newton con grado variabile

Il grafico complessivo mostra ancora una volta come l'errore nel caso di nodi di Chebychev sia minore di quello valutato nel caso di nodi equispaziati per ogni polinomio rappresentato.

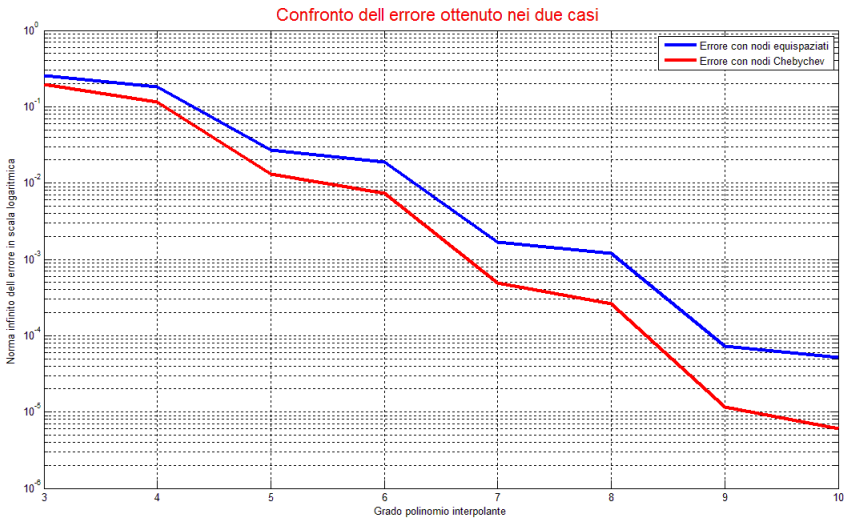


Figura 2.10: Confronto errore al variare del grado del polinomio interpolante di Newton e dei nodi

Applicazione 3 : Funzione di Runge con Lagrange e Newton

Per la terza applicazione utilizziamo la funzione di Runge presentata alla sezione 2.2.7.

La costruzione dell'errore è stata fatta utilizzando sia i polinomi di Lagrange che di Newton ma siccome i risultati sono pressochè gli stessi si presentano solo i grafici relativi al polinomio interpolante di Lagrange; i parametri scelti inizialmente sono:

- $f(x) = \frac{1}{1+25x^2}$ nell'intervallo di riferimento $[-1, 1]$
- $n = 3 : 10$
- 201 *punti per la visualizzazione e valutazione dell' errore*
- *listati di riferimento* : *Programma* 09, *Programma* 10 .

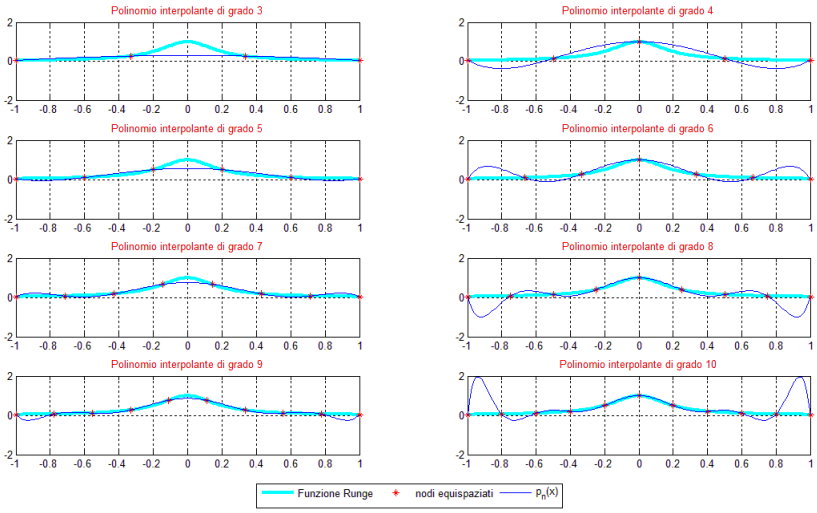


Figura 2.11: Nodi Equispaziati: polinomi interpolanti Lagrange con grado variabile

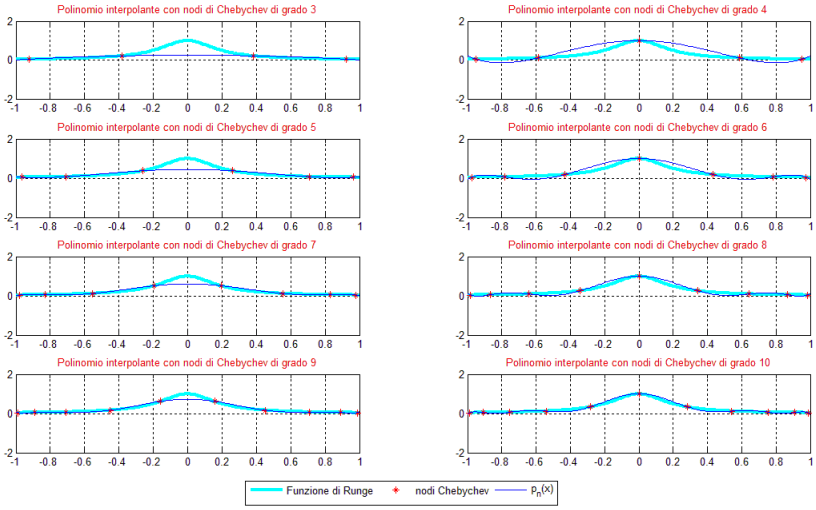


Figura 2.12: Nodi Chebychev:polinomi interpolanti Lagrange con grado variabile

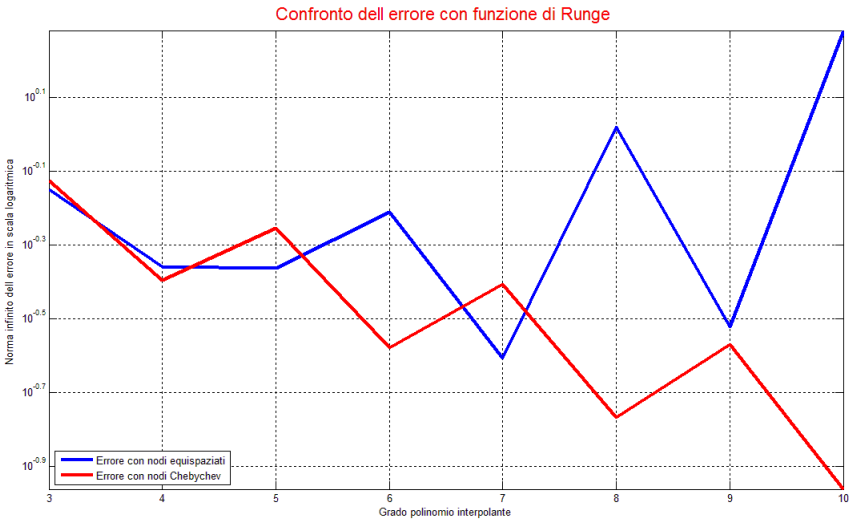


Figura 2.13: Confronto errore al variare del grado del polinomio interpolante di Lagrange e dei nodi

Le figure 2.11, 2.12, 2.13 sono state prodotte dal listato nel *Programma* 09 allegato alla tesina.

Applicazione 4 : Limiti e osservazioni sulle applicazioni presentate

Le applicazioni precedenti dimostrano come l'errore ottenuto nel caso di nodi di Chebychev sia sempre inferiore a quello ottenuto considerando nodi equispaziati, erano dei risultati prevedibili sostenuti anche dal 2.2.5.

Aumentando ulteriormente il grado del polinomio di Lagrange, l'errore valutato coi nodi equispaziati staziona a valori molto elevati, mentre quello valutato coi nodi di Chebychev è praticamente nullo:

- $n = 68 : 75$
- *listati di riferimento* : *Programma* 07, *Programma* 08,
- *Programma* 11, *Programma* 12.

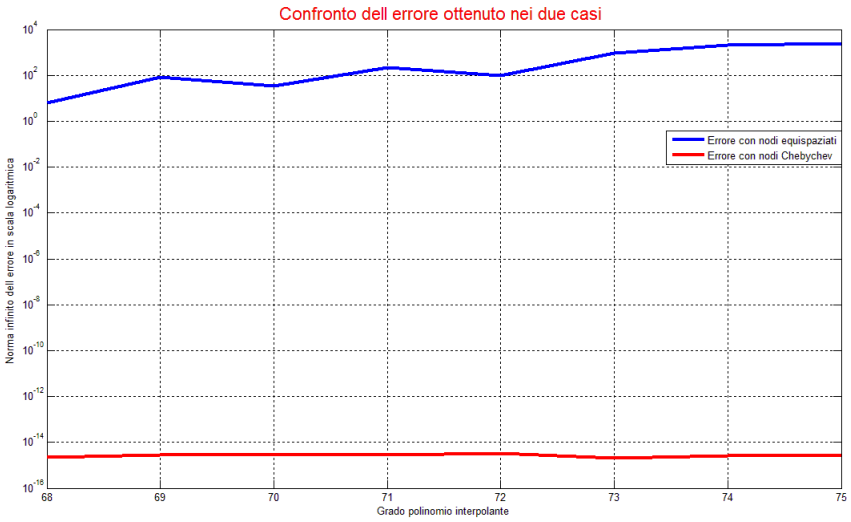


Figura 2.14: Funzione seno, confronto errore al variare del grado e dei nodi del polinomio interpolante di Lagrange

Utilizzando invece i polinomi di Newton, con la stessa funzione e lo stesso intervallo di variazione del grado si ha che nel caso di nodi di Chebychev la convergenza non è garantita e l'errore diventa addirittura superiore al caso di nodi equispaziati.

Infine la figura 2.16 mostra come approssimando la funzione di Runge con il polinomio di Lagrange l'errore calcolato usando i nodi equispaziati è molto elevato, mentre quello coi nodi di Chebychev è praticamente nullo.

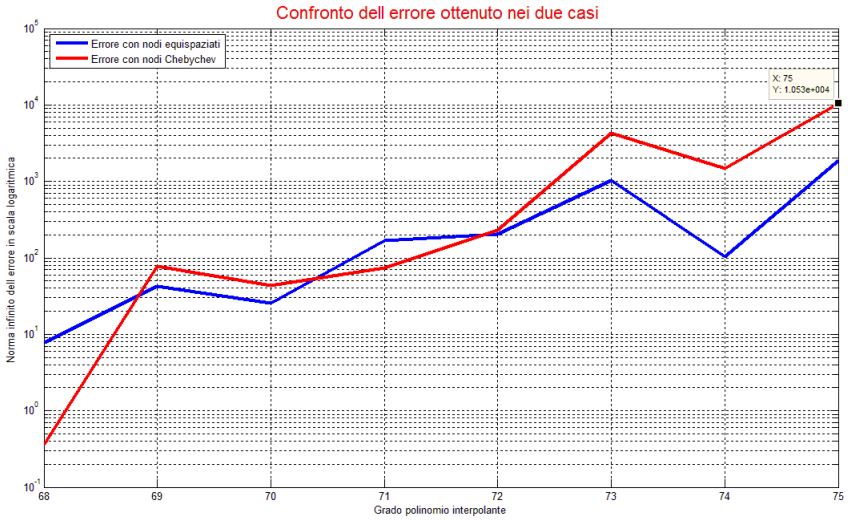


Figura 2.15: Funzione seno, confronto errore al variare del grado del polinomio interpolante di Newtone dei nodi

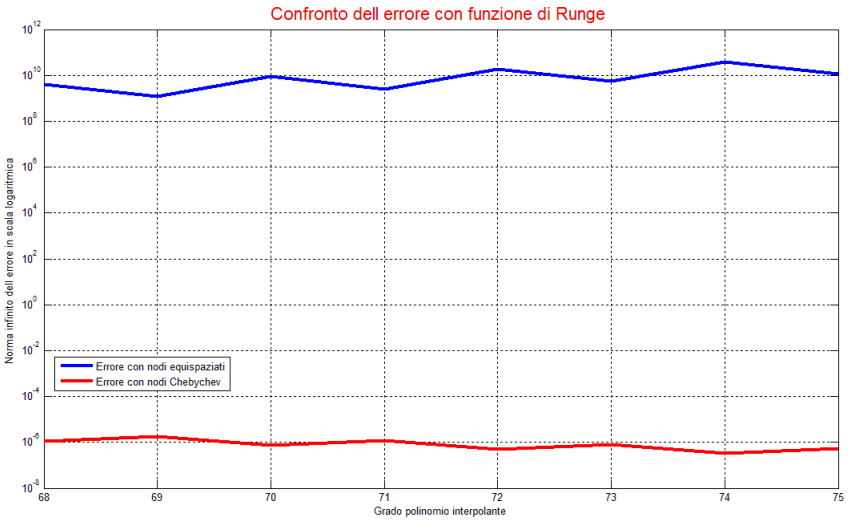


Figura 2.16: Confronto errore al variare del grado del polinomio interpolante di Lagrange e dei nodi con la funzione di Runge

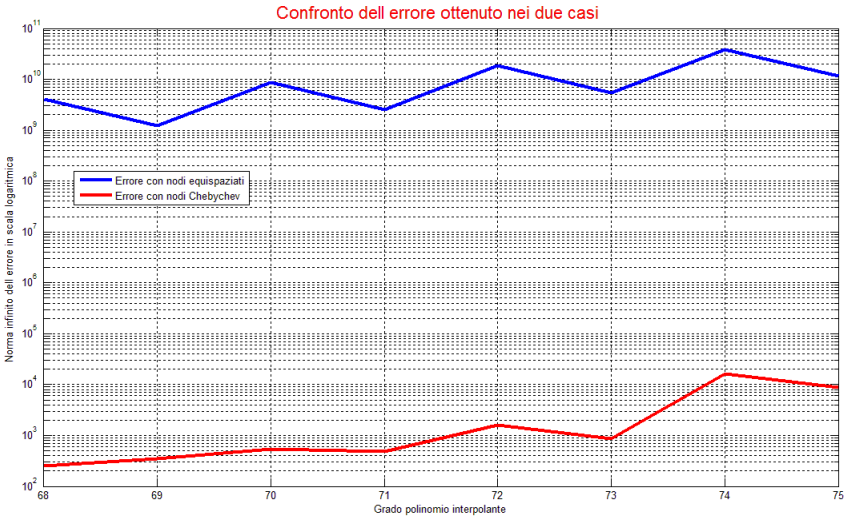


Figura 2.17: Confronto errore al variare del grado del polinomio interpolante di Newton e dei nodi con la funzione di Runge

Approssimando invece la funzione di Runge con il polinomio di Newton e mantenendo la stessa variazione del grado del polinomio ottengo la figura 2.17 dove entrambi gli errori convergono a valori alti tanto da far perdere di significato l'interpolazione stessa.

Per concludere si noti come i nodi di Chebychev all'aumentare del grado, in prossimità del valore -1 siano talmente vicini, da causare un effetto analogo a quelli equispaziati, infatti i nodi di Chebychev a differenza di quelli equispaziati si dispongono molto più vicini agli estremi dell'intervallo tanto da risultare a parità di grado del polinomio interpolante più vicini di quelli equispaziati . La figura2.18 mostra questo effetto per $n = 75$ estrapolata dal *listato di riferimento* : *Programma* 12.

2.3.1 Conclusioni

Con le sezioni precedenti si è visto come applicare correttamente i polinomi di Lagrange e di Newton, evidenziando per ognuno i limiti di applicazione.

Si consideri il caso della funzione $f(x) = sin(2\pi x)$ nell'intervallo di riferimento $[0, 1]$, interpolata con il polinomio di Newton scelto con un grado molto elevato. É stata ottenuta la figura 2.19 imponendo i seguenti parametri:

- $n = 75$;
- 51 *punti per visualizzare la funzione*

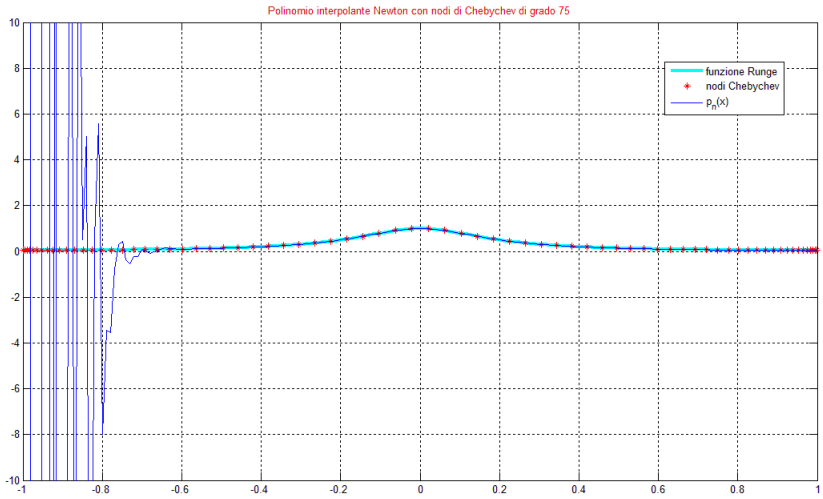


Figura 2.18: Polinomio interpolante di Newton con n=75 e nodi di Chebychev

- *escursione asse y limitata all'intervallo* $[-4, 4]$
- 76 *nodi equispaziati*
- *listato di riferimento* : *Programma* 13

Sia i polinomi di Lagrange che quelli di Newton per come sono definiti, sono utilizzati principalmente quando il numero dei punti di interpolazione non è troppo elevato. Si può notare, indipendentemente dal tipo di nodi scelti, che quando il grado cresce in maniera troppo elevata il polinomio diverge (non interpola nemmeno i punti in cui dovrebbe valere la condizione di appartenenza). Lo studio condotto fa emergere i problemi numerici che mostrano il limite di applicazione di tali polinomi. Se si vuole approssimare una funzione con un numero di punti di interpolazione elevato si ricorre ad un altra classe di funzioni, le funzioni $SPLINE$, che approssimano la funzione con polinomi di grado basso raccordati adeguatamente per soddisfare un numero di punti interpolanti molto grande.

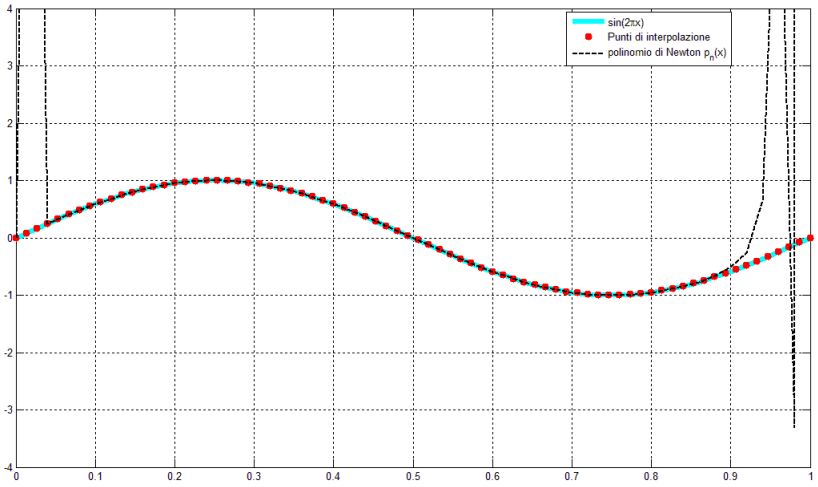


Figura 2.19: Funzione seno approssimata con un polinomio di Newton di grado n=75

Capitolo 3

Interpolazione 3D

3.1 Interpolazione di Lagrange in 3D

In questo caso si è trattato di effettuare un'interpolazione bidimensionale con proiezione lungo l'asse z. Siano date le coppie di punti (n_x, n_y) che vanno a formare una griglia sul piano xy. Mentre nel paragrafo 2.2.2 si andavano a calcolare n polinomi interpolatori ora questi sono dati in questa forma

$$p_{n_x,n_y}(x,y) = \sum_{i=1}^{n_x+1} \sum_{j=1}^{n_y+1} z(x_i, y_j) f_{i,j}(x,y), \tag{3.1}$$

dove $f_{i,j}(x,y) = L_i(x)L_j(y)$.

3.1.1 Applicazione interpolazione di Lagrange in 3D

É stata costruita una funzione per effettuare l'interpolazione in 3D in riferimento al paragrafo 3.1.
Sono stati impostati i seguenti parametri:

- $m = 41$
- $n_x = 4$
- $n_y = 4$
- *nodi equispaziati sia per asse x che per asse y.*
- *listati di riferimento* : *Programma* 14, *Programma* 15

La funzione errore è riportata in figura 3.2.

Listing 3.1: Funzione new_lagint_3D

```
%function [px,py,Lx,Ly,P,Zf]=new_lagint_3D(x,y,t_x,t_y,fx,fy)
function [P,Zf]=new_lagint_3D(x,y,t_x,t_y,fx,fy)
m = length(x);%numero dei punti scelti per valutare la funzione
n_x= length (t_x)-1;%numero dei punti di interpolazione
n_y= length (t_y)-1;
%Si richiama la funzione new_lagint per la costruzione dei polinomi
%interpolanti px,py e le funzioni di Lagrange Lx,Ly
[px,Lx]=new_lagint(x,t_x,fx);
[py,Ly]=new_lagint(y,t_y,fy);

[Xt,Yt]=meshgrid(t_x,t_y);
%Zt=zeros(n_y+1,n_x+1);
Zf=(sin(2*pi*Xt).*sin(2*pi*Yt))';

P=zeros(m); %matrice quadrata contenente il polinomio interpolante
for k=1:m           %k indice riga matrice P
    for l=1:m       %l indice colonna
        for i=1:n_x+1   %indice riga matrice Zf contenente la
          funzione Z
                        %calcolata nei punti di interpolazione
            for j=1:n_y+1 % indice colonna
                P(k,l)=P(k,l)+(Zf(i,j)*(Lx(k,i)*Ly(l,j)));

            end
        end
    end
end
```

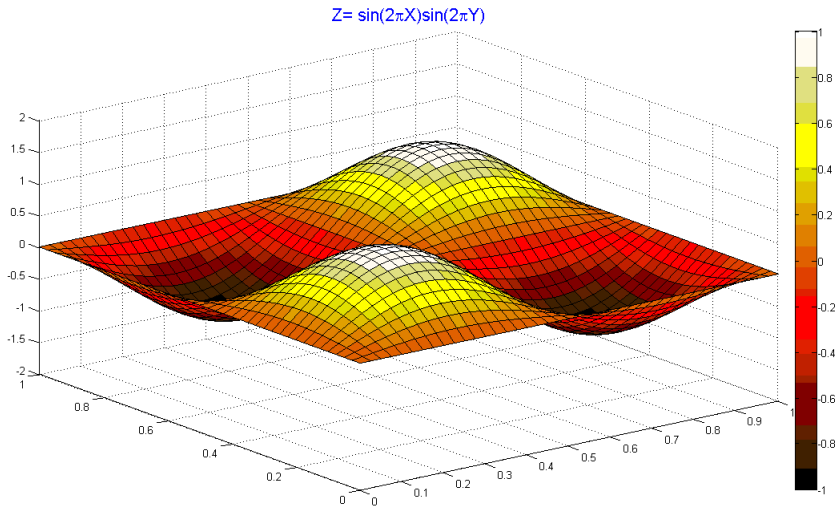


Figura 3.1: Funzione da interpolare

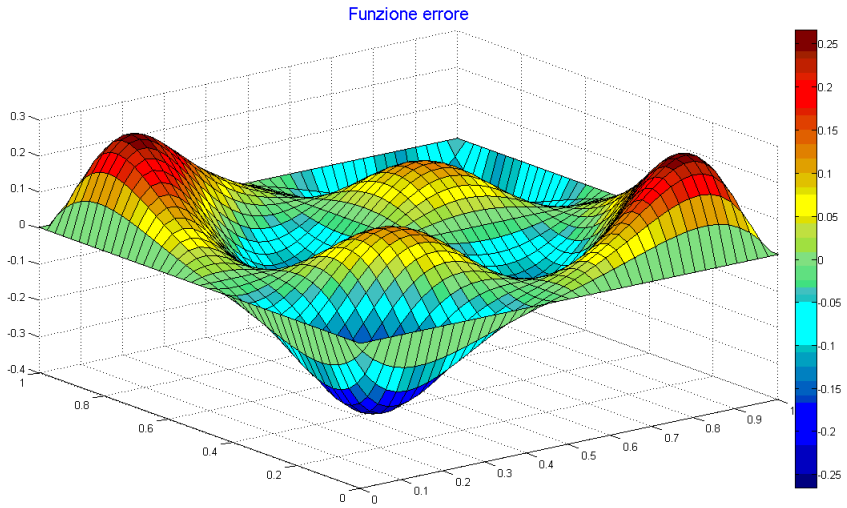


Figura 3.2: Funzione errore con $n_x = 4,\ n_y = 4$

Osservazioni figura 3.3

Come si vede nel listato, la funzione *new lagint* $3D$ è stata implementata per permettere un campionamento dei nodi diverso per i due assi. In particolare nella figura in esame la scelta è stata la seguente:

- $n_x = 60$ ovvero 61 punti di interpolazione
- $n_y = 65$ ovvero 66 punti di interpolazione

Il fenomeno che si osserva nella figura è analogo a quello che avviene nell'approssimazione di funzioni in 2D. All'aumentare del grado i polinomi caratteristici di Lagrange diventano sempre più instabili specialmente quando calcolati per valori vicini agli estremi dell'intervallo. Infatti per come sono definiti (equazione 2.7), il rapporto interno alla produttoria avrà un numeratore sempre più grande e un denominatore sempre più piccolo al crescere del grado dei polinomi e dunque dell'indice k. Per valori molto elevati come quelli in esame il valore dei polinomi di Lagrange nei punti agli angoli della figura diventa estremamente grande. L'effetto è stato tale da non permettere inizialmente la visualizzazione della figura, per evitare questo sono stati introdotti dei cicli *for* per limitare i valori della matrice contenente i punti utili per la visualizzazione del polinomio interpolante nell'intervallo $[-2, 2]$, in questo modo è stato possibile rendere visibile la figura. In realtà le oscillazioni agli angoli sono dunque molto più ampie ed evidenziano ancora una volta i limiti dell'interpolazione di Lagrange quando il numero dei nodi scelti è molto elevato.

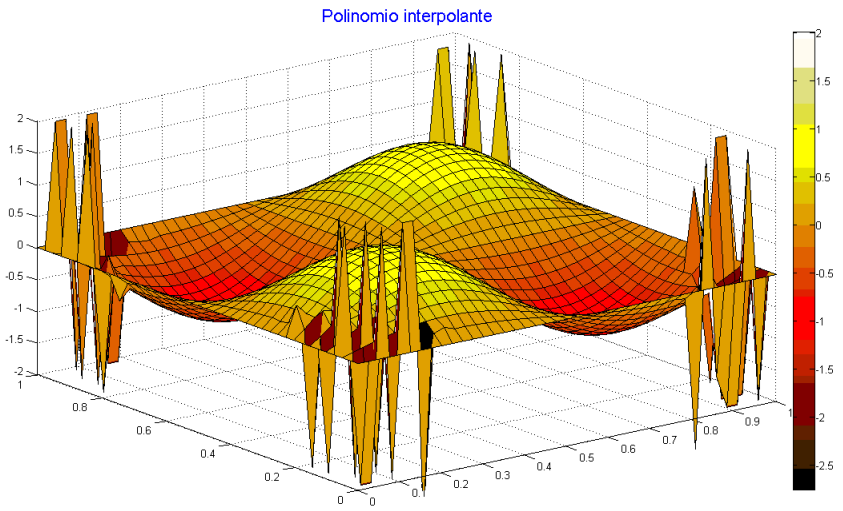


Figura 3.3: Polinomio interpolante $n_x = 60,\ n_y = 65$

3.1.2 Funzione di Runge e polinomio interpolante in 3D

I grafici precedenti sono riferiti alla funzione $f(x,y) = \sin(2\pi x)\sin(2\pi y)$ che nel caso di identico campionamento degli assi equivale al $\sin^2(2\pi x)$. Si consideri ora la funzione $f(x,y) = \frac{1}{1+25x^2}\ \frac{1}{1+25y^2}$ e per semplicità si utilizzi lo stesso campionamento degli assi, e dunque lo stesso grado del polinomio interpolante per le due dimensioni. Siccome i nodi equispaziati scelti per la funzione di Runge non danno una buona approssimazione sono state sviluppate due applicazioni.

Applicazione 1 Sono stati scelti i seguenti parametri:

- $m = 41$ *punti in cui si valutano la funzione*
- $n = 10$
- *nodi Chebychev*
- *listato di riferimento* : *Programma* 16, *Programma* 17.

L'errore è calcolato come differenza della matrice Z rappresentativa della funzione di Runge in due variabili e della matrice P rappresentativa del polinomio interpolante. La matrice Z è ottenuta adoperando per prima cosa la funzione in matlab $meshgrid(x,y)$ (dove x e y sono i due vettori di dimensione m) per ottenere le matrici X e Y; poi $Z = (1./(1+25*X.^2)).*(1./(1+25*Y.^2))$.

La funzione che si usa per rappresentare tali matrici in 3D è $surf(X,Y,Z)$, tale grafico è in figura 3.4, in figura 3.5 è invece rappresentato il polinomio

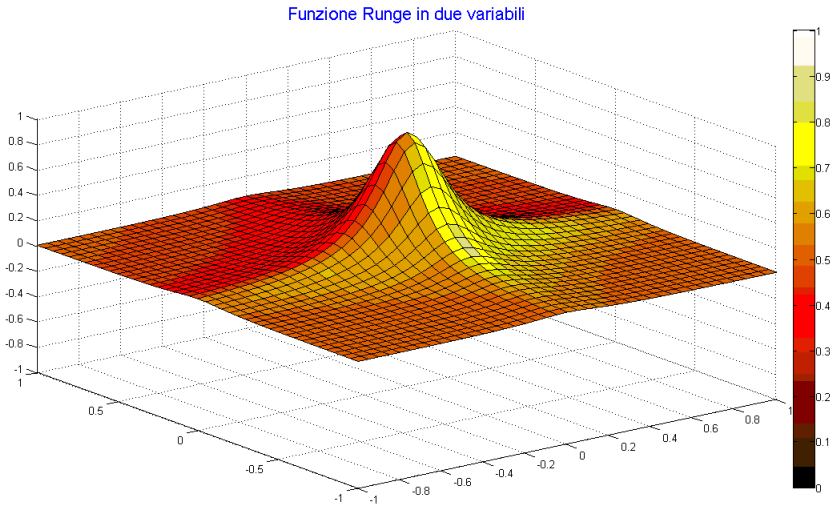


Figura 3.4: Funzione di Runge in due variabili

interpolante di grado 10 che si ottiene con la sequenza di comandi presente nel *Programma* 16 allegato alla tesina; infine in figura3.6 è riportato l'errore.

Applicazione 2 In analogia alle funzioni interpolanti sviluppate nel capitolo 2, si propone un confronto fra la funzione di Runge (in due variabili) interpolata con polinomi coi nodi equispaziati e con polinomi che utilizzano i nodi di Chebychev; il grado del polinomio sarà fatto variare nell'intervallo $[5, 8]$ ritenuto sufficiente per mostrare come gli stessi fenomeni osservati nel capitolo 2 si ripresentino anche nel caso 3D. La funzione di Runge è stata valutata nell'intervallo $[-1, 1]$ e tutti gli assi sono stati settati utilizzando il comando $axis([-1\ 1\ \ -1\ 1\ \ -1\ 1])$.

Nella figura 3.7 sono presentati quattro polinomi interpolanti della funzione di Runge in due variabili, nella figura 3.8 è rappresentato l'errore. Nelle figure 3.9 e 3.10 sono rappresentati rispettivamente i polinomi interpolanti valutati utilizzando i nodi di Chebychev e il relativo errore valutato come spiegato nella prima applicazione presentata.

Sono stati riscontrati gli stessi problemi incontrati nel capitolo 2, all'aumentare del grado del polinomio interpolante e dunque del numero di nodi si presentano due situazioni differenti a seconda della disposizione di essi:

- Nodi equispaziati: all'aumentare del grado si può notare come le oscillazioni del polinomio interpolante aumentino e di come l'errore diventi sempre più grande. La scelta dei nodi equispaziati non garantisce la convergenza dell'errore a zero e con tale disposizione risulta inutile avere molti punti di

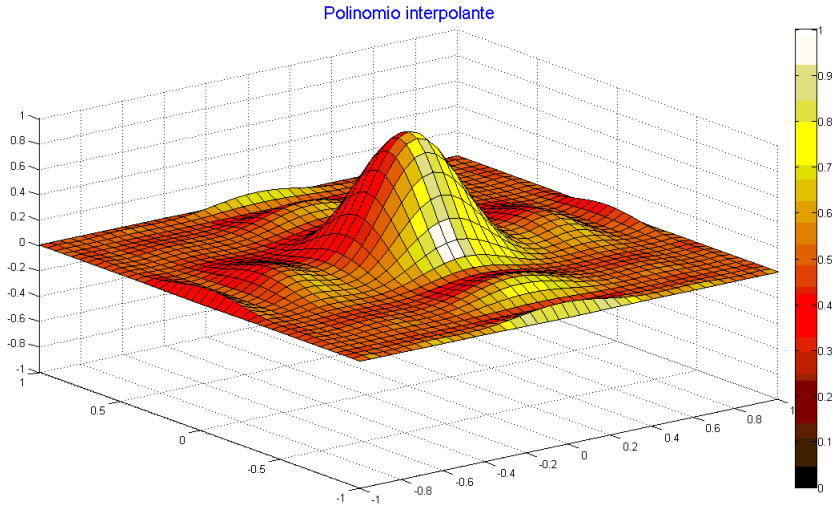


Figura 3.5: Polinomio interpolante con $n = 10$

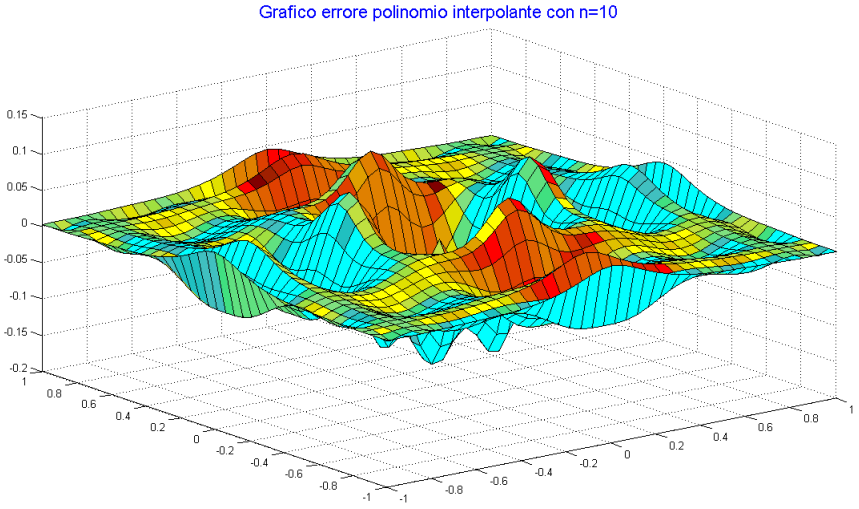


Figura 3.6: Funzione errore $Z - P$

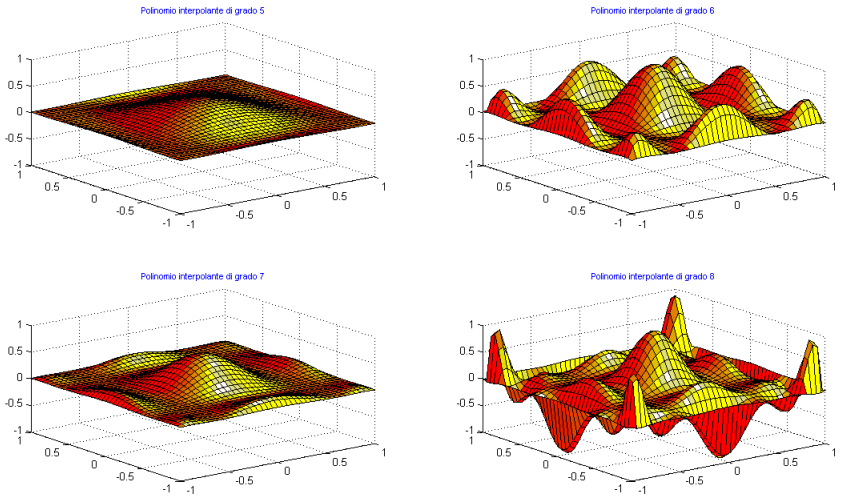


Figura 3.7: Nodi Equispaziati: polinomi interpolanti

campionamento. Pertanto conoscere meglio la funzione in termini di punti da interpolare non si traduce in una maggiore precisione del polinomio interpolante.

- Nodi Chebychev: all'aumentare del grado si evidenziano i problemi numerici incontrati nel capitolo 2 . Nonostante ciò la scelta dei nodi di Chebychev garantisce ancora una volta una approssimazione migliore rispetto ai nodi equispaziati.

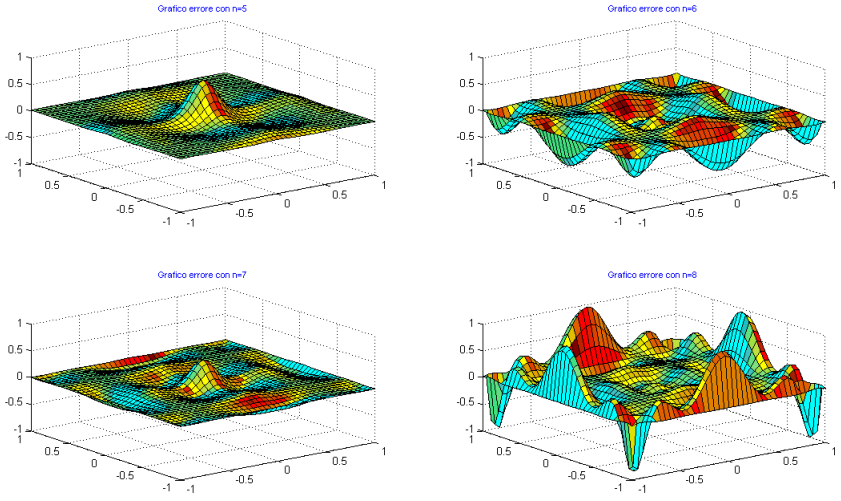


Figura 3.8: Nodi Equispaziati:errore

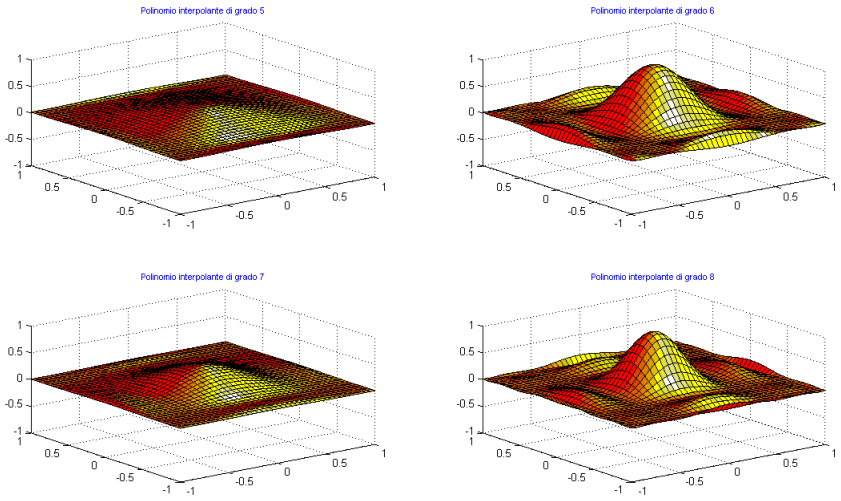


Figura 3.9: Nodi Chebychev:Polinomi interpolanti

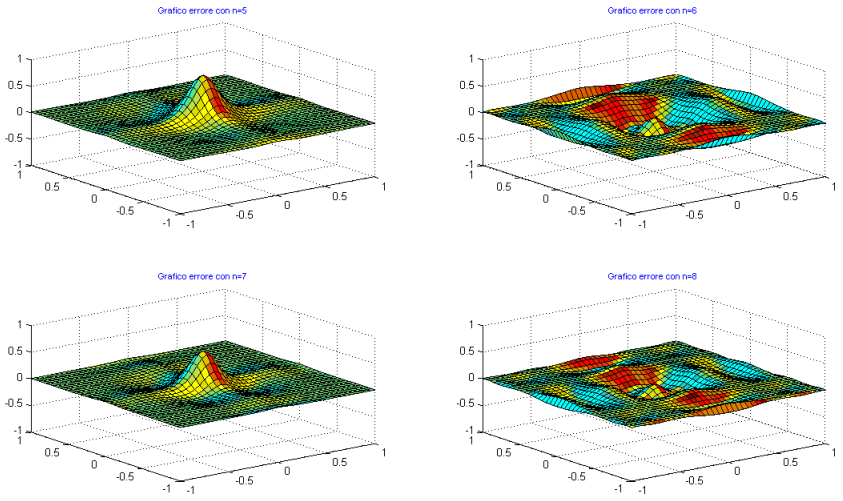


Figura 3.10: Nodi Chebychev: Errore

Capitolo 4

Approssimazione ai minimi quadrati

Il polinomio di **migliore approssimazione** per una funzione f è quel polinomio di grado n che minimizza la norma dell'errore

$$\min_{p_n \epsilon\, \Pi_n} \|p_n - f\| .$$

Dove utilizzando la norma infinito si parla di **migliore approssimazione uniforme** e l'espressione assume questa forma

$$\|p_n - f\|_\infty = \max_{x \epsilon\, [a,b]} |p_n(x) - f(x)| ,$$

invece il polinomio che minimizza la norma di $L^2[a,b]$ è così dato

$$\|p_n - f\|_2 = \left(\int_a^b [p_n(x) - f(x)]^2 dx \right)^{\frac{1}{2}}$$

chiamato di **migliore approssimazione nel senso dei minimi quadrati**.

Tra i due problemi quello più utilizzato, per l'approssimazione di funzioni, è l'ultimo in quanto richiede un minor costo computazionale.

4.1 Caso discreto

In diversi problemi la funzione da approssimare è conosciuta tramite un certo numero di valori affetti da errore. Data la forte presenza di errori sui dati non ha senso interpolare ma si effettua un'approssimazione ai minimi quadrati tramite una discretizzazione della norma-2.

Si considerino la ascisse $\{x_0, x_1, ..., x_m\}$ degli $m+1$ punti per i quali sono noti i valori $\{y_0, y_1, ..., y_m\}$ pertanto $y_i = f(x_i)$. Fissato un numero naturale $n \leq m$ per ogni $p_n \epsilon\ \Pi_n$ si considera la norma

$$\|p_n - f\|_2 = \left(\sum_{i=0}^{m} [p_n(x) - f(x)]^2 \right)^{\frac{1}{2}}. \tag{4.1}$$

Si vuole determinare il polinomio $p_n^*(x)$ di grado n che risolva il problema di minimo

$$\min_{p_n \epsilon\, \Pi_n} \|p_n - f\|_2^2, \tag{4.2}$$

poichè i problemi sono equivalenti si minimizza il quadrato della norma.

Se $m = n$ la soluzione coincide con il polinomio interpolante, se $m > n$ invece fornisce la miglior approssimazione nel senso dei minimi quadrati rispetto alla norma discreta (4.1).

Si esprime la formula e poi la si minimizza; usando la base canonica si ottiene

$$p_n(x_i) = \sum_{j=0}^{n} a_j x_i^j = (X\mathbf{a})_i, \qquad i = 0, ..., m,$$

dove $\mathbf{a} = (a_0, ..., a_n)^T\ \epsilon\ \mathbb{R}^{n+1}$ è il vettore dei coefficienti del polinomio e X è la matrice di Vandermonde rettangolare di dimensione $(m+1) \times (n+1)$

$$X = \begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_m & x_m^2 & \cdots & x_m^n \end{bmatrix}.$$

Quindi si può scrivere

$$\|p_n - f\|_2^2 = \sum_{i=0}^{m} \left[\sum_{j=0}^{n} a_j x_i^j - y_i \right]^2 = \sum_{i=0}^{m} [(X\mathbf{a})_i - y_i]^2 = \|X\mathbf{a} - \mathbf{y}\|_2^2,$$

pertanto il problema (4.2) è equivalente alla risoluzione del problema

$$\min_{a\, \epsilon\, \mathbb{R}^n} \|X\mathbf{a} - \mathbf{y}\|_2^2,$$

cioè alla risoluzione nel senso dei minimi quadrati del sistema lineare sovradeterminato

$$X^T X\mathbf{a} = X^T \mathbf{y} \tag{4.3}$$

oppure usando la fattorizzazione Cholesky della matrice $X^T X$ o la fattorizzazione QR della matrice X.

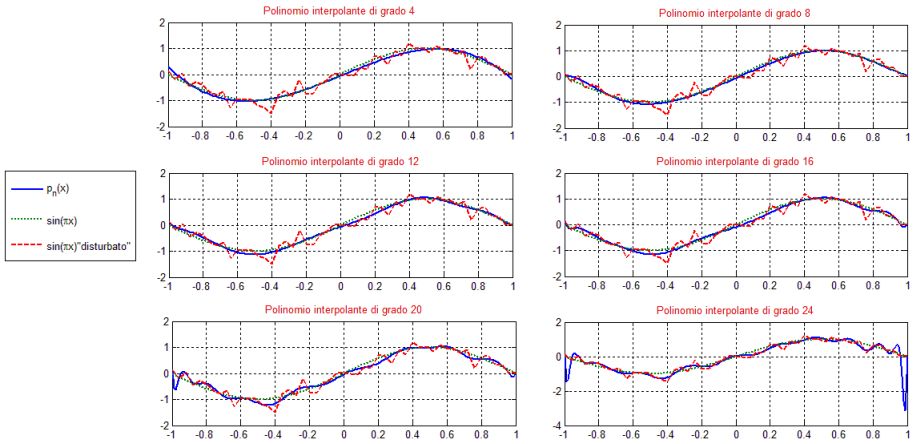


Figura 4.1: Approssimazione ai minimi quadrati per alcuni valori di n

4.2 Applicazione in Matlab

Questo capitolo aggiuntivo ha l' obiettivo di esplorare alcune funzioni matlab utili per risolvere dei problemi ai minimi quadrati. Supponiamo di avere la funzione $f(x) = \sin(\pi x)$ e di disturbarla in $N = 50$ punti con delle variazioni random, in modo da avere non più un $sin(\pi x)$ ma una funzione simile disturbata. L'obiettivo è quello di approssimare tale funzione disturbata non più considerando i punti di interpolazione ma nel senso dei minimi quadrati.

In particolare sono state utilizzate le due funzioni: polyfit e polyval.

- $polyfit(t, b, n)$: crea il polinomio di grado n che approssima il polinomio nel senso dei minimi quadrati.
- $polyval(a, x)$: calcola il valore del polinomio negli m punti scelti per la visualizzazione.

Il programma matlab di riferimento è il *Programma* 18.

É stato possibile implementare l'algoritmo soprastante tramite l'uso di funzioni fornite da Matlab. In particolare sono state utilizzate le due funzioni: polyfit e polyval.

- Polyfit: generalmente è usata per costruire il polinomio interpolante ma in questo caso il risultato non è un polinomio che passa per i punti dati, ma un polinomio che cerca di ricostruire l'insieme dei dati.
- Polyval: permette di determinare il polinomio in determinati punti.

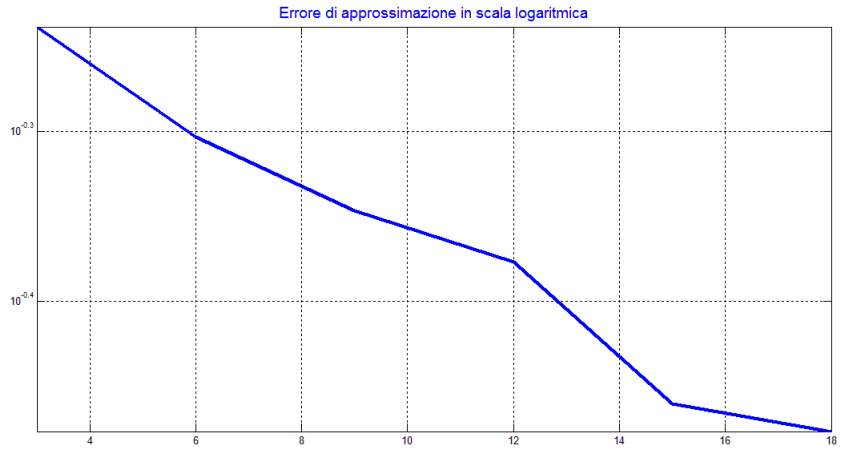


Figura 4.2: Grafico dell' errore al variare del grado

La figura 4.1 mostra la valutazione del polinomio interpolante per diversi valori del grado n , l'errore associato a tali polinomi è riportato in scala logaritmica nella figura 4.2.

Elenco delle figure