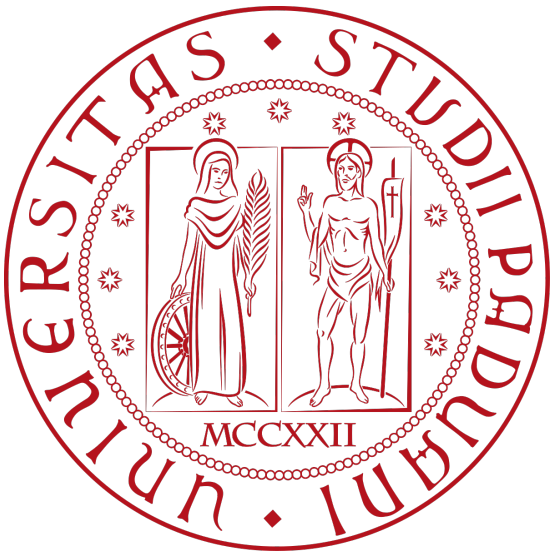
Università degli Studi di Padova



Relazione per:

INTERPOLAZIONE POLINOMIALE CON NODI DI LEJA APPROSSIMATI

Realizzata da:
Riccardo Pontello, matricola 2076441
Mihaela-Mariana Romascu, matricola 2079726
Szymon Fyda, matricola 2101080

Corso: Programmazione/Calcolo Numerico (MATLAB)

Data: 30 gennaio 2026

1 Introduzione

Supponiamo di avere delle valutazioni $x \to f(x)$ e delle misurazioni $(x_i, y_i = f(x_i))$, $i = 0, 1, ..., m$. L'interpolazione è il processo con cui si cerca di costruire una funzione g, tale che $g(x_i) = y_i \ \forall i$, con g scelta in maniera tale da renderla facile da costruire, valutare, derivare e integrare.

L'obiettivo del progetto è implementare in MATLAB l'interpolazione polinomiale sfruttando i *nodi di Leja* su $[-1, 1]$, ma essendo il calcolo dei veri nodi computazionalmente molto oneroso, ci basiamo sui *nodi di Leja approssimati* estratti da una mesh fitta di 5000 punti.
Nella soluzione verranno implementati due metodi per il calcolo dell'interpolazione con tali nodi, di cui si stimerà la *costante di Lebesgue* di ciascuno per valutarne la stabilità e successivamente, verrà comparata l'accuratezza dell'interpolante, confrontandola con quella ottenuta utilizzando i *nodi equispaziati*.

1.1 Algoritmo 1: DLP (massimizzazione della produttoria)

Fissato $z_0 = x(1)$ come da specifica del progetto, il successivo nodo massimizza la produttoria delle distanze:

$$z_s = \arg\max_{x \in X_M} \prod_{i=0}^{s-1} |x - z_i|, \qquad s = 1, \ldots, d.$$

L'implementazione si basa su un vettore (inizializzato con tutti uno) che memorizza, per ogni punto della mesh, il prodotto cumulativo delle distanze dai nodi di Leja già selezionati. In seguito verrà aggiornato moltiplicando elemento per elemento con il vettore contenente le distanze (in modulo) dall'ultimo nodo scelto a tutti i punti di x. Questa è un'operazione vettoriale efficiente in MATLAB, che aggiorna tutti gli elementi simultaneamente senza bisogno di loop interni. Senza vettorializzazione, si dovrebbe calcolare il prodotto per ogni punto individualmente avendo così una complessità maggiore.

1.2 Algoritmo 2: DLP2 (LU su *Vandermonde–Chebyshev*)

In ambito di interpolazione di dati, concetto fondamentale è la *Matrice di Vandermonde V*, tale per cui $V * c = y$. Il condizionamento del polinomio interpolante λ_d dipende esclusivamente dai nodi, mentre per $V(i, j) = \phi_j(x_i)$ base di P^d, il condizionamento di V dipende sia dai nodi che dalla base ϕ_j.
Per $d \gg 1$ (già a partire da 10/12) la matrice di Vandermonde diventa malcondizionata, ma è possibile contenere il condizionamento di V utilizzando la base dei *polinomi di Chebyshev* per cui, preso $\{z_i\}_{i=0}^{d}$ l'insieme dei nodi, l'interpolante costruito su base di **Chebyshev** di primo tipo è:

$$V_{ij} = \cos\big((j-1)\arccos(z_i)\big), \qquad i = 1, \ldots, d{+}1, \ j = 1, \ldots, d{+}1,$$

i cui coefficienti si ottengono risolvendo il sistema lineare $V * c = f(z)$ con l'operatore \ di MATLAB, senza usare `polyfit`.

Si costruisce la matrice di Vandermonde in base di Chebyshev su tutta la mesh e si applica la fattorizzazione LU con pivoting sulle righe forzando $z_0 = x(1)$ come richiesto dalla specifica, successivamente l'ordinamento dei pivot fornisce i primi $d{+}1$ punti.

1.3 Costante di Lebesgue

Nell'analisi matematica, la costante di Lebesgue dà un'idea di quanto sia buono l'interpolante di una funzione (in base ai nodi considerati) rispetto alla miglior approssimazione polinomiale della funzione.
Il grado d dei polinomi viene fissato preventivamente e la costante di Lebesgue per polinomi di grado al massimo d e insieme di almeno $d + 1$ nodi T, è generalmente indicata con $\lambda_d(T)$ e tende a $+\infty$ se $d \to +\infty$.
La funzione che associa $x \to \sum_{i=0}^{d} |\ell_i(x)|$ prende il nome di funzione di Lebesgue, da cui si ricava la costante di Lebesgue :

$$\lambda_d(x) = MAX_{x \in [a,b]} \sum_{i=0}^{d} |\ell_i(x)|,$$

con $[a, b] = [-1, 1]$ nel nostro caso ed ℓ_i sono i polinomi di Lagrange.
La quantità λ_d misura dunque l'amplificazione dell'errore dei dati nell'interpolazione e un valore contenuto implica stabilità numerica.
La funzione MATLAB per il calcolo della costante è `leb_con`, la quale utilizza la formula *baricentrica* usando al massimo un ciclo (per i pesi), mentre il resto è vettorializzato.

2 Implementazione (MATLAB)

File principali

- `DLP.m` — Estrazione nodi di Leja per massimizzazione della produttoria.
  `[INPUT]` : DLP(x,d) con x vettore dei punti della mesh in [-1,1] e d grado del polinomio
  `[OUTPUT]` : vettore riga con i d+1 punti di Leja approssimati

- `DLP2.m` — Estrazione nodi via LU su Vandermonde–Chebyshev
  `[INPUT]` e `[OUTPUT]` come sopra

- `leb_con.m` — Calcolo della costante di Lebesgue su griglia di valutazione.
  `[INPUT]` : leb_con(z, x) con z vettore riga dei nodi e x vettore colonna dei punti in [-1,1]
  `[OUTPUT]` : scalare reale contenente il massimo della funzione di Lebesgue su tutti i punti x

- `main.m` — Sperimentazione automatica: tempi, costanti di Lebesgue e accuratezze.

3 Sperimentazione

Setup

Inizializzazione della mesh per l'estrazione dei nodi di Leja con M $= 10^5$ sull'intervallo $[-1, 1]$ e gradi $d = 1, \ldots, 50$.
Infine per ciascun n dell'algoritmo più efficiente, si testa l'interpolante costruita su tali nodi con la funzione ben definita su $[-1, 1]$:

$$f(x) = \frac{1}{x - 1.3},$$

3.1 Tempi computazionali

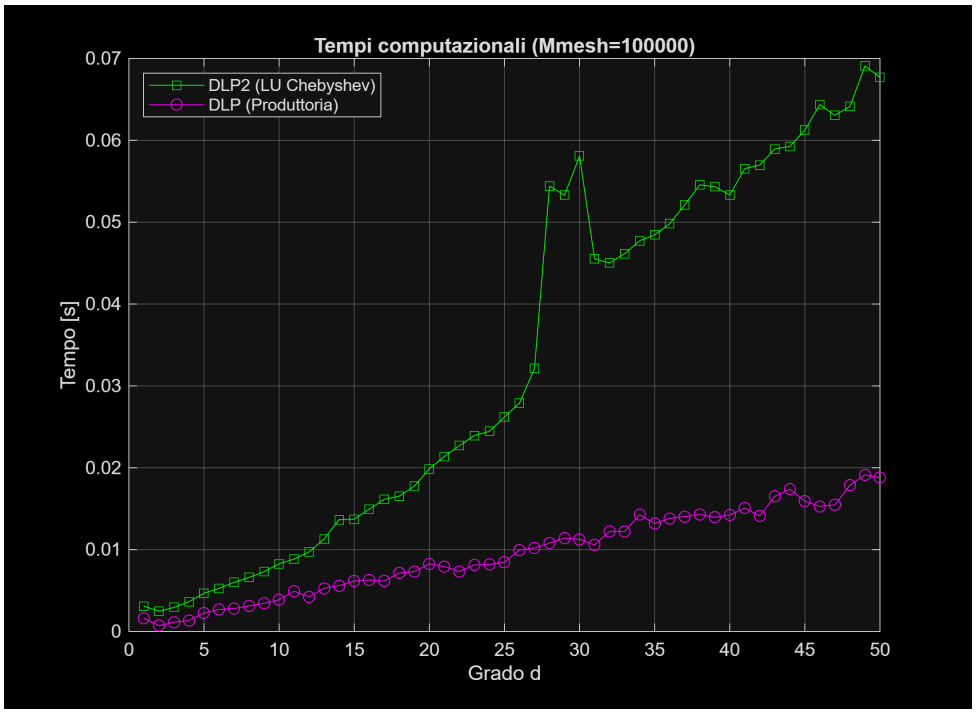


Figura 1: Tempi per l'estrazione dei nodi di Leja: DLP (produttoria) vs DLP2 (LU–Chebyshev)

A parità di condizioni, si nota come l'algoritmo DLP sia più veloce rispetto a $DLP2$. Questa differenza è dovuta all'operazione di fattorizzazione LU con pivoting della matrice di Vandermonde di grado d utilizzante i polinomi di Chebyshev richiesta nell'algoritmo prima dell'esecuzione.

3.2 Costante di Lebesgue

La costante di Lebesgue λ_d è stimata con la quantità $MAX_{x_k \in M} \lambda_d(x_k)$ su una griglia densa M di 5000 punti equispaziati nell'intervallo $[-1, 1]$.
Presi dei nodi "buoni" (Chebyshev, Leja) la crescita attesa è $O(\log d)$ e ci si aspetta sia più lenta rispetto ai nodi equispaziati, per i quali la costante cresce rapidamente (quasi esponenziale) causando il fenomeno di Runge.

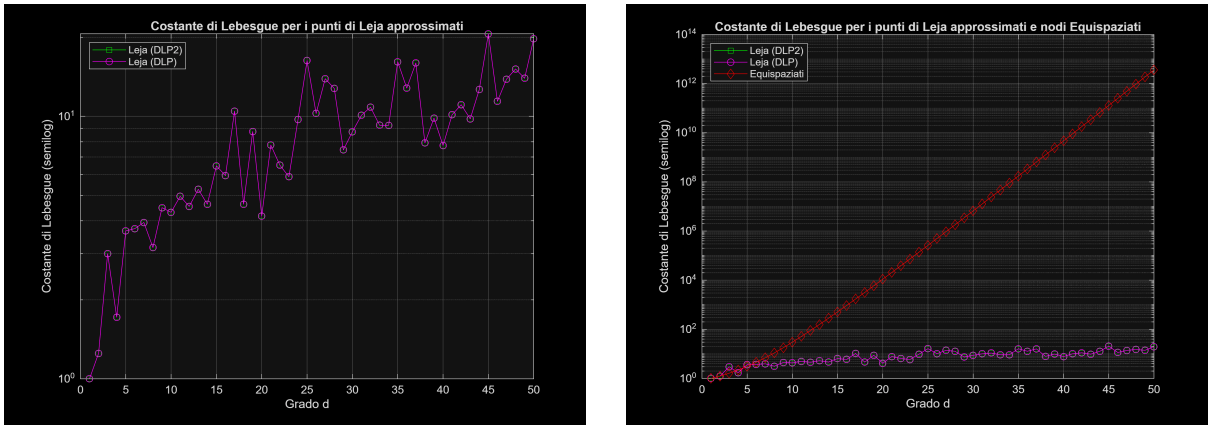


Figura 2: Costante di Lebesgue (scala semilog): Confronto tra DLP, DLP2 e nodi equispaziati

Analisi dei grafici

- gli algoritmi DLP e DLP2 producono la stessa sequenza di nodi e di conseguenza anche l'andamento della costante di Lebesgue coincide;
- la curva relativa al valore della costante di Lebesgue per DLP (e di conseguenza anche DLP2) cresce lentamente e conferma la previsione di un andamento pseudo-logaritmico stabilizzandosi per gradi $d \geq 40$. Per questo motivo, si conferma che la scelta dei nodi di Leja portano a un buon condizionamento;
- il grafico relativo all'andamento della costante di Lebesgue per l'interpolazione prodotta con nodi equispaziati cresce in maniera lineare, ma essendo in scala semilogaritmica, la costante di Lebesgue ha una crescita esponenziale e dunque la funzione è malcondizionata;
- il confronto tra gli algoritmi, mostra come le interpolazioni con nodi di Leja siano soluzioni decisamente migliori rispetto all'interpolazione ottenuta con nodi equispaziati.

In conclusione, il valore relativamente contenuto di Λ_d implica che l'errore dell'interpolante sui nodi di Leja è dominato dall'errore di approssimazione della funzione e non dall'amplificazione numerica.

3.3 Accuratezza dell'interpolante

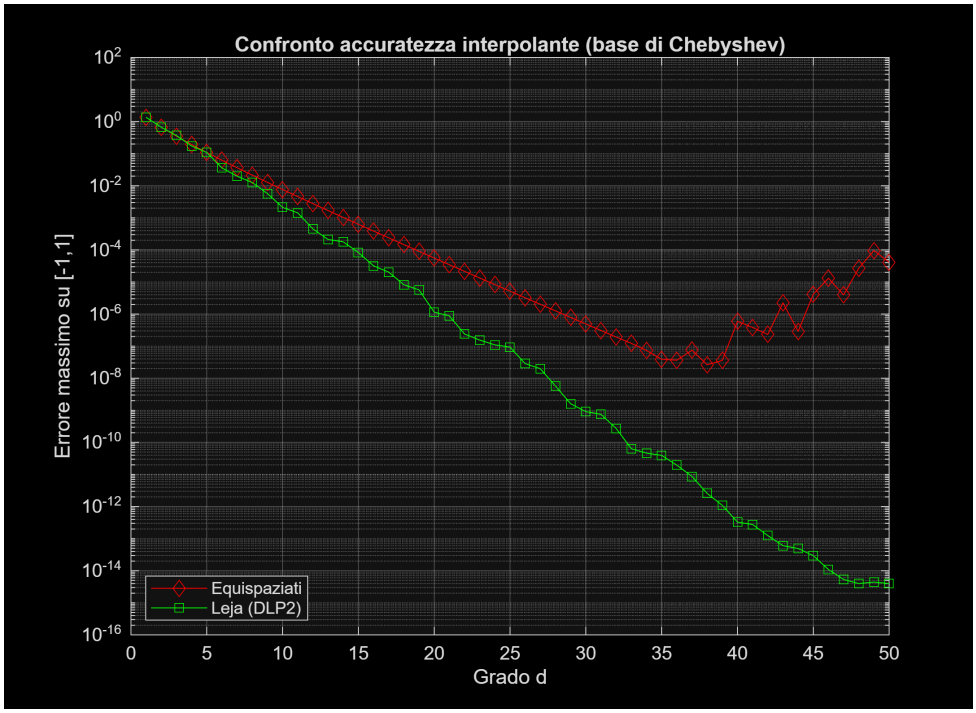


Figura 3: Massimo errore dell'interpolante su $[-1, 1]$: Leja vs nodi equispaziati

Considerazione principale che si ricava dal grafico è che l'errore massimo dell'interpolante ottenuto dai nodi di Leja è stabilmente inferiore all'errore dell'interpolante derivante da nodi equispaziati e quindi gode di una migliore accuratezza.
Inoltre, al crescere di d, l'errore descresce in maniera lineare per l'interpolazione con nodi di Leja, mentre per nodi equispaziati, con grado $d \approx 35 - 40$, l'errore si stabilizza e per grado $d > 40$, l'errore torna ad aumentare.

4 Conclusioni

Riassumendo i risultati ottenuti, possiamo affermare che:

- i due algoritmi (`DLP`, `DLP2`) estraggono la stessa sequenza di nodi di Leja (con $z_0 = x(1)$), come confermato dall'identità delle costanti di Lebesgue;
- l'algoritmo `DLP` è più veloce per i gradi considerati, mentre `DLP2` risulta leggermente più lento, ma guadagna in stabilità per gradi d alti (≥ 100);
- la costante di Lebesgue sui nodi di Leja cresce con andamento pseudo-logaritmico, indicando buona stabilità dell'interpolazione;
- l'interpolazione sui nodi di Leja è nettamente più accurata rispetto a quella sui nodi equispaziati, con l'errore che decresce fino al limite macchina.

5 Altro

Informazioni utili riguardo al codice

Per riprodurre i grafici, eseguire `main.m` dopo aver messo i file `DLP.m`, `DLP2.m`, `leb_con.m` nella stessa cartella. Per salvare automaticamente i grafici in `doc/img/`, rimuovere i commenti alle righe 67–69, 78, 87, 108, 110 del `main.m`. Se si vuole riprodurre e salvare anche il grafico del confronto tra DLP, DLP2 e i nodi equispaziati, togliere i commenti alle righe 28, 34, 89–99.