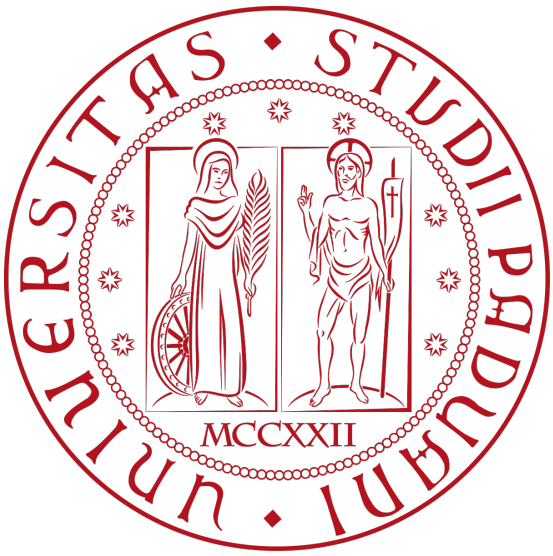
Università degli Studi di Padova



Relazione per:

INTERPOLAZIONE POLINOMIALE CON NODI DI LEJA APPROSSIMATI

Realizzata da:
Oliver Florin Stiglet, matricola 2044895

Corso: Programmazione/Calcolo Numerico (MATLAB)

Data: August 17, 2025

1 Introduzione

Obiettivo del progetto: implementare in MATLAB l'interpolazione polinomiale su $[-1, 1]$ utilizzando *punti di Leja approssimati* estratti da una mesh fitta, calcolare la *costante di Lebesgue* per valutarne la stabilità, e confrontare l'accuratezza dell'interpolante con quella ottenuta usando *nodi equispaziati*.

Sia $\{z_i\}_{i=0}^{d}$ l'insieme dei nodi. L'interpolante è costruito su base di **Chebyshev** di primo tipo:

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

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

2 Algoritmi implementati

2.1 Algoritmo 1: DLP (produttoria)

A partire dal primo nodo $z_0 = x(1)$ (primo elemento della mesh), il nodo successivo 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 sfrutta aggiornamenti vettoriali della produttoria per ridurre i costi.

2.2 Algoritmo 2: DLP2 (LU su Vandermonde–Chebyshev)

Si costruisce la matrice di Vandermonde in base di Chebyshev su tutta la mesh e si applica la fattorizzazione LU con pivoting sulle righe; l'ordinamento dei pivot fornisce i primi $d+1$ punti. Si forza $z_0 = x(1)$ per rispettare la specifica.

2.3 Costante di Lebesgue

La funzione di Lebesgue è

$$\lambda_d(x) = \sum_{i=0}^{d} |\ell_i(x)|\,,$$

dove ℓ_i sono i polinomi di Lagrange. La costante è $\Lambda_d = \max_{x\in[-1,1]} \lambda_d(x)$. La funzione `leb_con` la calcola con formula *baricentrica* usando al massimo un ciclo (per i pesi), il resto è vettorializzato.

3 Implementazione (MATLAB)

File principali

- `DLP.m` — Estrazione nodi di Leja per massimizzazione della produttoria.
- `DLP2.m` — Estrazione nodi via LU su Vandermonde–Chebyshev (con $z_0 = x(1)$ forzato).
- `leb_con.m` — Calcolo della costante di Lebesgue su griglia di valutazione.
- `main.m` — Sperimentazione automatica: tempi, costanti di Lebesgue, accuracies.

Estratti di codice

Listing 1: Risoluzione dei coefficienti in base di Chebyshev

```
% z: nodi (Leja o equispaziati), f: funzione campionata in z
j = 0:d;
V  = cos(acos(z) * j);  % Vandermonde-Chebyshev
c  = V \ f(z);          % coeff. senza polyfit, risoluzione del sistema
```

Listing 2: Costante di Lebesgue (schema baricentrico)

```
% w_i = 1 / prod_{j!=i} (z_i - z_j)
% ell_i(x) = (w_i/(x - z_i)) / sum_j (w_j/(x - z_j))
% lambda(x) = sum_i |ell_i(x)|
```

4 Sperimentazione

Setup. Mesh per l'estrazione dei Leja: $M = 10^4$ (poi ripetuto con 10^5 per i grafici finali); gradi $d = 1, \ldots, 50$; griglia di valutazione densa su $[-1, 1]$. Funzione test:

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

ben definita su $[-1, 1]$. I tempi sono misurati con `tic/toc` per entrambi gli algoritmi.

4.1 Tempi computazionali

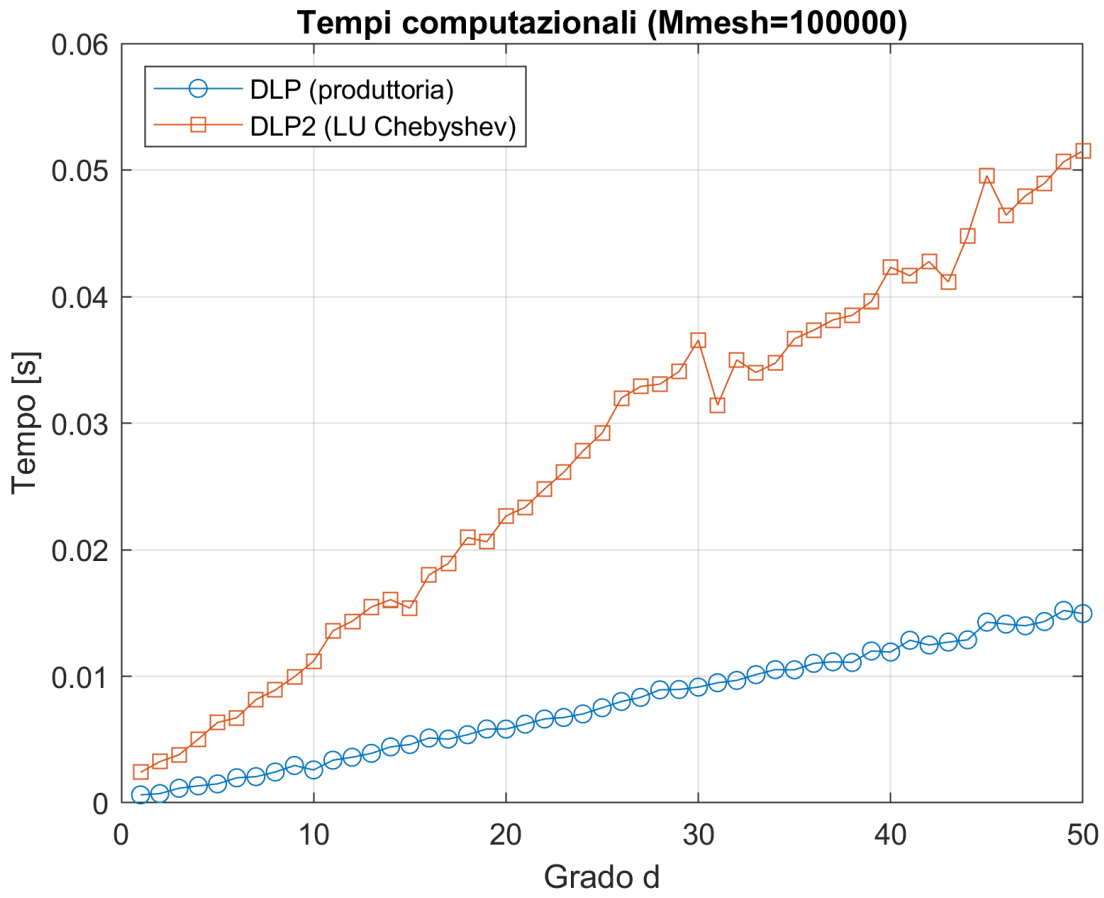


Figure 1: Tempi per l'estrazione dei nodi di Leja: DLP (produttoria) vs DLP2 (LU–Chebyshev).

4.2 Costante di Lebesgue

La costante di Lebesgue Λ_d misura l'amplificazione dell'errore dei dati nell'interpolazione: un valore moderato implica stabilità numerica. Per nodi "buoni" (Chebyshev, Leja) la crescita attesa è $O(\log d)$ o comunque molto più lenta rispetto ai nodi equispaziati, per i quali la costante cresce rapidamente (quasi esponenziale) causando il fenomeno di Runge.

Nel nostro esperimento Λ_d è stimata come $\max_{x_k \in \mathcal{X}} \lambda_d(x_k)$ su una griglia densa $\mathcal{X}$ di 5000 (o 10000 nelle repliche finali) punti equispaziati in $[-1, 1]$. Questa è un'approssimazione per difetto della vera costante continua, ma sufficiente a evidenziare l'andamento.

Osservazioni dai grafici:

- l'andamento cresce lentamente e non mostra esplosioni: conferma che i nodi di Leja selezionati mantengono un buon condizionamento;
- le due implementazioni (DLP e DLP2) producono la stessa sequenza di nodi, quindi la curva coincide (differenze numeriche al di sotto di 10^{-13});
- per gradi oltre 40 la crescita si appiattisce leggermente a causa della discretizzazione della griglia e dell'arrotondamento floating point (doppia precisione);
- il confronto (non riportato in figura) con nodi equispaziati mostrerebbe valori decisamente maggiori già per gradi medi, giustificando i maggiori errori interpolatori visti nella Sez. 4.3.

Dal punto di vista pratico, 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.

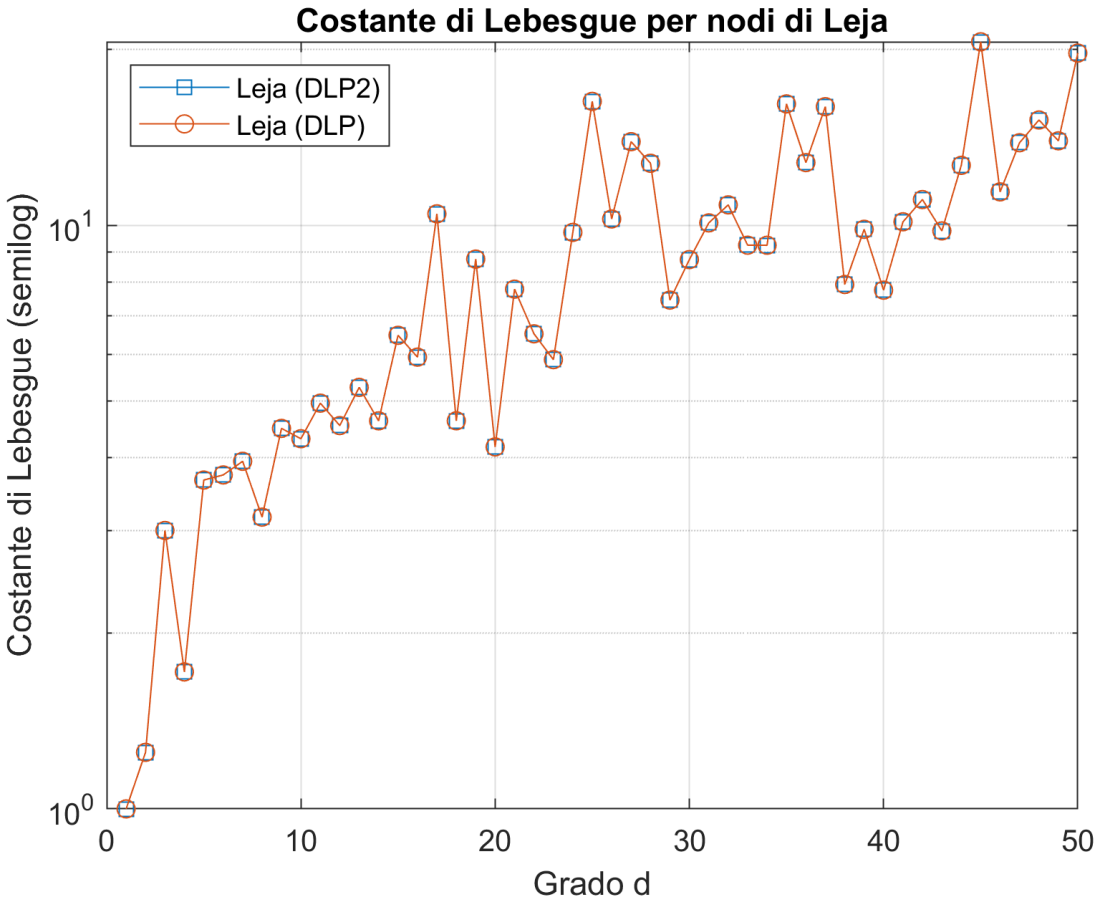


Figure 2: Costante di Lebesgue (scala semilog): i due metodi producono gli stessi nodi di Leja.

4.3 Accuratezza dell'interpolante

Interpolante costruita in base di Chebyshev con i nodi Leja (dall'algoritmo più efficiente) e con nodi equispaziati; confronto dell'errore massimo su griglia densa.

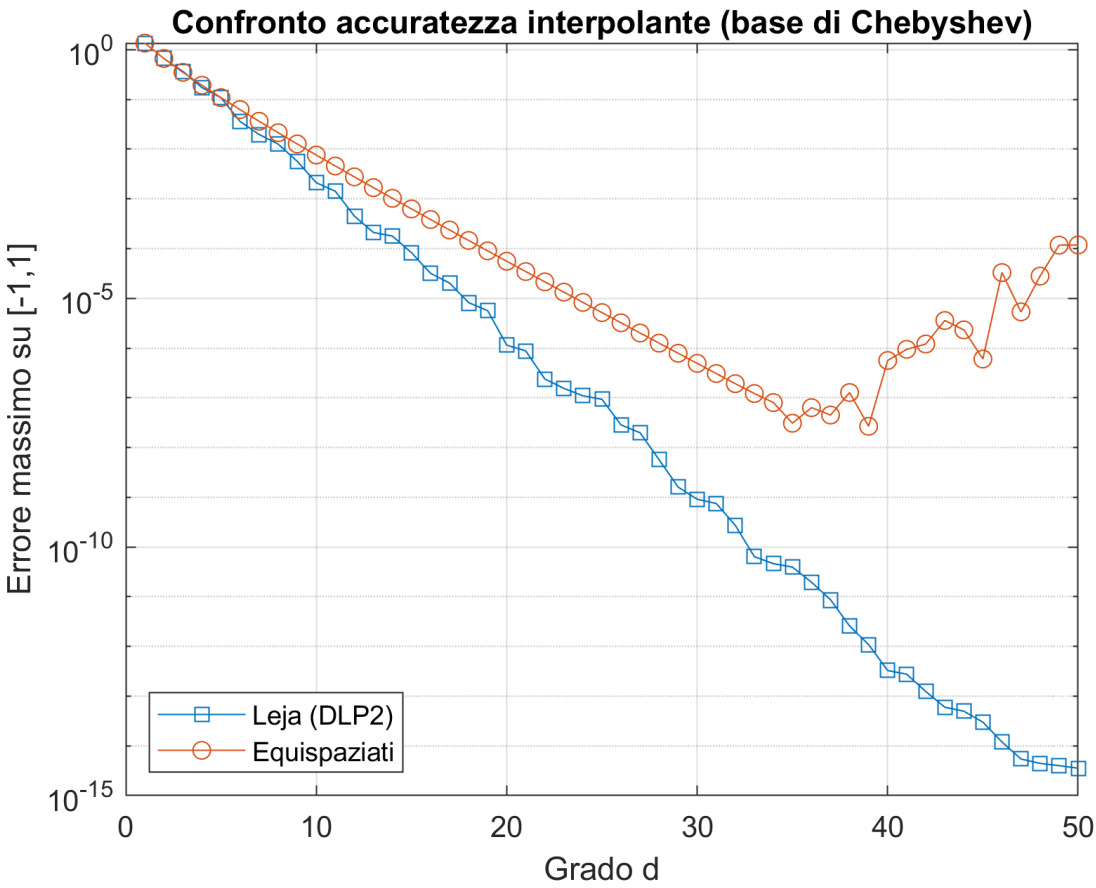


Figure 3: Errore massimo dell'interpolante su $[-1, 1]$: Leja vs equispaziati.

5 Conclusioni

I risultati mostrano 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 comunque competitivo e più strutturato;
- la costante di Lebesgue sui nodi di Leja cresce moderatamente, indicando buona stabilità dell'interpolazione;
- l'interpolante su nodi di Leja è nettamente più accurata rispetto a quella su nodi equispaziati, con errori che decrescono fino al limite di macchina.

Appendice: riproducibilità

Per riprodurre i grafici, eseguire `main.m` dopo aver posto i sorgenti `DLP.m`, `DLP2.m`, `leb_con.m` nella stessa cartella. Le figure sono salvate automaticamente in `doc/img/` (o spostate manualmente prima di compilare).