Interpolazione Polinomiale

Guglielmo Bartelloni

30 marzo 2022

23 Marzo 2022

Noi sappiamo che:

$$||e|| \leq a(1 + \Delta_n)w(f_i; \frac{b-a}{n}).$$

Supponiamo di scegliere una partizione dell'intervallo [a,b]:

$$\Delta = a = x_0 < x_1 < ... < x_n = b.$$

Quindi consideriamo n sottointervalli, $[x_{i-1}, x_i]$ $i = 1, ..., n$ tale che, detta: $h = max(x_i, x_{i-1}) \to 0 \quad n \to \infty$ e supponiamo che in ciascuno sottointervallo la f(x) sia interpolata da un polinomio di grado n **fissato** (quindi interpolo ogni sottoinrevallo).

Ottengo quindi una funzione **polinomiale a tratti**, chiamiamola s_m, tale che, detto $e(x) = s_m(x) - f(x)$ si avra:

$$||e|| \leq a(1 + \Delta_m)w(f; h) \to 0 \; quando \; h \to 0 \implies n \to \infty.$$

Definizione 1. Assegnato la partizione $\Delta = a = x_0 < x_1 < ... < x_n = b$ diremo che s_m e' una **spline di grado n** sulla partizione Δ se:

1. $sm_{|_{[x_{i-1}, x_i]}}(x) \in \Pi_m$
2. $S_m \in C^{m-1}[a, b] \to$ (derivabile m-1 volte e continua)

Osservazione 1. Il requisito 2) significa richiedere che (la derivata j-esima):

$$sm^j_{|_{[x_{i-1}, x_i]}}(x_i) = sm^j_{|_{[x_i, x_{i+1}]}}(x_i).$$

con $i = 1, ..., n-1$ e $j = 0, ..., m-1$

Quindi nel punto x_i le derivate devono essere uguali

Osservazione 2.
1. Se s_m fosse un polinomio allora sarebbe una spline

2. Se α,$\beta \in R$, s_m $g_m \in S_m(\Delta)$: su Δ

$$\alpha S_m(x) + B g_m(x) \in S_m(\Delta).$$

Sono anch'essi spline e pertanto $S_m(\Delta)$ e' **uno spazio vettoriale**

Teorema 0.1. $dim(S_m(\Delta) = m + n)$, pertanto sono necessarie n+m condizioni indipendenti per individuare univocamente una spline interpolante di grado m su una partizione Δ

Definizione 2. Una spline di grado m sulla partizione Δ $S_m(x)$, si dice **interpolante una funzione** $f(x)$ se:

$$s_m(x_i) = f_i = f(x_i) \quad i = 0, ..., n.$$

Osservazione 3. Le condizioni di interpolazione sono n+1, pertanto esse permettono di individuare univocamente solo una spline di grado 1 o lineare (n+m dove m=1) ovvero la spezzata congungente i punti (x_i, f_i) $i = 0, ..., n$. La sua espressione e' data da:

$$s1_{|_{[x_{i-1},x_i]}}(x) = \frac{(x_i - x_{i-1})f_i + (x_i - x)f_{i-1}}{x_i - x_{i-1}} \quad per \quad i = 1, ..., n.$$

1 Spline Cubiche

$s_3(x)$ sara tale che:

$$s3_{|_{[x_{i-1},x_i]}}(x) \leq \Pi_3, \ s_3(x) \in C^2[a, b].$$

Per individuare in maniera univoca si ha bisogno di **n+3** condizioni.

Le condizioni di interpolazione sono n+1 quindi ci mancano due condizioni. Vediamo quindi alcuni modi di interpolare: ciascuno di essi dara origine ad una spline cubica interpolante **diversa**

1.1 Spline Cubica Naturale

$s_3''(a) = 0$ e $s_3''(b) = 0$

1.2 Spline Cubica Completa

$s_3'(a) = f'(a)$ e $s_3'(b) = f'(b)$

1.3 Spline Cubica Periodica

Si applica nel caso in cui $f(x)$ sia periodica e l'intervallo [a,b] contenga m numero intero di periodi (es.1) imponendo le condizioni. $s_3'(a) = s_3'(b)$ e $s_3''(a) = s_3''(b)$

1.4 Spline not-a-knot (implementato da matlab spline())

In questo caso si richiede che le restrizioni di $s_3(x)$ sugli intervalli $[x_0, x_1]$ e $[x_1, x_2]$ siano lo **stesso polinomio**, si richiede che le restrizioni di $s_3(x)$ sugli intervalli $[x_{n-2}, x_{n-1}]$ e $[x_{n-1}, x_n]$ siano lo stesso polinomio.

Ragioniamo su $[x_0, x_1]$ e $[x_1, x_2]$. Supponiamo che $s_3(x) \in C^2[a, b]$, pertanto:

$$s3^j_{|_{[x_0,x_1]}}(x_1) = s3^j_{|_{[x_1,x_2]}}(x_1) \quad j = 0, 1, 2.$$

se imponiamo che la $s3^j_{|_{[x_0,x_1]}}(x_1) = s3^j_{|_{[x_1,x_2]}}(x_2)$ allora i **due polinomi coincideranno** (perche 4 ingonite).

Questo imopone che

$$\frac{s_3^2(x_1) - s_3^2(x_0)}{x_1 - x_0} = \frac{s_3^2(x_2) - s_3^2(x_1)}{x_2 - x_1}.$$

Simmetricamente si richiedera per gli ultrimi 2 sottointervalli che:

$$\frac{s_3^2(x_{n-1}) - s_3^2(x_{n-2})}{x_{n-1} - x_{n-2}} = \frac{s_3^2(x_n) - s_3^2(x_{n-1})}{x_n - x_{n-1}}.$$

24 Marzo 2022

Calcolo Numerico

2 Calcolo Spline Cubiche naturali

Oggi vediamo come si calcano le spline cubiche naturali. Le spline cubiche non hanno il problema del malcondizionamento crescente, al crescere di n, e si puo dimostrare che, se $f \in C^{(4)}[a, b]$:

$$||f^{(j)} - s_3^{(j)}|| = O(h^{4-j}).$$

Per $j = 0, 1, 2$

Questo vale per tutte le spline cubiche viste, tranne quella naturaleche ha un piccolo errore agli estremi dell'intervallo (per le condizioni imposte), tuttavia, si smezza esponenzialmente agli estremi dell'intervallo. Pertanto abbiamo necessita di definire un algoritmo efficiente per il calcolo di $s_3(x)$. Esamineremo in dettaglio il calcolo di una spline cubica naturale anche se gli argomenti si possono generalizzare alle spline cubiche di altro tipo (sul libro not-a-knot).

Preliminarmente diamo il seguente risultato:

Teorema 2.1. $\forall m \geq 2$: se s_m e' una spline di grado m su Δ, allora s_m e' una spline di grado m-1 su Δ

Dimostrazione. Immediata dalla definizione di spline di grado m su Δ. □

3 Calcolo di una spline cubica naturale

($s_3''(a) = s_3''(x_0) = 0$, $s_3''(b) = s_3''(x_n) = 0$)

Se $s_3(x)$ e' una spline cubica su Δ allora la $s_3'(x)$ e' una spline quadratica su Δ (per il teorema sopra) e, quindi, $s_3''(x)$ e' una spline lineare su Δ.

$$s_3(x)\ cubica \rightarrow s_3'(x)\ quadratica \rightarrow s_3''(x)\ lineare.$$

Supponiamo di conoscere i valori di $f_i = f(x_i)$ per i=0,..,n della funzione interpolante.

Poiche $s_3''(x)$ e' una spline lineare, detta $m_i = s_3''(x)$ per i=0,...,n si avra:

$$f_{|_{[x_{i-1},x_i]}}(x) = \frac{(x - x_{i-1})^2 m_i - (x_i - x)^2 m_{i-1}}{2h_i} + q_i \ i = 1, ..., n.$$

(2)

in cui q_i e' una costante di integrazione

Integrando ancora la (2), otteniamo:

$$s3_{|_{[x_{i-1},x_i]}}(x) = \frac{(x - x_{i-1})^3 m_i + (x_i - x)^3 m_{i-1}}{6h_i} + q_i(x - x_{i-1}) + r_i \ i = 1, ..., n.$$

Dove r_i e' una nuova costante di integrazione. Imponendo le condizioni di interpolazione in x_{i-1} e x_i, si ottiene:

$$s3_{|_{[x_{i-1},x_i]}}(x_{i-1}) = \frac{h_i^2}{6} m_{i-1} + r_i = f_{i-1}.$$

Faccio la formula inversa

$$r_i = f_{i-1} - \frac{h_{i-1}^2}{6} m_{i-1}.$$

$$s3_{|_{[x_{i-1},x_i]}}(x_i) = \frac{h_i^2}{6} m_i + q_i h_i + r_i = f_i.$$

$$h_i q_i = f_i - f_{i-1} + \frac{h_i^2}{6} m_i.$$

$$h_i q_i = f_i - f_{i-1} + \frac{h_i^2}{6} m_{i-1} - \frac{h_i^2}{6} m_i.$$

$$q_i = \frac{f_i - f_{i-1}}{h_i} - \frac{h_i}{6}(m_i - m_{i-1}).$$

La prima frazione del secondo membro e' una differenza divisisa in particolare $f[x_{i-1}, x_i]$

Pertanto potremo calcolare la spline cubica in ogni sottointervallo a patto di calcolare m_i per $i = 1, ..., n-1$ ($m_0 = m_n = 0$). Per calcolare questi valori essendo $s_3'(x)$ una spline qudratica su Δ, imponendo che:

$$s3'_{|_{[x_{i-1},x_i]}}(x_i) = s3'_{|_{[x_i,x_{i+1}]}}(x_i) \; per \; i = 1, ..., n-1.$$

Quindi abbiamo un sistema di n-1 equazioni in n-1 incognite, esplicitiamolo:

$$\frac{h_i}{2}m_i + q_i = -.\frac{h_{i+1}}{2}m_i + q_{i+1}.$$

Raccolgo il tutto

$$\frac{h_i + h_{i+1}}{2}m_i = q_{i+1} - q_i.$$

$$= f[x_i, x_{i+1}] - f[x_{i-1}, x_i] - \frac{h_{i+1}}{6}(m_{i+1} - m_i) + \frac{h_i}{6}(m_i - m_{i+1}).$$

moltiplico per 6 e porto al primo membro tutto quello che non e' una differenza divisa

$$3(h_i + h_{i+1})m_i + h_{i+1}m_{i+1} + h_i m_{i-1} = 6(f[x_i, x_{i+1}] - f[x_{i-1}, x_i]).$$

Divido membro a mebro per $h_i - h_{i+1}$

$$\frac{h_i}{h_i + h_{i+1}}m_{i+1} + 2m_i + \frac{h_{i+1}}{h_i + h_{i+1}}m_{i+1} = 6\frac{f[x_i, x_{i+1}] - f[x_{i-1}, x_i]}{h_i + h_{i+1}}.$$

Ponendo

$$q_i = \frac{h_i}{h_i + h_{i+1}} \qquad \psi_i = \frac{h_{i+1}}{h_i + h_{i+1}}.$$

E osservando che

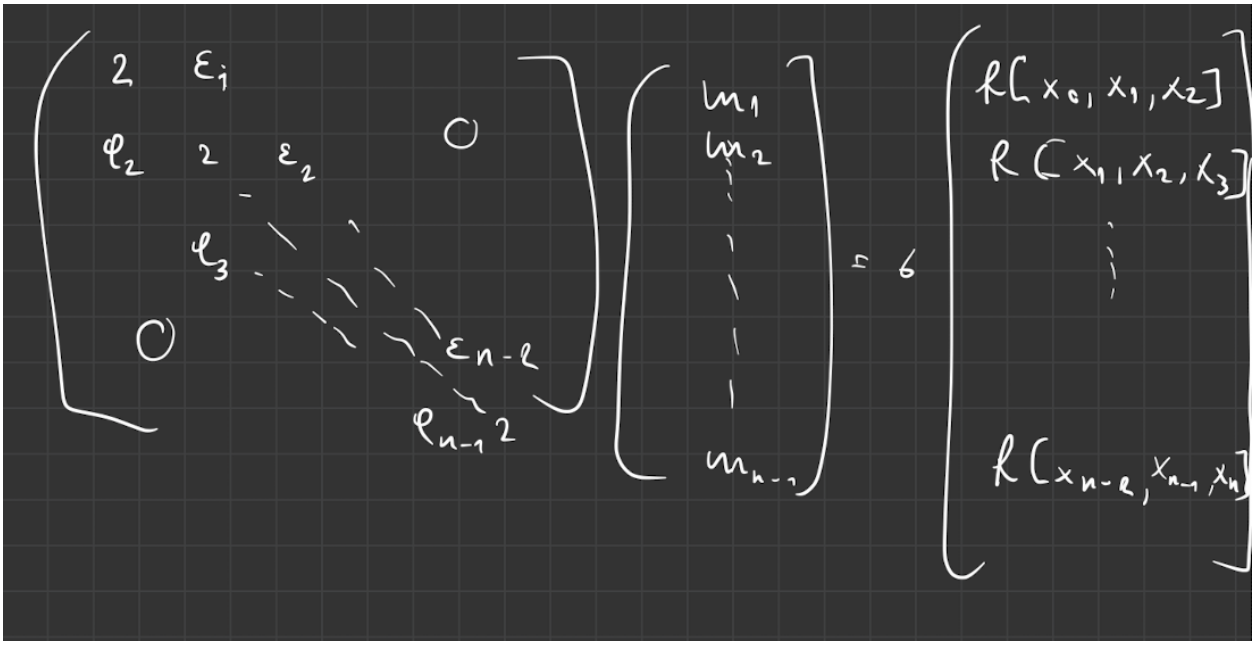
$$\frac{f[x_i, x_{i+1}] - f[x_{i-1}, x_i]}{h_i + h_{i+1}} = f[x_{i-1}, x_i, x_{i+1}].$$

dove $h_i + h_{i+1} = x_{i+1} - x_{i-1}$

Si ottiene che il sistema lineare tridiagonale

$$q_i m_{i-1} + 2m_i + \psi_i m_{i+1} = 6f[x_{i-1}, x_i, x_{i+1}] \quad i = 1, ..., n-1.$$

In forma vettoriale, si ottiene, ricodando che $m_0 = 0 \; e \; m_n = 0$:



Osserviamo che q_i e $\psi_i > 0$ per $\forall i = 1, ..., n$ Inoltre

$$q_i + \psi_i = 1.$$

Pertanto la matrice dei coefficienti e' diagonale dominante per righe. Pertanto la sua fattorizzazione LU e' sempre definita (inoltre si puo vedere essere sempre ben condizionata, indipendentemente dalla sua dimensione). Andiamo a vedre come si possa risolvere con complessita lineare.

Infatti, sono necessari, in tutto 4 vettori:

Uno che contiene in ingresso il termine noto e puo essere riscritto con la soluzione del sistema lineare. Tre che contengono in ingresso gli elementi sulle 3 diagonali dei fattori L e U che al pari della matrice, sono sparsi

30 Marzo 2022

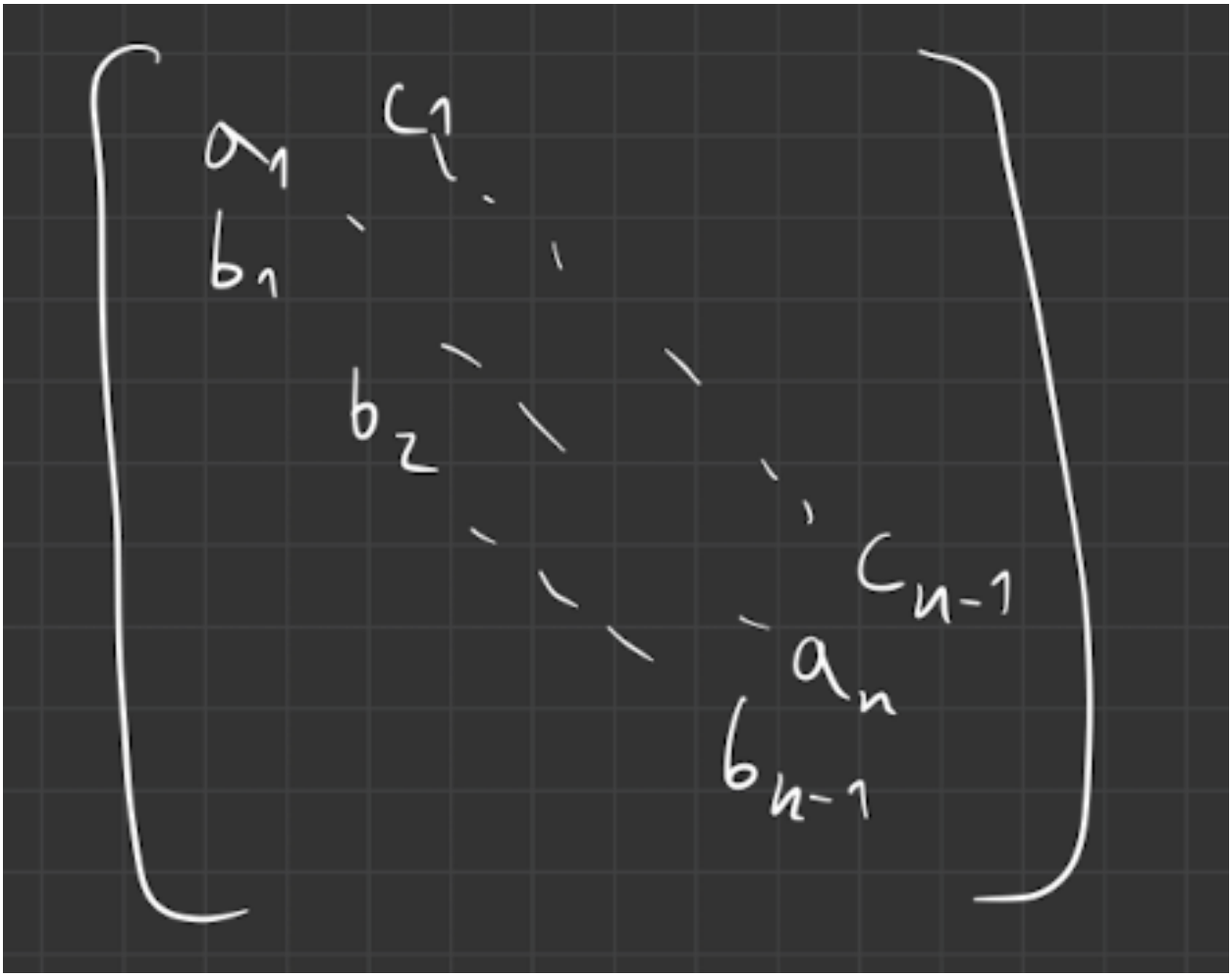
Calcolo Numerico

4 Risoluzione del sistema lineare della spline cubica naturale

Il sistema lineare della lezione precedente e' **sparso**, nel senso che molti degli elementi della matrice dei coefficienti sono null. (In generale per una matrice sparsa il numero di elementi non nulli e' **lineare** nella dimensione, invece che essere una frazione del quadrato).

Questa matrice si puo fattorizzare con un costo lineare.

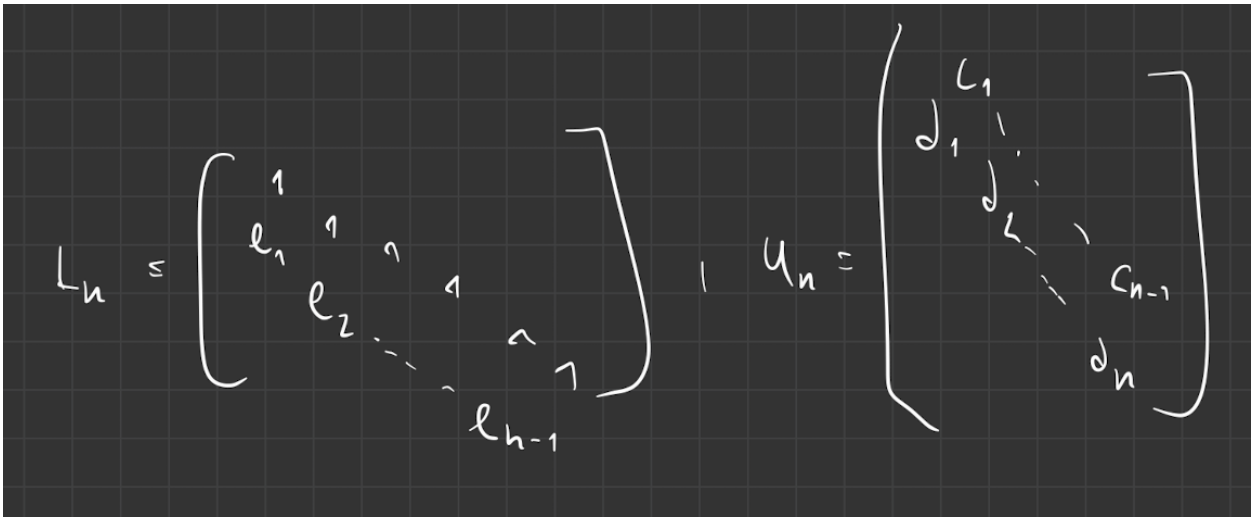
Motivati da quanto sopra, andiamo a definire un algoritmo di fattorizzazione LU efficiente per una **generica matrice tridiagonale**.



Osservazione 4. Pertanto, per memorizzare la matrice T_n saranno sufficienti 3 vettori di lunghezza n, Definiamo a, b e c, tali che:

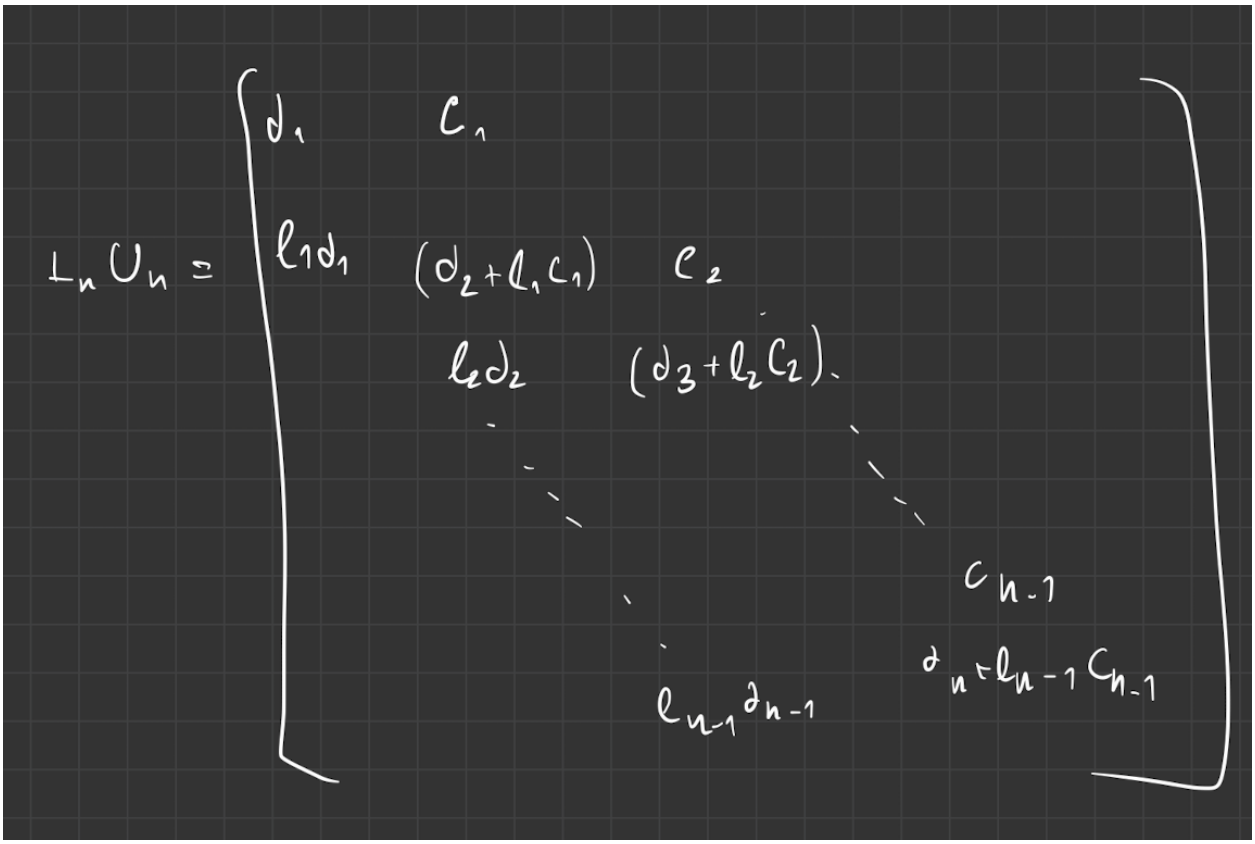
- a_i = i-esimo elemento della diagonale principlae,
- c_i = i-esimo elemento della sopradiagonale,
- b_i = i-esimo elemento della sottodiagonale

Verifichiamo che $T_n = L_n U_n$,



con $l_i = \frac{b_i}{d_i}$, $d_{i+1} = a_{i+1} - l_i c_i$ per $i = 1, ..., n-1$ e $d_1 = a_1$

Verifichiamo questo facendo il prodotto $L_n U_n$ ed uguagliando gli elementi omomloghi:



$d_1 = a_1$, $d_{i+1} + l_i c_i = a_{i+1}$, $i = 1, ..., n-1$ e $l_i d_i = b_i$

Pertanto:

$$d_1 = a_1.$$

$$l_i = \frac{b_i}{d_i}.$$
$$b_{i+1} = a_{i+1} - l_i c_i, \quad i = 1, ..., n-1.$$

Costo totale: $3(n-1)$ flops.

Se utilizziamo questa fattorizzazione per risolvere.

$T_n x = f = [f_1, ..., f_n]^T$

Allora dobbiamo risolvere $L_n y = f$ e, quindi $U_n x = y$. Nella pratica, lo stesso vettore che contiene il termine noto puo essere sovrascritto con i vettori y e x. (Possiamo sovrascrivere i tre vettori iniziali con i tre vettori di cui sopra)

$$L_n y = f \Leftrightarrow y_1 = f_1.$$

```
for i=1:n-1
```

$y_{i+1} = f_{i+1} - l_i y_i$

```
end
```

Questo da luogo ad 2(n-1) flops

$$U_n x = y \Leftrightarrow x_n = \frac{y_n}{d_n}.$$

```
for i=n-1:-1:1
```

$x_i = \frac{f_i - c_i x_{i+1}}{d_i}$

```
end
```

Questo costa 3(n-1) flops.

In tutto il costo e' di 8n flops quindi un costo lineare.

Su moodle c'e' un'implementazione efficiente dell'algoritmo di risoluzione di un sistema tridiagonale.