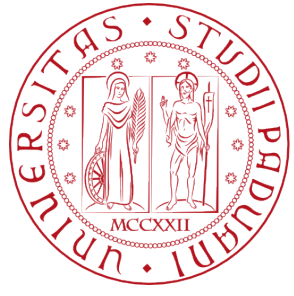
i

Implementazione in Python di un algoritmo per la cubatura numerica su elementi poligonali con un lato curvo

GIOVANNI TRAVERSIN

Laurea Triennale in Matematica

17/09/2025



UNIVERSITÀ
DEGLI STUDI
DI PADOVA

L'obiettivo

L'obiettivo di questo lavoro è fornire una fedele traduzione in Python di routines Matlab implementate in

Algebraic cubature on polygonal elements with a circular edge

da E. Artioli, A. Sommariva e M. Vianello.

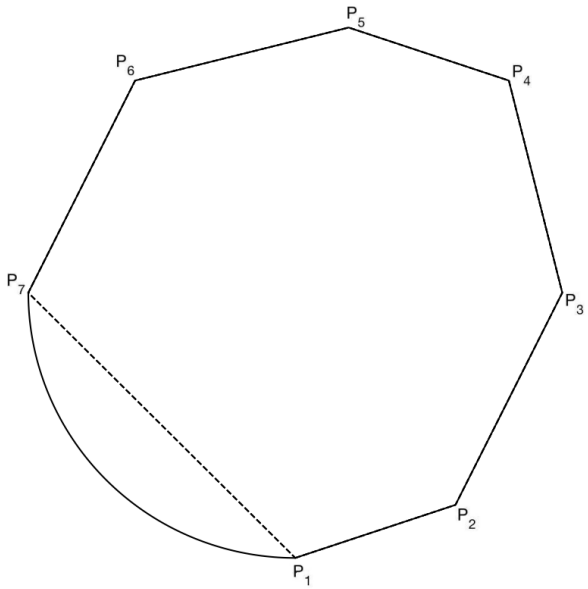
Tali codici calcolano una formula di quadratura algebrica per funzioni aventi come dominio un poligono circolare, sfruttando tecniche quali

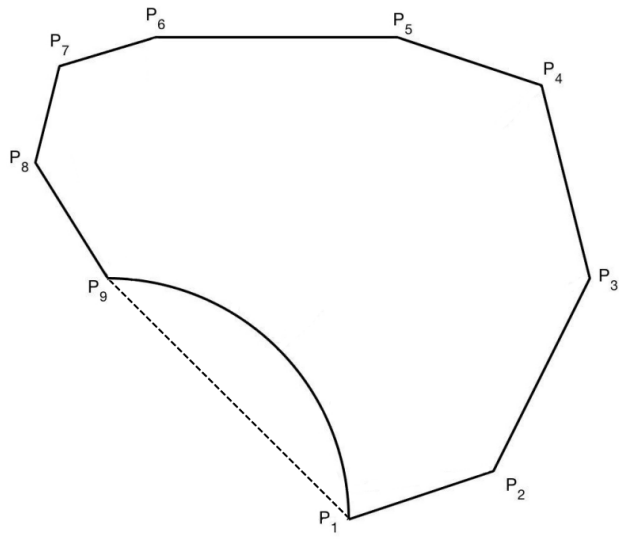
- il *linear blending di archi ellittici*,
- la *compressione della formula di quadratura*.

Nota: Queste formule hanno impiego nei *virtual elements* (risoluzione di certi problemi alle derivate parziali).

I domini considerati

Con *elemento poligonale con un lato curvo* considereremo un qualsiasi poligono convesso $\mathcal{P}$ a cui è stato sostituito uno dei lati con un arco di circonferenza. Tale arco può essere sia convesso che concavo.





Le formule di integrazione

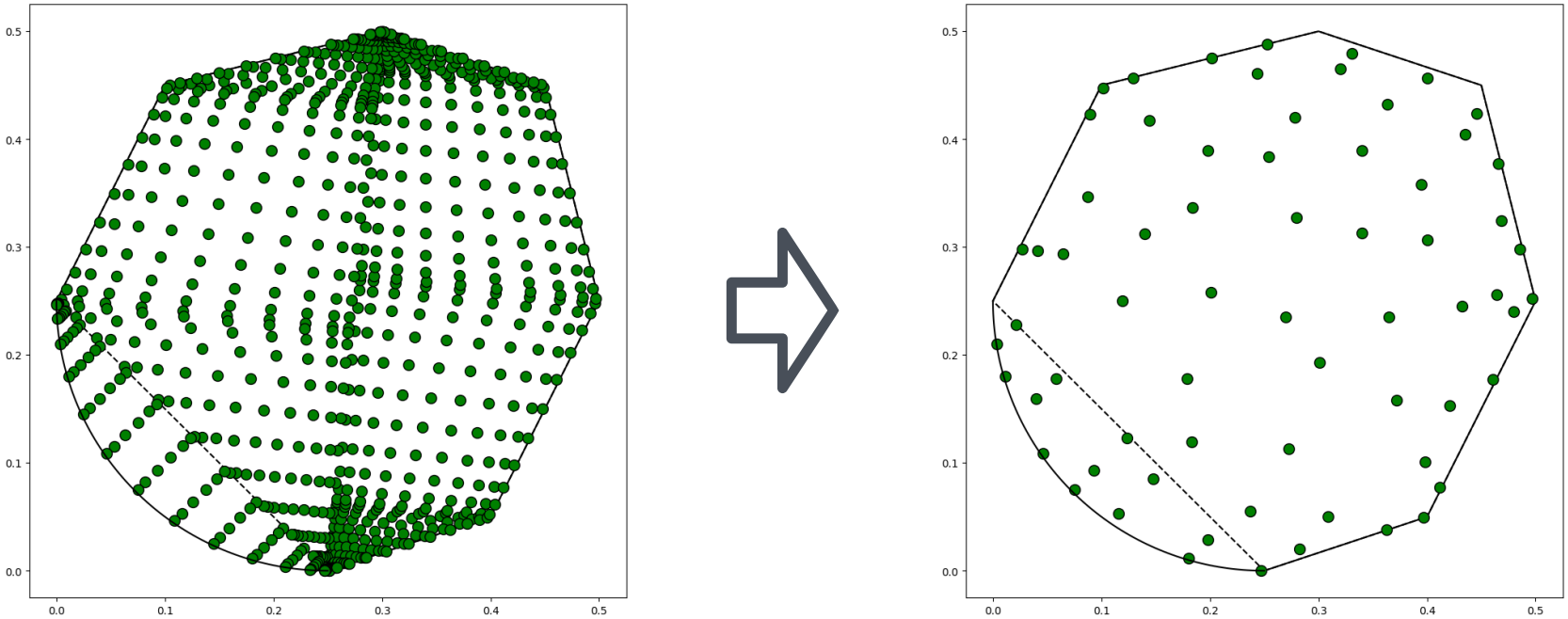
Nello specifico, i programmi costruiti restituiscono due formule di quadratura di tipo algebrico (*integrano esattamente polinomi algebrici fino ad un certo grado*), entrambe a nodi interni e pesi positivi.

- La prima ha grado di precisione n e cardinalità dell'insieme dei nodi pari a $M_n \geq \frac{(n+1)(n+2)}{2}$:

$$\int\int_{\Omega} p(x,y)\ dxdy = \sum_{i=1}^{M_n} \omega_i\ p(x_i,y_i),\ \forall\ p \in \mathbb{P}_n^2\ .$$

- La seconda ha nuovamente grado di precisione n, cardinalità $M_n^{(c)} = \frac{(n+1)(n+2)}{2}$ ed è una compressione della prima tramite l'algoritmo di Lawson-Hanson.

Compressione delle formule



Il fulcro della quadratura

Punto chiave: sfruttando la proprietà additiva dell'integrale, suddividiamo i poligoni circolari in sottodomini più semplici per definire progressivamente su ogni suddivisione una formula di quadratura del tipo richiesto.

Sui sottodomini disponiamo di formule di quadratura algebriche di grado prefissato avente nodi interni e pesi positivi e quindi, per l'additivitá dell'operatore di integrazione, abbiamo simili formule anche sull'elemento poligonale con un lato curvo.

Elementi poligonali con lato curvo convesso

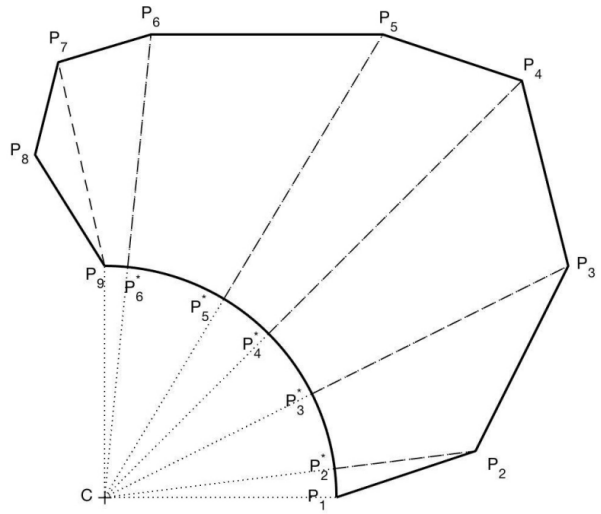
Per elementi poligonali con lato curvo convesso, quindi, consideriamo:

- **suddivisione**: un poligono e un settore circolare; il poligono viene quindi triangolato in modo minimale;
- **quadratura**: formule algebriche, a bassa cardinalità e del grado richiesto, su ogni dominio della suddivisione:
  1. **sul triangolo**: formule quasi minimali;
  2. **sul settore circolare**: formule di tipo *linear blending di archi ellittici*.

Elementi poligonali con lato curvo concavo

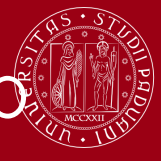
Per elementi poligonali con lato curvo concavo, quindi, consideriamo:

Suddivisione: due poligoni e alcuni quadrilateri circolari.



- $\{P_1, ..., P_{\ell_1}\}$, che contiene i vertici consecutivi raccolti a partire da P_1 esterni all'angolo (θ_1, θ_ℓ);
- $\{P_{\ell_2}, ..., P_\ell\}$, che contiene i vertici consecutivi raccolti a partire da P_ℓ esterni all'angolo (θ_1, θ_ℓ);
- $\{P_{\ell_1+1}, ..., P_{\ell_2-1}\}$ che contiene tutti gli eventuali vertici restanti.

Elementi poligonali con lato curvo concavo



Formule: di tipo algebrico, a bassa cardinalità e del grado richiesto su ogni dominio della suddivisione:

1. **triangolo**: formule quasi-minimali (*cardinalità prossima alla minima possibile per il grado di precisione fissato*);
2. **quadrilateri circolari**: algebrico di tipo *linear blending di archi ellittici*.

Esperimenti numerici

Di seguito facciamo degli esperimenti su un elemento poligonale con lato curvo di tipo concavo per mostrare che

- le formule non compresse dei due sistemi sono numericamente identiche,
- le formule compresse hanno le proprietà richieste ma possono essere diverse viste le differenti routines che implementano l'algoritmo di Lawson-Hanson nei due sistemi di calcolo.

Completeremo con alcuni esperimenti numerici su una certa integranda, ottenendo risultati comparabili offerti dai codici MATLAB e PYTHON.

Nodi e pesi delle formule

Mettendo a confronto, tramite la norma infinito, i nodi e i pesi trovati in MATLAB con quelli raggiunti in PYTHON, si nota che:

- i nodi e i pesi trovati per le formule originali sono gli stessi in entrambi i linguaggi;
- i nodi e i pesi trovati per le formule compresse possono non essere gli stessi.

ADE	E_{nodes}	$E_{weights}$	$E^{(c)}_{nodes}$	$E^{(c)}_{weights}$
2	$5.0 \cdot 10^{-16}$	$4.7 \cdot 10^{-16}$	$4.4 \cdot 10^{-16}$	$4.9 \cdot 10^{-16}$
4	$5.6 \cdot 10^{-16}$	$5.0 \cdot 10^{-16}$	$4.4 \cdot 10^{-16}$	$4.8 \cdot 10^{-16}$
6	$5.6 \cdot 10^{-16}$	$5.0 \cdot 10^{-16}$	$3.3 \cdot 10^{-1}$	$1.7 \cdot 10^{-2}$
8	$5.3 \cdot 10^{-16}$	$5.0 \cdot 10^{-16}$	$1.9 \cdot 10^{-1}$	$5.0 \cdot 10^{-3}$
10	$5.6 \cdot 10^{-16}$	$5.0 \cdot 10^{-16}$	$3.0 \cdot 10^{-1}$	$6.7 \cdot 10^{-3}$

Formule non completamente equivalenti

Nonostante le formule compresse siano differenti sono entrambe accettabili, in quanto soddisfano la condizione dei momenti:

ADE	$\texttt{momerr}_M$	$\texttt{momerr}_P$
2	$7.4 \cdot 10^{-18}$	$5.9 \cdot 10^{-18}$
4	$9.0 \cdot 10^{-18}$	$7.1 \cdot 10^{-18}$
6	$9.9 \cdot 10^{-18}$	$6.8 \cdot 10^{-18}$
8	$9.3 \cdot 10^{-18}$	$6.3 \cdot 10^{-18}$
10	$1.4 \cdot 10^{-17}$	$5.5 \cdot 10^{-18}$

Il parametro `momerr` valuta l'errore sui momenti in norma 2 di una certa base polinomiale a grado n e, qualora sia vicino alla precisione di macchina, indica che la formula è numericamente esatta.

Equivalenza degli integrali calcolati

Calcoliamo l'integrale su tale elemento poligonale di tipo concavo Ω della funzione

$$\int_{\Omega} e^{-(x-0.2)^2-(y-0.2)^2}\, dxdy \approx 0.14517446422077326 3845. \qquad (1)$$

ADE	ER_M	ER_P	ERC_M	ERC_P
2	$6.3 \cdot 10^{-5}$	$6.3 \cdot 10^{-5}$	$1.2 \cdot 10^{-4}$	$1.2 \cdot 10^{-4}$
4	$2.0 \cdot 10^{-7}$	$2.0 \cdot 10^{-7}$	$3.3 \cdot 10^{-7}$	$3.3 \cdot 10^{-7}$
6	$4.7 \cdot 10^{-10}$	$4.7 \cdot 10^{-10}$	$1.0 \cdot 10^{-9}$	$1.0 \cdot 10^{-9}$
8	$8.6 \cdot 10^{-13}$	$8.6 \cdot 10^{-13}$	$1.9 \cdot 10^{-12}$	$1.9 \cdot 10^{-12}$
10	$2.5 \cdot 10^{-15}$	$2.3 \cdot 10^{-15}$	$9.6 \cdot 10^{-16}$	$1.1 \cdot 10^{-15}$

La tabella mostra che i risultati sono numericamente equivalenti.

Tempi di calcoli

Concludiamo coi tempi di calcolo che indicano che i codici in MATLAB sono leggermente più rapidi di quelli in PYTHON.

Formule originali

ADE	MATLAB	PHYTON
2	$2.545 \cdot 10^{-3}$	$5.410 \cdot 10^{-3}$
4	$1.546 \cdot 10^{-3}$	$5.432 \cdot 10^{-3}$
6	$1.845 \cdot 10^{-3}$	$6.905 \cdot 10^{-3}$
8	$1.991 \cdot 10^{-3}$	$1.698 \cdot 10^{-2}$
10	$2.786 \cdot 10^{-3}$	$2.169 \cdot 10^{-2}$

Formule compresse

ADE	MATLAB	PHYTON
2	$2.406 \cdot 10^{-3}$	$5.375 \cdot 10^{-3}$
4	$2.337 \cdot 10^{-3}$	$6.218 \cdot 10^{-3}$
6	$3.376 \cdot 10^{-3}$	$8.459 \cdot 10^{-3}$
8	$4.514 \cdot 10^{-3}$	$2.845 \cdot 10^{-2}$
10	$9.514 \cdot 10^{-3}$	$5.807 \cdot 10^{-2}$

Grazie per l'attenzione

Appendice: Quadratura su quadrilateri circolari

Il processo di quadratura per un quadrilatero circolare $\mathcal{Q}$ è più complesso rispetto a quello per un poligono. Innanzitutto, si riscrive $\mathcal{Q}$ in forma parametrica mediante la tecnica di arc blending:

$$\mathcal{Q} = \{(x, y) = \sigma(t, \theta) = tP(\theta) + (1 - t)Q(\theta) \quad , \ (t, \theta) \in [0, 1] \times [\alpha, \beta]\}.$$

con

$$P(\theta) = A_1 \cos(\theta) + B_1 \sin(\theta) + C_1, \quad Q(\theta) = A_2 \cos(\theta) + B_2 \sin(\theta) + C_2.$$

Definiti i vertici del quadrilatero circolare come $V_i = (\xi_i, \eta_i)$ per $i \in \{1, 2, 3, 4\}$, con $\widehat{V_1 V_4}$ come lato curvo e α e β le rispettive coordinate angolari di V_1 e V_4, si definiscono

$$s = \sin\left(\frac{\beta - \alpha}{2}\right) \qquad \phi = \frac{\beta + \alpha}{2}.$$

Appendice: Quadratura su quadrilateri circolari

È ora possibile quindi definire ogni poligono circolare come funzione bilineare ponendo

$$A_1 = (r, 0), \qquad B_1 = (0, r), \qquad C_1 = (x_0, y_0),$$

$$A_2 = \frac{\sin(\phi)}{2s}(V_3 - V_2), \qquad B_2 = \frac{\cos(\phi)}{2s}(V_3 - V_2), \qquad C_2 = \frac{1}{2}(V_3 + V_2).$$

Da qui è immediata la formula di cubatura con le caratteristiche richieste:

$$\int\int_{\Omega} p(x, y)\ dxdy = \int\int_{[0,1]\times[\alpha,\beta]} p(\sigma(t, \theta))[\pm \det(J\sigma(t, \theta))]\ dtd\theta\ .$$

Appendice: Corrispondenze con i codici

- **Formule di quadratura**
  - **Polygauss**: produce nodi e pesi per la formula di quadratura gaussiana sui triangoli che compongono le parti poligonali dell'elemento circolare.
  - **Circtrap**: produce nodi e pesi per la formula di quadratura gaussiana sulle componenti di lato curvo dell'elemento circolare. La tecnica di *arc blending* è implementata direttamente all'interno della funzione, mentre il calcolo dei valori è affidato alla funzione `gqellblend`.

Appendice: Corrispondenze con i codici

- **Compressione**
  - **Chebvand**: costruisce la matrice di Chebyshev-Vandermonde per il programma lineare da risolvere.
  - **Comprexcub**: risolve il sistema lineare a partire da Chebvand.
- **Visualizzazione**
  - **Demo_polygcirc**: permette di visualizzare graficamente i gli elementi circolari usati per l'integrazione, e di stampare i valori riportati all'interno delle tabelle, oltre alle specifiche sulle formule utilizzate (grado di precisione, numero di nodi, algoritmo di compressione, etc.)

Appendice: Note sulla traduzione

Per mantenere in Python uno stile quanto più simile possibile al codice sorgente, all'interno della traduzione sono state prese le seguenti accortezze:

- tutti i valori definiti in Matlab tramite il comando `nargin` sono definiti a priori come input facoltativi durante la costruzione delle varie funzioni in Python;
- i valori non definiti ma non calcolati in Matlab tramite il comando `nargout` sono stati riproposti in modo facoltativo anche nella traduzione tramite un comando `if` controllato da una `flag` facoltativa in input;
- all'interno dei cicli `for` che vagliano gli indici delle variabili, il valore analizzato di volta in volta è già diminuito di 1 rispetto agli indici Matlab, salvo rare eccezioni in cui tali indici hanno effettivo valore numerico.

Appendice: Compressione delle formule

Per un teorema di Tchakaloff, da formule a pesi positivi con grado di precisione n e un numero di nodi maggiore della dimensione $\texttt{dim}_n = \frac{(n+1)(n+2)}{2}$ dello spazio polinomiale $\mathbb{P}_n$, possiamo ricavare una formula con cardinalità al più $\texttt{dim}_n$, pesi positivi e grado di esattezza nuovamente n.

Tale compressione avviene mediante l'applicazione dell'algoritmo di Lawson-Hanson

$$V^t u = b \ , \quad u \geq 0,$$

dove V é la matrice di Chebyshev-Vandermonde sui nodi della formula iniziale.