QUADRATURA SUBPERIODICA IN PYTHON

MATTIA STORGATO
Laurea Triennale in Matematica
19/09/2025

UNIVERSITÀ DEGLI STUDI DI PADOVA

Introduzione

In questo lavoro implementiamo un algoritmo per il calcolo di formule di quadratura trigonometrica subperiodica in Python.

Ció significa determinare nodi $\{x_i\}_{i=1,\dots,n+1}$ e pesi $\{w_i\}_{i=1,\dots,n+1}$ tali che

$$\int_\alpha^\beta f(x)dx = \sum_{k=1}^{n+1} w_i f(x_i), \quad [\alpha,\beta] \subseteq [0,2\pi]$$

per ogni f appartenente allo spazio $\mathbb{T}_n$ dei polinomi trigonometrici di grado n.

- Analizzaremo la sua implementazione nei codici `trigauss` in Matlab, offrendo una traduzione in Python come alternativa.
- Dopo aver verificato l'equivalenza numerica delle formule univariate implementate nei due sistemi di calcolo, valuteremo una applicazione di `trigauss` legata al calcolo di integrali in certi domini bivariati.

Quadratura subperiodica

Fissati $0 \leq \alpha < \beta \leq 2\pi$ e posto $\omega = \frac{\beta-\alpha}{2}$, siano $\{\xi_j, \lambda_j\}_{1\leq j\leq n+1}$ i nodi e i pesi della formula di Gaussiana per la funzione peso sono

$$w(x) = \frac{2\sin(\omega/2)}{\sqrt{1 - x^2\sin^2(\omega/2)}}, \quad x \in (-1, 1). \tag{1}$$

Quadratura subperiodica

Fissati $0 \leq \alpha < \beta \leq 2\pi$ e posto $\omega = \frac{\beta-\alpha}{2}$, siano $\{\xi_j, \lambda_j\}_{1\leq j\leq n+1}$ i nodi e i pesi della formula di Gaussiana per la funzione peso sono

$$w(x) = \frac{2\sin(\omega/2)}{\sqrt{1 - x^2\sin^2(\omega/2)}}, \quad x \in (-1, 1). \tag{1}$$

Definiti

$$\theta_j = 2\arcsin(\xi_j \sin(\omega/2)) \in (-\omega, \omega), \quad j = 1, 2, ..., n+1, \quad \omega = \frac{\beta - \alpha}{2} \tag{2}$$

si ha che

$$\int_\alpha^\beta f(\theta)d\theta = \sum_{j=1}^{n+1} \lambda_j f(\theta_j + \mu), \qquad \mu = \frac{\alpha + \beta}{2}, \tag{3}$$

per ogni $f \in \mathbb{T}_n$.

Implementazione

Nella fase di implementazione vogliamo trovare nodi e pesi di quadratura in modo efficiente. Come proposto in

G. Da Fies, M. Vianello, Trigonometric Gaussian quadrature on subintervals of the period,
Electron. Trans. Numer. Anal. 39 (2012), 102-112.

Implementazione

Nella fase di implementazione vogliamo trovare nodi e pesi di quadratura in modo efficiente. Come proposto in

G. Da Fies, M. Vianello, Trigonometric Gaussian quadrature on subintervals of the period,
Electron. Trans. Numer. Anal. 39 (2012), 102-112.

utilizziamo:

- Algoritmo di Chebyshev modificato, che sfrutta il momento misto:

$$\sigma_{k,l} = \int_{\alpha}^{\beta} \pi_k(x) p_l(x) w(x) dx. \tag{4}$$

Implementazione

Nella fase di implementazione vogliamo trovare nodi e pesi di quadratura in modo efficiente. Come proposto in

G. Da Fies, M. Vianello, Trigonometric Gaussian quadrature on subintervals of the period,
Electron. Trans. Numer. Anal. 39 (2012), 102-112.

utilizziamo:

- Algoritmo di Chebyshev modificato, che sfrutta il momento misto:

$$\sigma_{k,l} = \int_{\alpha}^{\beta} \pi_k(x) p_l(x) w(x) dx. \tag{4}$$

- Algoritmo di Golub-Welsch, che partendo dai coefficienti di ricorsione di una famiglia di polinomi ortogonali ricava i pesi e nodi della formula gaussiana associata a w.

Test su intervalli univariati

Per confrontare i codici scritti nei due linguaggi abbiamo fatto dei test su due intervalli univariati, esprimendo

- il massimo errore su nodi e pesi:

$$E_{nodi} = \|x^{(m)} - x^{(p)}\|_\infty, \tag{5}$$

  dove $(x^{(m)}, w^{(m)})$, $(x^{(p)}, w^{(p)})$ sono i nodi e i pesi rispettivamente in Matlab e in Python;

- l'errore relativo $\tilde{E} = \frac{\left|I - I_n^{(p)}\right|}{I}$ dove fissato n

$$\begin{aligned} f(x) &= 5 + \frac{1}{2}\sin(17x) - 6\cos(14x), \\ I &= \int_{\pi/6}^{\pi/4} f(x)dx \approx 2.062453518370601, \\ I_n^{(p)} &= \sum_{i=1}^{n+1} w_i^{(p)} f(x_i^{(p)}). \end{aligned}$$

Risultati

n	E_{nodi}	E_{pesi}
5	0	9.02×10^{-17}
10	1.11×10^{-16}	1.35×10^{-16}
15	1.11×10^{-16}	1.70×10^{-16}
20	0	4.51×10^{-17}

Massimi errori nei nodi e nei pesi, tra le formule ottenute in Matlab e Python, relativamente al primo intervallo $(\pi/6, \pi/4)$.

n	$E_{assoluto}$	$E_{relativo}$
5	1.78×10^{-15}	3.31×10^{-11}
10	4.44×10^{-16}	6.46×10^{-16}
15	8.88×10^{-16}	1.51×10^{-15}
20	4.44×10^{-16}	1.29×10^{-15}

Errori relativi agli integrali calcolati sul primo intervallo $(\pi/6, \pi/4)$.

Blending lineare di archi ellittici

Una applicazione è il calcolo di formule algebriche, con grado di precisione fissato su domini *blending lineare di archi ellittici*.

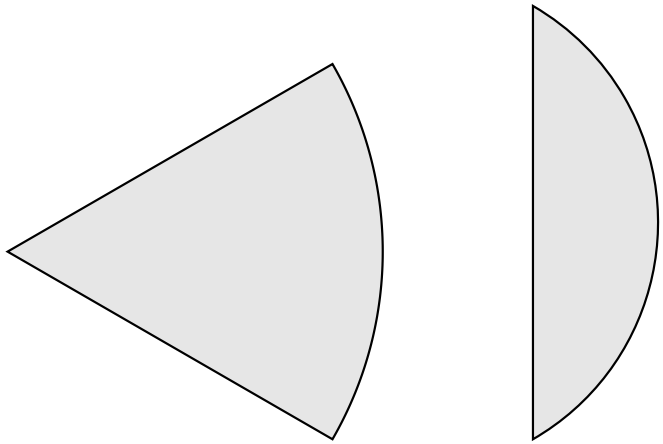
Date 2 curve

$$\begin{aligned} P(\theta) &= A_1 \cos(\theta) + B_1 \sin(\theta) + C_1 \\ Q(\theta) &= A_2 \cos(\theta) + B_2 \sin(\theta) + C_2 \end{aligned}$$

con $A_i, B_i, C_i \in \mathbb{R}^2$, $i = 1, 2$, definiamo il dominio

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

Blending lineare di archi ellittici



Esempi di domini che sono blending lineare di archi ellittici. A sinistra un settore circolare, a destra un segmento circolare.

Formula di quadratura per linear blending di archi ellittici

In

G. Da Fies, A. Sommariva, M. Vianello, Algebraic cubature by linear blending of elliptical arcs, Applied Numerical Mathematics, 74 (2013), 49-61.

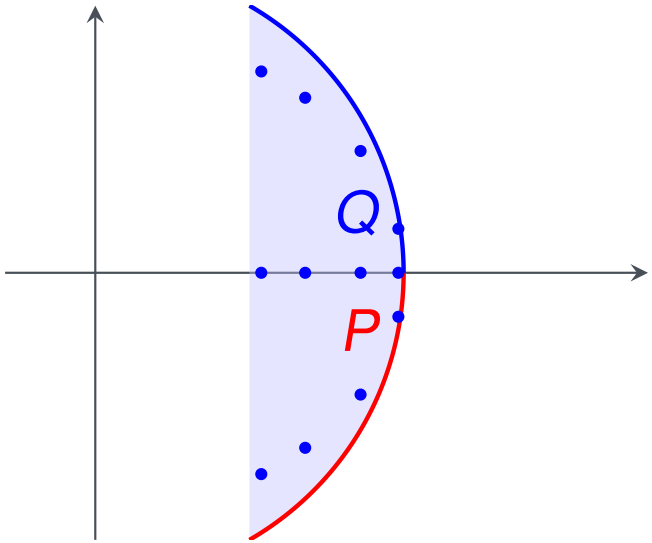
gli autori hanno descritto come, utilizzando `trigauss`, si possano ottenere su domini di questo tipo formule di cubatura a nodi interni, pesi positivi e grado di precisione fissato, proponendo dei codici Matlab.

Abbiamo tradotto le relative routines in Python e verificato che corrispondano numericamente a quelle implementate nell'articolo sopracitato.

Esempio 1

Quale primo esempio consideriamo un segmento circolare Ω_1 (linear blending di due archi P e Q).

Utilizziamo la routine `gqcircsegm` per produrre una formula di grado di precisione n con cardinalità di $\lceil \frac{n+1}{2} \rceil \cdot \lceil \frac{n+2}{2} \rceil$ e pesi positivi.



Segmento circolare e formula di grado 5.

Esempio 1

Approssimiamo con le formule indicate

$$I = \int_{\Omega_1} x - y^3 + x^7 y \quad dxdy$$

sul dominio bivariato.

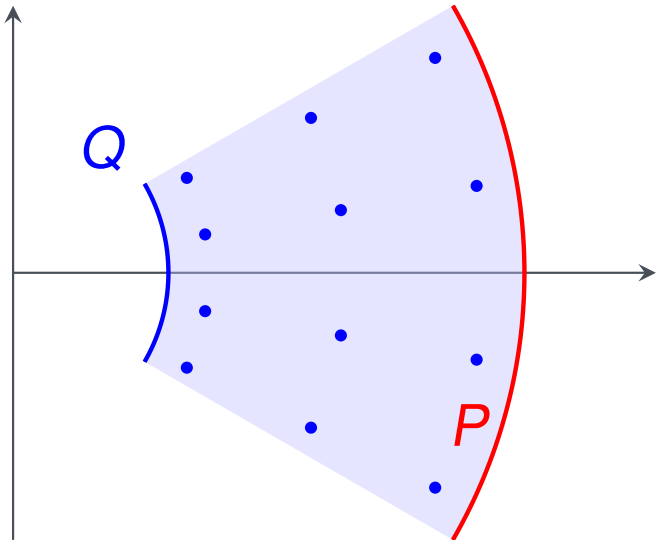
n	nodi	$E_{assoluto}$	$E_{relativo}$
5	12	1.13×10^{-16}	5.13×10^{-16}
10	36	2.56×10^{-16}	3.85×10^{-16}
15	72	3.89×10^{-16}	2.18×10^{-15}
20	121	3.41×10^{-16}	6.41×10^{-16}

Errori relativi all'approssimazione di I, calcolati sul segmento circolare generato da linear blending, al variare di n.

Esempio 2

Come secondo esempio consideriamo una sezione di una corona circolare Ω_2 (linear blending di due archi P e Q).

Utilizziamo la routine `gqcircsect` che determina formule di cardinalità $\lceil \frac{n+2}{2} \rceil \cdot (n+1)$ e grado di precisione n.



Formula di grado 3 sulla sezione di una corona circolare Ω_2.

Esempio 2

n	nodi	$E_{assoluto}$	$E_{relativo}$
5	24	2.56×10^{-13}	7.09×10^{-14}
10	66	1.55×10^{-13}	3.98×10^{-14}
15	144	1.88×10^{-13}	8.42×10^{-14}
20	231	1.20×10^{-13}	2.37×10^{-14}

Errori relativi all'approssimazione di $I = \int_{\Omega_2} x - y^3 + x^7 y \quad dxdy$, al variare di n.

Confronto tempi di calcolo

Per concludere il lavoro abbiamo paragonato i tempi di calcolo nei due sistemi.

Si riscontra una maggiore rapidità in Matlab ma i risultati Python sono comunque rimasti sull'ordine di 10^{-2}. Elenchiamo i risultati ottenuti relativamente al primo dominio di tipo linear blending.

n	$t_p[s]$	$t_m[s]$
5	1.07×10^{-3}	3.31×10^{-4}
10	1.80×10^{-3}	3.06×10^{-4}
15	5.69×10^{-3}	3.41×10^{-4}
20	7.22×10^{-3}	4.55×10^{-4}

Tempi medi di applicazione t_p, t_m del programma `gqcircsegm` nei 2 linguaggi (rispettivamente Python e Matlab).

Università
degli Studi
di Padova

Grazie!

Appendice: Formula di cubatura per domini linear blending di archi ellittici

$$\iint_{\Omega} f(x,y)\,dxdy = I_n(f) = \sum_{j=1}^{n+k+1} \sum_{i=1}^{\left\lceil \frac{n+h+1}{2} \right\rceil} W_{ij}\, f(x_{ij}, y_{ij}), \quad \forall f \in \mathbb{P}_n^2$$

$$(x_{ij}, y_{ij}) = U(t_i^{GL}, \theta_j + \mu), \qquad W_{ij} = \left|\det JU\left(t_i^{GL}, \theta_j + \mu\right)\right| \omega_i^{GL}\, \lambda_j,$$

dove $\{(\theta_j + \mu, \lambda_j)\}$ sono i nodi e i pesi della formula Gaussiana trigonometrica su $[\alpha, \beta]$, e $\{(t_i^{GL}, \omega_i^{GL})\}$ i nodi e i pesi della formula di Gauss-Legendre su $[0, 1]$.

Appendice: Formula di cubatura per domini linear blending di archi ellittici

$$\iint_{\Omega} f(x,y)\,dxdy = I_n(f) = \sum_{j=1}^{n+k+1} \sum_{i=1}^{\left\lceil \frac{n+h+1}{2} \right\rceil} W_{ij}\, f(x_{ij}, y_{ij}), \quad \forall f \in \mathbb{P}_n^2$$

$$(x_{ij}, y_{ij}) = U(t_i^{GL}, \theta_j + \mu), \qquad W_{ij} = \left|\det JU\left(t_i^{GL}, \theta_j + \mu\right)\right| \omega_i^{GL}\, \lambda_j,$$

dove $\{(\theta_j + \mu, \lambda_j)\}$ sono i nodi e i pesi della formula Gaussiana trigonometrica su $[\alpha, \beta]$, e $\{(t_i^{GL}, \omega_i^{GL})\}$ i nodi e i pesi della formula di Gauss-Legendre su $[0, 1]$.
Paragoniamo i 2 linguaggi nel calcolo di

$$\int_{-\omega}^{\omega} \int_{r_1}^{r_2} g(r,\theta) r\; dr\; d\theta, \tag{6}$$

con $g(r, \theta) = f(r\cos(\theta), r\sin(\theta))$.