UNIVERSITÀ
DI PARMA

DIPARTIMENTO DI SCIENZE MATEMATICHE, FISICHE E INFORMATICHE

Corso di Laurea Triennale in Matematica

Tesi di Laurea in
Analisi Numerica

RISOLUZIONE NUMERICA DI EQUAZIONI INTEGRALI DI FREDHOLM DI SECONDA SPECIE CON IL METODO DI KULKARNI

Relatore
Chiar.ma Prof.ssa Alessandra AIMI

Candidato
Alexei CONDURACHI

Settembre 2020

Indice

Parte I
Introduzione

Nell'ambito scientifico capita frequentemente di imbattersi in equazioni differenziali dalla più svariata struttura, ma sono altrettanto frequenti e di elevato interesse le equazioni integrali, ovvero quelle in cui l'incognita compare anche sotto il segno di integrale. Esse hanno le varie applicazioni in ambito fisico, e non solo, a tal punto che si rende necessario poter trovare la soluzione (o le molteplici) a queste. Spesso però la risoluzione di questi problemi matematici risulta ostica e complessa. Grazie agli sviluppi tecnologici degli ultimi decenni alla Matematica è stato fornito un nuovo strumento per poter affrontare suddetti problemi: il calcolatore. Questo adoperato in maniera oculata, permette di risolvere problemi di elevata complessità in tempi drasticamente inferiori.
Proprio della risoluzione di questi ultimi si occupa l'Analisi Numerica. Non bisogna pensare che quest'ultima nasca con l'avvento dei calcolatori moderni: varie figure chiave della Matematica, quali Newton, Lagrange, Jacobi, Gauss ed Eulero, si erano interessate allo studio e alla creazione di algoritmi secoli prima che il calcolatore elettronico venisse inventato.

Parte II
Risoluzione numerica

1 Obiettivi

Per poter affrontare un problema reale da un punto di vista matematico è necessario creare quello che viene detto un *modello* che traduce in linguaggio matematico un fenomeno reale. Da questo si passa ad un problema numerico che viene risolto attraverso *algoritmi*. In questi passaggi vengono effettuate varie semplificazioni e approssimazioni che possono avere ripercussioni sulla soluzione ottenuta. Così come accade in laboratorio dove è necessario tenere conto degli errori e della tolleranza degli strumenti, allo stesso modo in Analisi Numerica bisogna tenere conto degli *errori di rappresentazione*, ovvero i valori che si possono considerare in aritmetica di macchina dovuti alla natura finita del calcolatore, degli *errori analitici*, dovuti al passaggio da continuo a discreto, e degli errori causati dalle semplificazioni effettuate.
Lo studio degli errori analitici è cruciale nella fase di modellizzazione in quanto ci fornisce in prima battuta una iniziale stima dell'errore che intercorre fra la soluzione esatta e quella approssimata.
Risulta ovvio che in molti casi si possono fare migliorie sulla soluzione per ridurre l'errore commesso. Per esempio, possiamo approssimare un cerchio con un poligono più semplice di n lati: se aumentiamo il numero n otteniamo una approssimazione più accurata che però richiede un numero maggiore di passaggi. Allo stesso modo possiamo migliorare la nostra soluzione, ma questo spesso si traduce in un tempo computazionale maggiore. Diventa fondamentale dunque analizzare gli algoritmi e trovarne di più efficienti.
Di seguito confronteremo il *metodo di Galërkin* e il *metodo di Kulkarni* per risolvere equa-

zioni di Fredholm di seconda specie, osservando quale converge più rapidamente.
Il metodo di Galërkin, fondamentale in vari ambiti dell'Analisi Numerica, verrà adoperato come elemento di riferimento e paragone per il più recente metodo di Kulkarni.

2 Presentazione del problema

Introduciamo la nozione di equazione integrale di Fredholm di seconda specie.

Definizione 2.1. Sia T un operatore *lineare compatto* definito sullo spazio X *complesso* e di *Banach*[1]. Siano f e u elementi di X. Viene detta **equazione integrale di Fredholm di seconda specie** la seguente

$$u - Tu = f \tag{1}$$

Assumiamo che $(I - T)$ sia invertibile, in modo che (1) abbia un'*unica soluzione.*
Facciamo un maggior raffinamento su X e supponiamo sia uno spazio di *Hilbert*[2].

Definizione 2.2. Un operatore lineare P su X, tale che $P^2 = P$ si dice *proiezione.*

Consideriamo la successione di proiezioni $\pi_n : X \to X_n$ convergente puntualmente all'operatore identità I.
Esse operano nel seguente modo: ad $x \in X$ associano un elemento $x_n \in X_n$.

Per le proprietà dell'operatore di proiezione si dimostra che ogni elemento di X si può scrivere come somma di un elemento di X_n e di uno del suo complementare X_n^c.

Sia $\varphi_i \quad i = 0, \ldots, n$ una base del sottospazio vettoriale X_n di dimensione finita. Consideriamo la *matrice dei pesi* $\langle \varphi_i, \varphi_j \rangle$, indichiamo con $A := (\alpha_{ij})$ la sua inversa. In questa tesi considereremo proiezioni del seguente tipo

$$\pi_n(f) := \sum_{i,j=1}^{n} \alpha_{ij} < f, \varphi_i > \varphi_j$$

Proposizione 2.1. L'operatore π_n appena definito, è una proiezione. Ovvero verifica

$$\pi_n^2 = \pi_n$$

[1] uno spazio X si dice *Banach* se e solo se è uno spazio *normato completo* rispetto alla metrica indotta dalla norma.

[2] ogni spazio di Hilbert è anche di Banach

Dimostrazione. Sia $u \in X$ e $<,>$ il prodotto interno classico su X, allora

$$\begin{aligned}
\pi_n(\pi_n u) &= \sum_{l,k=0}^{n} \langle \sum_{i,j=0}^{n} \alpha_{ij} \langle u, \varphi_i \rangle \varphi_j, \varphi_l \rangle \varphi_k \\
&= \sum_{l,k=0}^{n} \left(\alpha_{lk} \sum_{i,j=0}^{n} \langle u, \varphi_i \rangle \langle \varphi_j, \varphi_l \rangle \varphi_k \right) \\
&= \sum_{k=0}^{n} \left(\sum_{i,j=0}^{n} \alpha_{ij} \langle u, \varphi_i \rangle \varphi_k \sum_{l=0}^{n} \alpha_{l,k} \langle \varphi_j, \varphi_l \rangle \right) \\
&= \sum_{i,k=0}^{n} \alpha_{ik} \langle u, \varphi_i \rangle \varphi_k = \pi_n u
\end{aligned}$$

□

3 Risoluzione numerica con metodo di Galërkin

Il metodo di Galërkin fornisce una soluzione discreta al problema continuo. Questo passaggio si rivelerà cruciale, poiché ci permette di poter implementare un algoritmo che darà come risultato la soluzione approssimata sostituendo all'operatore T un operatore di *rango finito* T_n.
Il metodo di Galërkin prevede quindi che l'operatore T venga sostituito da

$$T_n^G = \pi_n T \pi_n$$

e che f sia sostituita da $\pi_n f$.
Otteniamo così la nuova equazione

$$u_n^G - \pi_n T \pi_n u_n^G = \pi_n f \tag{2}$$

Risulta immediatamente che vale il seguente

$$\lim_{n \to \infty} u_n = u$$

Procediamo ora alla ricerca del sistema lineare da risolvere per trovare le soluzioni approssimate.
Per il metodo di Galërkin u_n viene scritto nel seguente modo

$$u_n^G := \pi_n f + \sum_{i=1}^{n} X_i \varphi_i$$

sostituendo in (2), otteniamo

$$\pi_n f + \sum_{i=1}^{n} X_i \varphi_i = \pi_n f + \pi_n (T(\pi_n f + \sum_{i=1}^{n} X_i \varphi_i)) \tag{3}$$

La (3) diventa dunque

$$\sum_{j=1}^{n} X_j \varphi_j = (\sum_{i,h=1}^{n} \alpha_{ih} < T \pi_n f, \varphi_i > \varphi_h) + (\sum_{i,h=1}^{n} \alpha_{ih} < T(\sum_{j=1}^{n} X_j \varphi_j), \varphi_i > \varphi_h)$$

Sviluppando per l'elemento $j-esimo$ di X, otteniamo

$$X_j = (\sum_{i=1}^{n} \alpha_{ij} < T(\sum_{t,s=1}^{n} \alpha_{ts} < f, \varphi_t > \varphi_s), \varphi_i >) + (\sum_{p=1}^{n} X_p \sum_{i=1}^{n} \alpha_{ij} < T(\varphi_p), \varphi_i >)$$

$$X_j = (\sum_{i=1}^{n} \alpha_{ij} (\sum_{t,s=1}^{n} \alpha_{ts} < f, \varphi_t >) < T(\varphi_s), \varphi_i >) + (\sum_{p=1}^{n} X_p \sum_{i=1}^{n} \alpha_{ij} < T(\varphi_p), \varphi_i >)$$

Definiamo il vettore $\mathbf{g}$ e la matrice $\mathbf{B}$ come segue

$$\mathbf{g}(k) := \sum_{h=1}^{n} \alpha_{hk} \sum_{i,j=1}^{n} < f, \varphi_i >< T(\varphi_j), \varphi_h >$$

$$\mathbf{B} := (b_{ij}) := \sum_{h=1}^{n} \alpha_{hi} < T(\varphi_j), \varphi_h >$$

Con queste notazioni la (3) si riduce al seguente sistema lineare

$$(\mathbf{I} - \mathbf{B})X = \mathbf{g}$$

4 Risoluzione numerica con metodo di Kulkarni

Il metodo di Kulkarni prevede che T venga sostituito da

$$T_n = \pi_n T \pi_n + \pi_n T(I - \pi_n) + (I - \pi_n) T \pi_n$$

Dunque si ha

$$\|T - T_n\| = \|(I - \pi_n) T (I - \pi_n)\| \to 0 \quad per \quad n \to \infty$$

Si ottiene quindi la seguente approssimazione dell'equazione (1)

$$u_n - (\pi_n T \pi_n + \pi_n T(I - \pi_n) + (I - \pi_n) T \pi_n) u_n = f \tag{4}$$

Applicando π_n ad entrambi i membri

$$\pi_n u_n - (\pi_n T \pi_n + \pi_n T(I - \pi_n) + \pi_n (I - \pi_n) T \pi_n) u_n = \pi_n f \tag{5}$$

Osservazione 4.1. Per le proprietà di π_n si ha $\pi_n(I - \pi_n) = 0$

Dunque la precedente equazione si riduce a

$$\pi_n u_n - \pi_n T \pi_n u_n - \pi_n T(I - \pi_n) u_n = \pi_n f \tag{6}$$

Per ricavare il valore di $(I - \pi_n)u_n$, applichiamo $(I - \pi_n)$ alla (4)

$$(I - \pi_n)u_n - (I - \pi_n) T \pi_n u_n = (I - \pi_n) f$$

ovvero otteniamo che

$$(I - \pi_n)u_n = (I - \pi_n) T \pi_n u_n + (I - \pi_n) f$$

Sostituendo nella (6) otteniamo

$$\pi_n u_n - \pi_n T \pi_n u_n - \pi_n T(I - \pi_n) T \pi_n u_n - \pi_n T(I - \pi_n) f = \pi_n f$$

Ponendo $\omega_n = \pi_n u_n$, abbiamo la seguente equazione

$$\omega_n - (\pi_n T + \pi_n T(I - \pi_n) T)\omega_n = \pi_n f + \pi_n T(I - \pi_n) f \tag{7}$$

Poiché u_n è combinazione della sua proiezione su X_n e sul suo complementare, si scriverà nel seguente

$$u_n = \pi_n u_n + (I - \pi_n) u_n = \omega_n + (I - \pi_n) T \omega_n + (I - \pi_n) f \tag{8}$$

Esplicitiamo quindi i termini di (7) considerando che $\omega_n = \sum_{i=1}^{n} X_i \varphi_i$

- $\pi_n T \omega_n = \pi_n \sum_{t=1}^{n} X_t T(\varphi_t) = \sum_{i,h=1}^{n} \alpha_{ij} \left(\sum_{t=1}^{n} < T(\varphi_t), \varphi_i > \varphi_j \right)$
- $\pi_n T^2 \omega_n = \pi_n T \left(\sum_{t=1}^{n} T(\varphi_t) \right) = \sum_{i,j=1}^{n} \alpha_{ij} \left(\sum_{t=1}^{n} < T^2(\varphi_t), \varphi > \varphi_j \right)$
- $\pi_n T \pi_n T \omega_n = \sum_{i,j=1}^{n} \alpha_{ij} \left(\sum_{k,l}^{n} \alpha_{kl} \left(\sum_{t=1}^{n} X_t < T(\varphi_t), \varphi_k > \right) < T(\varphi_l), \varphi_i > \varphi_j \right)$
- $\pi_n f = \sum_{i,j=1}^{n} (\alpha_{ij} < f, \varphi_i > \varphi_j)$
- $\pi_n T f = \sum_{i,j=1}^{n} (\alpha_{ij} < T(f), \varphi_i > \varphi_j)$
- $\pi_n T \pi_n f = \sum_{i,j=1}^{n} \alpha_{ij} \left(\sum_{k,l=1}^{n} (\alpha_{kl} < f, \varphi_k >) < T(\varphi_l), \varphi_i > \varphi_j \right)$

Introduciamo dunque le matrici $\mathbf{A}$, $\mathbf{B}$, $\mathbf{M}$ e i vettori $\mathbf{b}$, $\mathbf{g}$ definiti come segue

- $\mathbf{A} :=< T(\varphi_j), \varphi_i >$
- $\mathbf{B} :=< T^2(\varphi_j), \varphi_i >$
- $\mathbf{M} :=< \varphi_i, \varphi_j >$
- $\mathbf{b} :=< f, \varphi_i >$
- $\mathbf{g} :=< T(f), \varphi_i >$

Possiamo riscrivere la (7) in forma matriciale

$$(\mathbf{I} - \mathbf{M}^{-1}\mathbf{A} - \mathbf{M}^{-1}\mathbf{B} + (\mathbf{M}^{-1}\mathbf{A})^2)\mathbf{X} = \mathbf{M}^{-1}\left(\mathbf{b} + \mathbf{A} - \mathbf{A}\mathbf{M}^{-1}\mathbf{b}\right) \tag{9}$$

o equivalentemente moltiplicando per $\mathbf{M}$

$$(\mathbf{M} - \mathbf{A} - \mathbf{B} + \mathbf{M}(\mathbf{M}^{-1}\mathbf{A})^2)\mathbf{X} = \left(\mathbf{b} + \mathbf{A} - \mathbf{A}\mathbf{M}^{-1}\mathbf{b}\right)$$

Una volta risolto il sistema è sufficiente sostituire il valore di ω_n in (8) per trovare la soluzione discretizzata u_n.

5 Stime degli errori

Consideriamo la seguente definizione per l'operatore integrale

$$(Tx)(s) = \int_0^1 k(s,t)x(t)dt \quad , \quad s \in [0,1] \tag{10}$$

con $k(\cdot,\cdot) \in C([0,1] \times [0,1])$. Allora $T : L^2[0,1] \to L^2[0,1]$ è un *operatore lineare compatto.*
Sia $r \geq 1$. Se $k(\cdot,\cdot) \in C^r([0,1] \times [0,1])$, allora $R(T) \subset C^r[0,1]$, dove $R(T)$ indica il range dell'operatore T. Quindi, se $k(\cdot,\cdot) \in C^r([0,1] \times [0,1])$ e $f \in C^r[0,1]$, allora $u \in C^r[0,1]$.

Per $u \in C^r[0,1]$, si indichi con $u^{(r)}$ la r-esima derivata di u. Definiamo

$$D^{i,j}k(s,t) := \frac{\partial^{i+j}}{\partial s^i \partial t^j}k(s,t) \quad , \quad s,t \in [0,1]$$

$$\|k\|_{r,\infty} = \sum_{i=0}^{r}\sum_{j=0}^{r}\|D^{i,j}k\|_\infty$$

e

$$\|u\|_{r,\infty} = \sum_{i=0}^{r}\|u^{(i)}\|_\infty$$

La stima dell'errore per u_n^G e u_n sono ottenute dal seguente

Teorema 5.1. Per n sufficientemente grandi

$$\|u - u_n^G\| \leq C1\|(I - \pi_n)u\| \tag{11}$$

e

$$\|u - u_n\| \leq C_2\|(I - \pi_n)T(I - \pi_n)u\| \tag{12}$$

Dimostrazione. Siccome $\|T - T_n^G\| \to 0$ per $n \to \infty$, per n grandi, $(I - T_n^G)$ è invertibile, $\|T - T_n^G\| \leq C_1$ e, poiché $(I - T)$ è invertibile, $\|T + \pi_n T\| \leq C_2$ con C_1, C_2 indipendenti da n. Dunque

$$\begin{aligned}
u - u_n^G &= [(I-T)^{-1} - (I - \pi_n T\pi_n)^{-1}]f = \\
&= [(I-T)^{-1} - (I - \pi_n T\pi_n)^{-1}](I-T)u = \\
&= [I - (I - \pi_n T\pi_n)^{-1}(I-T)]u = \\
&= [(I - \pi_n T\pi_n)^{-1}(I - \pi_n T\pi_n) - (I - \pi_n T\pi_n)^{-1}(I-T)]u = \\
&= (I - \pi_n T\pi_n)^{-1}[I - \pi_n T\pi_n - I + T]u = \\
&= (I - \pi_n T\pi_n)^{-1}[T - \pi_n T\pi_n]u
\end{aligned}$$

Studiamo ora $T - \pi_n T \pi_n$

$$\begin{aligned} T - \pi_n T \pi_n &= T - \pi_n T + \pi_n T - \pi_n T \pi_n = (I - \pi_n)T + \pi_n T(I - \pi_n) = \\ &= T - T\pi_n + T\pi_n - \pi_n T \pi_n = T(I - \pi_n) + (I - \pi_n)T\pi_n \end{aligned}$$

Dunque, poiché i termini sono equivalenti

$$\begin{aligned} &(I - \pi_n)T + \pi_n T(I - \pi_n) = T(I - \pi_n) + (I - \pi_n)T\pi_n \\ &\Rightarrow (I - \pi_n)T(I - \pi_n) = T(I - \pi_n)(I - pi_n) \\ &\Rightarrow (I - \pi_n)T = T(I - \pi_n) \end{aligned}$$

Sostituendo il risultato ottenuto nella precedente, abbiamo che

$$\begin{aligned} T - \pi_n T \pi_n &= T(I - \pi_n) + \pi_n T(I - \pi_n) = \\ &= (T + \pi_n T)(I - \pi_n) \end{aligned}$$

In conclusione abbiamo che

$$u - u_n^G = (I - \pi_n T \pi_n)^{-1}(T + \pi_n T)(I - \pi_n)u$$

Dunque

$$\begin{aligned} \|u - u_n^G\| &\leq \|(I - \pi_n T \pi_n)^{-1}\| \|T + \pi_n T\| \|(I - \pi_n)u\| \leq \\ &\leq C \|(I - \pi_n)u\| \end{aligned}$$

che conclude la dimostrazione per (11), posto $C := C_1 \cdot C_2$.
Dimostriamo ora la (12).
Poiché $\|T - T_n\| \to 0$ per $n \to \infty$, per n grandi, $(I - T_n)$ è invertibile e $\|(I - T_n)^{-1}\| \leq C_2$, costante indipendente da n.
Abbiamo

$$\begin{aligned} u - u_n &= [(I - T)^{-1} - (I - T_n)^{-1}]f = \\ &= [(I - T)^{-1} - (I - T_n)^{-1}](I - T)u = \\ &= [I - (I - T_n)^{-1}(I - T)]u = \\ &= [(I - T_n)^{-1}(I - T_n) - (I - T_n)^{-1}(I - T)]u = \\ &= (I - T_n)^{-1}[I - T_n - I + T]u = \\ &= (I - T_n)^{-1}(T - T_n)u \end{aligned}$$

Quindi,

$$\begin{aligned} \|u - u_n\| &\leq \|(I - T_n)^{-1}\| \|(I - \pi_n)T(I - \pi_n)u\| \\ &\leq C_2 \|(I - \pi_n)T(I - \pi_n)u\| \end{aligned}$$

□

6 Ordini di convergenza

Analizziamo ora gli ordini di convergenza dei due metodi.

Sia $X = L^2[0,1]$ e indichiamo con $< , >$ il prodotto interno classico su X. Sia T un operatore integrale, come definito in (10), con nucleo $k(\cdot,\cdot) \in C^r([0,1]\times[0,1])$. Per ogni intero n, sia

$$0 = t_0 < t_1 < \cdots < t_n = 1$$

una partizione di $[0,1]$ e per $i = 1,\ldots,n$. Definiamo $h_i := t_i - t_{i-1}$, $h := \max\limits_{i=0,\ldots,n} h_i$. Si assume inoltre che $h \to 0$ per $n \to \infty$.
Sia $X_n = S^\nu_{r,n}$, dove $S^\nu_{r,n}$ indica lo spazio dei polinomi a tratti di ordine r (ovvero di grado $\leq r-1$) sui punti di interruzione $t_1,\ldots,t_{n-1}$ e con ν derivate continue ($-1 \leq \nu \leq r-2$).
Sia $\pi_n : X \to X_n$ la proiezione *ortogonale*.
Per $\nu = -1$ o $\nu = 0$, è noto, senza alcuna restrizione sulla partizione di $[0,1]$, che (vedere Richter ([11]) e de Boor ([4]), rispettivamente)

$$\|\pi_n\|_{L^\infty \to L^\infty} \leq c \tag{13}$$

Poiché $\pi_n x \to x$ per $n \to \infty$ per ogni $x \in X$, i risultati ottenuti nella Sezione 5 sono ancora validi.
Per l'analisi degli ordini è cruciale la seguente stima.

Proposizione 6.1. Per $x \in C^r[0,1]$, (vedere Chatelin ([3]))

$$\|(I-\pi_n)x\|_\infty \leq C_1 \|x^{(r)}\|_\infty h^r \tag{14}$$

con C_1 costante indipendente da n e h.

Proposizione 6.2. Per $x \in C^r[0,1]$ abbiamo

$$\|T(I-\pi_n)x\|_{r,\infty} \leq (C_1)^2 (r+1) \|k\|_{r,\infty} \|x^{(r)}\|_\infty h^{2r}$$

Dimostrazione. Per j fissato tale che $0 \leq j \leq r$, si abbiamo che la derivata j-esima rispetto ad s di $T(I-\pi_n)x$ è

$$[T(I-\pi_n)x]^{(j)}(s) = \int_0^1 \frac{\partial^j}{\partial s^j} k(s,t)(I-\pi_n)x(t)dt$$

Definiamo

$$l(s,t) := \frac{\partial^j}{\partial s^j} k(s,t), \quad s,t \in [0,1]$$

Per $s \in [0,1]$ fissato, indichiamo con $l_s(t) := l(s,t)$, $t \in [0,1]$ per mettere in evidenza la dipendenza da t. Allora

$$\begin{aligned}
[T(I-\pi_n)x]^{(j)}(s) &= \int_0^1 l_s(t)(I-\pi_n)x(t)dt \\
&= \langle (I-\pi_n)x, l_s(t)\rangle \\
&= \langle (I-\pi_n)x, (I-\pi_n)l_s(t)\rangle
\end{aligned}$$

poichè π_n è la proiezione ortogonale e poiché siamo in $\mathbb{R}$. Dunque per ogni $s \in [0,1]$ otteniamo

$$\begin{aligned}|[T(I-\pi_n)x]^{(j)}(s)| &\leq \|(I-\pi_n)x\|_\infty \|(I-\pi_n)l_s\|_{i\infty} \\ &\leq C_1\|x^{(r)}\|_\infty h \cdot C_1\|l_s^{(r)}\|_\infty h \\ &\leq C_1\|x^{(r)}\|_\infty \|k\|_{r,\infty} h^{2r}\end{aligned}$$

Passando all'estremo superiore su $s \in [0,1]$ abbiamo

$$\|[T(I-\pi_n)x]^{(j)}\|_\infty \leq (C_1)^2\|k\|_{r,\infty}\|x^{(r)}\|_\infty h^{2r} \tag{15}$$

e dunque

$$\|T(I-\pi_n)x\|_{r,\infty} = \sum_{j=0}^{r}\|[T(I-\pi_n)x]^{(j)}\|_\infty \leq (C_1)^2(r+1)\|k\|_{r,\infty}\|x^{(r)}\|_\infty h^{2r}$$

□

Stimiamo ora l'errore per (12)

Proposizione 6.3. Per $x \in C^r[0,1]$, si ottiene

$$\|(I-\pi_n)T(I-\pi_n)x\|_\infty \leq C_2 h^{3r} \tag{16}$$

Dimostrazione. Grazie alla stima (14) abbiamo

$$\|(I-\pi_n)T(I-\pi_n)x\|_\infty \leq C_1\|[T(I-\pi_n)x]^{(r)}\|_\infty h^r$$

Per la (15), si ottiene

$$\begin{aligned}\|(I-\pi_n)T(I-\pi_n)x\|_\infty &\leq C_1 \cdot (C_1)^2\|k\|_{r,\infty}\|x^{(r)}\|_\infty h^{2r} \cdot h^r \\ &\leq (C_1)^3\|k\|_{r,\infty}\|x^{(r)}\|_\infty h^{3r}\end{aligned}$$

che conclude la dimostrazione, ponendo $C_2 := (C_1)^3\|k\|_{r,\infty}\|x^{(r)}\|_\infty$ □

Risultati analoghi si ottengono nel caso π_n sia una *proiezione interpolatoria* (vedere Kulkarni ([16])).
Enunciamo dunque il seguente teorema.

Teorema 6.1. Sia π_n proiezione ortogonale o proiezione interpolatoria su X_n. Nel caso della proiezione ortogonale si considerino $k(\cdot,\cdot) \in C^r([0,1]\times[0,1])$ e $f \in C^r[0,1]$, mentre per la proiezione interpolatoria si considerino $k(\cdot,\cdot) \in C^{2r}([0,1]\times[0,1])$ e $f \in C^{2r}[0,1]$. Allora

$$\|u-u_n\| = O(h^{3r}) \tag{17}$$

Dimostrazione. Studiamo il caso della proiezione ortogonale, da (12) e da (16) segue la (17)

$$\begin{aligned}\|u-u_n\|_2 &\leq C_2\|(I-\pi_n)T(I-\pi_n)u\|_2 \\ &\leq C_2\|(I-\pi_n)T(I-\pi_n)u\|_\infty \\ &\leq (C_2)^2 h^{3r}\end{aligned}$$

ovvero la tesi. □

7 Esempi numerici con proiezione ortogonale

Vediamo alcuni esempi di equazioni di Fredholm di seconda specie e confrontiamo la soluzione esatta, la soluzione approssimata con metodo di Galërkin u_n^G e quella con il metodo di Kulkarni u_n.
Nei seguenti esempi ci siamo posti nello spazio L^2 sull'intervallo $[a, b]$. Consideriamo come sottospazio, l'insieme dato dalle splines costanti a tratti, ovvero dei polinomi di grado $r = 0$ e $\nu = -1$.
Ci aspettiamo quindi ordine di convergenza 1 per il metodo di Galërkin e 3 per quello di Kulkarni.
Ricordiamo l'equazione di Fredholm

$$u(s) - Tu(s) = f(s), \qquad T = \int_a^b k(s,t)u(t)dt, \qquad s,t \in [a,b]$$

7.1 Esempio 1

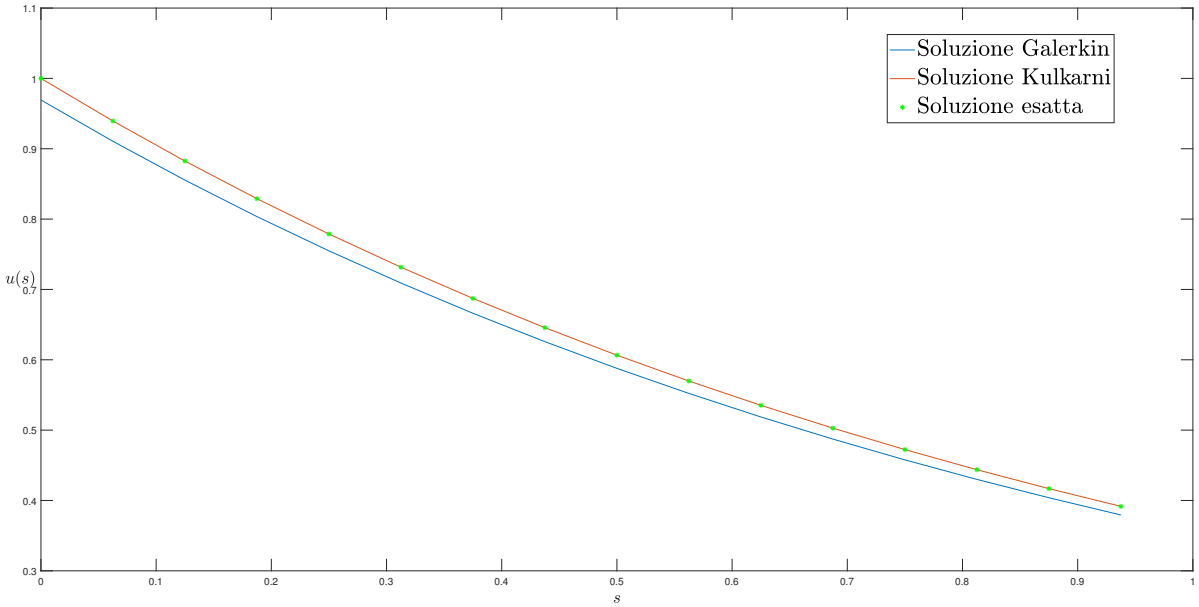
Consideriamo il seguente caso

$$u(s) - \int_0^1 \frac{1}{2}(s+1)e^{-st} \cdot u(t)dt = e^{-s} + \frac{1}{2}(e^{-s-1} - 1)$$

che ha la seguente soluzione

$$u(s) = e^{-s}$$

Le soluzioni approssimate con i due metodi sono le seguenti



Vediamo in modo evidente come il metodo di Kulkarni risulti un grande miglioramento rispetto quello di Galërkin, che comunque fornisce una buona approssimazione anche per n non elevati.
Confrontiamo, al raddoppiare del numero di sottointervalli n, l'andamento degli *errori* e degli *ordini* dei due metodi nella seguente tabella (N.B. per mettere in evidenza p, il rapporto tra due errori successivi è stato normalizzato dividendo per $\log 2$)

N	h	Errore Galerkin	Errore Kulkarni	Ordine Galerkin	Ordine Kulkarni
2	5.0000e−01	2.2006e−01	2.5147e−03	—	—
4	2.5000e−01	1.1666e−01	3.2438e−04	9.1557e−01	2.9546e+00
8	1.2500e−01	6.0305e−02	4.0781e−05	9.5199e−01	2.9917e+00
16	6.2500e−02	3.0687e−02	5.0974e−06	9.7464e−01	3.0000e+00
32	3.1250e−02	1.5483e−02	6.3666e−07	9.8699e−01	3.0012e+00

La tabella mostra come i risultati numerici siano in perfetto accordo con le stime teoriche presentate nelle sezioni precedenti. Inoltre mette in evidenza la maggior precisione del metodo di Kulkarni.

7.2 Esempio 2

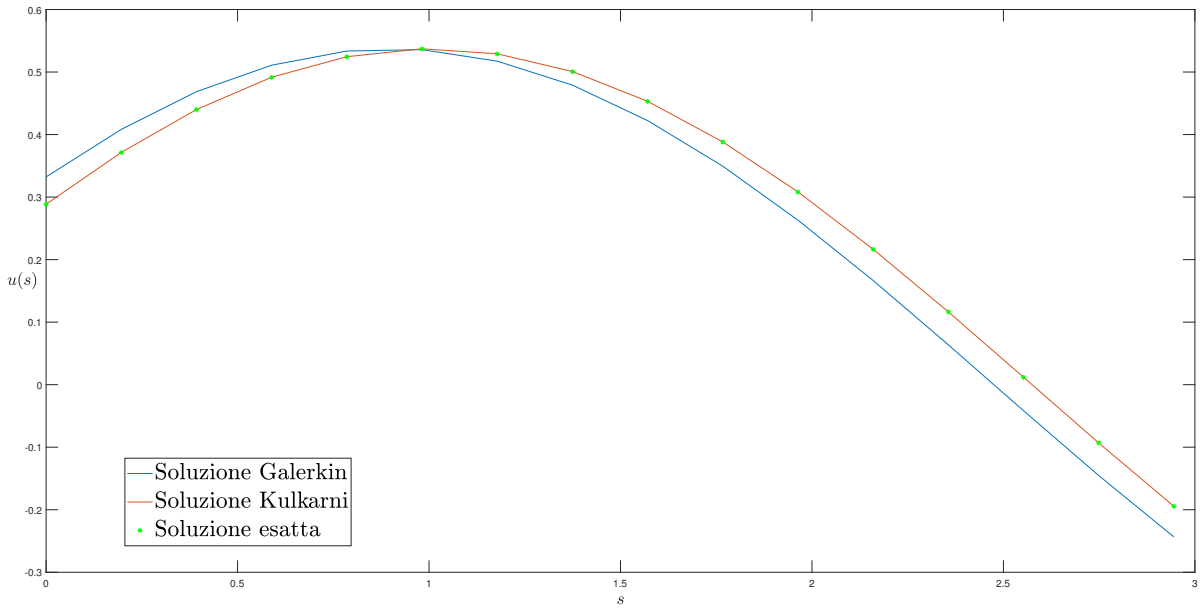
Studiamo il comportamento dei due metodi sull'intervallo $[0, \pi]$ e mostriamo che i risultati ottenuti nelle sezioni precedenti si estendono in modo naturale su intervalli del tipo $[a, b]$ L'equazione di Fredholm in questo caso è la seguente

$$u(s) - \int_0^{\pi} sin(s-t)u(t)dt = cos(s)$$

che ha soluzione

$$u(s) = \frac{2}{4+\pi^2}(2cos(s) + \pi sin(s))$$

Le soluzioni sono mostrate di seguito



Vediamo dunque che anche spostandosi dall'intervallo $[0, 1]$ i due metodi risultano efficaci. Anche gli errori e gli ordini continuano ad essere in linea con le stime calcolate in precedenza, come si nota dalla seguente tabella

N	Errore Galerkin	Errore Kulkarni	Ordine Galerkin	Ordine Kulkarni
2	3.8665e−01	1.0874e−01	—	—
4	2.2307e−01	1.7007e−02	7.9353e−01	2.6766e+00
8	1.0763e−01	2.1455e−03	1.0514e+00	2.9868e+00
16	5.3491e−02	2.6896e−04	1.0087e+00	2.9958e+00
32	2.6557e−02	3.3475e−05	1.0102e+00	3.0062e+00

7.3 Esempio 3

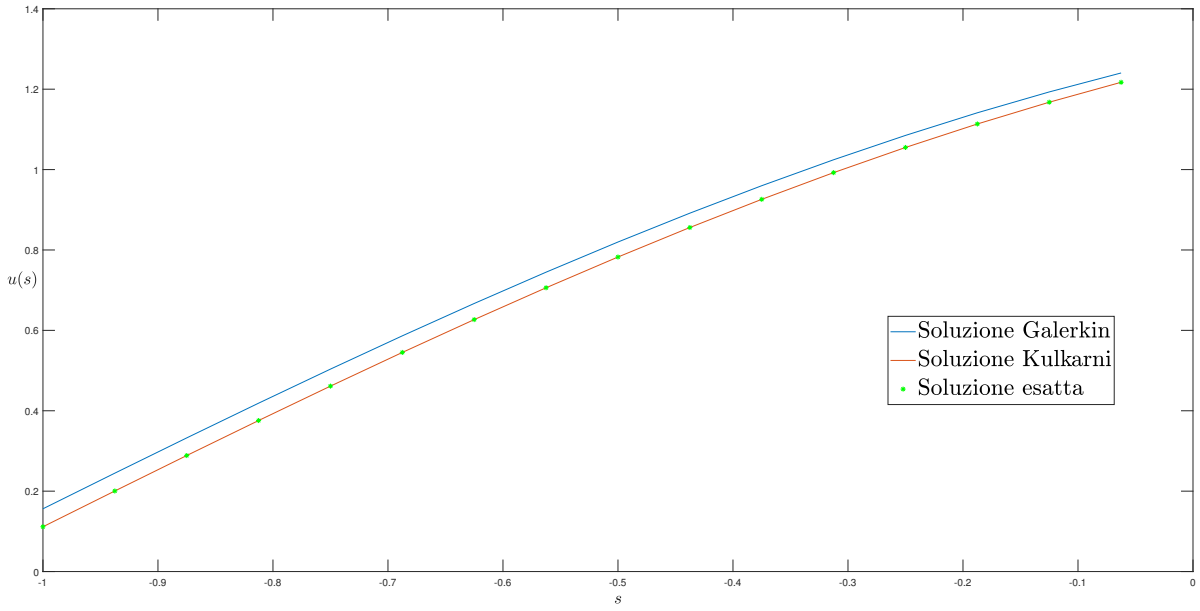
Studiamo nuovamente l'equazione dell'esempio precedente, ponendoci però nell'intervallo $[-1, 0]$ in cui k e f non godono di particolari simmetrie

$$u(s) - \int_{-1}^{0} sin(s-t)u(t)dt = cos(s)$$

che ha soluzione

$$u(s) = -\frac{2}{9 + cos(2)}(-2sin(s) - 5cos(s) + +cos(s+2))$$

Ancora una volta entrambi i metodi forniscono soluzioni consistenti ed efficaci per il problema in analisi



Allo stesso modo non si evincono anomalie nemmeno per gli errori e gli ordini di convergenza

N	Errore Galerkin	Errore Kulkarni	Ordine Galerkin	Ordine Kulkarni
2	3.6149e−01	1.6365e−03	—	—
4	1.7975e−01	2.7387e−04	1.0080e+00	2.5790e+00
8	8.9581e−02	4.1105e−05	1.0047e+00	2.7361e+00
16	4.4711e−02	5.5470e−06	1.0025e+00	2.8895e+00
32	2.2335e−02	7.1827e−07	1.0013e+00	2.9491e+00

7.4 Esempio 4

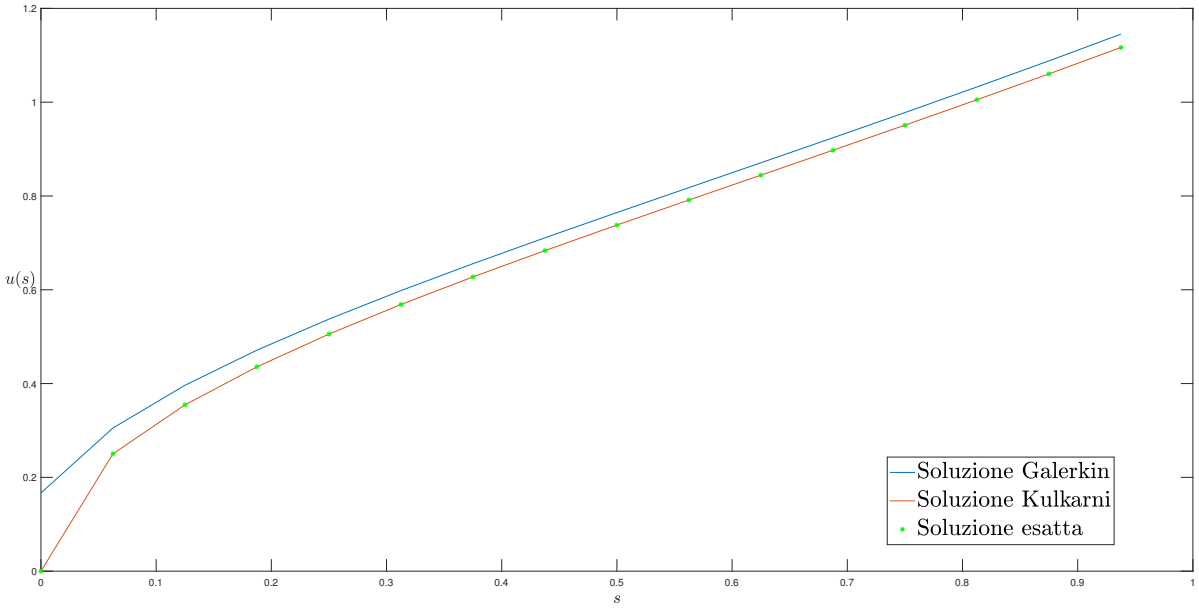
Analizziamo ora il caso in cui la regolarità del nucelo k e di f non sono quelle attese, ovvero non sono di classe $C^1[a.b]$. Un esempio di questa situazione è la seguente equazione

$$u(s) - \int_0^{\pi} s^{5/2} t^5 u(t) dt = \sqrt{s}$$

con soluzione

$$u(s) = \sqrt{s} + \frac{34}{195} s^{5/2}$$

Notiamo che le soluzioni ottenute con i due metodi implementati approssimano in modo efficace la soluzione.



Inoltre notiamo che i due metodi continuano ad essere convergenti nonostante f non sia di classe C^1 in $a = 0$.

N	Errore Galerkin	Errore Kulkarni	Ordine Galerkin	Ordine Kulkarni
2	4.7928e−01	5.4903e−03	0.0000e+00	0.0000e+00
4	3.3484e−01	1.0430e−03	5.1739e−01	2.3961e+00
8	2.3598e−01	1.5539e−04	5.0486e−01	2.7468e+00
16	1.6672e−01	2.1034e−05	5.0125e−01	2.8851e+00
32	1.1786e−01	2.7306e−06	5.0032e−01	2.9455e+00

La discontinuità in uno dei due estremi non inficia dunque la convergenza, ma riduce l'ordine per entrambi i metodi (vedere Kulkarni ([16]), Osservazione 4.7, p 525).

8 Esempi numerici con proiezione interpolatoria

Studiamo il comportamento dei due algoritmi per operatori di proiezione *interpolatoria* invece che di proiezione ortogonale. Ci poniamo dunque nello spazio $X_n = S^0_{2,n}$ delle splines lineari.
Il passaggio a questo spazio implica un aumento dal punto di vista del costo computazionale, ma viene ripagato con ordini di convergenza doppi rispetto al caso con operatori di proiezione ortogonale.

8.1 Esempio 1

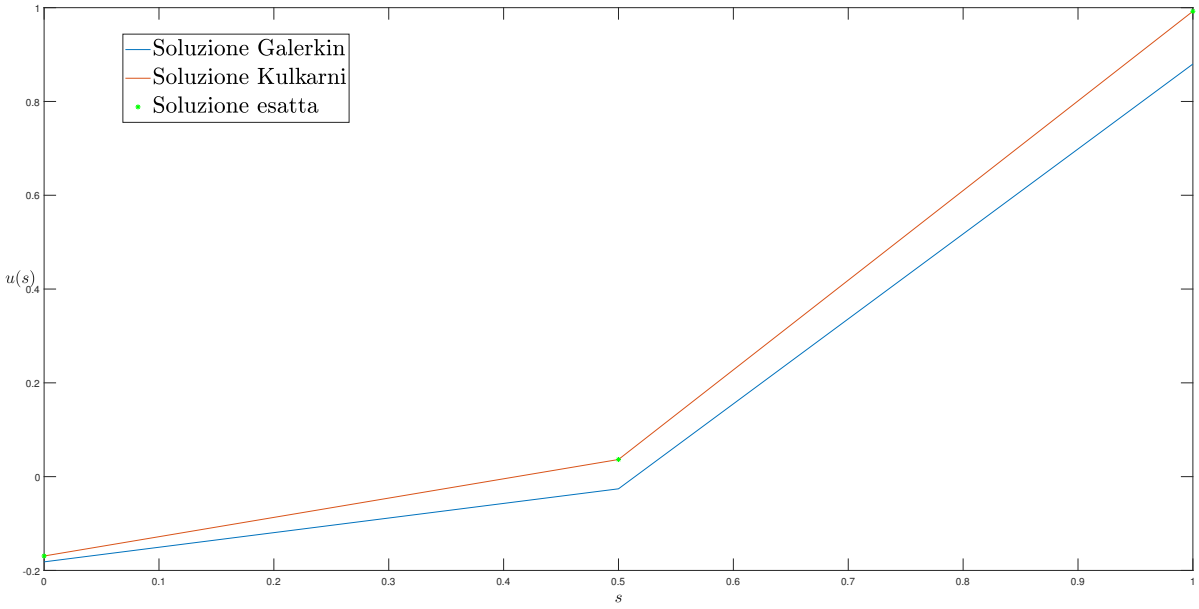
Cerchiamo le soluzioni approssimate della seguente

$$u(s) - \int_0^1 (s-t)u(t)dt = s^3$$

che ha la seguente soluzione

$$u(s) = s^3 + \frac{21}{130}s - \frac{11}{65}$$

Otteniamo dunque le seguenti soluzioni approssimate



La tabella degli errori pare dare risultati non coerenti con quelli del grafico, infatti

N	Errore Galerkin	Errore Kulkarni
2	1.1250e−01	2.1213e−12
4	2.9464e−02	2.4469e−12

Notiamo immediatamente che l'errore per il metodo di Kulkarni è dell'ordine di 10^{-12} già per $n = 2$ e non migliora all'aumentare del numero di nodi. Questo fenomeno è dovuto alle limitazioni del calcolatore. Poiché la soluzione u_n è data da un contributo dovuto a f e da una combinazione degli elementi della base questo fa sì che già per $n = 2$ la soluzione

di Kulkarni vada a coincidere con quella esatta. L'errore non cala perché i contributi sono dell'ordine di 10^{-16}, inferiori al valore dell'*eps* di macchina, ovvero il più piccolo numero rappresentabile in aritmetica di macchina.
Lo stesso ragionamento non è valido per il metodo di Galërkin in quanto non è presente il contributo f, ma $\pi_n f$ che genera un errore dovuto alla proiezione.

8.2 Esempio 2

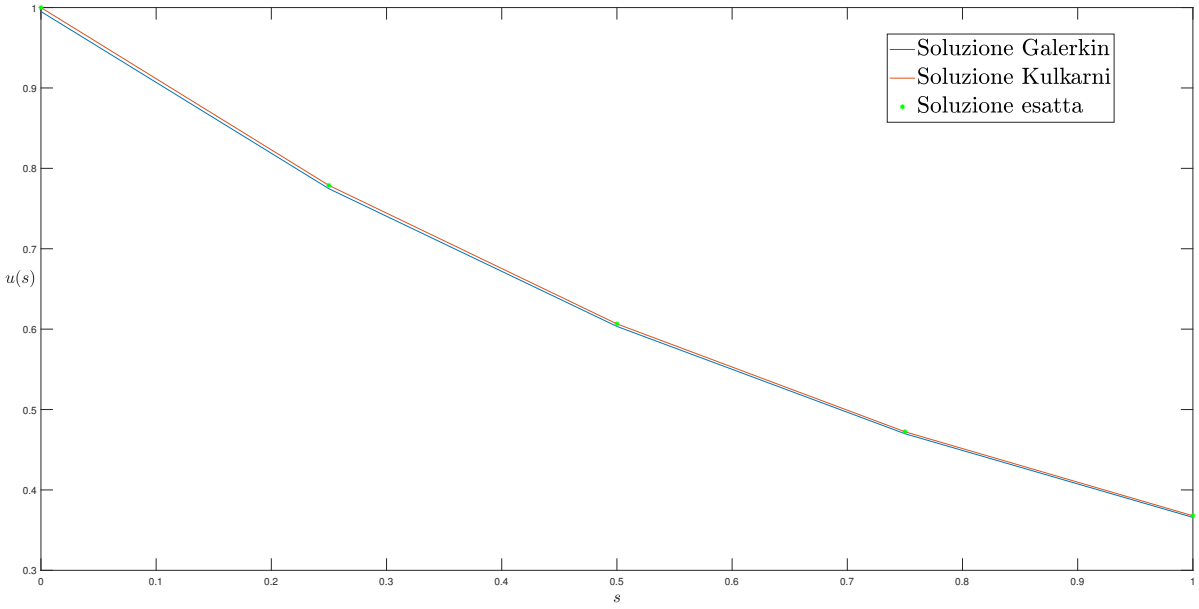
Consideriamo l'Esempio 1 della sezione precedente e confrontiamo i risultati

$$u(s) - \int_0^1 \frac{1}{2}(s+1)e^{-st} \cdot u(t)dt = e^{-s} + \frac{1}{2}(e^{-s-1} - 1)$$

con soluzione

$$u(s) = e^{-s}$$

Le soluzioni approssimate su una base di splines lineari sono le seguenti



e la relativa tabella

N	Errore Galerkin	Errore Kulkarni	Ordine Galerkin	Ordine Kulkarni
2	1.8432e−02	2.0108e−06	—	—
4	4.9009e−03	3.0513e−08	1.9111e+00	6.0422e+00
8	1.2639e−03	4.7732e−10	1.9551e+00	5.9983e+00
16	3.2079e−04	2.0094e−11	1.9782e+00	4.5701e+00

E' evidente che le soluzioni in questo caso sono ben più precise di quelle ottenute nella sezione precedente.
Notiamo che vengono nuovamente rispettate le stime attese per gli errori e per gli ordini di convergenza.

8.3 Esempio 3

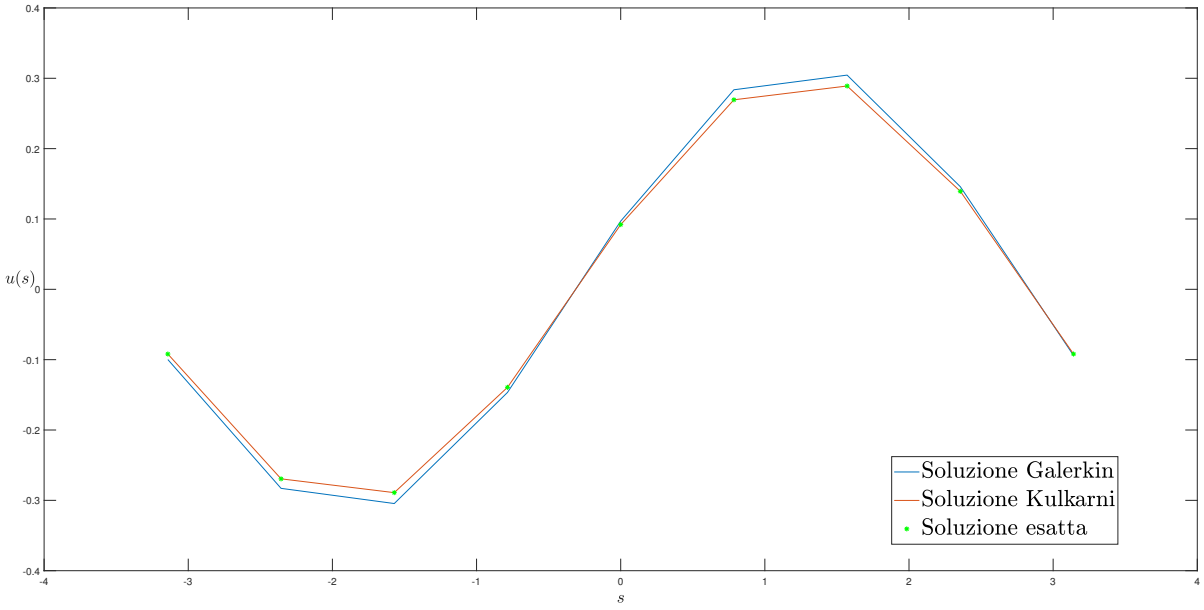
Prendiamo in analisi come ultimo esempio la seguente equazione, definita sull'intervallo $[-\pi, \pi]$

$$u(s) - \int_{-\pi}^{pi} sin(s-t)u(t)dt = cos(s)$$

la cui soluzione esatta è

$$\frac{\pi sin(s) + cos(s)}{1+\pi^2}$$

Otteniamo dunque le seguenti approssimazioni della soluzione



La tabella degli errori è la seguente

N	h	Errore Galerkin	Errore Kulkarni	Ordine Galerkin	Ordine Kulkarni
2	3.1416e+00	5.1155e−01	9.4879e−02	—	—
4	1.5708e+00	5.8484e−02	2.8243e−03	3.1288e+00	5.0701e+00
8	7.8540e−01	1.5496e−02	3.0450e−05	1.9162e+00	6.5353e+00
16	3.9270e−01	3.9129e−03	4.3334e−07	1.9856e+00	6.1348e+00

Notiamo che gli errori questa sono dell'ordine di 10^{-1}, ma ciò è dovuta all'ampiezza dell'intervallo e quindi di conseguenza da h che risulta essere dell'ordine dell'unità anche per $n = 4$.

9 Conclusioni

Abbiamo potuto osservare dunque come il metodo di Kulkarni fornisca un miglioramento importante per la risoluzione di equazioni integrali di Fredholm di seconda specie. Entrambi i metodi richiedono la risoluzione di un sistema lineare di pari dimensione. Nonostante ciò, il metodo di Kulkarni risulta avere un costo computazionale maggiore dovuto al calcolo di $< T^2(\varphi_j), \varphi_i >$ e di $< T(f), \varphi_i >$.
Questo però viene ripagato con una precisione maggiore e con un ordine di convergenza **triplo** rispetto al metodo di Galërkin.
E' stato inoltre mostrato che gli algoritmi sono efficaci su intervalli del tipo $[a.b]$ senza alcuna particolare limitazione.
E' opportuno sottolineare che le stime teoriche effettuate nelle sezioni precedenti erano validi per n sufficientemente grandi. Questo ragionamento viene a perdersi in aritmetica di macchina, emerso anche dagli esempi. Dunque aumentare in maniera non oculata n non garantisce la convergenza delle soluzioni fornite dai due algoritmi.

10 Appendice

Si forniscono in questa sezione i codici Matlab utilizzati nella presente tesi

Programma per risoluzione con metodo di Galërkin per splines costanti

```
function [u] = Galerkin_T_costanti(x_i,x_f,N,k,f)
%Metodo di Galerkin per risoluzione di equazioni integrale di seconda specie
    %Metodo di Galerkin per risoluzione di equazioni integrale di seconda
    %specie (di Fredholm){u-T(u)=f} su un intervallo [x_i,x_f] con k(x,y)
    %nucleo dell'operatore integrale T
    %Si sfruttano risultati analitici e il che la proiezione sia
    %ortogonale, che ci permette di dire che la matrice dei pesi e'
    %diagonale di costante h, con inversa a sua volta diagonale di
    %costante 1/h
%INPUT:     -x_i,x_f estremi
%           -k nucleo di T
%           -f
%           -N numero di elementi della base
%OUTPUT:    -u soluzione approssimata

%Inizializziamo il vettore x per la creazione della base di splines
%costanti
x=linspace(x_i,x_f,N+1);
%Definiamo il passo della mesh
h=(x_f-x_i)/N;
%Inizializzazione del vettore g, che rappresenta la proiezione di T(P_n(f))
g=zeros(1,N);
for i=1:N
    for j=1:N
        g(i)=g(i)+1/h^2*integral(f,x(j),x(j+1))*integral2(k,x(i),x(i+1),x(j),x(j+1));
    end
end
%Matrice B le cui colonne sono la proiezione di T(v_j)
B=zeros(N,N);
for i=1:N
    for j=1:N
        B(i,j)=1/h*integral2(k,x(i),x(i+1),x(j),x(j+1));
    end
end
%Risoluzione del sistema lineare associato al metodo di Galerkin
X=(eye(size(B))-B)\g';
%Inizializzazione della soluzione
u=zeros(1,N+1);
for i=1:N
    %La soluzione u e' data dalla proiezione di f a cui si somma il vettore
    %la i-esima spline per il coefficiente opportuno dato da X
    u(i)=1/h*integral(f,x(i),x(i+1))+X(i);
end
%Definizione dell'ultimo nodo per rispettare le dimensioni di partenza
u(N+1)=u(N);
%Conclusione del programma
end
```

Programma per risoluzione con metodo di Kulkarni per splines costanti

```
function [u] = Kulkarni_T_costanti(x_i,x_f,N,k,f)
%Metodo di Kulkarni per risoluzione di equazioni integrale di seconda specie
    %Metodo di Galerkin per risoluzione di equazioni integrale di seconda
    %specie (di Fredholm){u-T(u)=f} su un intervallo [x_i,x_f] con k(x,y)
    %nucleo dell'operatore integrale T

%INPUT:     -x_i,x_f estremi
%           -k nucleo di T
%           -f
%           -N numero di elementi della base
%OUTPUT:    -u soluzione approssimata

%Inizializziamo il vettore x per la creazione della base di splines lineari
x=linspace(x_i,x_f,N+1);
%Definiamo il passo della mesh
h=(x_f-x_i)/N;
%Definiamo la matrice M dei pesi e la sua inversa M^-1
M=eye(N,N)*h;
M_1=eye(N,N)*1/h;
%Inizializzazione delle matrici A e C che rappresentano rispettivamente
%<T(v_j),v_i> e <T^2(v_j),v_i>
A=zeros(N,N);
C=zeros(N,N);
%Inizializzazione dei vettori b e g, che rispettivamente rappresentano
%<f,v_i> e <T(f),v_i>
b=zeros(1,N);
g=zeros(1,N);
%Calcolo di A,B, b e g
for l=1:N
    for j=1:N
        A(l,j)=integral2(@(s,t) k(s,t),x(l),x(l+1),x(j),x(j+1));
        C(l,j)=integral3(@(s,t,v) k(s,t).*k(t,v),x(l),x(l+1),x_i,x_f,x(j),x(j+1));
    end
    b(l)=integral(@(s) f(s),x(l),x(l+1));
    g(l)=integral2(@(s,t) k(s,t).*f(t),x(l),x(l+1),x_i,x_f);
end
%Risoluzione del sistema lineare relativo per trovare w_n
X=(M*eye(size(A))-A-C+M*(M_1*A)^2)\(b'+g'-A*M_1*b');
%Inizializzazione vettori di supporto per il calcolo di u a partire da
%omega_n. Y indica il termine P_n(T(w_n)), b_1 il termine P_n(f)
Y=M_1*A*X;
b_1=M_1*b';
%Inizializzazione di u
u=zeros(1,N+1);
for l=1:N
    %Introduciamo una variabile di appoggio per calcolare T(w_n)
    sum=0;
    for j=1:N
        %E' necessario calcolare il valore di T(w_n) per in ogni nodo della
        %mesh
        sum=sum+X(j).*integral(@(t) k(x(l),t), x(j),x(j+1));
    end
    %Soluzione approssimata, scritta come somma della sua proiezione su X_n
    %e del suo ortogonale
    u(l)=f(x(l))+X(l)+sum-Y(l)-b_1(l);
end
%E' necessario un ulteriore ciclo per il calcolo del valore di u
%nell'ultimo nodo
sum=0;
for j=1:N
    sum=sum+X(j).*integral(@(t) k(x(N+1),t), x(j),x(j+1));
end
%Definizione dell'ultimo nodo per rispettare le dimensioni di partenza
u(N+1)=f(x(N+1))+X(N)+sum-b_1(N)-Y(N);
%Conclusione del programma
end
```

Programma per risoluzione con metodo di Galërkin per splines lineari

```
function [u] = Galerkin_T_lineari(x_i,x_f,N,k,f)
%Metodo di Galerkin per risoluzione di equazioni integrale di seconda specie
    %Metodo di Galerkin per risoluzione di equazioni integrale di seconda
    %specie (di Fredholm){u-T(u)=f} su un intervallo [x_i,x_f] con k(x,y)
    %nucleo dell'operatore integrale T

%INPUT:     -x_i,x_f estremi
%           -k nucleo di T
%           -f
%           -N numero di elementi della base
%OUTPUT:    -u soluzione approssimata

%Inizializziamo il vettore x per la creazione della base di splines lineari
x=linspace(x_i,x_f,N+1);
%Inizializzazione delle splines lineari, definite come function handles
Base=Splines_Lineari(x);
%Definizione di un vettore di nodi ausiliario z, estensione di x, per
%permettere un calcolo piu' agevole
z=[x(1) x x(N+1)];
%Inizializzazione matrice dei pesi
Alfa_1=zeros(N+1,N+1);
for i=1:N+1
    for j=1:N+1
        Alfa_1(i,j)=integral(@(s)Base{i}(s).*Base{j}(s),max(z(i),z(j)),min(z(i+2),z(j+2)));
    end
end
%Inizializzazione dell'inversa della matrice dei pesi
Alfa=inv(Alfa_1);
%Inizializzazione del vettore g, che rappresenta la proiezione di T(P_n(f))
g=zeros(1,N+1);
for l=1:N+1
    for h=1:N+1
        for i=1:N+1
            for j=1:N+1
               g(l)=g(l)+Alfa(h,l)*Alfa(i,j)*integral(@(s)f(s).*Base{i}(s),z(i),z(i+2))*integral2(@(v,r) k(v,r)
                    .*Base{j}(r).*Base{h}(v),z(h),z(h+2),z(j),z(j+2));
            end
        end
    end
end
%Matrice B le cui colonne sono la proiezione di T(v_j)
B=zeros(N+1,N+1);
for i=1:N+1
    for j=1:N+1
            for l=1:N+1
                B(i,j)=B(i,j)+Alfa(l,i)*integral2(@(s,t) k(s,t).*Base{j}(t).*Base{l}(s), z(l),z(l+2),z(j),z(j
                     +2));
            end
    end
end
%Risoluzione del sistema lineare associato
X=(eye(size(B))-B)\g';
%Calcolo della soluzione $u_n^G$
u=zeros(1,N+1);
for i=1:N+1
    for j=1:N+1
        %Sommiamo a $u_n^G$ il primo elemento che la compone: $\pi_n f$
        u=u+Alfa(i,j)*integral(@(s) f(s).*Base{i}(s), z(i),z(i+2)).*Base{j}(x);
    end
end
%Aggiungiamo a $u_n^G$ il secondo elemento: una combinazione di splines
%lineari
for i=1:N+1
    u=u+X(i).*Base{i}(x);
end
%Conclusione del programma
end
```

Programma per risoluzione con metodo di Kulkarni per splines lineari

```
function [u] = Kulkarni_T_lineari(x_i,x_f,N,k,f)
%Metodo di Kulkarni per risoluzione di equazioni integrale di seconda specie
    %Metodo di Kulkarni per risoluzione di equazioni integrale di seconda
    %specie (di Fredholm){u-T(u)=f} su un intervallo [x_i,x_f] con k(x,y)
    %nucleo dell'operatore integrale T

%INPUT:     -x_i,x_f estremi
%           -k nucleo di T
%           -f
%           -N numero di elementi della base
%OUTPUT:    -u soluzione approssimata

%Inizializziamo il vettore x per la creazione della base di splines lineari
x=linspace(x_i,x_f,N+1);
%Inizializzazione delle splines lineari, definite come function handles
Base=Splines_Lineari(x);
%Definizione di un vettore di nodi ausiliario z, estensione di x, per
%permettere un calcolo piu' agevole
z=[x(1) x x(end)];
%Inizializzazione matrice dei pesi
M=zeros(N+1,N+1);
for i=1:N+1
    for j=1:N+1
        M(i,j)=integral(@(s)Base{i}(s).*Base{j}(s), x_i,x_f);
    end
end
%Inizializzazione dell'inversa della matrice dei pesi
M_1=inv(M);
%Inizializzazione delle matrici A e C che rappresentano rispettivamente
%<T(v_j),v_i> e <T^2(v_j),v_i>
A=zeros(N+1,N+1);
C=zeros(N+1,N+1);
%Inizializzazione dei vettori b e g, che rispettivamente rappresentano
%<f,v_i> e <T(f),v_i>
b=zeros(1,N+1);
g=zeros(1,N+1);
%Calcolo di A,B, b e g
for i=1:N+1
    for j=1:N+1
        A(i,j)=integral2(@(s,t) k(s,t).*Base{j}(t).*Base{i}(s),z(i),z(i+2),z(j),z(j+2));
        C(i,j)=integral3(@(s,t,v) k(s,t).*k(t,v).*Base{j}(v).*Base{i}(s),z(i),z(i+2),x_i,x_f,z(j),z(j+2));
    end
    b(i)=integral(@(s) f(s).*Base{i}(s),z(i),z(i+2));
    g(i)=integral2(@(s,t) k(s,t).*f(t).*Base{i}(s),z(i),z(i+2),x_i,x_f);
end
%Risoluzione del sistema lineare relativo per trovare w_n
X=(M*eye(size(A))-A-C+M*(M_1*A)^2)\(b'+g'-A*M_1*b');
%Inizializzazione vettori di supporto per il calcolo di u a partire da
%omega_n. Y indica il termine P_n(T(w_n)), b_1 il termine P_n(f)
Y=M_1*A*X;
b_1=M_1*b';
%Inizializzazione di u
u=zeros(1,N+1);
for l=1:N+1
    %Introduciamo una variabile di appoggio per calcolare T(w_n)
    sum=0;

    for j=1:N+1
        %E' necessario calcolare il valore di T(w_n) per in ogni nodo della
        %mesh
        sum=sum+X(j).*integral(@(t) k(x(l),t).*Base{j}(t),z(j),z(j+2));
    end
    %Soluzione approssimata, scritta come somma della sua proiezione su X_n
    %e del suo ortogonale
    u(l)=f(x(l))+X(l)+sum-Y(l)-b_1(l);
end
%Conclusione del programma
end
```

Sottoprogramma per la creazione della base di splines

```
function [f]=Splines_Lineari(x)
%Creazione di splines lineari sui nodi forniti da x
    %Creazione di splines lineari sui nodi forniti da x
%INPUT:     -x vettore dei nodi

%OUTPUT:    -f cell array con function handle corrispondenti alle splines
%            lineari

%Dichiarazione N, numero di elementi di x
N=length(x);
%Dichiarazione prima splines lineare
f{1}=@(s) (x(2)-s)/(x(2)-x(1)).*((x(1)<=s & s<x(2)));
%Dichiarazione splines intermedie
for i=2:N-1
    f{i}=@(s) (s-x(i-1))/(x(i)-x(i-1)).*(x(i-1)<=s & s<x(i))+(x(i+1)-s)/(x(i+1)-x(i)).*((x(i)<=s &s<x(i+1)));
end
%Dichiarazione splines finale
f{N}=@(s) (s-x(N-1))/(x(N)-x(N-1)).*(x(N-1)<=s &s<=x(N));
end
```

Sottorogramma per il confronto delle soluzioni

```
%Dichiarazione dell'intervallo [a,b]
a=0; b=1;
%Definizione del nucleo k(s,t) dell'operatore T
k=@(s,t) s-t
%Definizione di f(s)
f=@(s) s.^3
%Soluzione esatta
sol=@(s) s.^3+21/130.*s-11/65

format short e;

%Inizializzazione di N, numero di nodi
N=16;
%Definizione del vettore dei nodi
x=linspace(a,b,N+1);
%Soluzione con metodo di Galerkin
u_g=Galerkin_T_costanti(a,b,N,k,f);
%Soluzione con metodo di Kulkarni
u_k=Kulkarni_T_costanti(a,b,N(1),k,f);
%Soluzione esatta calcolata su x
soluzione=sol(x);
%Grafico di confronto
plot(x(1:end-1),u_g(1:end-1),x(1:end-1),u_k(1:end-1),x(1:end-1),soluzione(1:end-1),'* g')
ax=gca;
ax.FontSize = 12;
legend('Soluzione Galerkin', 'Soluzione Kulkarni', 'Soluzione esatta','interpreter','latex','location','best',
     'fontSize', 28)
xlabel('$s$','interpreter','latex','fontsize',20)
ylabel('$u(s)$','interpreter','latex','fontsize',20,'rotation',0)
```

Sottorogramma per tabelle di confronto degli errori e degli ordini delle soluzioni

```
%Dichiarazione dell'intervallo [a,b]
a=0; b=1;
%Definizione del nucleo k(s,t) dell'operatore T
k=@(s,t) s-t
%Definizione di f(s)
f=@(s) s.^3
%Soluzione esatta
sol=@(s) s.^3+21/130.*s-11/65

format short e;
%Numero di iterate
M=5;
%Inizializzazione del vettore del numero di intervalli N
N=zeros(1,M);
%Inizializzazione vettore dei passi della mesh h
h=linspace(1,M);
%Definizione del primo elemento di N
N(1)=2;
%Calcolo del primo elemento di h
h=(b-a)/N(1);
%Definizione del vettore dei nodi
x=linspace(a,b,N(1)+1);
%Soluzione con metodo di Galerkin
u_g=Galerkin_T_costanti(a,b,N(1),k,f);
%Soluzione con metodo di Kulkarni
u_k=Kulkarni_T_costanti(a,b,N(1),k,f);
%Soluzione esatta calcolata su x
soluzione=sol(x);
%Definizione degli errori al 1 passo
errore_g=zeros(1,M);
errore_k=zeros(1,M);
%Errore Galerkin (escludiamo l'ultimo nodo per via della struttura delle
%splines)
errore_g(1)=max(abs(u_g(1:end-1)-soluzione(1:end-1)));
%Errore Kulkarni (escludiamo l'ultimo nodo per via della struttura delle
%splines)
errore_k(1)=max(abs(u_k(1:end-1)-soluzione(1:end-1)));
%Calcolo degli errori al raddoppiare di N
for i=2:M
    %Raddoppiamo N
    N(i)=2*N(i-1);
    %Aggiornamento di h
    h(i)=(b-a)/(N(i));
    %Aggiornato il vettore dei nodi
    x=linspace(a,b,N(i)+1);
    %Calcolo delle due soluzioni approssimate e di quella esatta
    u_g=Galerkin_T_costanti(a,b,N(i),k,f);
    u_k=Kulkarni_T_costanti(a,b,N(i),k,f);
    soluzione=sol(x);
    %Calcolo degli errori all'i-esimo passo
    errore_g(i)=max(abs(u_g(1:end-1)-soluzione(1:end-1)));
    errore_k(i)=max(abs(u_k(1:end-1)-soluzione(1:end-1)));
end
%Confrontiamo i due errori e i relativi ordini
Errore=table(int16(N)',h', errore_g', errore_k',[0,log((errore_g(1:end-1)./errore_g(2:end)))./log(2)]', [0,(log
     ((errore_k(1:end-1)./errore_k(2:end)))./log(2))]');
Errore.Properties.VariableNames={'N','h', 'Errore Galerkin','Errore Kulkarni', 'Ordine Galerkin', 'Ordine
     Kulkarni'};
Errore
```

Riferimenti bibliografici

[1] G.A. Chandler, *Superconvergence of numerical solutions of second kind integral equations*, Ph.D. Thesis (Australian National University, ACT, Australia, 1979).

[2] F. Chatelin, *Spectral approximation of linear operators* (Academic Press, New York, 1983).

[3] F. Chatelin and R. Lebbar, 'The iterated projection solution for the Fredholm integral equation of second kind', *J. Austral. Math. Soc. Ser. B* **22** (1981), 439-451.

[4] C. de Boor, 'A bound on the L_∞ norm of L_2-approximation by splines in terms of a global mesh ratio', *Maths. Comput.* **30** (1976), 765-771.

[5] C. de Boor and B. Swartz, 'Collocation at Gaussin points', *SIAM J. Numer. Anal.* **10** (1973), 582-606.

[6] J. Douglas, Jr., T. Dupont and L. Wahlbin, 'Optimal L_∞ error estimates for Galerkin approximations to solutions of two point boundary value problems', *Math. Comp.* **29** (1975), 475-483.

[7] I.G. Graham, S. Joe and I.H. Sloan, 'Iterated Galerkin versus iterated collocation for integral equations of the second kind', *IMA J. Numer. Anal.* **5** (1985), 355-369.

[8] Q. Hu, 'Interpolation correction for collocation solutions of Fredholm integro-differential equations', *Math. Comp.* **67** (1998), 987-999.

[9] R.P. Kulkarni, 'A New Superconvergent projection method for approximate solutions of eigenvalue problems', *Numer. Funct. Anal. Optim.* **24** (2003), 75-84.

[10] Q. Lin, S. Zhang and N. Yan, 'An acceleration method for integral equations by using interpolation post-processing', *Adv. Comput. Math.* **9** (1998), 117-129.

[11] G.R. Richter, 'Superconvergence of piecewise polynomial Galerkin approximations for Fredholm integral equations of the second kind', *Num. Math.* **31** (1978), 63-70.

[12] E. Schock, 'Galerkin like methods for equations of the second kind', *J. Integral Equations Appl.* **4** (1982), 361-364.

[13] I.H. Sloan, 'Improvement by iteration for compact operator equations', *Math. Comp.* **30** (1976), 758-764.

[14] I.H. Sloan, 'Four variants of the Gaterkin method for Integral equations of the second kind', *IMA J. Numer. Anal.* **4** (1984), 9-17.

[15] A. Spence and K.S. Thomas, 'On superconvergence properties of Galerkin's method for compact operator equations', *IMA J. Numer. Anal.* **3** (1983), 253-271.

[16] Rekha P. Kulkarni (2003). 'A superconvergence result for solutions of compact operator equations', *Bulletin of the Australian Mathematical Society*, 68, (2003) 517-528, doi:10.1017/S0004972700037916

[17] H. Brezis, 'Functional Analysis, Sobolev Spaces and Partial Differential Equations', Springer (2011)