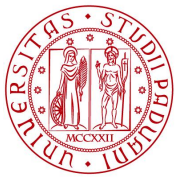


UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI MATEMATICA "TULLIO LEVI-CIVITA"

**Corso di Laurea Triennale in Matematica**

**Tesi di Laurea**

# Iperinterpolazione ibrida su domini multivariati


**RELATORE:**

**Prof. Alvise Sommariva**

**LAUREANDA:**

**Anna Duse**
**1189659**


**Anno Accademico 2022/2023**

**15 dicembre 2023**

# Indice

# Introduzione

In questo lavoro si discutono alcuni tipi di iperinterpolazione su domini $\Omega$ di tipo poligonale.

L'iperinterpolazione è un concetto introdotto nel 1995 da Ian Sloan, in cui si intende approssimare una funzione $f \in C(\Omega)$ tramite un polinomio $p_L \in \mathbb{P}_L$ mediante minimi-quadrati discreti, costruiti con una formula di quadratura con grado di precisione $2L$. Nel valutare i coefficienti di Fourier di $f$, rispetto a una base ortonormale di $\mathbb{P}_L$, l'autore considera una formula a pesi positivi e nodi interni. Inoltre, mostrando che con questa scelta si ha la convergenza in norma 2, sottolinea come talora l'iperinterpolante coincida con una interpolante.

Uno degli aspetti caratteristici è la necessità di una base polinomiale ortonormale, che per molti classici domini quale intervallo, disco, quadrato, triangolo, cubo, sfera, è nota esplicitamente.

Nei domini trattati in questa tesi la questione risulta più complessa in quanto tanto le formule di cubatura quanto le basi ortonormali vengono calcolate numericamente.

Nei primi due approcci introdotti (classico e filtrato), le iperinterpolanti a grado $N$ risultano proiezioni su spazi polinomiali, rispettivamente di polinomi di grado totale $L$ e $\lfloor \frac{L}{2} \rfloor$. Questo non avviene per l'iperinterpolante di Lasso e ibrido.

Nonostante questo, il vantaggio di quest'ultimo risulta evidente qualora i campionamenti della funzione $f$ nei nodi di cubatura siano soggetti a un rumore non trascurabile. Esperimenti numerici sottolineano tutto ciò con test in classici domini come quelli sopra menzionati.

In questo lavoro consideriamo l'applicazione di varie iperinterpolanti, come la classica, la filtrata, la Lasso e la ibrida, a un dominio convesso, concavo e non semplice.

Come anticipato, in mancanza di una esplicita base triangolare ed ortonormale, questa viene calcolata numericamente.

Per quanto concerne la cubatura numerica, necessaria al calcolo dell'iperinterpolante, utilizzamo l'approccio suggerito in [3] (senza compressione).

Nella sezione numerica, abbiamo verificato numericamente le proprietà di proiezione sullo spazio dei polinomi di grado $L$ di quella classica e filtrata (utilizzando una formula di cubatura con grado di precisione $2L$) e successivamente mostrato che in assenza di rumore è da preferirsi l'iperinterpolazione classica mentre, in presenza di rumore gaussiano o impulsivo, quella di tipo ibrida è un ottimo compromesso tra sparsità dei coefficienti del polinomio iperinterpolatore e ricostruzione della funzione.

Gli esperimenti numerici sono stati eseguiti in Matlab e resi disponibili *open-source* sulla piattaforma *GITHUB* in [6].

# Capitolo 1

# Iperinterpolazione

Il termine *iperinterpolazione* è stato introdotto dal matematico australiano Ian H. Sloan nel 1995 nel [4]. L'iperinterpolazione è un metodo di approssimazione polinomiale per funzioni continue su insiemi compatti o varietà. In particolare, usa una opportuna formula di quadratura (ad alto grado di precisione) per approssimare una espansione in serie di Fourier troncata (più precisamente i suoi coefficienti) in una serie di polinomi ortogonali per qualche misura in un dato dominio multivariato. Per darne una definizione rigorosa, bisogna prima definire dei concetti relativi alla teoria dell'approssimazione.

## 1.1 Nozioni Preliminari

**Definizione 1.1.** *(Misura finita di un insieme compatto)*
*Sia $\Omega \subset \mathbb{R}^n$, $n \geq 1$ un insieme compatto. Quest'ultimo ha misura (positiva) finita $V$ rispetto ad una data misura $d\omega$ se:*

$$\int_\Omega d\omega = V < +\infty \tag{1.1}$$

Indicando con $C(\Omega)$ lo spazio delle funzioni continue su $\Omega$ e $L^2(\Omega)$ lo spazio delle funzioni quadrato-integrabili sempre su $\Omega$, possiamo definire su $L^2(\Omega)$ il prodotto interno.

**Definizione 1.2.** *(Prodotto interno)*
*Siano dati $\Omega$ un insieme compatto e $d\omega$ misura. Il prodotto interno, rispetto alla misura $d\omega$, tra le due funzioni $f$ e $g$ è definito da:*

$$\langle f, g \rangle = \int_\Omega f g \, d\omega \qquad \forall f, g \in C(\Omega) \tag{1.2}$$

*Il prodotto interno induce la norma* $\|f\|_2 := \langle f, f\rangle^{\frac{1}{2}}$.

Indicando con $\mathbb{P}_{\mathrm{L}}(\Omega)$ lo spazio dei polinomi di grado minore non strettamente di L ristretto a $\Omega$, la cui dimensione è $d = dim(\mathbb{P}_{\mathrm{L}}(\Omega))$, si nota che $\mathbb{P}_{\mathrm{L}}(\Omega) \subset L^2(\Omega)$.

Di seguito verranno enunciate una serie di definizioni: base ortonormale e proiezione ortogonale di $L^2(\Omega)$, formula di quadratura e il suo grado di precisione, il prodotto interno e il prodotto interno discreto.

**Definizione 1.3.** *(Base ortonormale)*
*Una base* $\{\phi_i\}_{i=1,\dots,d}$ *di* $\mathbb{P}_L(\Omega)$ *si dice ortonormale se:*

$$\langle \phi_i, \phi_j \rangle = \delta_{ij} \qquad 1 \leq i, j \leq d \tag{1.3}$$

*con* $\delta_{ij}$ *delta di Kronecker.*

**Definizione 1.4.** *(Proiezione ortogonale)*
*La proiezione ortogonale* $\mathcal{T}_L : L^2(\Omega) \to \mathbb{P}_L(\Omega)$ *che prende una funzione* $f \in L^2(\Omega)$ *e la proietta nello spazio dei polinomi è definita (unicamente) da:*

$$\mathcal{T}_L f := \sum_{i=1}^{d} \langle f, \phi_i \rangle \, \phi_i \tag{1.4}$$

*con* $\langle f, \phi_i \rangle = \int_\Omega f \phi_i d\omega$ *coefficiente i-esimo di Fourier.*

Per valutare numericamente il prodotto scalare presente nel calcolo dei coefficienti di Fourier (presenti nella proiezione di f), è fondamentale definire:

**Definizione 1.5.** *(Formula di quadratura)*
*Una formula di quadratura a N punti* $\{x_1, \dots, x_N\} =: \mathbf{X}_N \in \Omega$ *è*

$$\sum_{i=1}^{N} w_i f(\mathbf{x}_i) \approx \int_\Omega f \, d\omega \qquad \forall f \in C(\Omega) \tag{1.5}$$

*in cui* $\{x_1, \dots, x_N\}$ *sono i nodi e* $\{w_1, \dots, w_N\}$ *i rispettivi nodi.*

*Osservazione.* Si parla di *formula di cubatura* quando l'obiettivo è quello di approssimare integrali doppi.

**Definizione 1.6.** *(Grado di precisione di una formula di quadratura)*
*Data una formula di quadratura come (1.5), questa ha grado di precisione* $\delta$ *se*

$$\sum_{i=1}^{N} w_i \, p(\mathbf{x}_i) = \int_\Omega p \, d\omega \qquad \forall p \in \mathbb{P}_\delta(\Omega). \tag{1.6}$$

In altre parole, una formula di quadratura ha grado di precisione $\delta$ almeno $N-1$ *se e solo se* è esatta per ogni polinomio di grado inferiore o uguale a $N-1$.

**Definizione 1.7.** *(Prodotto interno discreto)*
*Il prodotto interno discreto su $C(\Omega)$ è definito nel seguente modo:*

$$\langle f, g\rangle_{\mathrm{N}} := \sum_{i=1}^{N} w_i \, f(\mathbf{x}_i)\, g(\mathbf{x}_i) \qquad \forall f, g \in C(\Omega) \tag{1.7}$$

Notiamo che il prodotto interno e il prodotto interno discreto sono soggetti alle seguenti considerazioni dovute alla formula di quadratura (1.5):

- si ha che
$$\boxed{\langle f, g\rangle_N \approx \langle f, g\rangle} \qquad \forall f, g \in C(\Omega).$$
Cioè l'approssimazione tra i due prodotti vale sempre poiché se
$$f, g \in C(\Omega) \Rightarrow fg \in C(\Omega).$$

- si ha che
$$\boxed{\langle p, q\rangle_N = \langle p, q\rangle} \qquad \forall p, q \in \mathbb{P}_L(\Omega).$$
Cioè si ha l'uguaglianza quando le due funzioni sono polinomi di grado L poiché
$$p, q \in \mathbb{P}_L(\Omega) \Rightarrow pq \in \mathbb{P}_{2L}(\Omega).$$

Infine si può così definire:

**Definizione 1.8.** *(Operatore di iperinterpolazione)*
*L'operatore di iperinterpolazione $\mathcal{L}_L f$ è la proiezione di $f$ in $\mathbb{P}_L(\Omega)$ ottenuta sostituendo il prodotto interno (1.2) con il prodotto interno discreto (1.7) ed è definito da:*

$$\mathcal{L}_L f := \sum_{i=1}^{d} \langle f, \phi_i\rangle_N \, \phi_i \qquad \forall f \in C(\Omega) \tag{1.8}$$

Si può dimostrare che $\mathcal{L}_L f$ è esatta, cioè $\mathcal{L}_L f = f$, se $f$ è un polinomio di grado $\leq L$. Questo teorema prende il nome di teorema di proiezione.

**Teorema 1.1.** *(Teorema di proiezione)*
*Se $f \in \mathbb{P}_L(\Omega)$ allora $\mathcal{L}_L f = f$.*

*Dimostrazione.* Dall'ipotesi $f \in \mathbb{P}_L(\Omega)$ si ha:

$$f = \sum_{j=1}^{d} a_j \phi_j$$

Per definizione di iperinterpolazione (1.8) si ha che:

$$\begin{aligned}
\mathcal{L}_L f &= \sum_{j=1}^{d} \langle f, \phi_j \rangle_N \, \phi_j = \\
&= \sum_{j=1}^{d} \left\langle \sum_{i=1}^{d} a_i \phi_i, \, \phi_j \right\rangle_N \phi_j = \\
&= \sum_{j=1}^{d} \sum_{i=1}^{d} a_i \langle \phi_i, \phi_j \rangle_N \, \phi_j = \\
&= \sum_{j=1}^{d} a_j \phi_j = f
\end{aligned}$$

□

Altre definizioni utili sono le seguenti:

**Definizione 1.9.** *(Norma infinito)*
*Siano $\Omega$ un dominio e $f \in C(\Omega)$. La norma infinito (o norma uniforme) è definita:*

$$\|f\|_\infty = \sup_{x \in \Omega} |f(x)| \tag{1.9}$$

**Definizione 1.10.** *(Norma $L_2$)*
*Siano $\Omega$ dominio, $f \in L_2(\Omega)$ e $d\omega$ misura come definita in (1.1). La norma $L_2$ si definisce nel seguente modo:*

$$\|f\|_2 = \left( \int_\Omega |f|^2 \, d\omega \right)^{1/2} \tag{1.10}$$

**Definizione 1.11.** *(Polinomio di miglior approssimazione)*
*Dati $\Omega$ dominio e $f \in C(\Omega)$, allora il polinomio di miglior approssimazione di $f$ si indica con $\phi^* \in \mathbb{P}_L$ si definisce:*

$$\mathcal{E}_L(f) := \inf_{\phi \in \mathbb{P}_L} \|f - \phi\|_\infty = \|f - \phi^*\|_\infty \tag{1.11}$$

*con $\mathcal{E}_L(f)$ errore di miglior approssimazione uniforme di $f$.*

## 1.2 Varianti dell'Iperinterpolazione

Nelle seguente sezione si approndiranno tre varianti dell'operatore di interpolazione prese da [1]. Inizialmente si definisce cosa si intende per funzione filtro e per *iperinterpolazione filtrata*.

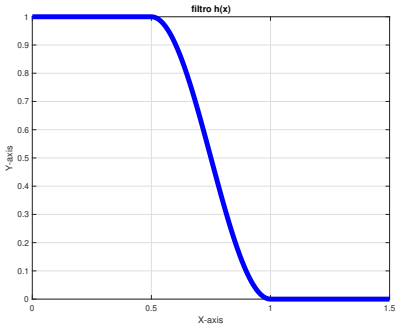
**Definizione 1.12.** *(Filtro)*
*La funzione filtro $h \in C([0, +\infty])$ è una funzione che soddisfa:*

$$h(x) = \begin{cases} 1 & per\ x \in [0, \frac{1}{2}] \\ 0 & per\ x \in [1, \infty) \end{cases}$$

Si osserva subito che si possono definire diversi filtri in base al comportamento di $h$ in $[1/2, 1]$.

In questo lavoro, si considera in particolare:

$$h(x) = \begin{cases} 1 & \text{per } x \in [0, \frac{1}{2}] \\ \sin^2(\pi x) & \text{per } x \in (\frac{1}{2}, 1) \\ 0 & \text{per } x \in [1, \infty) \end{cases} \tag{1.12}$$




Esistono altri esempi della funzione filtro ma tutti hanno in comune le seguenti proprietà:

- sono decrescenti nell'intervallo $(\frac{1}{2}, 1)$;
- maggiore è il grado del filtro, più decadono lentamente nell'intervallo $(\frac{1}{2}, 1)$;
- il supporto dei filtri $supp(h)$ è uguale:

  $$supp(h) = [0, a]$$

  con $a < 1$ e $supp(\cdot)$ indica il supporto della funzione.

Quelli indicati di seguito sono presi dall'articolo di Ian H. Sloan e Robert S. Womersley [5].

1. Funzione lineare:

$$h(x) = \begin{cases} 1 & \text{se } x \in [0, \frac{1}{2}] \\ 2 - 2x & \text{se } x \in (\frac{1}{2}, 1) \\ 0 & \text{se } x \in [1, +\infty] \end{cases}$$

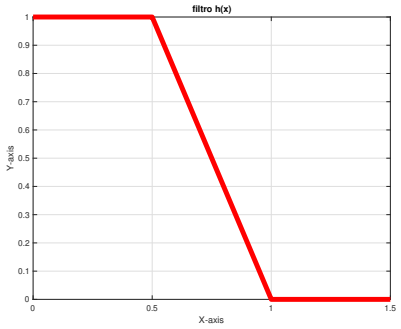



Figura 1.1: Filtro di tipo *funzione lineare*.

2. Funzione cubica:

$$h(x) = \begin{cases} 1 & \text{se } x \in [0, \frac{1}{2}] \\ 16x^3 - 36x^2 + 24x - 4 & \text{se } x \in (\frac{1}{2}, 1) \\ 0 & \text{se } x \in [1, +\infty] \end{cases}$$

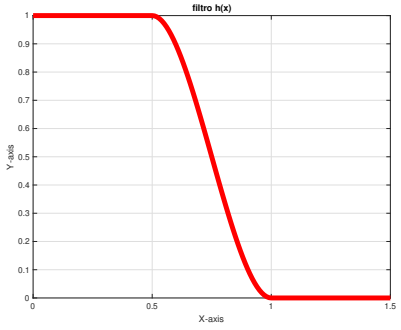



Figura 1.2: Filtro di tipo *funzione cubica*.

3. Funzione quadratica a tratti

$$h(x) = \begin{cases} 1 & \text{se } x \in [0, \frac{1}{2}] \\ -8x^2 + 8x - 1 & \text{se } x \in (\frac{1}{2}, \frac{3}{4}) \\ 8x^2 - 16x + 8 & \in [\frac{3}{4}, 1) \\ 0 & \text{se } x \in [1, +\infty] \end{cases}$$

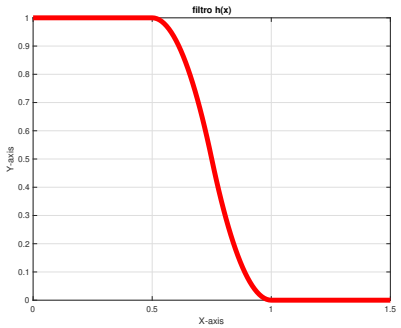



Figura 1.3: Filtro di tipo *funzione quadratica a tratti*.

4. Funzione esponenziale

$$h(x) = \begin{cases} 1 & \text{se } x \in [0, \frac{1}{2}] \\ \exp\left(\frac{2\exp\left(\frac{1}{1-2x}\right)}{x-1}\right) & \text{se } x \in (\frac{1}{2}, 1) \\ 0 & \text{se } x \in [1, +\infty] \end{cases}$$

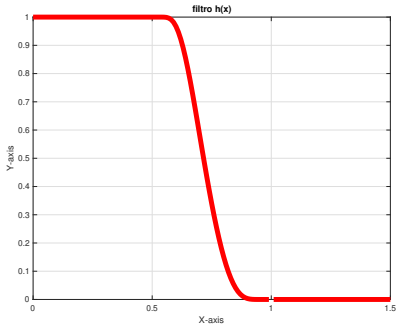



Figura 1.4: Filtro di tipo *funzione esponenziale*.

**Definizione 1.13.** *(Iperinterpolazione filtrata)*
*Sia $\Omega$ un dominio compatto e si supponga che il prodotto scalare discreto $\langle f, g\rangle_N$ definito in (1.7) sia determinato da una formula di quadratura a $N$ punti con grado di precisione $L - 1 + \lfloor L + 2 \rfloor$. L'interpolazione filtrata di $f$ in $\mathbb{P}_L(\Omega)$ è definita da:*

$$\mathcal{F}_{\mathrm{L,N}} f := \sum_{l=1}^{d} h_l \left( \langle f, \phi_l \rangle_N \right) \phi_l \tag{1.13}$$

*con*

- $d = dim(\mathbb{P}_L(\Omega))$;
- $h_l := h\left(\frac{\deg \phi_l}{L}\right)$ *per semplicità di notazione e* $h(\cdot)$ *funzione filtro definita in (1.12).*

L'iperinterpolazione filtrata modifica quindi i coefficienti discreti di Fourier dell'interpolazione classica applicandoci davanti il filtro.

*Osservazione.* Data che si è assundo il grado di precisione della formula di quadratura pari a $L - 1 + \lfloor \frac{L}{2} \rfloor$, si ha che:

$$\mathcal{F}_{L,N} f = f \qquad \forall f \in \mathbb{P}_{\lfloor \frac{L}{2} \rfloor}(\Omega).$$

L'osservazione può essere formalizzata (e dimostrata) nel seguente teorema di proiezione di proezione. È necessario enunciare preliminarmente questa proposizione:

**Proposizione 1.1.** *$\mathcal{F}_{L,N} f$ è un polinomio di grado totale al più $\lceil aL \rceil - 1$.*

**Teorema 1.2.** *(Teorema di proiezione)*
*Se $f \in \mathbb{P}_{\lfloor \frac{L}{2} \rfloor}(\Omega)$ allora $\mathcal{F}_{L,N} f = f$.*

*Dimostrazione.* Si ponga $d_{\lfloor \frac{L}{2} \rfloor} = \dim(\mathbb{P}_{\lfloor \frac{L}{2} \rfloor})$. Per ipotesi si ha che:

$$f = \sum_{j=1}^{d_{\lfloor L/2 \rfloor}} \langle f, \phi_j \rangle \, \phi_j$$

Quindi per definizione di iperinterpolazione filtrata (1.13) segue che:

$$\begin{aligned} \mathcal{F}_{L,N} f &= \sum_{j=1}^{d} h\left(\frac{deg(\phi_j)}{L}\right) \langle f, \phi_j \rangle_N \, \phi_j = \\ &= \sum_{j=1}^{d_{\lfloor L/2 \rfloor}} h\left(\frac{deg(\phi_j)}{L}\right) \langle f, \phi_j \rangle_N \, \phi_j + \sum_{j=d_{\lfloor L/2 \rfloor}+1}^{d} h\left(\frac{deg(\phi_j)}{L}\right) \langle f, \phi_j \rangle_N \, \phi_j \end{aligned}$$

Dato che con $\mathbb{P}_k$ si intende lo spazio dei polinomi di grado $\leq k$ con $d_k = \dim(\mathbb{P}_k)$, allora $\forall\, 1 \leq j \leq d_{\lfloor \frac{L}{2} \rfloor}$ si ha $0 \leq deg(\phi_j) \leq \lfloor \frac{L}{2} \rfloor$.
Quindi $0 \leq \frac{deg(\phi_j)}{L} \leq \frac{1}{2}$. Inoltre dalla condizione per la funzione filtro $h(x) = 1$ per ogni $x \in [0, \frac{1}{2}]$, si ha che $h\big(\frac{deg(\phi_j)}{L}\big) = 1$ per ogni $1 \leq j \leq d_{\lfloor \frac{L}{2} \rfloor}$ e quindi:

$$\mathcal{F}_{L,N} f = \sum_{j=1}^{d_{\lfloor L/2 \rfloor}} \langle f, \phi_j \rangle_N \, \phi_j + \sum_{j=d_{\lfloor L/2 \rfloor}+1}^{d} h\left(\frac{deg(\phi_j)}{L}\right) \langle f, \phi_j \rangle_N \, \phi_j$$

Ora, grazie al fatto che $h$ una funzione non negativa e decrescente con $supp(h) = [0, a]$ allora $\forall j \geq d_{\lceil aL \rceil - 1} + 1$, ovvero tale che $deg(\phi_j) \geq aL$, si ha: $0 \leq h\big(\frac{deg(\phi_j)}{L}\big) \leq h\big(\frac{aL}{L}\big) = h(a) = 0$ e quindi $h\big(\frac{deg(\phi_j)}{L}\big) = 0$. Pertanto si ricava:

$$\begin{aligned}
\sum_{j=d_{\lfloor L/2 \rfloor}+1}^{d} h\left(\frac{deg(\phi_j)}{L}\right) \langle f, \phi_j \rangle_N \, \phi_j &= \sum_{j=d_{\lfloor L/2 \rfloor}+1}^{d_{\lceil aL \rceil - 1}} h\left(\frac{deg(\phi_j)}{L}\right) \langle f, \phi_j \rangle_N \, \phi_j \\
&\quad + \sum_{j=d_{\lceil aL \rceil - 1}+1}^{d} h\left(\frac{deg(\phi_j)}{L}\right) \langle f, \phi_j \rangle_N \, \phi_j = \\
&= \sum_{j=d_{\lfloor L/2 \rfloor}+1}^{d_{\lceil aL \rceil - 1}} h\left(\frac{deg(\phi_j)}{L}\right) \langle f, \phi_j \rangle_N \, \phi_j
\end{aligned}$$

e quindi:

$$\mathcal{F}_{L,N} f = \sum_{j=1}^{d_{\lfloor L/2 \rfloor}} \langle f, \phi_j \rangle_N \, \phi_j + \sum_{j=d_{\lfloor L/2 \rfloor}+1}^{d_{\lceil aL \rceil - 1}} h\left(\frac{deg(\phi_j)}{L}\right) \langle f, \phi_j \rangle_N \, \phi_j$$

Infine per ipotesi il grado di precisione è almeno $\lceil aL \rceil - 1 + \lfloor \frac{L}{2} \rfloor$, allora $\forall j \leq d_{\lceil aL \rceil - 1 + \lfloor L/2 \rfloor}$ si ha $\langle f, \phi_j \rangle_N = \langle f, \phi_j \rangle$.
Inoltre $f \in \mathbb{P}_{\lfloor L/2 \rfloor}(\Omega)$ quindi $\forall j \geq d_{\lfloor L/2 \rfloor} + 1$ si ha $\langle f, \phi_j \rangle = 0$. Pertanto si ottiene:

$$\begin{aligned}
\mathcal{F}_{L,N} f &= \sum_{j=1}^{d_{\lfloor l/2 \rfloor}} \langle f, \phi_j \rangle \, \phi_j + \sum_{j=d_{\lfloor L/2 \rfloor}+1}^{d_{\lceil aL \rceil - 1}} h\left(\frac{deg(\phi_j)}{N}\right) \langle f, \phi_j \rangle \, \phi_j = \\
&= \sum_{j=1}^{d_{\lfloor L/2 \rfloor}} \langle f, \phi_j \rangle \, \phi_j = f
\end{aligned}$$

□

Una alternativa alla iperinterpolazione filtrata è l'*iperinterpolazione di Lasso* definita nel seguente modo:

**Definizione 1.14.** *(Iperinterpolazione di Lasso)*
*Sia $\Omega$ un dominio compatto e si supponga che il prodotto scalare discreto $\langle f, g\rangle_N$ definito in (1.7) sia determinato da una formula di quadratura a $N$ punti con grado di precisione $2L$. La interpolazione di Lasso di $f$ in $\mathbb{P}_L(\Omega)$ è definita da:*

$$\mathcal{L}_{\mathrm{L}}^{\lambda} f := \sum_{l=1}^{d} \delta_{\lambda,\mu_1}\left(\langle f, \phi_l\rangle_N\right) \phi_l \tag{1.14}$$

*con*

- $\lambda > 0$*; $\lambda$ parametro di regolarizzazione;*
- $\{\mu_l\}_{l=1,\dots,d}$ *insieme di parametri positivi e* $d = dim(\mathbb{P}_L(\Omega))$*;*
- $\delta_{\lambda\mu_l}\left(\langle f, \phi_l\rangle_N\right)$ *operatore di* soft thresholding *di soglia $\langle f, \phi_l\rangle_N$ definito:*

$$\delta_{\lambda\mu_l}\left(\langle f, \phi_l\rangle_N\right) := \max\{0, \langle f, \phi_l\rangle_N - \lambda\mu_l\} + \max\{0, \langle f, \phi_l\rangle_N + \lambda\mu_l\}$$

L'iperinterpolazione di Lasso modifica quindi i coefficienti discreti di Fourier dell'iperinterpolazione classica applicandoci davanti una funzione soglia. Questa funzione soglia manda a zero alcuni coefficienti, in particolare quelli che:

$$|\langle f, \phi_l\rangle| \leq \lambda\mu_l \Rightarrow \delta_{\lambda,\mu_1}(\langle f, \phi_l\rangle_N) = 0.$$

La terza variante è una combinazione delle due precedenti e prende il nome di *iperinterpolazione ibrida.*

**Definizione 1.15.** *(Iperinterpolazione ibrida)*
*Sia $\Omega$ un dominio compatto e si supponga che il prodotto scalare discreto $\langle f, g\rangle_N$ definito in (1.7) sia determinato da una formula di quadratura a $N$ punti con grado di precisione $2L$. La interpolazione ibrida di $f$ in $\mathbb{P}_L(\Omega)$ è definita da:*

$$\mathcal{H}_{\mathrm{L}}^{\lambda} f := \sum_{l=1}^{d} h_l\, \delta_{\lambda,\mu_1}\left(\langle f, \phi_l\rangle_N\right) \phi_l \tag{1.15}$$

*con*

- $\lambda > 0$*; $\lambda$ parametro di regolarizzazione;*
- $h_l = h\left(\frac{deg\Phi_l}{L}\right)$*, $h$ funzione filtro definita in (1.12);*
- $\{\mu_l\}_{l=1,\dots,d}$ *insieme di parametri positivi;*

- $\{\delta_{\lambda\mu_l}(\cdot)\}_{\mathrm{l}=1}^{\mathrm{d}}$ *operatori di soft thresholding definiti come in (1.14).*

È evidente che l'interpolazione ibrida $\mathcal{H}_L^\lambda f$ può essere vista come la composizione dell'interpolazione filtrata $\mathcal{F}_{L,N} f$ e l'interpolazione di Lasso $\mathcal{L}_L^\lambda f$:

$$\mathcal{H}_L^\lambda f = \mathcal{F}_{L,N}(\mathcal{L}_L^\lambda f).$$

Si osserva che per definizione $h : [0,1] \to [0,1]$, quindi è necessario che:

$$|h_l \langle f, \phi_l \rangle| \leq |\langle f, \phi_l \rangle|.$$

Inoltre in generale non vale la proprietà commutativa, cioè può essere che:

$$\mathcal{F}_{L,N}(\mathcal{L}_L^\lambda f) \neq \mathcal{L}_L^\lambda(\mathcal{F}_{L,N} f).$$

# Capitolo 2

# Problema ai minimi quadrati

È rilevante introdurre il problema ai minimi quadrati (discreto) per poter dimostrare alcune proprietà importanti dell'iperinterpolazione. Ne esistono 3 varianti che servono per caratterizzare le 3 varianti dell'iperinterpolazione.

## 2.1 Problema ai minimi quadrati (standard)

Dato un polinomio $\phi(x) = \sum_{l=1}^{d} \alpha_l \phi_l(x) \in \mathbb{P}_L(\Omega)$ di grado L, questo approssima f se soddisfa il *problema discreto ai minimi quadrati*:

$$\min_{\phi \in \mathbb{P}_L(\Omega)} \left\{ \frac{1}{2} \sum_{j=1}^{N} w_j (\phi(x_j) - f(x_j))^2 \right\}. \tag{2.1}$$

Una scrittura equivalente di questo problema è:

$$\min_{\boldsymbol{\alpha} \in R^d} \frac{1}{2} \|\mathbf{W}^{\frac{1}{2}}(\mathbf{A}\boldsymbol{\alpha} - \mathbf{f})\|_2^2 \tag{2.2}$$

con

- $\mathbf{W} = diag(w_1, \ldots, w_N) \in \mathbb{R}^{N \times N}$ matrice diagonale;
- $\mathbf{A} = (\phi_l(x_j))_{l,j} \in \mathbb{R}^{N \times d}$ la matrice di *Vandermonde*;
- $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_d]^T \in \mathbb{R}^d$ il vettore dei coefficienti utilizzati per costruire $\phi$;
- $\mathbf{f} = [f(x_1), \ldots, f(x_N)]^T \in \mathbb{R}^N$ il vettore dei valori della funzione nei nodi di campionamento $x_1, \ldots, x_N$.

Nell'articolo [4] si dimostra la relazione tra l'iperinterpolante classico $\mathcal{L}_L f$ e la miglior approssimazione ai minimi quadrati (pesata dai pesi della formula di quadratura) di $f$ nei nodi di campionamento. Questo formalmente diventa:

**Teorema 2.1.** *(Caratterizzazione iperinterpolazione classica)*
*Data $f \in \mathcal{C}(\Omega)$, sia $\mathcal{L}_L f \in \mathbb{P}_L(\Omega)$ iperinterpolazione (classica) di $f$ definita in (1.8) con $\langle f, g\rangle_N$ prodotto scalare interno discreto definito in (1.7) da una formula di quadratura a $N$ punti in $\Omega$ con grado di precisione $2L$. Allora $\mathcal{L}_L f$ è l'unica soluzione del problema di approssimazione (2.1).*

*Dimostrazione.* Dato che la funzione da minimizzare in (2.1) è strettamente convessa nella variabile $\alpha$, allora basta calcolare il gradiente (rispetto a $\alpha$) per ottenere la condizione (*equazioni normali*):

$$\mathbf{A}^T\mathbf{W}\mathbf{A}\boldsymbol{\alpha} - \mathbf{A}^T\mathbf{W}\mathbf{f} = 0 \tag{2.3}$$

Si noti che:

- $\mathbf{A}^T\mathbf{W}\mathbf{A} = \mathbb{I}^{d\times d}$ poiché $\{\phi_k\}_{k=1,\dots,d}$ base ortonormale e il grado di precisione $\delta = 2L$:

$$[\mathbf{A}^T\mathbf{W}\mathbf{A}]_{i,k} = \sum_{j=1}^{N} w_j\phi_i(x_j)\phi_k(x_j) = \langle\phi_i,\phi_j\rangle_N = \delta_{i,k} \qquad 1 \le i,k \le d;$$

- $[\mathbf{A}^T\mathbf{W}\mathbf{f}]_l = \sum_{j=1}^{N} w_j\phi_l(x_j)f(x_j) = \langle\phi_l, f\rangle_N \qquad l = 1,\dots,d.$

Quindi la soluzione unica di (2.3) è $\alpha$ che ha coefficienti $\alpha_l = \langle\phi_l, f\rangle_N$. Cioè la soluzione unica è $\mathcal{L}_L f$. □

*Osservazione.* Si noti che $N \ge d$. Questo deriva dal fatto che le colonne della matrice A sono ortonormali e il rango di A è $rank(A) = d$. Inoltre se $N = d$ allora l'iperinterpolante $\mathcal{L}_L f$ è interpolante.

## 2.2 Problema ai minimi quadrati regolarizzati (o penalizzati)

### 2.2.1 Iperinterpolazione filtrata

Dato un polinomio $\phi(x) = \sum_{l=1}^{d} \beta_l^\lambda \phi_l(x) \in \mathbb{P}_L(\Omega)$ di grado L, possiamo introdurre il seguente problema ai minimi quadrati regolarizzati (o penalizzati):

$$\min_{\phi_\lambda\in\mathbb{P}_L(\Omega)} \left\{ \frac{1}{2}\sum_{j=1}^{N} w_j(\phi_\lambda(x_j) - f(x_j))^2 + \sum_{l=1}^{d} w_j(\mathcal{R}_L\phi(x_j))^2 \right\} \tag{2.4}$$

con $\mathcal{R}_L : \mathbb{P}_L \to \mathbb{P}_L$ operatore di regolarizzazione che manda i $\phi_k \in \mathbb{P}_L$ in $\mu_k \, \phi_k \in \mathbb{P}_L$.

Una scrittura equivalente di questo problema è:

$$\min_{\boldsymbol{\beta}^\lambda \in R^d} \frac{1}{2} \|\mathbf{W}^{\frac{1}{2}}(\mathbf{A}\boldsymbol{\beta}^\lambda - \mathbf{f})\|_2^2 + \|\mathbf{W}^{\frac{1}{2}}\mathbf{R}\mathbf{A}\boldsymbol{\beta}^\lambda\|_2^2 \tag{2.5}$$

con

- $\lambda \geq 0$ parametro di regolarizzazione
- $\mathbf{W} = diag(w_1, \ldots, w_N) \in \mathbb{R}^{N \times N}$ matrice diagonale;
- $\mathbf{A} = (\phi_l(x_j))_{l,j} \in \mathbb{R}^{N \times d}$ la matrice di *Vandermonde*;
- $\boldsymbol{\beta}^\lambda = [\beta_1^\lambda, \ldots, \beta_d^\lambda]^T \in \mathbb{R}^d$ vettore dipendente dal parametro di regolarizzazione $\lambda$. Sono i coefficienti da minimizzare del polinomio $\phi(x)$;
- $\mathbf{f} = [f(x_1), \ldots, f(x_N)]^T \in \mathbb{R}^N$ il vettore dei valori della funzione nei nodi di campionamento $x_1, \ldots, x_N$;
- $\mathbf{R} = diag(\mu_1, \ldots, \mu_d) \in \mathbb{R}^{d \times d}$ matrice diagonale.

In maniera analoga alla dimostrazione del teorema di caratterizzazione dell'interpolazione classica (2.1) della sezione precedente, si ricava la condizione:

$$\mathbf{A}^T\mathbf{W}\mathbf{A}\boldsymbol{\beta}^\lambda - \mathbf{A}^T\mathbf{W}\mathbf{f} + \mathbf{A}^T\mathbf{R}^T\mathbf{W}\mathbf{R}\mathbf{A}\boldsymbol{\beta}^\lambda = 0 \tag{2.6}$$

Si era già dimostrato sempre nella dimostrazione del teorema (2.1) che:

- $[\mathbf{A}^T\mathbf{W}\mathbf{A}]_{i,k} = \delta_{i,k} \qquad 1 \leq i, k \leq d;$
- $[\mathbf{A}^T\mathbf{W}\mathbf{f}]_l = \langle \phi_l, f \rangle_N = \alpha_l \qquad l = 1, \ldots, d.$

Inoltre si osserva che: $\forall 1 \leq i, k \leq d$

$$[\mathbf{A}^T\mathbf{R}^T\mathbf{W}\mathbf{R}\mathbf{A}\boldsymbol{\beta}^\lambda]_{i,k} = \sum_{j=1}^{N} \mu_i^2 w_j \phi_i(x_j)\phi_k(x_j) = \mu_i^2 \langle \phi_i, \phi_k \rangle_N = \mu_i^2 \delta_{i,k}. \tag{2.7}$$

Quindi la condizione (2.6) è equivalente a

$$\beta_l^\lambda - \alpha_l + \mu_l^2 \beta_l^\lambda = 0 \tag{2.8}$$

che diventa

$$\boxed{\beta_l^\lambda = \frac{1}{1 + \mu_l^2}\alpha_l}.$$

Questi sono i coefficienti del polinomio che soddisfa il problema ai minimi quadrati regolarizzati (2.4).

*Osservazione.* Si nota che per $\mu_l = 0$ per $l = 1, \ldots, d$ si hanno i coefficienti dell'interpolante classico $\alpha_l$. Questo poiché se non si introduce l'operatore di regolarizzazione $\mathcal{R}_l$, il problema ai minimi quadrati regolarizzati (2.4) diventa equivalente al problema ai minimi quadrati standard (2.1).

*Osservazione.* Si nota inoltre che se si sceglie $\mu_l = 0$ per $l = 1, \ldots, \lfloor L/2 \rfloor$, l'operatore di regolarizzazione $\mathcal{R}_L$ coincide con un particolare filtro $h$. In maniera analoga, ogni filtro $h$ che abbia immagine in $[0, 1]$ coincide con un operatore di regolarizzazione.

### 2.2.2 Iperinterpolazione di Lasso

Dato un polinomio $\phi(x) = \sum_{l=1}^{d} \gamma_l^\lambda \phi_l(x) \in \mathbb{P}_L(\Omega)$ di grado L, possiamo introdurre il problema ai minimi quadrati regolarizzati (o penalizzati):

$$\min_{\phi_\lambda \in \mathbb{P}_L(\Omega)} \left\{ \frac{1}{2} \sum_{j=1}^{N} w_j (\phi_\lambda(x_j) - f(x_j))^2 + \lambda \sum_{l=1}^{d} \mu_l |\gamma_l^\lambda| \right\} \tag{2.9}$$

Una scrittura equivalente di questo problema è:

$$\min_{\boldsymbol{\gamma}^\lambda \in R^d} \frac{1}{2} \|\mathbf{W}^{\frac{1}{2}} (\mathbf{A}\boldsymbol{\gamma}^\lambda - \mathbf{f})\|_2^2 + \lambda \|\mathbf{R}_1 \boldsymbol{\gamma}^\lambda\|_1 \tag{2.10}$$

con

- $\lambda \geq 0$, $\lambda$ parametro di regolarizzazione;
- $\mathbf{W} = diag(w_1, \ldots, w_N) \in \mathbb{R}^{N \times N}$ matrice diagonale;
- $\mathbf{A} = (\phi_l(x_j))_{l,j} \in \mathbb{R}^{N \times d}$ la matrice di *Vandermonde*;
- $\boldsymbol{\gamma}^\lambda = [\gamma_1^\lambda, \ldots, \gamma_d^\lambda]^T \in \mathbb{R}^d$ vettore dipendente dal parametro $\lambda$;
- $\mathbf{f} = [f(x_1), \ldots, f(x_N)]^T \in \mathbb{R}^N$ il vettore dei valori della funzione nei nodi di campionamento $x_1, \ldots, x_N$;
- $\mathbf{R}_1 = diag(\mu_1, \ldots, \mu_d) \in \mathbb{R}^{d \times d}$ matrice diagonale.

Si può enunciare così un teorema di caraterizzazione per l'iperinterpolante di Lasso $\mathcal{L}_L^\lambda f$:

**Teorema 2.2.** *(Caratterizzazione iperinterpolazione di Lasso)*
*Data $f \in \mathcal{C}(\Omega)$, sia $\mathcal{L}_L^\lambda f \in \mathbb{P}_L(\Omega)$ iperinterpolazione di Lasso di $f$ definita in (1.14) con $\langle f, g \rangle_N$ prodotto scalare interno discreto definito in (1.7) da una formula di quadratura a N punti in $\Omega$ con grado di precisione 2L. Allora $\mathcal{L}_L^\lambda f$ è l'unica soluzione del problema di approssimazione (2.9).*

*Osservazione.* La iperinterpolazione di Lasso, a differenza delle iperinterpolazioni classica e filtrata, non è in generale una proiezione di un certo spazio polinomiale. Infatti non esiste un teorema di proiezione come invece (1.1) per l'iperinterpolazione classica e (1.2) per quella filtrata.

### 2.2.3 Iperinterpolazione ibrida

Considerando sempre un polinomio $\phi(x) = \sum_{l=1}^{d} \gamma_l^\lambda \phi_l(x) \in \mathbb{P}_L(\Omega)$ di grado L, possiamo introdurre una variazione del problema ai minimi quadrati regolarizzati che caratterizza l'iperinterpolazione ibrida:

$$\min_{\phi_\lambda \in \mathbb{P}_L(\Omega)} \Psi_\lambda(\phi_\lambda) = \frac{1}{2}\sum_{j=1}^{N} w_j(\phi_\lambda(x_j) - f(x_j))^2 + \frac{1}{2}\sum_{j=1}^{N} w_j(\mathcal{R}_L\phi_\lambda(x_j))^2 + \lambda \sum_{l=1}^{d} \mu_l |\beta_l^\lambda| \tag{2.11}$$

con

$$\mathcal{R}_L\phi_\lambda(x) = \sum_{l=1}^{d} b_l \langle \phi_l, \phi_\lambda \rangle_N \phi_l = \sum_{l=1}^{d} b_l \beta_l^\lambda \phi_l(x) \tag{2.12}$$

lo stesso operatore di regolarizzazione usato in (2.4). Inoltre

$$b_l = \begin{cases} 0 & \text{per } \frac{deg(\phi_l)}{L} \in [0, \frac{1}{2}] \\ \sqrt{\frac{1}{h_l} - 1} & \text{per } \frac{deg(\phi_l)}{L} \in [\frac{1}{2}, 1) \end{cases} \tag{2.13}$$

*Osservazione.* Con $h$ intendiamo la funzione filtro definita in (1.12). Inoltre si osserva che il caso $\frac{deg(\phi_l)}{L} = 1$ è escluso, poiché sarebbe $b_l = \infty$ e quindi $\beta_l^\lambda = 0$.

Una scrittura equivalente di questo problema è:

$$\min_{\boldsymbol{\beta}^\lambda \in R^d} \Psi_\lambda(\boldsymbol{\beta}^\lambda) = \frac{1}{2}\|\mathbf{W}^{\frac{1}{2}}(\mathbf{A}\boldsymbol{\beta}^\lambda - \mathbf{f})\|_2^2 + \frac{1}{2}\|\mathbf{W}^{\frac{1}{2}}\mathbf{R}_2\boldsymbol{\beta}^\lambda\|_2^2 + \lambda\|\mathbf{R}_1\boldsymbol{\beta}^\lambda\|_1 \tag{2.14}$$

con

- $\lambda \geq 0$, $\lambda$ parametro di regolarizzazione;
- $\mathbf{W} = diag(w_1, \ldots, w_N) \in \mathbb{R}^{N\times N}$ matrice diagonale;
- $\mathbf{A} = (\phi_l(x_j))_{l,j} \in \mathbb{R}^{N\times d}$ la matrice di *Vandermonde*;
- $\boldsymbol{\beta}^\lambda = [\beta_1^\lambda, \ldots, \beta_d^\lambda]^T \in \mathbb{R}^d$ vettore dipendente dal parametro $\lambda$;
- $\mathbf{f} = [f(x_1), \ldots, f(x_N)]^T \in \mathbb{R}^N$ il vettore dei valori della funzione nei nodi di campionamento $x_1, \ldots, x_N$;

- $\mathbf{R}_1 = diag(\mu_1, \ldots, \mu_d) \in \mathbb{R}^{d\times d}$ matrice diagonale;
- $\mathbf{R}_2 = \mathbf{AB} \in \mathbb{R}^{N\times d}$.

**Teorema 2.3.** *(Caratterizzazione iperinterpolazione ibrida)*
*Data $f \in \mathcal{C}(\Omega)$, sia $\mathcal{H}_L^\lambda f \in \mathbb{P}_L(\Omega)$ iperinterpolazione ibrida di $f$ definita in (1.15) con $\langle f, g\rangle_N$ prodotto scalare interno discreto definito in (1.7) da una formula di quadratura a $N$ punti in $\Omega$ con grado di precisione $2L$. Siano $\mu_1, \ldots, \mu_d > 0$ e $\mathcal{R}_L$ definito in (2.12) con $b_l$ definito in (2.13). Allora $\mathcal{H}_L^\lambda f$ è l'unica soluzione del problema di approssimazione (2.11).*

*Dimostrazione.* Come si è visto nella dimostrazione del teorema (2.1), dato che la funzione da minimizzare in (2.1) è strettamente convessa nella variabile $\alpha$, allora il punto stazionario del problema (2.1) rispetto a $\alpha$ soddisfa l'equazione del primo ordine:

$$\mathbf{A}^T\mathbf{WA}\boldsymbol{\alpha} - \mathbf{A}^T\mathbf{Wf} = 0 \tag{2.15}$$

L'assunzione $f \neq 0$ assicura che $\|\mathbf{A}^T\mathbf{Wf}\|_\infty \neq 0$.

Si noti che $\mathbf{A}^T\mathbf{WA} = \mathbb{I}^{d\times d}$ poiché $\{\phi_k\}_{k=1,\ldots,d}$ base ortonormale e il grado di precisione $\delta = 2L$, infatti:

$$[\mathbf{A}^T\mathbf{WA}]_{i,k} = \sum_{j=1}^{N} w_j\phi_i(x_j)\phi_k(x_j) = \langle\phi_i, \phi_j\rangle_N = \delta_{i,k} \qquad 1 \leq i, k \leq d \tag{2.16}$$

Per la condizione del primo ordine (2.15) e per la condizione di identità (2.16) si ha: $\boldsymbol{\alpha} = \mathbf{A}^T\mathbf{Wf}$.
Quindi $\boldsymbol{\beta}^\lambda = [\beta_1^\lambda, \ldots, \beta_d^\lambda]^T$ è soluzione di (2.11) se e solo se:

$$\mathbf{0} \in \mathbf{A}^T\mathbf{WA}\boldsymbol{\beta}^\lambda - \mathbf{A}^T\mathbf{Wf} + \lambda\mu_l\partial(|\beta_l^\lambda|) + \mathbf{B}^T\mathbf{A}^T\mathbf{WAB}\boldsymbol{\beta}^\lambda \tag{2.17}$$

con $\partial(\cdot)$ derivate partziali. Questo implica:

$$0 \in \beta_l^\lambda - \alpha_l + \lambda\mu_l\partial(|\beta_l^\lambda|) + b_l^2\beta_l^2, \qquad \forall l = 1, \ldots, d.$$

dove

$$\partial(|\beta_l^\lambda|) = \begin{cases} 1, & \text{se } \beta_l^\lambda > 0 \\ \in [-1, 1], & \text{se } \beta_l^\lambda = 0 \\ -1, & \text{se } \beta_l^\lambda < 0 \end{cases}$$

Sia $\boldsymbol{\beta}^* = [\beta_1^*, \ldots, \beta_d^*]$ la soluzione ottima del problema ai minimi quadrati regolarizzati (2.11). Allora si ha:

$$\beta_l^* = \frac{1}{1 + b_l^2}\left[\alpha_l - \lambda\mu_l\partial(|\beta_l^\lambda|)\right], \qquad \forall l = 1, \ldots, d.$$

Ci sono ora 3 casi da prendere in considerazione:

1. se $\alpha_l > \lambda\mu_l \;\Rightarrow\; \alpha_l - \lambda\mu_l\partial(|\beta_l^\lambda|) > 0$:
   allora $\beta_l^* > 0 \;\Rightarrow\; \partial(|\beta_l^\lambda|) = 1 \;\Rightarrow\; \beta_l^* = (\alpha_l - \lambda\mu_l)(1 + b_l^2) > 0$;
2. se $\alpha_l < -\lambda\mu_l \;\Rightarrow\; \alpha_l - \lambda\mu_l\partial(|\beta_l^\lambda|) < 0$:
   allora $\beta_l^* < 0 \;\Rightarrow\; \partial(|\beta_l^\lambda|) = -1 \;\Rightarrow\; \beta_l^* = (\alpha_l + \lambda\mu_l)(1 + b_l^2) < 0$;
3. se $-\lambda\mu_l \leq \alpha_l \leq \lambda\mu_l$
   allora $\beta_l^* > 0$ vuol dire che $\beta_l^* = (\alpha_l - \lambda\mu_l)(1 + b_l^2) < 0$,
   mentre $\beta_l^* > 0$ vuol dire $\beta_l^* = (\alpha_l + \lambda\mu_l)(1 + b_l^2) > 0$.
   C'è una contraddizione quindi $\beta_l^* = 0$.

Dato che $\alpha_l = \langle f, \phi_l\rangle_N$ e $b_l$ definito come in (2.13) per ogni $l = 1, \ldots, d$, allora si deduce, per definizione di soft thresholding (1.14), che:

$$\beta_l^\lambda = \frac{\delta_{\lambda,\mu_l}(\langle f, \phi_l\rangle_N)}{1 + b_l^2} = h_l\delta_{\lambda,\mu_l}(\langle f, \phi_l\rangle_N). \tag{2.18}$$

Con questo si è dimostrati che $\mathcal{H}_L^\lambda f$ l'iperinterpolatore ibrido di $f$ risolve il problema ai minimi quadrati regolarizzati (2.11). □

# Capitolo 3

# Analisi dell'errore

In questo capitolo riporteremo delle proprietà relative a dati soggetti (o meno) a rumore.

## 3.1 Dati non soggetti a rumore

Per prima cosa viene enunciato, senza dimostrazione, un lemma sulla norma $L^2(\Omega)$ dell'iperinterpolatore classico.

**Lemma 3.1.** *(Convergenza e stabilità di $\mathcal{L}_L$)*
*Data $f \in \mathcal{C}(\Omega)$, sia $\mathcal{L}_L f \in \mathbb{P}_L(\Omega)$ iperinterpolazione (classica) di $f$ definita in (1.8) con $\langle f, g\rangle_N$ prodotto scalare interno discreto definito in (1.7) da una formula di quadratura a $N$ punti in $\Omega$ con grado di precisione $2L$. Siano inoltre $\|\cdot\|_\infty$ norma infinito definita in* (1.9)*, $\|\cdot\|_2$ la norma $L_2$ definita in* (1.9) *e $\mathcal{E}_L(\cdot)$ l'errore di miglior approssimazione definito in* (1.11)*. Allora se*

$$\|\mathcal{L}_L f\|_2 \leq V^{\frac{1}{2}} \|f\|_\infty$$

*e*

$$\|\mathcal{L}_L f - f\|_2 \leq 2V^{\frac{1}{2}} \mathcal{E}_L(f),$$

*si ha $\|\mathcal{L}_L f - f\|_2 \to 0$ per $L \to \infty$.*

Si procede dando delle proprietà dell'iperinterpolatore ibrido (oggetto della tesi). Preliminarmente si può introdurre, sotto le ipotesi del teorema di caratterizzazione dell'iperinterpolazione ibrida (2.3), l'operatore:

$$K(f) := \sum_{l=1}^{d} \{h_l \, \delta_{\lambda,\mu_l}(\alpha_l) \cdot \alpha_l - [h_l \, \delta_{\lambda,\mu_l}(\alpha_l)]^2\} \geq 0 \tag{3.1}$$

con $\alpha_l = \langle f, \phi_l \rangle_N = \sum_{j=1}^{N} w_j \, f(x_j) \, \phi_l(x_j)$.
Si nota, quindi, che:

$$\text{se} \quad \boxed{\lambda \geq \max_{l=1,\ldots,d} \frac{|\alpha_l|}{\mu_l}} \quad \text{sufficientemente grande} \;\Rightarrow\; K(f) = 0.$$

Si possono ora enunciare e dimostrare le seguenti proprietà:

**Lemma 3.2.** *Sotto le condizioni del teorema di caratterizzazione dell'iperinterpolazione ibrida* (2.3) *si ha:*

1. $\langle f - \mathcal{H}_L^\lambda f, \mathcal{H}_L^\lambda f \rangle_N = K(f)$
2. $\langle \mathcal{H}_L^\lambda f, \mathcal{H}_L^\lambda f \rangle_N + \langle f - \mathcal{H}_L^\lambda f, f - \mathcal{H}_L^\lambda f \rangle_N = \langle f, f \rangle_N - 2K(f)$
3. $K(f) \leq \frac{\langle f, f \rangle_N}{2}$
4. $\langle \mathcal{H}_L^\lambda f, \mathcal{H}_L^\lambda f \rangle_N \leq \langle f, f \rangle_N - 2K(f)$

*Dimostrazione. 1.* Calcolando il primo membro si ottiene:

$$\begin{aligned}
&\langle f - \mathcal{H}_L^\lambda, \mathcal{H}_L^\lambda \rangle_N = \\
&= \left\langle f - \sum_{l=1}^{d} h_l \, \delta_{\lambda\mu_l}(\alpha_l) \, \phi_l, \, \sum_{k=1}^{d} h_k \, \delta_{\lambda\mu_k}(\alpha_k) \, \phi_k \right\rangle_N = \\
&= \sum_{k=1}^{d} h_k \, \delta_{\lambda\mu_k}(\alpha_k) \left\langle f - \sum_{l=1}^{d} h_l \, \delta_{\lambda\mu_l}(\alpha_l) \, \phi_l, \, \phi_k \right\rangle_N
\end{aligned}$$

e dato che

$$\begin{aligned}
&\left\langle f - \sum_{l=1}^{d} h_l \, \delta_{\lambda\mu_l}(\alpha_l) \, \phi_l, \, \phi_k \right\rangle_N = \\
&= \langle f, \phi_k \rangle_N - \left\langle \sum_{l=1}^{d} h_l \, \delta_{\lambda\mu_l}(\alpha_l) \, \phi_l, \, \phi_k \right\rangle_N = \\
&= \alpha_k - h_k \, \delta_{\lambda\mu_l}(\alpha_l),
\end{aligned}$$

quindi si ha

$$\begin{aligned}
&\langle f - \mathcal{H}_L^\lambda, \mathcal{H}_L^\lambda \rangle_N = \\
&= \sum_{k=1}^{d} h_k \, \delta_{\lambda\mu_k}(\alpha_k) \left\langle f - \sum_{l=1}^{d} h_l \, \delta_{\lambda\mu_l}(\alpha_l) \, \phi_l, \, \phi_k \right\rangle_N \\
&= \sum_{k=1}^{d} h_k \, \delta_{\lambda\mu_k}(\alpha_k) \cdot [\alpha_k - h_k \, \delta_{\lambda\mu_l}(\alpha_l)] = \\
&= \sum_{k=1}^{d} \left\{ h_k \, \delta_{\lambda\mu_k}(\alpha_k) \, \alpha_k - [h_k \, \delta_{\lambda\mu_l}(\alpha_l)]^2 \right\} = \\
&= K(f)
\end{aligned}$$

*2.* Dal fatto che $\left\langle \mathcal{H}_L^\lambda, \mathcal{H}_L^\lambda \right\rangle_N = \left\langle f, \mathcal{H}_L^\lambda f \right\rangle_N - K(f)$ e

$$\left\langle f - \mathcal{H}_L^\lambda, f - \mathcal{H}_L^\lambda \right\rangle_N = \left\langle f, f \right\rangle_N - 2\left\langle f, \mathcal{H}_L^\lambda \right\rangle_N + \left\langle \mathcal{H}_L^\lambda, \mathcal{H}_L^\lambda \right\rangle_N$$

si ha immediatamente che

$$\langle \mathcal{H}_L^\lambda f, \mathcal{H}_L^\lambda f \rangle_N + \langle f - \mathcal{H}_L^\lambda f, f - \mathcal{H}_L^\lambda f \rangle_N = \langle f, f \rangle_N - 2K(f) \tag{3.2}$$

*3.* Dall'uguaglianza (3.2) e da $\langle f, f \rangle_N \geq 0$ per tutte le $f \in C(\Omega)$, si ottiene:

$$\langle f, f \rangle_n - 2K(f) \geq 0,$$

cioè si è dimostrata la terza proprietà.

*4.* Dall'uguaglianza (3.2) e da $\langle f, f \rangle_N \geq 0$ per tutte le $f \in C(\Omega)$, si ottiene:

$$\left\langle \mathcal{H}_L^\lambda, \mathcal{H}_L^\lambda \right\rangle_N \leq \langle f, f \rangle_N - 2K(f).$$

□

Ora si enuncia un teorema di convergenza e stabilità simile a quello per l'iperinterpolazione classica (3.1) ma per l'iperinterpolazione ibrida.

**Teorema 3.1.** *Si assumano le condizioni del teorema di caratterizzazione dell'iperinterpolazione ibrida* (2.3)*. Allore esiste* $\tau_1 < 1$ *inversamente proporzionale a* $K(f)$ *tale che:*

$$\|\mathcal{H}_L^\lambda f\|_2 \leq \tau_1 V^{\frac{1}{2}} \|f\|_\infty, \tag{3.3}$$

*con* $V = \int_\Omega d\omega$ *misura definita come in* (1.1)*. Inoltre esiste* $\tau_2 < 1$ *inversamente proporzionale a* $K(f - \phi^*)$ *tale che:*

$$\|\mathcal{H}_L^\lambda f - f\|_2 \leq (1 + \tau_2) V^{\frac{1}{2}} \mathcal{E}_L(f) + \|\phi^* - \mathcal{H}_L^\lambda \phi^*\|_2. \tag{3.4}$$

*con* $\phi^*$ *il polinomio di miglior approssimazione di* $f$ *in* $\mathbb{P}(\Omega)$ *definito in* (1.11)*.*

*Dimostrazione.* Per qualsiasi $f \in C(\Omega)$ si ha $\mathcal{H}_L^\lambda f \in \mathbb{P}_L(\Omega)$.
Inoltre usando la proprietà *4.* del lemma (3.2) si ottiene:

$$\begin{aligned}\|\mathcal{H}_L^\lambda f\|_2^2 &= \left\langle \mathcal{H}_L^\lambda f, \mathcal{H}_L^\lambda f \right\rangle = \left\langle \mathcal{H}_L^\lambda f, \mathcal{H}_L^\lambda f \right\rangle_N \leq \langle f, f \rangle_N - 2K(f), \\ &= \sum_{j=1}^{d} w_j (f(x_j))^2 - 2K(f) \leq \sum_{j=1}^{d} w_j \|f\|_\infty^2 - 2K(f), \\ &= V\|f\|_\infty^2 - 2K(f).\end{aligned}$$

Quindi si deduce che $\|\mathcal{H}_L^\lambda f\|_2 \leq \sqrt{V\|f\|_\infty^2 - 2K(f)}$.
Per questo e per il fatto che $K(f) \geq 0$, si ha che esiste $\tau_1 = \tau_1(K(f)) < 1$ che inversamente proporzionale a K(f) ed è tale che:

$$\|\mathcal{H}_L^\lambda f\|_2 \leq \sqrt{V\|f\|_\infty^2 - 2K(f)} = \tau_1 V^{\frac{1}{2}} \|f\|_\infty. \tag{3.5}$$

E con questo abbiamo dimostrato la prima parte.

Ora dato che per ogni polinomio $\phi \in \mathbb{P}_L(\Omega)$ si ha:

$$\begin{aligned}\|\mathcal{H}_L^\lambda f - f\|_2 &= \|\mathcal{H}_L^\lambda (f - \phi) - (f - \phi) - (\phi - \mathcal{H}_L^\lambda \phi)\|_2, \\ &\leq \|\mathcal{H}_L^\lambda (f - \phi)\|_2 + \|f - \phi\|_2 + \|\phi - \mathcal{H}_L^\lambda \phi\|_2.\end{aligned}$$

Si prenda ora $\phi = \phi^*$ polinomio di miglior approssimazione di $f$.
Inoltre ricordiamo una relazione tra *norma 2* (1.10) e *norma infinito* (1.9):

$$\|g\|_2 = \sqrt{\langle g, g \rangle} \leq \|g\|_\infty \sqrt{\langle 1, 1 \rangle} = V^{\frac{1}{2}} \|G\|_\infty \qquad \forall g \in C(\Omega). \tag{3.6}$$

Allora per (3.5) e per (3.6) esiste $\tau_2 = \tau_2(K(f - \phi^*)) < 1$ inversamente proporzionale a $K(f - \phi^*)$ tale che:

$$\begin{aligned}\|\mathcal{H}_L^\lambda f - f\|_2 &\leq \|\mathcal{H}_L^\lambda (f - \phi^*)\|_2 + \|f - \phi^*\|_2 + \|\phi^* - \mathcal{H}_L^\lambda \phi^*\|_2 \\ &\leq \tau_2 V^{\frac{1}{2}} \|f - \phi^*\|_\infty + V^{\frac{1}{2}} \|f - \phi^*\|_\infty + \|\phi^* - \mathcal{H}_L^\lambda \phi^*\|_2 \\ &= (1 + \tau_2) V^{\frac{1}{2}} \|f - \phi^*\|_\infty + \|\phi^* - \mathcal{H}_L^\lambda \phi^*\|_2\end{aligned}$$

□

## 3.2 Dati soggetti a rumore

Nel seguente teorema si descrive l'errore $L^2(\Omega)$ dell'iperinterpolazione ibrida in presenza di rumore:

**Teorema 3.2.** *Si assumano le condizioni del teorema di caratterizzazione dell'iperinterpolazione ibrida* (2.3). *Siano $f^\epsilon$ una versione perturbata di $f$ e $\mathcal{H}_L^\lambda f \in \mathbb{P}_L(\Omega)$ l'iperinterpolazione ibrida di $f$ definita in* (1.15).

*Allora esistono $\tau_3 < 1$ e $\tau_4 < 1$ inversamente proporzionali rispettivamente a $K(f^\epsilon - f)$ e $K(f^\epsilon - \phi^*)$ tali che:*

$$\|\mathcal{H}_L^\lambda f^\epsilon - f\|_2 \leq \tau_3 V^{\frac{1}{2}} \|f - f^\epsilon\|_\infty + (1 + \tau_4) V^{\frac{1}{2}} \mathcal{E}_L(f) + \|\phi^* - \mathcal{H}_L^\lambda \phi^*\|_2 \quad (3.7)$$

*con $V = \int_\Omega d\omega$ misura definita come in* (1.1) *e $\mathcal{E}_L f$ l'errore di miglior approssimazione di $f$ in $\mathbb{P}_L(\Omega)$ definito in* (1.11).

*Dimostrazione.* Si vede facilmente che

$$\begin{aligned} \|\mathcal{H}_L^\lambda f^\epsilon - f\|_2 &= \|\mathcal{H}_L^\lambda f^\epsilon - \mathcal{H}_L^\lambda f + \mathcal{H}_L^\lambda f - f\|_2 \\ &\leq \|\mathcal{H}_L^\lambda (f^\epsilon - f)\|_2 + \|\mathcal{H}_L^\lambda f - f\|_2. \end{aligned}$$

Per il teorema (3.1):

$$\|\mathcal{H}_L^\lambda f^\epsilon - f\|_2 \leq \tau_3 V^{\frac{1}{2}} \|f - f^\epsilon\|_\infty + (1 + \tau_4) V^{\frac{1}{2}} \mathcal{E}_L(f) + \|\phi^* - \mathcal{H}_L^\lambda \phi^*\|_2$$

dove $\tau_3 < 1$ e $\tau_4 < 1$ sono inversamente proporzionali rispettivamente a $K(f^\epsilon - f)$ e $K(f^\epsilon - \phi^*)$.

□

### 3.2.1 Parametro di regolarizzazione

È evidente dalla trattazione fatta fin'ora che è presente un parametro di regolarizzazione. In questa sezione si offre un criterio a priori per la scelta di questo parametro.
Questa regola riflette la relazione tra l'errore in *norma $L_2$* e la *sparsità* dei coefficienti delle varianti di iperinterpolazione.

*Nota.* Per *sparsità* dei coefficienti si intende che molti di questi sono nulli. Infatti si dice che una matrice è *sparsa* se i suoi valori sono quasi tutti uguali a zero.

**Teorema 3.3.** *Si assumano le condizioni del teorema di caratterizzazione dell'iperinterpolazione ibrida* (2.3). *Sia $\{\phi_l | l = 1, \ldots, d\}$ base ortonormale di $\mathbb{P}_L(\Omega)$ definita in* (1.3). *Una volta determinata la formula di quadratura* (1.5) *si hanno le matrici $\mathbf{A} = (\phi_l(x_j))_{j,l} \in \mathbb{R}^{N \times d}$ e $\mathbf{W} = diag(w_1, \ldots, w_N) \in \mathbb{R}^{N \times N}$ matrice dei pesi. Siano inoltre:*

- $\boldsymbol{\epsilon} = [\epsilon_1, \ldots, \epsilon_N]^T \in \mathbb{R}^N$;

- $\mathbf{f}^\epsilon, \mathbf{f} \in C(\Omega)$ *con* $\mathbf{f}^\epsilon = [f^\epsilon(x_1), \ldots, f^\epsilon(x_N)]^T \in \mathbb{R}^N$ *la funzione con rumore valutata nei nodi di campionamento e* $\mathbf{f} = [f(x_1), \ldots, f(x_N)]^T \in \mathbb{R}^N$ *la funzione (senza rumore) valutata nei nodi di campionamento t.c.* $f^\epsilon - f = \epsilon$ *;*

- $\mu_l = 1$ *per ogni* $l = 1, \ldots, d$*;*

- $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_d] \in \mathbb{R}^d$ *vettore dei coefficienti* $\alpha_l = \langle f^\epsilon, \phi_l \rangle_N$*;*

- $\boldsymbol{\zeta} = [\zeta_1, \ldots, \zeta_d]^T \in \mathbb{R}^d$ *t.c.* $\zeta_1 \geq \cdots \geq \zeta_d$.
  $\zeta_s$ *valori assoluti degli elementi di* $\boldsymbol{\alpha}$ *per* $s = 1, \ldots, d$*;*

- *per* $\mathbf{z} = [z_1, \ldots, z_d]^T \in \mathbb{R}^d$*:*

$$J(z) := \sum_{l=1}^{d} (z_l^2 - 2 z_l \, \alpha_l) \quad e \quad H(z) := 2 \sum_{l=1}^{d} z_l \sum_{j=1}^{N} w_j \, \epsilon_j \, \phi_l(x_j)$$

*Se i parametri di regolarizzazione nei problemi ai minimi quadrati regolarizzati* (2.9) *e* (2.11) *sono scelti in questo modo:* $\lambda^* = \lambda(s) := \zeta_s$*, allora si ha:*

1. *per l'iperinterpolazione ibrida:*

$$\|\mathbf{W}^{\frac{1}{2}}(\mathbf{A}\boldsymbol{\beta}^{\lambda(s)} - \mathbf{f})\|_2^2 = J(\boldsymbol{\beta}^{\lambda(s)}) + H(\boldsymbol{\beta}^{\lambda(s)}) + \|\mathbf{W}^{\frac{1}{2}} f\|_2^2 \tag{3.8}$$

   *Osservazione.* Il valore del primo termine del membro di destra $J(\boldsymbol{\beta}^{\lambda(s)})$ è tale che $J(\boldsymbol{\beta}^{\lambda(s)}) \leq 0$ e decresce al crescere di s.

2. *per l'iperinterpolazione di Lasso:*

$$\|\mathbf{W}^{\frac{1}{2}}(\mathbf{A}\boldsymbol{\gamma}^{\lambda(s)} - \mathbf{f})\|_2^2 = J(\boldsymbol{\gamma}^{\lambda(s)}) + H(\boldsymbol{\gamma}^{\lambda(s)}) + \|\mathbf{W}^{\frac{1}{2}} f\|_2^2 \tag{3.9}$$

   *Osservazione.* Il valore del primo termine del membro di destra $J(\boldsymbol{\gamma}^{\lambda(s)})$ è tale che $J(\boldsymbol{\gamma}^{\lambda(s)}) \leq 0$ e decresce al crescere di s.

3. *vale la sequente disuguaglianza:*

$$J(\boldsymbol{\gamma}^{\lambda(s)}) \leq J(\boldsymbol{\beta}^{\lambda(s)}) \leq 0. \tag{3.10}$$

*Nota.* Per (3.8) e (3.9) e l'esistenza della funzione filtro (1.12), l'iperinterpolazione ibrida performa molto vicino a quella di Lasso quando $s$, l'indice scelto tale che $\lambda^* = \lambda(s)$, è piccolo.

# Capitolo 4

# Risultati Numerici

In questo capitolo si verificherà numericamente in Matlab (versione *"Matlab R2022b"*) quanto discusso nei capitoli precedenti.

Negli esempi riportati in [1] sono stati considerati quali domini $\Omega$

- l'intervallo $[-1, 1]$,
- il disco unitario $B((0, 0), 1)$,
- la sfera unitaria $\mathbb{S}^2$,
- il cubo unitario $[-1, 1]^3$,
- un unione di dischi $\Omega = \cup_{k=1}^{M} B(C_k, r_k) \in \mathbb{R}^2$, aventi rispettivamenti centri e raggi uguali a $C_k$, $r_k$ con $k = 1, \ldots, M$.

Mentre nei primi 4 casi la base ortonormale dello spazio dei polinomi di grado totale $L$, ovvero $\mathbb{P}_L(\Omega)$, è nota esplicitamente, nell'ultimo esempio $\{\phi_l\}$ è determinata grazie ad un procedimento utilizzante fattorizzazione **QR**. A quanto visto precedentemente aggiungiamo una simile analisi a domini di tipo poligonale per i quali una base ortonormale non è al momento nota in forma chiusa. A tal proposito abbiamo utilizzato quanto descritto in [1], nella sezione relativa all'unione di dischi.

In dettaglio, sia $\{\varphi_k\}_{k=1}^{d}$ una base polinomiale di tipo triangolare di $\mathbb{P}_\delta$, cioè $deg(\varphi_k) = k-1$. Sia inoltre $\mathbf{V}_\varphi = (\varphi_j(\mathbf{x}_i)) \in \mathbb{R}^{N \times d}$ la matrice di Vandermonde valutata nei nodi di cubatura della formula utilizzata, che supponiamo di tipo PI, con grado algebrico di precisione $\delta = 2L$:

$$\int_\Omega p(\mathbf{x}) dx_1 dx_2 = \sum_{k=1}^{N_\delta} w_k \, p(\mathbf{x_k}) \qquad p \in \mathbb{P}_\delta(\Omega).$$

Sia $\sqrt{\mathbf{W}} \in \mathbb{R}^{d\times d}$ la matrice diagonale con componenti diagonali $\sqrt{\mathbf{W}}_{i,i} = \sqrt{w_i}$, $i = 1, \ldots, N$. Sotto queste ipotesi, si vede che $\mathbf{V}_\varphi$ ha rango pieno ed è possibile utilizzare la fattorizzazione $QR$, ottenendo $\sqrt{\mathbf{W}}\mathbf{V}_\varphi = \mathbf{QR}$ con $\mathbf{Q} \in \mathbb{R}^{N\times d}$ matrice ortogonale e $\mathbf{R} \in \mathbb{R}^{d\times d}$ matrice triangolare superiore, non singolare. Questo comporta che la base $(\phi_1, \ldots, \phi_d) = (\varphi_1, \ldots, \varphi_d)\mathbf{R}^{-1}$ è ortonormale rispetto al prodotto scalare $\langle f, g\rangle_N$ e conseguentemente a quello di $L_2(\Omega)$. Un aspetto fondamentale è che $\mathbf{R}^{-1}$ è una matrice triangolare superiore non singolare e quindi anche la base $(\phi_1, \ldots, \phi_d)$ è triangolare $\deg(\phi_k) = \deg(\varphi_k)$, $k = 1, \ldots, d$. Questo risulta essenziale nell'applicazione dell'iperinterpolazione filtrata nel dominio poligonale scelto.

Nei test numerici, quale base $\{\varphi_k\}_{k=1}^d$ con $d = \frac{(L+1)(L+2)}{2}$ di $\mathbb{P}_L(\Omega)$ abbiamo considerato la base prodotto di tipo Chebyshev scalata sul più piccolo rettangolo $[a_1, b_1] \times [a_2, b_2]$ contenente $\Omega$, utilizzando l'ordinamento lessicografico.

## 4.1 Formula di cubatura su domini poligonali

Relativamente alla formula di cubatura utilizzata, l'idea è di suddividere il poligono dato $\mathcal{P}$ in domini più semplici $\Omega_1, \ldots, \Omega_\nu$ tali che $\mathcal{P} = \cup_{i=1}^{\nu}\Omega_i$ e, su ognuno di questi $\Omega_i$, applicare una formula di quadratura avente nodi interni, pesi positivi e grado di precisione predefinito.

In virtù di queste osservazioni si implementa la seguente procedura (approfondita nell'articolo [3]):

- 1° step: si triangola il poligono dato tramite le routine `polyshape` in Matlab; a tal proposito, il numero di triangoli $\nu$ che si ottengono è minimale (o quasi minimale) ed è noto che i poligoni semplici e semplicemente connessi con $n$ vertici si ha $\nu = n - 2$ (e più in generale $\nu \approx n$).

- 2° step: su ogni triangolo trovato al 1° step si determina una formula di quadratura (quasi minima) a pesi positivi e nodi interni con grado di precisione $\delta$ (1.6).

In questo modo si ottiene una formula di cubatura *composta* su tutto il poligono $\mathcal{P}$.

*Nota.* Nel processo di compressione di una formula ad alta cardinalità con pesi positivi e grado di precisione $\delta$ in una compressa con cardinalità al massimo $N_\delta = \frac{(\delta+1)(\delta+2)}{2}$ dimensione di $\mathbb{P}_\delta$, è rilevante il seguente teorema:

*Teorema* 4.1. *Sia $\mu$ misura (multivariata) il cui supporto è un polinomio in $\mathbb{P}_\delta$ determinato da un insieme finito $X = \{\xi_i\} \subset \mathbb{R}^k$ e i corrispondenti pesi positivi $\lambda = \{\lambda_i\}_{i=1,\dots,M}$ con $M = card(X) > N_\delta$. Allora esiste una formula di cubatura per la misura discreta $\mu$, con nodi $T_\delta = \{t_j\} \subset X$ e pesi positivi $w = \{w_j\}_{j=1,\dots,m}$ con $m \leq N_\delta$, tale che*

$$\int_X p(x)\, d\mu = \sum_{i=1}^{M} \lambda_i\, p(\xi_i) = \sum_{j=1}^{m} w_i\, p(t_j) \qquad \forall p \in \mathbb{P}_\delta$$

Nel contesto della cubatura, un'intepretazione di questo teorema è:

se si ha $(X, \lambda)$ una formula a pesi positivi e nodi interni con $\delta$ grado di precisione e cardinalità $M > N_\delta$, allora si può comprimere in $(T_\delta, w)$ formula a pesi positivi e nodi interni con cardinalità al più $N_\delta$.

Per implementare questo risultato si consideri:

- $\{\phi_j\}$ base polinomiale di $\mathbb{P}_\delta$;
- $V = V_n(X) = \{\phi_j(\xi_i)\}$ con $i = 1, \dots, M$ e $j = 1, \dots, N_\delta$ simil matrice di Vandermonde;
- $\gamma = V^T\lambda$ vettore dei momenti della base $\{\phi_j\}$;
- $V^T u = \gamma$ sistema sottodeterminato (dei momenti $\gamma_i$).

Per il teorema (4.1) esiste una soluzione non negativa $u^*$ del sistema sottodeterminato $V^T u = \gamma$ le cui componenti non nulle sono al massimo $N_\delta$ (i.e. i pesi $\{w_j\}$) e sono determinate dai corrispondenti punti di campionamento $T_\delta = \{t_j\} \subset X$.

Esistono due approcci per ottenere le regole compresse:

1. *Programmazione Lineare (PL)*: consiste nel risolvere il sistema

$$\begin{cases} \min c^T u \\ V^T u = \gamma\ ,\ u \geq 0. \end{cases} \tag{4.1}$$

   I vincoli identificano un politopo in $\mathbb{R}^M$ e il vettore $c$ viene scelto indipendente dalle righe della matrice $V^T$ (per evitare che la funzione obiettivo sia costante nel politopo).

2. *Programmazione Quadratica (PQ)*: consiste nella risoluzione del problema ai minimi quadrati (*NonNegative Least Square problem*, NNLS)

$$\|V^T u^* - \gamma\|_2 = min\{\|V^T u - \gamma\|_2\ ,\ u \geq 0\}. \tag{4.2}$$

   Con l'applicazione di questo approccio si genera un residuo $\epsilon = \|V^T u^* - \gamma\|_2$ estremamente piccolo (dell'ordine di $10^{-14}$ per $\delta \leq 30$).

Negli esperimenti riportati in questa tesi si è ritenuto non necessario utilizzare formule di quadrature compresse.

Ecco uno schema di come si articoleranno le sezioni successive:

1. si considera prima un dominio $\Omega_i$ di tipo poligonale, poi una funzione $f$ almeno $C(\Omega_i)$.

2. I campionamenti di $f$ potranno essere (o meno) soggetti a rumore. In questo caso, si avranno $\{(x_j, f(x_j) + \epsilon_j)\}_{j=1,\dots,N}$ dati perturbati in cui $\{x_j\}_{j=1,\dots,N}$ sono i nodi di quadratura e $\epsilon_j$ le perturbazioni.

3. Mediante alcune varianti dell'iperinterpolazione, si approssima $f$ in $\Omega_i$.

4. Si riporterà in Tabelle gli errori commessi degli iperinterpolanti nell'approssimare la funzione $f$.

Una volta definiti questi elementi, l'algoritmo esegue una batteria di 100 tests. Di seguito valuterà gli errori compiuti tra funzione $f$ e iperinterpolante $p_n$ in un insieme fitto di punti di riferimento nel dominio, calcolando un approssimazione delle norme $\|f - p_n\|_\infty$, $\|f - p_n\|_2$. Nelle Tabelle esporremo le medie di questi valori nei 100 tests numerici eseguiti.

## 4.2 Esempi su un poligono convesso

Quale primo dominio $\Omega_1$ consideriamo il poligono convesso a 6 lati rappresentato nella Figura 4.1.

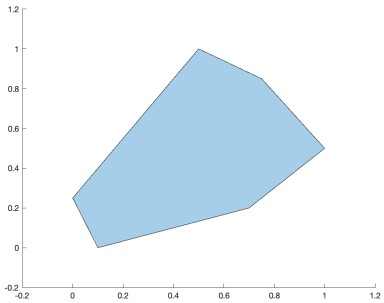



Figura 4.1: Dominio *convesso* $\Omega_1$.

Nella Tabella 4.1 abbiamo indicato la cardinalità delle formule composte utilizzate. Visto che intendiamo calcolare iperinterpolanti di grado $L = 3, 4, \dots, 10$ tali formule hanno grado algebrico di precisione $\delta = 2L$.

| $L$ | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|
| $M$ | 44 | 64 | 96 | 128 | 168 | 208 | 264 | 312 |

Tabella 4.1: Cardinalità $M$ della formula composta avente grado di precisione $\delta = 2L$, utile per calcolare l'iperinterpolante a grado $L$.

### 4.2.1 Polinomi di grado $L$ e $\lfloor \frac{L}{2} \rfloor - 1$

In prima istanza, consideriamo quale funzione $f$ un polinomio bivariato di grado $L$ della forma

$$f(x, y) = (c_0 + c_1 x + c_2 y)^L \tag{4.3}$$

con $\{c_i\}_{i=0,1,2}$ coefficienti random generati dalla funzione Matlab `rand(1)`. Come anticipato, il grado di approssimazione della formula $\delta$ varia al variare di $L$: nei test eseguiti $\delta = 2L$ e $L = 3, 4, \ldots, 10$.

| $L$ | $\mathcal{L}_L f$ | | $\mathcal{F}_{L,N} f$ | | $\mathcal{L}_L^\lambda f$ | | $\mathcal{H}_L^\lambda f$ | |
|---|---|---|---|---|---|---|---|---|
| | $\|\cdot\|_\infty$ | $\|\cdot\|_2$ | $\|\cdot\|_\infty$ | $\|\cdot\|_2$ | $\|\cdot\|_\infty$ | $\|\cdot\|_2$ | $\|\cdot\|_\infty$ | $\|\cdot\|_2$ |
| 3 | 2.77e-15 | 5.93e-16 | 1.24e-1 | 2.42e-2 | 1.96e-2 | 2.87e-3 | 1.31e-1 | 2.53e-2 |
| 4 | 4.63e-15 | 7.03e-16 | 9.58e-2 | 1.52e-2 | 8.36e-2 | 1.17e-2 | 1.36e-1 | 2.11e-2 |
| 5 | 7.55e-15 | 1.09e-15 | 7.28e-2 | 9.55e-3 | 2.33e-1 | 3.21e-2 | 2.53e-1 | 3.55e-2 |
| 6 | 2.95e-14 | 3.29e-15 | 7.89e-2 | 1.06e-2 | 1.35 | 1.77e-1 | 1.35 | 1.77e-1 |
| 7 | 4.77e-14 | 4.94e-15 | 1.06e-1 | 1.04e-2 | 2.68 | 3.37e-1 | 2.68 | 3.38e-1 |
| 8 | 9.47e-14 | 9.31e-15 | 9.35e-2 | 9.62e-3 | 7.33 | 9.07e-1 | 7.33 | 9.07e-1 |
| 9 | 2.29e-13 | 2.31e-14 | 8.11e-2 | 6.86e-3 | 1.60e | 1.89 | 1.60e | 1.89 |
| 10 | 3.47e-13 | 3.24e-14 | 5.06e-2 | 4.32e-3 | 2.53e | 2.79 | 2.53e | 2.79 |

Tabella 4.2: Approssimazione della funzione $f(x, y) = (c_0 + c_1 x + c_2 y)^L \in C(\Omega_1)$, tramite gli iperinterpolatori analizzati nei capitoli precedenti, in assenza di rumore. I parametri $\lambda^* = \lambda(10)$ selezionati variano tra $4.59e{-}4$ e $2.13e{-}1$.

*Nota.* Sottolineamo che i risultati potrebbero cambiare, non in maniera consistente, a causa della casualità di $c_0, c_1, c_2$.

Questa batteria di 100 esperimenti risulta un *test di esattezza* per l'iperinterpolante classico. Si è verificato infatti numericamente il teorema di proiezione ortogonale (1.1).

In Tabella 4.2 è esposto l'errore assoluto medio tra i vari test, sia in norma infinito (1.9) che in norma 2 (1.10). Risulta evidente che quello dell'iperinterpolante classica $\mathcal{L}_L$, a differenza di tutti gli altri, è molto vicino alla precisione di macchina (`eps=2.2204e-16`).

In seconda istanza, abbiamo considerato polinomi bivariati di grado $\lfloor \frac{L}{2} \rfloor - 1$ della forma

$$f(x, y) = (c_0 + c_1 x + c_2 y)^{L_1} \tag{4.4}$$

con $L_1 = \max(\lfloor \frac{L}{2} \rfloor - 1, 0)$ e, analogamente, $\{c_i\}_{i=0,1,2}$ coefficienti random generati dalla funzione Matlab `rand(1)`.
Variando il grado di approssimazione della formula si ottengono i risultati esposti nella Tabella 4.3.

Da questa si ottiene una dimostrazione numerica dell'*esattezza* dell'iperinterpolante classica e filtrata. In altri termini, $\mathcal{L}_L f$ e $\mathcal{F}_L f$ sono numericamente uguali a $f$, qualora questi sia un polinomio di grado $\leq \lfloor \frac{L}{2} \rfloor$ (e nel nostro caso $\lfloor \frac{L}{2} \rfloor - 1 < \lfloor \frac{L}{2} \rfloor$).

| $L$ | $\mathcal{L}_L f$ | | $\mathcal{F}_{L,N} f$ | | $\mathcal{L}_L^\lambda f$ | | $\mathcal{H}_L^\lambda f$ | |
|---|---|---|---|---|---|---|---|---|
| | $\|\cdot\|_\infty$ | $\|\cdot\|_2$ | $\|\cdot\|_\infty$ | $\|\cdot\|_2$ | $\|\cdot\|_\infty$ | $\|\cdot\|_2$ | $\|\cdot\|_\infty$ | $\|\cdot\|_2$ |
| 3 | 2.00e-15 | 4.40e-16 | 1.11e-15 | 4.19e-16 | 2.00e-15 | 4.40e-16 | 1.11e-15 | 4.19e-16 |
| 4 | 1.04e-15 | 2.42e-16 | 8.31e-16 | 2.10e-16 | 8.75e-16 | 2.24e-16 | 7.59e-16 | 2.08e-16 |
| 5 | 8.12e-16 | 1.85e-16 | 6.05e-16 | 1.80e-16 | 5.62e-16 | 1.75e-16 | 5.27e-16 | 1.74e-16 |
| 6 | 1.90e-15 | 2.54e-16 | 1.43e-15 | 2.33e-16 | 1.57e-15 | 2.62e-16 | 1.55e-15 | 2.61e-16 |
| 7 | 1.86e-15 | 2.61e-16 | 1.33e-15 | 2.31e-16 | 1.43e-15 | 2.49e-16 | 1.35e-15 | 2.46e-16 |
| 8 | 3.83e-15 | 6.14e-16 | 2.53e-15 | 5.36e-16 | 2.10e-2 | 3.07e-3 | 2.10e-2 | 3.07e-3 |
| 9 | 4.37e-15 | 5.93e-16 | 3.23e-15 | 5.33e-16 | 2.08e-2 | 3.04e-3 | 2.08e-2 | 3.04e-3 |
| 10 | 8.99e-15 | 8.45e-16 | 6.21e-15 | 7.70e-16 | 1.22e-1 | 1.72e-2 | 1.22e-1 | 1.72e-2 |

Tabella 4.3: Approssimazione della funzione $f(x, y) = (c_0 + c_1 x + c_2 y)^{L_1} \in C(\Omega_1)$, tramite gli iperinterpolatori analizzati nei capitoli precedenti, in assenza di rumore. I parametri $\lambda^* = \lambda(10)$ selezionati sono compresi tra $1.0e{-}16$ e $2.5e{-}3$.

Un discorso a parte deve essere fatto per l'interpolante di Lasso e ibrido: $\mathcal{L}_L^\lambda f$ e $\mathcal{H}_L^\lambda f$ sono esatti fino al grado 7 dopo il quale l'errore (sia in norma infinito che in norma 2) cresce allontanandosi dalla precisione di macchina. Quindi fino a $L = 7$ anche $\mathcal{L}_L^\lambda$ e $\mathcal{H}_\mathcal{L}^\lambda$ sono numericamente uguali a $f$.

*Nota.* Per scegliere il parametro di regolarizzazione $\lambda$ utilizzato per l'iperinterpolazione di Lasso e ibrida si è usato il breve codice

```
lambda_index=10;
lambdas=sort(abs(coeff0,'descend'))
lambdaL=lambdas(lambda_index)
```

dove `coeff0` è il vettore dei coefficienti dell'iperinterpolazione classica.

Quindi nelle tabelle, per $L \geq 3$, $\lambda$ è pari al decimo coefficiente di $\mathcal{L}_L$ per valore assoluto.

### 4.2.2 Funzione bivariata senza rumore

Quale ulteriore test approssimiamo numericamente la funzione bivariata

$$f(x,y) = (1 - x^2 - y^2)e^{x\cos(y)} \tag{4.5}$$

sul dominio convesso precedentemente indicato. Supponiamo che i campionamenti siano esenti da rumore. Si ottengono i risultati in Tabella 4.4.

Si nota che gli errori di tutte le approssimanti decrescono all'aumentare di $L$. In particolare

1. nel caso delle *Iperinterpolanti classiche o filtrate*, seppur con velocità diverse, gli errori decrescono fino ad arrivare alla precisione di macchine; infatti $\mathcal{L}_L f$ è dell'ordine di `eps` già per $L = 18$ mentre $\mathcal{F}_{L,N} f \approx f$ quando $L = 30$.

2. nel caso delle *Iperinterpolanti di tipo Lasso o ibrido* si può notare nelle tabelle che gli errori di $\mathcal{L}_L^\lambda$ e $\mathcal{H}_L^\lambda$ decrescono e tendono a stabilizzarsi allo stesso risultato da $L = 6$.

| $L$ | $\mathcal{L}_L f$ | | $\mathcal{F}_{L,N} f$ | | $\mathcal{L}_L^\lambda f$ | | $\mathcal{H}_L^\lambda f$ | |
|---|---|---|---|---|---|---|---|---|
| | $\|\cdot\|_\infty$ | $\|\cdot\|_2$ | $\|\cdot\|_\infty$ | $\|\cdot\|_2$ | $\|\cdot\|_\infty$ | $\|\cdot\|_2$ | $\|\cdot\|_\infty$ | $\|\cdot\|_2$ |
| 3 | 2.57e-2 | 3.66e-3 | 2.99e-1 | 3.82e-2 | 3.32e-2 | 4.22e-3 | 3.04e-1 | 3.89e-2 |
| 4 | 4.08e-3 | 5.22e-4 | 9.08e-2 | 9.80e-3 | 7.36e-2 | 8.73e-3 | 1.29e-1 | 1.35e-2 |
| 5 | 4.56e-4 | 6.81e-5 | 2.71e-2 | 2.98e-3 | 7.36e-2 | 8.72e-3 | 8.42e-2 | 9.44e-3 |
| 6 | 1.35e-4 | 1.25e-5 | 6.75e-3 | 9.88e-4 | 7.36e-2 | 8.72e-3 | 7.36e-2 | 8.72e-3 |
| 7 | 1.06e-5 | 1.25e-6 | 1.80e-3 | 2.75e-4 | 7.36e-2 | 8.72e-3 | 7.36e-2 | 8.72e-3 |
| 8 | 2.56e-6 | 2.08e-7 | 7.07e-4 | 8.36e-5 | 7.36e-2 | 8.72e-3 | 7.36e-2 | 8.72e-3 |
| 9 | 3.61e-7 | 2.52e-8 | 1.78e-4 | 2.40e-5 | 7.36e-2 | 8.72e-3 | 7.36e-2 | 8.72e-3 |
| 10 | 2.99e-8 | 2.87e-9 | 7.24e-5 | 7.74e-6 | 7.36e-2 | 8.72e-3 | 7.36e-2 | 8.72e-3 |
| 18 | 3.11e-15 | 4.90e-16 | 9.72e-9 | 8.30e-10 | 7.36e-2 | 8.72e-3 | 7.36e-2 | 8.72e-3 |
| 30 | 6.61e-15 | 6.04e-16 | 1.13e-14 | 9.55e-16 | 7.36e-2 | 8.72e-3 | 7.36e-2 | 8.72e-3 |

Tabella 4.4: Approssimazione della funzione $f(x,y) = (1 - x^2 - y^2)e^{x\cos(y)} \in C(\Omega_1)$, tramite gli iperinterpolatori analizzati nei capitoli precedenti, in assenza di rumore. Inoltre per $L \geq 4$ si è utilizzato $\lambda^* = 2.6e{-}3$, mentre per $L = 3$ $\lambda^* = 6.77e{-}4$.

### 4.2.3 Funzione bivariata con rumore

In questa sezione intendiamo *ricostruire* la funzione (4.5)

$$f(x,y) = (1 - x^2 - y^2)e^{x\cos(y)}$$

considerata già nella sezione precedente, qualora questa sia perturbata da un rumore di tipo *gaussiano*, cioè i campionamenti della funzione sono soggetti ad una perturbazione casuale la cui funzione di distribuzione coinciderà con quella di una variabile aleatoria normale $N(0, \sigma^2)$ con media $\mu = 0$ e deviazione standard $\sigma$.

Abbiamo effettuato una batteria di 100 esperimenti, a grado fissato $L = 15$, divisa in due fasi:

- fase 1: fissata la deviazione standard a $\sigma = 0.5$ riportiamo, nella prima parte della Tabella 4.5, la media degli errori in norma 2 effettuati dai vari iperinterpolanti al variare di $\lambda$ (parametro di regolarizzazione descritto nella sezione 3.2.1).

- fase 2: fissato $\lambda = \lambda^*$ riportiamo nella seconda parte della Tabella 4.5 la media degli errori in norma 2 effettuati dai vari iperinterpolanti al variare di $\sigma$.

| | $\sigma = 0.5$ con $\lambda^*$ | | | | $\lambda^* = \lambda(10) \approx 3.63e-2$ con $\sigma$ | | | |
|---|---|---|---|---|---|---|---|---|
| | $\lambda(10)$ | $\lambda(20)$ | $\lambda(40)$ | $\lambda(80)$ | 0.05 | 0.1 | 0.4 | 0.8 |
| *Classica* | 0.1880 | 0.1910 | 0.1880 | 0.1918 | 0.0196 | 0.0380 | 0.1481 | 0.3065 |
| *Filtrata* | 0.1293 | 0.1281 | 0.1304 | 0.1331 | 0.0135 | 0.0261 | 0.1025 | 0.2123 |
| *Lasso* | 0.0965 | 0.0823 | 0.0840 | 0.1237 | 0.0160 | 0.0255 | 0.0769 | 0.1471 |
| *Ibrida* | 0.0957 | 0.0778 | 0.0704 | 0.0910 | 0.0160 | 0.0254 | 0.0765 | 0.1461 |

Tabella 4.5: Approssimazione media dell'errore in norma 2 della funzione $f(x, y) = (1 - x^2 - y^2)e^{x\cos(y)} \in C(\Omega_1)$ in presenza di rumore gaussiano tramite gli iperinterpolatori presi in esame in questa tesi, tenendo fissata prima $\sigma$ e poi $\lambda^*$.

I risultati dimostrano quanto ci si aspettava dalla teoria esposta in [1]. Per valori di $\sigma$ superiori a 0.4, i metodi di tipo Lasso e Hybrid offrono migliori risultati rispetto l'interpolazione classica e filtrata, mentre per parametri minori quest'ultima risulta molto competitiva.

Aggiungiamo che, come indicato in precedenza, abbiamo fissato il parametro $\lambda^*$ uguale al decimo più grande coefficiente in modulo dell'iperinterpolante classica seppure altre scelte possano essere convenienti. Infatti per $\sigma = 0.5$ pure $\lambda(20)$ e $\lambda(40)$ hanno fornito ottimi risultati (cf. Tabella 4.5).

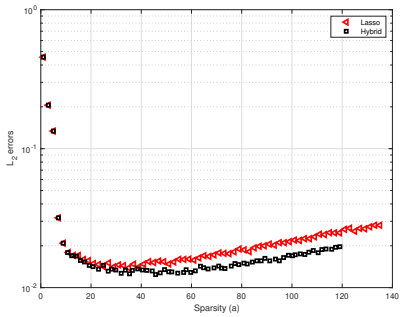



Figura 4.2: Rapporto tra l'errore in norma 2 e la sparsità per l'iperinterpolante di Lasso e ibrido in presenza di rumore gaussiano, nel caso del poligono convesso $\Omega_1$ e $f(x, y) = (1 - x^2 - y^2)e^{x\cos(y)}$.

In Figura 4.2 illustriamo il vantaggio in termini di sparsità del metodo ibrido rispetto a quello di tipo Lasso. Ció significa che con un errore comparabile, il numero di coefficienti non nulli del polinomio iperinterpolante è minore.

## 4.3 Esempi su un poligono concavo

Di seguito consideriamo il poligono concavo $\Omega_2$ avente 9 lati come in figura (4.3). Nella Tabella 4.6 abbiamo indicato la cardinalità $M$ delle formule composte utilizzate al variare del grado dell'iperinterpolante $L$.

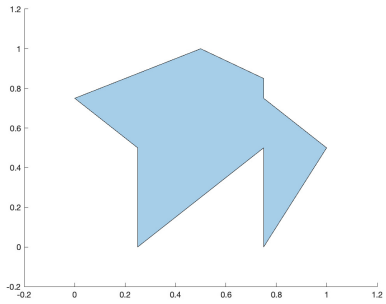


Figura 4.3: Il dominio *concavo* $\Omega_2$.

| $L$ | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|
| $M$ | 77 | 112 | 168 | 224 | 294 | 364 | 462 | 546 |

Tabella 4.6: Cardinalità $M$ della formula composta avente grado di precisione $2L$, utile per calcolare l'iperinterpolante a grado $L$.

### 4.3.1 Polinomi di grado $L$ e $\lfloor \frac{L}{2} \rfloor - 1$

Le funzioni che vogliamo approssimare in questa sezione, nel dominio $\Omega_2$ concavo, sono la famiglia di polinomi di grado $L$

$$f(x, y) = (c_0 + c_1 x + c_2 y)^L$$

e la famiglia di polinomi di grado $\lfloor \frac{L}{2} \rfloor - 1$

$$f(x, y) = (c_0 + c_1 x + c_2 y)^{\lfloor \frac{L}{2} \rfloor - 1}$$

dove i coefficienti $c_0$, $c_1$, $c_2$ sono numeri casuali.

Gli esperimenti sono stati effettuati in assenza di rumore, facendo variare il grado dell'iperinterpolante $L = 3, 4, \ldots, 10$ e riportando nelle Tabelle gli errori medi in norma infinito $\|\cdot\|_\infty$ e norma 2 $\|\cdot\|_2$ commessi dagli iperinterpolanti.

I dati della Tabella 4.7 riguardano l'approssimazione del polinomio di grado $L$, mentre quelli della Tabella 4.8 si riferiscono all'approssimazione del polinomio di grado $\lfloor \frac{L}{2} \rfloor - 1$.

| $L$ | $\mathcal{L}_L f$ | | $\mathcal{F}_{L,N} f$ | | $\mathcal{L}_L^\lambda f$ | | $\mathcal{H}_L^\lambda g$ | |
|---|---|---|---|---|---|---|---|---|
| | $\|\cdot\|_\infty$ | $\|\cdot\|_2$ | $\|\cdot\|_\infty$ | $\|\cdot\|_2$ | $\|\cdot\|_\infty$ | $\|\cdot\|_2$ | $\|\cdot\|_\infty$ | $\|\cdot\|_2$ |
| 3 | 3.01e-15 | 6.95e-16 | 1.07e-1 | 1.89e-2 | 2.07e-2 | 2.74e-3 | 1.15e-1 | 2.00e-2 |
| 4 | 3.78e-15 | 5.39e-16 | 7.56e-2 | 7.94e-3 | 5.92e-2 | 7.30e-3 | 1.01e-1 | 1.19e-2 |
| 5 | 1.01e-14 | 1.21e-15 | 6.14e-2 | 6.56e-3 | 2.43e-1 | 2.82e-2 | 2.64e-1 | 3.04e-2 |
| 6 | 1.68e-14 | 1.71e-15 | 5.12e-2 | 4.25e-3 | 6.54e-1 | 7.09e-2 | 6.58e-1 | 7.11e-2 |
| 7 | 4.54e-14 | 4.31e-15 | 5.32e-2 | 3.81e-3 | 2.31 | 2.47e-1 | 2.31 | 2.47e-1 |
| 8 | 1.42e-13 | 1.35e-14 | 6.98e-2 | 4.90e-3 | 8.70 | 9.01e-1 | 8.70 | 9.01e-1 |
| 9 | 2.66e-13 | 2.46e-14 | 9.91e-2 | 5.55e-3 | 2.28e | 2.20 | 2.28e | 2.20 |
| 10 | 7.43e-13 | 5.87e-14 | 7.35e-2 | 3.96e-3 | 5.22e | 5.11 | 5.22e | 5.11 |

Tabella 4.7: Approssimazione della funzione $f(x, y) = (c_0 + c_1 x + c_2 y)^L \in C(\Omega_2)$, tramite gli iperinterpolatori analizzati nei capitoli precedenti, in assenza di rumore. Inoltre i parametri $\lambda^*$ utilizzati sono compresi tra $5.12e{-}4$ e $6.23e{-}2$.

| $L$ | $\mathcal{L}_L f$ | | $\mathcal{F}_{L,N} f$ | | $\mathcal{L}_L^\lambda f$ | | $\mathcal{H}_L^\lambda f$ | |
|---|---|---|---|---|---|---|---|---|
| | $\|\cdot\|_\infty$ | $\|\cdot\|_2$ | $\|\cdot\|_\infty$ | $\|\cdot\|_2$ | $\|\cdot\|_\infty$ | $\|\cdot\|_2$ | $\|\cdot\|_\infty$ | $\|\cdot\|_2$ |
| 3 | 1.11e-15 | 3.22e-16 | 8.88e-16 | 3.15e-16 | 1.11e-15 | 3.22e-16 | 8.88e-16 | 3.15e-16 |
| 4 | 8.17e-16 | 1.40e-16 | 5.87e-16 | 1.30e-16 | 6.02e-16 | 1.29e-16 | 5.24e-16 | 1.26e-16 |
| 5 | 1.09e-15 | 1.71e-16 | 7.90e-16 | 1.55e-16 | 6.27e-16 | 1.43e-16 | 5.61e-16 | 1.42e-16 |
| 6 | 1.78e-15 | 2.23e-16 | 1.22e-15 | 2.00e-16 | 1.23e-15 | 2.02e-16 | 1.16e-15 | 2.01e-16 |
| 7 | 2.41e-15 | 2.63e-16 | 1.41e-15 | 2.16e-16 | 1.36e-15 | 2.39e-16 | 1.34e-15 | 2.38e-16 |
| 8 | 5.19e-15 | 4.75e-16 | 3.13e-15 | 4.05e-16 | 1.76e-2 | 2.33e-3 | 1.76e-2 | 2.33e-3 |
| 9 | 4.49e-15 | 4.73e-16 | 3.29e-15 | 4.33e-16 | 1.92e-2 | 2.55e-3 | 1.92e-2 | 2.55e-3 |
| 10 | 1.39e-14 | 1.06e-15 | 8.32e-15 | 9.61e-16 | 1.60e-1 | 1.99e-2 | 1.60e-1 | 1.99e-2 |

Tabella 4.8: Approssimazione della funzione $f(x,y) = (c_0 + c_1 x + c_2 y)^{L_1} \in C(\Omega_2)$, tramite gli iperinterpolatori analizzati nei capitoli precedenti, in presenza di rumore. I parametri $\lambda^*$ variano tra $1.0e{-}15$ e $4.2e{-}3$.

Risulta evidente che gli ordini di grandezza degli errori delle Tabelle 4.7 e 4.8 sono uguali rispettivamente a quelli delle Tabelle 4.2 e 4.3 della sezione 4.2.1. Quindi le considerazione sull'approssimazione di $f$ di grado $L$ e $\lfloor \frac{L}{2} \rfloor - 1$ su un dominio concavo sono del tutto analoghe a quelle su un dominio convesso:

- se $f \in \mathbb{P}_L$ allora $\mathcal{L}_L f \approx f$;
- se $f \in \mathbb{P}_{\lfloor \frac{L}{2} \rfloor - 1}$ allora $\mathcal{F}_{L,N} f \approx f$;
- se $f \in \mathbb{P}_{\lfloor \frac{L}{2} \rfloor - 1}$ con $L \leq 7$, allora $\mathcal{L}_L^\lambda f \approx f$ e $\mathcal{H}_L^\lambda f \approx f$.

### 4.3.2 Funzione bivariata senza rumore

Per parallelismo con la Sezione 4.2.2, approssimiamo in $\Omega_2$ per iperinterpolazione la funzione (4.5)

$$f(x,y) = (1 - x^2 - y^2) e^{x \cos(y)}$$

esente da rumore. In Tabella 4.9 riferiamo gli errori medi che compiono i vari interpolanti al variare di $L = 3, 4, \ldots, 10$.

Emergono le seguenti analogie con la Tabella 4.4 in cui $f$ è stata approssimata in $\Omega_1$ convesso:

- tutti gli errori decrescono (non strettamente) all'aumentare del grado di iperinterpolazione;

- L'errore dell'iperinterpolante classico raggiunge la precisione di macchina in modo più rapido rispetto all'errore di quello filtrato, alla quale entrambi convergono.

- gli interpolanti di Lasso e ibrido non raggiongono mai un errore dell'ordine $10^{-16}$, ma la norma infinito e la norma 2 convergono rispettivamente a `4.68e-2` e `6.14e-3` (stessi valori del caso convesso).

| $L$ | $\mathcal{L}_L f$ | | $\mathcal{F}_{L,N} f$ | | $\mathcal{L}_L^\lambda f$ | | $\mathcal{H}_L^\lambda f$ | |
|---|---|---|---|---|---|---|---|---|
| | $\|\cdot\|_\infty$ | $\|\cdot\|_2$ | $\|\cdot\|_\infty$ | $\|\cdot\|_2$ | $\|\cdot\|_\infty$ | $\|\cdot\|_2$ | $\|\cdot\|_\infty$ | $\|\cdot\|_2$ |
| 3 | 3.01e-2 | 3.08e-3 | 2.49e-1 | 2.80e-2 | 3.50e-2 | 4.30e-3 | 2.54e-1 | 2.89e-2 |
| 4 | 3.64e-3 | 2.91e-4 | 8.02e-2 | 8.78e-3 | 4.68e-2 | 6.14e-3 | 1.00e-1 | 1.12e-2 |
| 5 | 1.0e-3 | 4.90e-5 | 2.51e-2 | 2.56e-3 | 4.68e-2 | 6.14e-3 | 5.77e-2 | 6.89e-3 |
| 6 | 7.24e-5 | 5.07e-6 | 6.34e-3 | 7.97e-4 | 4.68e-2 | 6.14e-3 | 4.71e-2 | 6.18e-3 |
| 7 | 2.40e-5 | 6.62e-7 | 1.85e-3 | 1.92e-4 | 4.68e-2 | 6.14e-3 | 4.68e-2 | 6.14e-3 |
| 8 | 1.26e-6 | 8.56e-8 | 7.39e-4 | 4.87e-5 | 4.68e-2 | 6.14e-3 | 4.68e-2 | 6.14e-3 |
| 9 | 5.44e-7 | 1.06e-8 | 2.79e-4 | 1.52e-5 | 4.68e-2 | 6.14e-3 | 4.68e-2 | 6.14e-3 |
| 10 | 2.16e-8 | 1.34e-9 | 7.39e-5 | 4.98e-6 | 4.68e-2 | 6.14e-3 | 4.68e-2 | 6.14e-3 |

Tabella 4.9: Approssimazione della funzione $f(x,y) = (1-x^2-y^2)e^{x\cos(y)} \in C(\Omega_2)$, tramite gli iperinterpolatori analizzati nei capitoli precedenti in assenza di rumore. I parametri $\lambda^*$ sono tutti uguali a $1.8e{-}3$ per $L \geq 4$, mentre per $L = 3$ si ha $\lambda^* = 9.48e{-}4$.

### 4.3.3 Funzioni bivariate con rumore

Similmente a quanto già fatto per il precedente dominio convesso, approssimiamo su $\Omega_2$ la funzione esponenziale (4.5):

$$f(x,y) = (1-x^2-y^2)e^{x\cos(y)}$$

inserendo del rumore nella valuatazione di $f$ nei nodi di quadratura.

Nella batteria di esperimenti riferiti nella Tabella 4.10 il grado è stato fissato a $L = 15$. Il rumore introdotto è di tipo *impulsivo*: cioè la perturbazione a cui sono soggetti i campionamenti della funzione ha una funzione di distribuzione di una variabile aleatoria $U(\alpha)$ con valori random in $[-\alpha, \alpha]$ e probabilità $1/2$.

Nella prima parte della Tabella si fa variare il parametro di regolarizzazione $\lambda^* \in \{\lambda(10), \lambda(20), \lambda(40), \lambda(80)\}$ e si fissa il livello $\alpha = 0.5$.

Nella seconda parte, invece, si fissa $\lambda^* = \lambda(10)$ e si fa variare $\alpha$.

| | $\alpha = 0.5$ con $\lambda^*$ | | | | $\lambda^* = \lambda(10) \approx 1.41e-2$ con $\alpha$ | | | |
|---|---|---|---|---|---|---|---|---|
| | $\lambda(10)$ | $\lambda(20)$ | $\lambda(40)$ | $\lambda(80)$ | 0.05 | 0.1 | 0.4 | 0.8 |
| *Classica* | 0.0770 | 0.0773 | 0.0785 | 0.0775 | 0.0079 | 0.0155 | 0.0633 | 0.1269 |
| *Filtrata* | 0.0528 | 0.0535 | 0.0548 | 0.0553 | 0.0056 | 0.0108 | 0.0436 | 0.0876 |
| *Lasso* | 0.0393 | 0.0335 | 0.0372 | 0.0500 | 0.0075 | 0.0106 | 0.0313 | 0.0619 |
| *Ibrida* | 0.0391 | 0.0320 | 0.0319 | 0.0386 | 0.0075 | 0.0106 | 0.0311 | 0.0612 |

Tabella 4.10: Approssimazione media dell'errore in norma 2 della funzione $f(x,y) = (1-x^2-y^2)e^{x\cos(y)} \in C(\Omega_2)$ in presenza di rumore impulsivo tramite gli iperinterpolatori presi in esame in questa tesi, tenendo fissata prima $\alpha$ e poi $\lambda^*$.

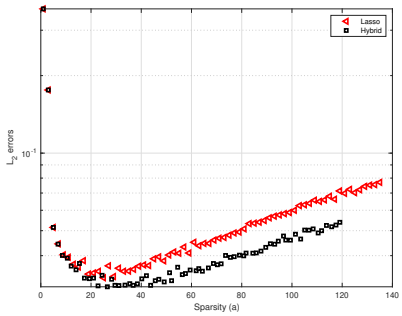



Figura 4.4: Rapporto tra l'errore in norma 2 e la sparsità per l'iperinterpolante di Lasso e ibrido in presenza di rumore impulsivo. Si evince che a parità di $\lambda$ quello della ibrida è generalmente migliore.

I risultati della Tabella 4.10 e della Figura 4.4 dimostrano quanto ci si aspettava dalla teoria, ovvero che la interpolante ibrida $\mathcal{H}_L^\lambda f$ a parità di errore ha maggiore sparsità rispetto a quella di tipo Lasso.

Si vede peraltro la ottima performance dell'iperinterpolante filtrata, paragonabile a quella di tipo Lasso e ibrida.

## 4.4 Dominio di tipo poligonale *non-semplice*

L'ultimo dominio che analizziamo $\Omega_3$ è il simil-quadrifoglio rappresentato in Figura 4.5, un poligono non semplice che presenta un'autointersezione in $(0,0)$.

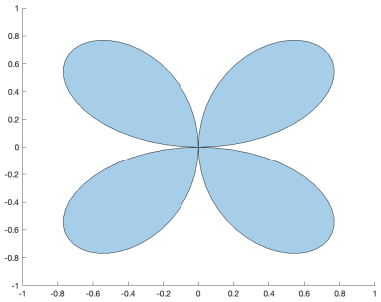


Figura 4.5: Dominio *non-semplice* $\Omega_3$.

Intendendo calcolare iperinterpolanti di grado $L = 3, 4, \ldots, 10$, ricordiamo che risulta necessario utilizzare formule con grado di precisione $2L$. Le formule di cubatura composte utilizzate richiedono il numero di nodi esposti in Tabella 4.11.

| $L$ | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|
| $M$ | 1342 | 1952 | 2928 | 3904 | 5124 | 6344 | 8052 | 9516 |

Tabella 4.11: Cardinalità $M$ della formula composta avente grado di precisione $\delta = 2L$, utile per calcolare l'iperinterpolante a grado $L$.

### 4.4.1 Polinomi di grado $L$ e $\lfloor \frac{L}{2} \rfloor - 1$

Come nelle sezioni precedenti, abbiamo considerato quale test una *batteria* di polinomi del tipo $(c_0 + c_1 \cdot x + c_2 \cdot y)^k$, con $c_0$, $c_1$, $c_2$ numeri random e $k$ rispettivamente uguale a $L$ e $\lfloor \frac{L}{2} \rfloor - 1$ .
Negli esperimenti non è stato introdotto rumore.

Quali errori, abbiamo considerato per ogni interpolante la media di quelli effettuati nei vari tests.

Dalle Tabelle (4.12) e (4.13) emorgono delle analogie (e non) con l'approssimazione delle stesse funzioni (sempre in assenza di rumore) nei poligoni $\Omega_1$ e $\Omega_2$ analizzati nelle Sezioni 4.2.1 e 4.3.1:

1. analogie: l'approssimazione migliore è data dall'iperinterpolante classico $\mathcal{L}_L f$;

2. differenze: l'errore aumenta (strettamente) all'aumentare del grado $L$ per tutti gli iperinterpolanti, però, se la crescita dell'errore in norma infinito e 2 di $\mathcal{F}_{L,N}f$ è contenuta, lo stesso non vale per $\mathcal{L}_L^\lambda f$ e $\mathcal{H}_L^\lambda f$.

| $L$ | $\mathcal{L}_L f$ | | $\mathcal{F}_{L,N} f$ | | $\mathcal{L}_L^\lambda f$ | | $\mathcal{H}_L^\lambda f$ | |
|---|---|---|---|---|---|---|---|---|
| | $\|\cdot\|_\infty$ | $\|\cdot\|_2$ | $\|\cdot\|_\infty$ | $\|\cdot\|_2$ | $\|\cdot\|_\infty$ | $\|\cdot\|_2$ | $\|\cdot\|_\infty$ | $\|\cdot\|_2$ |
| 3 | 1.44e-15 | 3.02e-16 | 3.47e-1 | 8.56e-2 | 1.35e-1 | 2.85e-2 | 4.08e-1 | 9.36e-2 |
| 6 | 1.69e-14 | 2.03e-15 | 5.19e-1 | 7.18e-2 | 3.47 | 5.17e-1 | 3.48 | 5.18e-1 |
| 9 | 1.20e-13 | 1.09e-14 | 1.27 | 1.14e-1 | 3.66e | 4.71 | 3.66e | 4.71 |
| 12 | 9.23e-13 | 5.45e-14 | 1.64 | 1.20e-1 | 2.24e2 | 2.66e | 2.24e2 | 2.66e |

Tabella 4.12: Approssimazione della funzione $f(x,y) = (c_0 + c_1 x + c_2 y)^L \in C(\Omega_3)$, tramite gli interpolanti presi in esame nei capitoli precedenti, in assenza di rumore. I parametri $\lambda^*$ selezionati variano tra $3.64e{-}4$ e 3.38.

| $L$ | $\mathcal{L}_L f$ | | $\mathcal{F}_{L,N} f$ | | $\mathcal{L}_L^\lambda f$ | | $\mathcal{H}_L^\lambda f$ | |
|---|---|---|---|---|---|---|---|---|
| | $\|\cdot\|_\infty$ | $\|\cdot\|_2$ | $\|\cdot\|_\infty$ | $\|\cdot\|_2$ | $\|\cdot\|_\infty$ | $\|\cdot\|_2$ | $\|\cdot\|_\infty$ | $\|\cdot\|_2$ |
| 3 | 3.33e-16 | 1.07e-16 | 2.22e-16 | 8.40e-17 | 3.33e-16 | 1.07e-16 | 2.22e-16 | 8.36e-17 |
| 6 | 1.38e-15 | 3.99e-16 | 1.29e-15 | 3.91e-16 | 1.04e-15 | 3.65e-16 | 1.04e-15 | 3.64e-16 |
| 9 | 2.91e-15 | 6.77e-16 | 2.69e-15 | 6.62e-16 | 1.31e-1 | 2.76e-2 | 1.31e-1 | 2.76e-2 |
| 12 | 9.74e-15 | 1.55e-15 | 9.45e-15 | 1.53e-15 | 1.85 | 2.93e-1 | 1.85 | 2.93e-1 |

Tabella 4.13: Approssimazione della funzione $f(x,y) = (c_0 + c_1 x + c_2 y)^{L_1} \in C(\Omega_3)$ tramite gli iperinterpolatori analizzati nei capitoli precedenti, in assenza di rumore. I parametri $\lambda^*$ selezionati variano tra $1.0\,e{-}15$ e $1.0\,e{-}14$.

*Nota.* Dal confronto con gli errori copiuti dagli iperinterpolanti di Lasso e ibrido nell'approssimare tali polinomi nei domini $\Omega_1$ e $\Omega_2$, emerge che nel caso di $\Omega_3$ non semplice l'errore diventa notevolmente maggiore.

In generale in assenza di rumore si puó concludere che $\mathcal{L}_L^\lambda$ e $\mathcal{H}_L^\lambda$ non sono performarti.

### 4.4.2 Funzione bivariata senza rumore

Dopo aver analizzato il caso di polinomi bivariati di tipo *random*, approssiamo in $\Omega_3$ la funzione esponenziale (4.5) già considerata in precedenza

$$f(x,y) = (1 - x^2 - y^2)e^{x\cos(y)}.$$

| $L$ | $\mathcal{L}_L f$ | | $\mathcal{F}_{L,N} f$ | | $\mathcal{L}_L^\lambda f$ | | $\mathcal{H}_L^\lambda f$ | |
|---|---|---|---|---|---|---|---|---|
| | $\|\cdot\|_\infty$ | $\|\cdot\|_2$ | $\|\cdot\|_\infty$ | $\|\cdot\|_2$ | $\|\cdot\|_\infty$ | $\|\cdot\|_2$ | $\|\cdot\|_\infty$ | $\|\cdot\|_2$ |
| 3 | 7.41e-2 | 2.27e-2 | 5.09e-1 | 1.54e-1 | 7.41e-2 | 2.27e-2 | 5.09e-1 | 1.54e-1 |
| 4 | 4.05e-2 | 8.57e-3 | 2.04e-1 | 6.88e-2 | 4.05e-2 | 8.57e-3 | 2.04e-1 | 6.88e-2 |
| 5 | 1.24e-2 | 2.56e-3 | 6.53e-2 | 2.04e-2 | 3.55e-2 | 1.29e-2 | 8.52e-2 | 2.61e-2 |
| 6 | 2.04e-3 | 4.27e-4 | 2.46e-2 | 8.46e-3 | 3.55e-2 | 1.29e-2 | 4.58e-2 | 1.58e.2 |
| 7 | 8.01e-4 | 1.13e-4 | 1.81e-2 | 3.95e-3 | 3.55e-2 | 1.29e-2 | 3.81e-2 | 1.37e-2 |
| 8 | 2.17e-4 | 3.44e-5 | 1.01e-2 | 1.78e-3 | 3.55e-2 | 1.29e-2 | 3.59e-2 | 1.31e-2 |
| 9 | 2.63e-5 | 4.29e-6 | 4.03e-3 | 7.27e-4 | 3.55e-2 | 1.29e-2 | 3.56e-2 | 1.29e-2 |
| 10 | 1.18e-5 | 1.40e-6 | 1.43e-3 | 2.91e-4 | 3.55e-2 | 1.29e-2 | 3.55e-2 | 1.29e-2 |

Tabella 4.14: Approssimazione della funzione $f(x,y) = (1-x^2-y^2)e^{x\cos(y)} \in C(\Omega_3)$, tramite gli iperinterpolatori presi in esame nei capitoli precendenti, in assenza di rumore. I parametri $\lambda^*$ selezionati sono $3.9\,e-3$ per $L=5$ in poi, mentre per $L \in \{3,4\}$ è dell'ordine di $10^{-10}$.

I risultati riferiti in Tabella 4.14 sono esenti da rumore.

Il comportamento dell'errore dato dall'approssimazione della funzione mediante i vari iperinterpolatori è coerente con l'analisi delle Tabelle (4.4) e (4.9). In particolare l'iperinterpolante classica, già a grado 10, fornisce una buona ricostruzione di $f$. Si nota peraltro che le interpolanti di tipo Lasso e ibrido numericamente *stallano* ad un errore in norma 2 di circa $10^{-2}$.

### 4.4.3 Funzione bivariata con rumore

Si prenda ora in esame la stessa

$$f(x,y) = (1-x^2-y^2)e^{x\cos(y)}$$

con l'aggiunta del rumore *gaussiano* o *impulsivo*. In questo modo si hanno $\{(x_j, f(x_j)+\epsilon_j)\}_{j=1,\dots,N}$ dati perturbati in cui $\{x_j\}_{j=1,\dots,N}$ nodi di quadratura e $\epsilon_j$ perturbazione dei dati che sta in $N(0,\sigma^2)$ (primo esperimento) o in $U(\alpha)$ (secondo esperimento).

Abbiamo effettuato 100 test numerici per ricostruire $f$, soggetta a rumore variabile da esperimento a esperimento, fissando il grado di iperinterpolazione uguale a $L=15$ (e quindi utilizzando formule aventi grado di precisione $\delta = 30$).

Si riportano di seguito prima le tabelle dei due esperimenti e poi due grafici che dimostrano la qualità dell'approssimazione data dall'iperinterpolante ibrida.

| | $\sigma = 0.5$ con $\lambda^*$ | | | | $\lambda^* = \lambda(10) \approx 1.24e-2$ con $\sigma$ | | | |
|---|---|---|---|---|---|---|---|---|
| | $\lambda(10)$ | $\lambda(20)$ | $\lambda(40)$ | $\lambda(80)$ | 0.05 | 0.1 | 0.4 | 0.8 |
| *Classica* | 0.0606 | 0.0596 | 0.0608 | 0.0611 | 0.0061 | 0.0122 | 0.0485 | 0.0964 |
| *Filtrata* | 0.0424 | 0.0412 | 0.0417 | 0.0431 | 0.0042 | 0.0085 | 0.0340 | 0.0677 |
| *Lasso* | 0.0371 | 0.0305 | 0.0293 | 0.0398 | 0.0128 | 0.0142 | 0.0307 | 0.0563 |
| *Ibrida* | 0.0370 | 0.0296 | 0.0257 | 0.0306 | 0.0128 | 0.0142 | 0.0306 | 0.0561 |

Tabella 4.15: Approssimazione media dell'errore in norma 2 della funzione $f(x, y) = (1 - x^2 - y^2)e^{x\cos(y)} \in C(\Omega_3)$ in presenza di rumore gaussiano tramite gli iperinterpolatori presi in esame in questa tesi, tenendo fissata prima $\sigma$ e poi $\lambda^*$.

| | $\alpha = 0.5$ con $\lambda^*$ | | | | $\lambda^* = \lambda(10) \approx 7.98e-3$ con $\alpha$ | | | |
|---|---|---|---|---|---|---|---|---|
| | $\lambda(10)$ | $\lambda(20)$ | $\lambda(40)$ | $\lambda(80)$ | 0.05 | 0.1 | 0.4 | 0.8 |
| *Classica* | 0.0356 | 0.0345 | 0.0351 | 0.0346 | 0.0035 | 0.0069 | 0.0285 | 0.0557 |
| *Filtrata* | 0.0249 | 0.0240 | 0.0241 | 0.0243 | 0.0025 | 0.0049 | 0.0199 | 0.0394 |
| *Lasso* | 0.0262 | 0.0191 | 0.0176 | 0.0228 | 0.0131 | 0.0130 | 0.0221 | 0.0351 |
| *Ibrida* | 0.0262 | 0.0187 | 0.0157 | 0.0174 | 0.0131 | 0.0130 | 0.0221 | 0.0351 |

Tabella 4.16: Approssimazione media dell'errore in norma 2 della funzione $f(x, y) = (1 - x^2 - y^2)e^{x\cos(y)} \in C(\Omega_3)$ in presenza di rumore impulsivo tramite gli iperinterpolatori presi in esame in questa tesi, tenendo fissata prima $\alpha$ e poi $\lambda^*$.

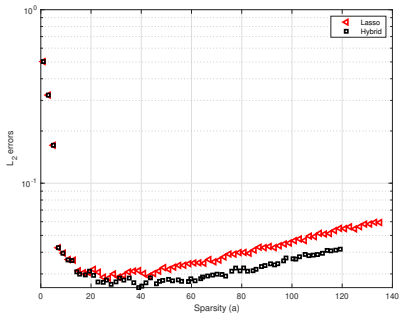



Figura 4.6: Rapporto tra l'errore in norma 2 e la sparsità per l'iperinterpolante di Lasso e ibrido in presenza di rumore gaussiano. Si evince che a parità di $\lambda$ quello della ibrida è generalmente migliore.

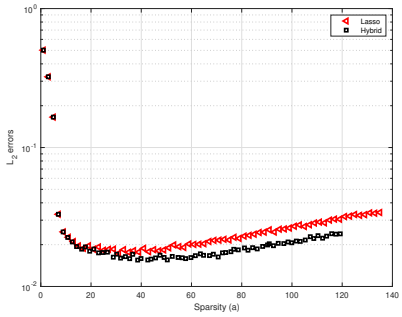



Figura 4.7: Rapporto tra l'errore in norma 2 e la sparsità per l'iperinterpolante di Lasso e ibrido in presenza di rumore impulsivo. Si evince che a parità di $\lambda$ quello della ibrida è generalmente migliore.

Come in precedenza, si vede che gli interpolanti di tipo Lasso e Ibrido offrono ottimi risultati in caso di rumore significativo $\sigma \geq 0.4$. Pure in questo esempio, le Figure 4.6 e 4.7 mostrano il vantaggio in termini di sparsità dell'iperinterpolante ibrida.

# Appendice A

# Codici

I codici Matlab utilizzati, ed eventuali algoritmi di supporto, sono resi disponibili *open-source* sulla piattaforma *GITHUB* [6].

Qui di seguito sono riportati le due routines principali per la realizzazione di questa tesi.

La prima, `demo_polygon`, è stata funzionale nella costruzione delle tabelle senza rumore, facendo variare il grado dell'iperinterpolante $LV$.

La routine, `demo_polygon_driver_02`, è stata funzionale nella costruzione delle tabelle (e grafici) delle funzioni bivariate in presenza di rumore nei 3 domini poligonali esaminati.

```
function [AEinfMV,AE2MV,beta0MV,lambdaV,XYW,XYWR,JzMV,HzMV,
    Wg_norm2]=...
     demo_polygon(lambda_index,a,sigma,XYW,XYWR)

%
% OBJECT
%
% Numerical experiment in "Hybrid hyperinterpolation over
    general regions".
% Region: polygon.
%
% Usage:
% >> demo_polygon
%
% Note:
% The routine uses 'binornd' that requires Statistics and
    Machine Learning
% Toolbox.
%
% Dates:
```

```
% Written on January 1, 2023: A. Sommariva.
% Modified on April 22, 2023: A. Sommariva.
%
% COPYRIGHT
%
% Copyright (C) 2023-
%
% Authors:
% Alvise Sommariva
%
% This program is free software: you can redistribute it and/
   or modify
% it under the terms of the GNU General Public License as
   published by
% the Free Software Foundation, either version 3 of the
   License, or
% (at your option) any later version.
%
% This program is distributed in the hope that it will be
   useful,
% but WITHOUT ANY WARRANTY; without even the implied warranty
    of
% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See
   the
% GNU General Public License for more details.
%
% You should have received a copy of the GNU General Public
   License
% along with this program.  If not, see <https://www.gnu.org/
   licenses/>.
%



%
% Degrees of precision in numerical experiments: can be a
   vector.
%
LV=15;      % Hyperinterpolant tot degree.
NV=2*LV;    % Degree of the rule used in hyperinterpolation.
NR=40;      % Degree of precision of the reference rule (
   estimate L2 error).


%
% Noise and choice of lasso, hybrid, hard threshold parameter
   .
%

noise=0;
```

```
if noise

    if nargin <2,a=0.02;end     % defining impulse noise (in
   experiment 2)
    if nargin <3,sigma=0.75;end % defining gaussian noise (in
    experiment 2)
else
    a=0; sigma=0; % no noise
end

% * Function to approximate:
% 1. degree L poly., 2. degree floor(L/2)-1 poly. 3. test
   functions
% (see line 324 approx).
funct_example=2;


% No table or stats.
display_stats=1;

%
% Special settings.
%

% Plot domain and nodes: do_plot=1 (yes), do_plot=0 (no).
do_plot=1;

% Number of tests for reconstructing functions of a type on
   this domain.
ntests=100;

if nargin < 1, lambda_index=10; end



% ......................... Main code below
   ..............................


% ......................... Define domain
   ..............................

[xv,yv]=example_polygon(1);


% ........ Numerical approximation, varying the degree in "nV
   " ............
```

```
AEinfMV=[]; AE2MV=[]; beta0MV=[]; % vectors used for
   statistics
JzMV=[]; HzMV=[];

for k=1:length(NV)
    N=NV(k); % Quadrature points.
    L=LV(k); % Hyperinterpolant degree.


    % Define quadrature rule for hyperinterpolation at L,
   with  ADE=N.
    if nargin < 4, XYW=cub_polygon(N,xv,yv); end
    if isempty(XYW), XYW=cub_polygon(N,xv,yv); end
    X=XYW(:,1); Y=XYW(:,2); W=XYW(:,3);

    % Test points
    if nargin < 5, XYWR=cub_polygon(NR,xv,yv); end
    if isempty(XYWR), XYWR=cub_polygon(NR,xv,yv); end
    XR=XYWR(:,1); YR=XYWR(:,2); WR=XYWR(:,3);


    % Compute bounding box
    xmin=min(xv); xmax=max(xv);
    ymin=min(yv); ymax=max(yv);
    dbox=[xmin xmax ymin ymax];

    % Compute orthonormal basis matrix at nodes.
    jvec=1:(L+1)*(L+2)/2;
    [U,~,Q,R,~,degs] = dORTHVAND(L,[X Y],W,jvec,[],dbox);

    % .. testing AE_L2err hyperinterpolation error for each "
   f" at "deg" ..
    poly_coeffs=[];
    lambdaV=[];

    for j=1:ntests

        % ... define function to approximate ...
        g=define_function(funct_example,L);

        % ... evaluate function to approximate ...
        gXY=feval(g,X,Y);

        % ... add noise (if present) ...

        % a) add impulse noise
        pert_impulse=0; %inizializzo il rumore impulsivo
        if a > 0
            pert_impulse=a*(1-2*rand(length(gXY),1))*binornd
```

```
(1,0.5);
        while norm(pert_impulse) == 0
            pert_impulse=a*(1-2*rand(length(gXY),1))*
binornd(1,0.5);
        end
    end

    % b) add gaussian noise
    pert_gauss=0; %inizializzo il rumore gaussiano
    if sigma > 0
        var=sigma^2;
        pert_gauss=sqrt(var)*randn(size(gXY));
        while norm(pert_gauss) == 0
            pert_gauss=sqrt(var)*randn(size(gXY));
        end
    end

    % add gaussian + impulse noise
    pert=pert_impulse+pert_gauss; %sommo i rumori

    % perturbed values
    gXY_pert=gXY+pert; %funzione+rumore=funzione
perturbata

    % ... determine polynomial hyperinterpolant ...
    % compute hyperinterpolant coefficients
    coeff0=Q'*(sqrt(W).*gXY_pert);

    % ... test hyperinterpolant with or withour filters
...

    lambdas=sort(abs(coeff0),'descend');
    lambdaL=lambdas(lambda_index);

    for ktest=[1 2 3 4 5 6] %questo ciclo for mi da in
tabella tutti i casi
        switch ktest
            case 1
                hypermode='tikhonov';
                parms.lambda=lambdaL;
                parms.mu=[];
                parms.b=ones(size(coeff0));
                coeff=hyperfilter(hypermode,coeff0,degs,
parms);
            case 2
                hypermode='filtered';
                parms.lambda=[];
                parms.mu=[];
                parms.b=[];
```

```
                coeff=hyperfilter(hypermode,coeff0,degs,
parms);
            case 3
                hypermode='lasso';
                parms.lambda=lambdaL;
                parms.mu=ones(size(coeff));
                parms.b=[];
                coeff=hyperfilter(hypermode,coeff0,degs,
parms);
            case 4
                hypermode='hybrid';
                parms.lambda=lambdaL;
                parms.mu=ones(size(coeff0));
                parms.b=ones(size(coeff0));
                parms.w=W;
                parms.pert=pert;
                parms.hybrid=0; % establishes it is a pre
-choosen parameter.
                coeff=hyperfilter(hypermode,coeff0,degs,
parms);
            case 5
                hypermode='hard';
                parms.lambda=lambdaL;
                parms.mu=[];
                parms.b=[];
                coeff=hyperfilter(hypermode,coeff0,degs,
parms);
            case 6
                hypermode='hyperinterpolation';
                parms.lambda=[];
                parms.mu=[];
                parms.b=[];
                coeff=coeff0;
        end

        % evaluate polynomial at reference points.
        gXYR=feval(g,XR,YR);
        VR=chebvand(L,[XR YR],dbox); %chebvand mi fa base
 di Chebyshev
        pXYR = (VR(:,jvec)/R)*coeff;

        % errors
        AEinfV(ktest,j)=norm(gXYR-pXYR,inf); % absolute
error (inf norm)
        AE2V(ktest,j)=sqrt(WR'*((gXYR-pXYR).^2)); %
absolute error (2 norm)
        beta0V(ktest,j)=sum(abs(coeff) > 0);
                    % evaluate J(coeff) and H(coeff),
that are error relevant
```

```
            % parameters, as observed in Thm 5.1.

            JzV(ktest,j)=evaluate_J(coeff,coeff0);
            HzV(ktest,j)=evaluate_H(coeff,W,pert,U);

        end

        lambdaV=[lambdaV lambdas];


    end



    % averages of the errors (vectors 5 x 1)
    AEinfM=mean(AEinfV,2);
    AE2M=mean(AE2V,2);
    beta0M=mean(beta0V,2);
    JzM=mean(JzV,2);
    HzM=mean(HzV,2);

    if display_stats
    fprintf('\n        ........ table at degree: %2.0f
   ........ \n \n ',N);
    HypType=categorical({ 'tikhonov'; 'filtered'; 'lasso'; '
   hybrid'; ...
         'hard'; 'hyperint.'});
    T = table(HypType,AEinfM,AE2M,beta0M,JzM,HzM); disp(T)
    end

    AEinfMV=[AEinfMV AEinfM]; AE2MV=[AE2MV AE2M]; beta0MV=[
   beta0MV beta0M];
    JzMV=[JzMV JzM]; HzMV=[HzMV HzM];

end

Wg_norm2=(norm(sqrt(W).*gXY,2))^2;









function [xv,yv,iv]=example_polygon(example)
```

```
switch example
    case 1
        polygon_sides=[0.1 0; 0.7 0.2; 1 0.5; 0.75 0.85; 0.5
   1; 0 0.25; 0.1 0];
        xv=polygon_sides(:,1); yv=polygon_sides(:,2);
        iv=length(xv); % This variable depends on the holes
   or not connected domain.
        % In these simple cases the domains are without holes
    and
        % domains are connected.

    case 2
        polygon_sides=(1/4)*[1 2; 1 0; 3 2; 3 0; 4 2; 3 3; 3
   0.85*4; 2 4;
             0 3; 1 2];
        xv=polygon_sides(:,1); yv=polygon_sides(:,2);
        iv=length(xv); % This variable depends on the holes
   or not connected domain.
        % In these simple cases the domains are without holes
    and
        % domains are connected.
    case 3
        % fprintf('\n \t [POLYGON]: QUATERFOIL LIKE');
        warning off;
        M=129;
        th=linspace(0,2*pi,M);
        %th=(th(1:end-1))';
        polygon_sides=[cos(th').*(sin(2*th')) sin(th').*(sin
   (2*th'))];
        polygon_sides=polygon_sides(1:end-1,:);
        xv=polygon_sides(:,1); yv=polygon_sides(:,2);
        iv=length(xv); % This variable depends on the holes
   or not connected domain.
        % In these simple cases the domains are without holes
    and
        % domains are connected.

    case 4 % domain not simply connected (optics)
        Nsides=100;
        y=[0          0    -0.1184    -0.1184    -0.3761];
        r=[1.0000     0.6120     0.5663     1.0761     1.2810];
        th=linspace(0,2*pi,Nsides); th=(th(1:end-1))';
        C1=[0 y(1)]; P1v=C1+r(1)*[cos(th) sin(th)]; P1=
   polyshape(P1v);
        C2=[0 y(2)]; P2v=C2+r(2)*[cos(th) sin(th)]; P2=
   polyshape(P2v);
        C3=[0 y(3)]; P3v=C3+r(3)*[cos(th) sin(th)]; P3=
   polyshape(P3v);
        C4=[0 y(4)]; P4v=C4+r(4)*[cos(th) sin(th)]; P4=
```

```
    polyshape(P4v);
        C5=[0 y(5)]; P5v=C5+r(5)*[cos(th) sin(th)]; P5=
    polyshape(P5v);
        Pout=intersect(P1,P4);
        Pout=intersect(Pout,P5);
        Pin=union(P2,P3);
        xv=subtract(Pout,Pin);
        yv=[]; iv=[];


end







function g=define_function(funct_example,L)

% function to test

switch funct_example

    case 1 % test exactness hyperinterpolation
        nexp=L;
        c0=rand(1); c1=rand(1); c2=rand(1);
        g=@(x,y) (c0+c1*x+c2*y).^nexp;

    case 2 % test exactness filt. hyperinterpolation
        nexp=max(floor(L/2)-1,0);
        c0=rand(1); c1=rand(1); c2=rand(1);
        g=@(x,y) (c0+c1*x+c2*y).^nexp;

    case 3 % function of that type

        funct_example_sub=1;

        fstring='Not available';

        switch funct_example_sub
            case 1
                g=@(x,y) (1-x.^2-y.^2).*exp(x.*cos(y));
                fstring='(1-x.^2-y.^2).*exp(x.*cos(y))';
            case 2
                g=@(x,y) exp(-(x.^2+y.^2));
                fstring='exp(-(x.^2+y.^2))';
            case 3
                g=@(x,y) sin(-(x.^2+y.^2));
```

```
                fstring='sin(-(x.^2+y.^2))';
            case 4
                g=@(x,y) 1+0*x+0*y;
                fstring='1+0*x+0*y';
            case 5
                g=@(x,y) sqrt((x-0.5).^2+(y-0.5).^2);
            case 6
                g=@(x,y) (0.2*x+0.5*y).^19;
            case 7
                g=@(x,y) exp((x-0.5).^2+(y-0.5).^2);
            case 8
                g=@(x,y) exp(-100*((x-0.5).^2+(y-0.5).^2));
            case 9
                g=@(x,y) cos(30*(x+y));
            case 10
                g=@(x,y) cos(5*(x+y));
            case 11
                g=@(x,y) exp((x-0.5).^1+(y-0.5).^1);
            case 12
                g=@(x,y) exp((x-0.5).^3+(y-0.5).^3);
            case 13
                g=@(x,y) (0.2*x+0.5*y).^15;
            case 14
                g=@(x,y) 1./(x.^2+y.^2);
            case 15
                % g=@(x,y) (x+y).^ade;
                g=@(x,y) (1+x+0.5*y).^ade;
            case 16
                x0=0.5; y0=0.5;
                g=@(x,y) exp(-((x-x0).^2+(y-y0).^2));
            case 17
                x0=0.5; y0=0.5;
                g=@(x,y) ((x-x0).^2 + (y-y0).^2).^(3/2);
            case 18 % franke
                g=@(x,y) .75*exp(-((9*x-2).^2 + (9*y-2).^2)
    /4) + ...
                    .75*exp(-((9*x+1).^2)/49 - (9*y+1)/10) +
    ...
                    .5*exp(-((9*x-7).^2 + (9*y-3).^2)/4) -
    ...
                    .2*exp(-(9*x-4).^2 - (9*y-7).^2);
        end


end
```

```
function Jz=evaluate_J(z,alpha)

%
% Object:
% Evaluate function J(z)=sum( (z(l))^2-2*z(l)*alpha(l) )
%
% Input:
% z     : vector of dimension d x 1
% alpha: vector of dimension d x 1
%
% Output:
% Jz: value of J(z)=sum( (z(l))^2-2*z(l)*alpha(l) )
%
% Reference:
% Quantity relevant in Thm. 5.1 of the paper
% "Hybrid hyperinterpolation over general regions"
%

% Jz=sum(z.^2-2*z.*alpha);
Jz=z'*z -2*z'*alpha;



function Hz=evaluate_H(z,w,err,V)

%
% Object:
% Evaluate function H(z)=2*sum_l( z(l) * sum_j( w(j)*err(j)*V
   (l,j) ) )
%
% Input:
% z     : vector of dimension d x 1
% w     : vector of dimension N x 1
% err   : vector of dimension N x 1
% V     : matrix of dimension d x N
%
% Output:
% Hz: value of H(z)=2*sum_l( z(l) * sum_j( w(j)*err(j)*V(l,j)
    ) )
%
% Reference:
% Quantity relevant in Thm. 5.1 of the paper
% "Hybrid hyperinterpolation over general regions"
%

inner_term=V'*(w.*err);
outer_term=z'*inner_term;
```

```
Hz=2*outer_term;




function demo_polygon_driver_02

%
% OBJECT.
% This function runs the experiments for hyperinterpolation
   over polygons.
% Several hyperinterpolants are use on possibly noisy
   functions.
%
% It makes the following numerical experiments:
% a) Fixed noise and several well-chosen lambda parameters (
   where useful).
% b) Fixed lambda and variable noise.
%
% At the end of the code
% 1. it produces the pertinent Latex files, making the
   relavant table.
% 2. it plots graphics comparing sparsity vs L2 errors on
   some hyp.variants;
%    all the latter data are saved on a file.
%
% In "Hybrid hyperinterpolation over general regions", we
   consider:
%
% 1. f(x,y)=(1-x.^2-y.^2).*exp(x.*cos(y)).
% 2. Maximum hyperinterpolant degree: 15 (rule: ade=30).
% 3. Reference cubature rule degree of precision: 40.
% 4. Experiment 1. Noise: a=0, sigma=0.075, lambda=[10 20 40
   80].
% 5. Experiment 2. Noise: a=0, sigma=[0.05 0.1 0.4 0.8],
   lambda=10.
% 6. Experiment 3. Noise: a=0, sigma=0.075, average Lambda:
   4.404e-03.
% 7. Cubature points for hyperinterpolation: 1197; note: no
   rule compress.
% 8. Cubature points for testing L2 errors: 2065; note: no
   rule compress.
% 9. Total amount of hyperinterpolation coefficients: 136.
%
% a) This routine runs the experiments cited in the paper
   above.
% Polygon: non convex polygon with vertices:
%   vertices=(1/4)*[1 2; 1 0; 3 2; 3 0; 4 2; 3 3; 3 0.85*4; 2
```

```
    4; 0 3; 1 2];
% b) The process took 207 seconds on a MacBook Pro, with
   Apple M1 processor
% and 16 GB of RAM (stage 3 may be time consuming).
%
% Dates:
% Written on January 1, 2023: A. Sommariva.
% Modified on April 24, 2023: A. Sommariva.
% Joint work with Cong-Pei An, Jia-Shu Ran.
%
% COPYRIGHT
%
% Copyright (C) 2023-
%
% Authors:
% Alvise Sommariva
%
% This program is free software: you can redistribute it and/
   or modify
% it under the terms of the GNU General Public License as
   published by
% the Free Software Foundation, either version 3 of the
   License, or
% (at your option) any later version.
%
% This program is distributed in the hope that it will be
   useful,
% but WITHOUT ANY WARRANTY; without even the implied warranty
    of
% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See
   the
% GNU General Public License for more details.
%
% You should have received a copy of the GNU General Public
   License
% along with this program.  If not, see <https://www.gnu.org/
   licenses/>.
%


clear;

% ..............................SETTINGS
   ..............................

% a) PART 1: fixed noise
kV=[10 20 40 80];
a1=0.0; sigma1=0.075;
```

```
% b) PART 2: fixed lambda
kS=10; % lambda is choosen equal to the "k"-th hyp. coeff. in
    magnitude.
a2=0.0; sigma2V=[0.05 0.1 0.4 0.8]; % noise parameters

% c) PART 3: sparsity plots
kP=2:2:100;
a3=0.0; sigma3=0.075;

% ......................FIRST STAGE OF EXPERIMENTS
    ......................

fprintf(2,'\n \n \n \t * <strong>Stage 1.</strong> \n \n \n')

AE2V=[]; betaV=[]; lambdaHIST=[];

XYW=[]; XYWR=[];

% note: cubature rule is computed only once.
for k=kV
    [AEinf,AE2,beta,lambdaV,XYW,XYWR]=demo_polygon(k,a1,
   sigma1,XYW,XYWR);
    AE2V=[AE2V AE2]; betaV=[betaV beta]; lambdaHIST=[
   lambdaHIST lambdaV];
end

AE2T=AE2V(1,:); BETAT=betaV(1,:); % tikhonov data storage
AE2F=AE2V(2,:); BETAF=betaV(2,:); % filtered hyp. data
   storage
AE2L=AE2V(3,:); BETAL=betaV(3,:); % lasso hyp. data storage
AE2H=AE2V(4,:); BETAH=betaV(4,:); % Hybrid hyp. data storage
AE2HA=AE2V(5,:); BETAHA=betaV(5,:); % Hard hyp. data storage
AE20=AE2V(6,:); BETA0=betaV(6,:); % Hyp. data storage


% Making tables
AE2V1=AE2V; betaV1=betaV; BETAH1=BETAH;
HypType=categorical({'tikhonov'; 'filtered'; 'lasso'; 'hybrid
   '; ...
    'hard'; 'hyperint.'; 'hyb.spars.'});
tablemat=[AE2V1; BETAH1];
T = table(HypType,tablemat); disp(T);

fprintf('\n \t * Lambda interval (for hyp. variants): [%1.2e
   ,%1.2e] \n',...
    min(min(lambdaHIST(kV,:))),max(max(lambdaHIST(kV,:))));

for k=1:length(kV)
    kVL=kV(k);
```

```
    fprintf('\n \t * Lambda %1.0f column (average): %1.4e'
    ,...
    k,mean(lambdaHIST(kVL,:)));
end



% ..................... SECOND STAGE OF EXPERIMENTS
    .....................

fprintf(2,'\n \n \n \t * <strong>Stage 2.</strong> \n ')
fprintf(2,'\n \n \t -> Average lambda: %1.3e \n \n',...
    mean(lambdaV(kS,:)));

% 2. Making experiments
AE2V=[]; betaV=[];

for sigma2=sigma2V
    % fprintf('\n \t sigma: %3.5e',sigma)
    [AEinf,AE2,beta,lambdaV,XYW,XYWR]=demo_polygon(kS,a2,
   sigma2,XYW,XYWR);
    AE2V=[AE2V AE2]; betaV=[betaV beta];
end

% 3. Making tables
AE2V2=AE2V; betaV2=betaV; BETAH2=betaV(4,:);
HypType=categorical({'tikhonov'; 'filtered'; 'lasso'; 'hybrid
   '; 'hard'; ...
    'hyperint.'; 'hyb.spars.'});
tablemat=[AE2V2; BETAH2];
T = table(HypType,tablemat); disp(T)



% ..................... PRODUCE LATEX TABLE
    ..........................

fprintf(2,'\n \n \t -> LaTeX table \n \n');

fileID=fopen('results_latex_polygon_stage1.txt','w');

strC={'Tikhonov'; 'Filtered'; 'Lasso'; 'Hybrid'; 'Hard'; '
   Hyperint.'; ...
    '$\|{\mathbf{\beta}}\|_{0}$'};
for k=1:7
    strL=strC{k};

    strNUM='';
```

```
    for s=1:size(AE2V1,2)

        if k< 7
            strNUML=['$',num2str(AE2V1(k,s),'%.4f'),'$ &'];
        else
            strNUML=[ '$',num2str(BETAH1(s),'%.1f'),'$ &'];
        end

        strNUM=[strNUM strNUML];

    end


    for s=1:size(AE2V2,2)

        if k< 7
            strNUML=['$',num2str(AE2V2(k,s),'%.4f'),'$ &'];
        else
            strNUML=[ '$',num2str(BETAH2(s),'%.1f'),'$ &'];
        end

        strNUM=[strNUM strNUML];

    end

    if k ==7
        fprintf('\n \t'); disp('\hline');
        fprintf(fileID,'\n \t \\hline');
    end

    strNUM=[extractBefore(strNUM,length(strNUM)),'\\'];

    strdisp=['{\em{',strL,'}} &',strNUM];
    fprintf('\n \t '); disp(strdisp);

    strdispfile=replace(strdisp,'\','\\');
    fprintf(fileID,strcat('\n \t',' ',strdispfile));
end

fclose(fileID);




% ..................... THIRD STAGE OF EXPERIMENTS
    ......................

fprintf(2,'\n \n \n \t * <strong>Stage 3.</strong> \n')
```

```
AE2P=[]; betaP=[]; JzHIST=[]; HzHIST=[];

% Making experiments.
for k=kP
    % fprintf('\n \t k: %3.0f',k)
    [AEinf,AE2,beta,lambdaV,XYW,XYWR,JzL,HzL,Wg_norm2]=
   demo_polygon(k,...
        a3,sigma3,XYW,XYWR);
    AE2P=[AE2P AE2]; betaP=[betaP beta];
    JzHIST=[JzHIST JzL]; HzHIST=[HzHIST HzL];
end


% ..................... PLOT FIGURE SPARSITY vs L2 ERROR
   ..................

fprintf(2,'\n \n \t -> Plotting sparsity/L2 errors figure.')

figure
subplot(1,2,1)


plot(betaP(3,:),JzHIST(3,:),'<','linewidth',1,'markersize'
   ,11,'color','r'), box on, set(gca,'fontsize',16),
set(gca, 'XMinorGrid', 'on'), set(gca, 'YMinorGrid', 'on'),
hold on

plot(betaP(3,:),HzHIST(3,:),'-<','linewidth',1,'markersize'
   ,11,'MarkerFaceColor','r','color','r'), box on, set(gca,'
   fontsize',16),
set(gca, 'XMinorGrid', 'on'), set(gca, 'YMinorGrid', 'on'),
hold on

plot(betaP(4,:),JzHIST(4,:),'s','linewidth',1,'markersize'
   ,11,'color','k'), box on, set(gca,'fontsize',16),
set(gca, 'XMinorGrid', 'on'), set(gca, 'YMinorGrid', 'on'),
hold on

plot(betaP(4,:),HzHIST(4,:),'-s','linewidth',1,'markersize'
   ,11,'MarkerFaceColor','k','color','k'), box on, set(gca,'
   fontsize',16),
set(gca, 'XMinorGrid', 'on'), set(gca, 'YMinorGrid', 'on'),
hold on


plot(betaP(5,:),JzHIST(5,:),'d','linewidth',1,'markersize'
   ,11,'color','b'), box on, set(gca,'fontsize',16),
set(gca, 'XMinorGrid', 'on'), set(gca, 'YMinorGrid', 'on'),
```

```
hold on

plot(betaP(5,:),HzHIST(5,:),'-d','linewidth',1,'markersize'
    ,11,'MarkerFaceColor','b','color','b'), box on, set(gca,'
    fontsize',16),
set(gca, 'XMinorGrid', 'on'), set(gca, 'YMinorGrid', 'on'),
hold on

legend({'Lasso (J)','Lasso (H)','Hybrid (J)','Hybrid (H)','
    Hard thresholding (J)','Hard thresholding (H)'},'
    interpreter','latex','fontsize',25);


xlabel({'Sparsity $(a)$'},'interpreter','latex','fontsize'
    ,25);ylabel({'Values'},'interpreter','latex','fontsize'
    ,25);
grid on;
yscale_symlog;
%set(gca,'position',[0.04 0.05 0.45 0.9])

subplot(1,2,2)



semilogy(betaP(3,:),AE2P(3,:),'r<','linewidth',2,'markersize'
    ,11),box on, set(gca,'fontsize',16),
set(gca, 'XMinorGrid', 'on'), set(gca, 'YMinorGrid', 'on')
hold on
semilogy(betaP(4,:),AE2P(4,:),'ks','linewidth',2,'markersize'
    ,11),box on, set(gca,'fontsize',16),
set(gca, 'XMinorGrid', 'on'), set(gca, 'YMinorGrid', 'on')
hold on
semilogy(betaP(5,:),AE2P(5,:),'bd','linewidth',2,'markersize'
    ,11),box on, set(gca,'fontsize',16),
set(gca, 'XMinorGrid', 'on'), set(gca, 'YMinorGrid', 'on')
hold on

legend({'Lasso', 'Hybrid', 'Hard thresholding'},'interpreter'
    ,'latex','fontsize',25);

xlabel({'Sparsity $(b)$'},'interpreter','latex','fontsize'
    ,25);ylabel({'$L_2$ Errors'},'interpreter','latex','
    fontsize',25);
grid on;
%set(gca,'position',[0.53 0.05 0.45 0.9])
```

# Bibliografia


[1] C. An, J.-S. Ran, A. Sommariva, *Hybrid Hyperinterpolation over general regions*, sottoposto a rivista.
https://arxiv.org/abs/2305.05863

[2] C. An and H.-N. Wu, *Lasso hyperinterpolation over general regions*. SIAM Journal on Scientific Computing 43(6), pp.A3967–A3991, (2021).
https://doi.org/10.1137/20M137793X

[3] B. Bauman, A. Sommariva, M. Vianello, *Compressed cubature over polygons with applications to optical design*, J. Comput. Appl. Math. 370 (2020).

[4] I.H. Sloan, *Polynomial interpolation and hyperinterpolation over general regions*. Journal of Approximation Theory 83(2), pp.238–254, (1995).
https://doi.org/10.1006/jath.1995.1119

[5] I.H. Sloan, R.S. Womersley, *Filtered hyperinterpolation: a constructive polynomial approximation on the sphere*. International Journal on Geomathematics 3(1), pp. 95–117, (2012).
https://doi.org/10.1007/s13137-011-0029-7

[6] A. Sommariva, A. Duse. Matlab codes for hyperinterpolation, filtered, Lasso and hybrid on convex, concave and non-simple polygonal domain.
https://github.com/annaduse/Codici_Tesi