UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI MATEMATICA "TULLIO LEVI-CIVITA"

Corso di Laurea in Matematica

Funzioni di base radiali per la soluzione approssimata di equazioni differenziali

Relatore:
Wolfgang Erb

Correlatore:
Antonia Larese De Tetto

Candidato:
Angelica Crepaldi
Matricola 1169732

Anno Accademico 2019/2020 - 11 Dicembre 2020

Abstract

La presente tesi tratta della soluzione approssimata di equazioni differenziali a partire da dati sparsi. I metodi studiati sono basati sulla collocazione multivariata dei suddetti dati mediante funzioni di base radiali (RBF). Sono stati analizzati due metodi: il metodo di Kansa - di natura non-simmetrica - e il metodo simmetrico basato sull'interpolazione generalizzata di Hermite. Gli algoritmi sono stati applicati a diversi problemi ellittici in due dimensioni al fine di esaminare la qualità dell'approssimazione, la stabilità numerica e la convergenza dei metodi. Si è giunti infine alla seguente conclusione: da un lato l'approccio di Hermite è vantaggioso nel caso si disponga di un risolutore per sistemi simmetrici, dall'altro il metodo di Kansa può essere impiegato nella risoluzione di diverse tipologie di problemi. Entrambi si rivelano, comunque, molto validi.

Indice

Introduzione

Molti metodi numerici per trovare la soluzione approssimata di equazioni differenziali utilizzano una mesh ben precisa. Per interpolare dati sparsi, tuttavia, risulta più comodo disporre di un metodo meshfree, cioè un metodo che operi su insiemi di nodi non strutturati. I metodi basati su funzioni di base radiali (RBF) sono esattamente di questo tipo. Un aspetto vantaggioso dell'impiego delle funzioni RBF è rappresentato dalla loro grande flessibilità: esse infatti - come ampiamente descritto in [8] - sono caratterizzate da un parametro di forma ε che può essere utilizzato per ottimizzarne l'accuratezza; bisogna tuttavia prestare attenzione al condizionamento della matrice che si ottiene, in quanto spesso accade che un parametro troppo piccolo fornisca un metodo molto accurato, ma che al tempo stesso causi un pessimo condizionamento.

Il primo metodo su cui concentreremo la nostra indagine è quello presentato da Kansa in [6]: di recente scoperta, è stato teorizzato riferendosi in particolare alle funzioni Multiquadratiche Inverse, ma è possibile estendere la teoria anche alle altre RBF. Per la natura della matrice di collocazione generata dal metodo, questo è conosciuto anche come *metodo non-simmetrico*. Affronteremo la questione della non-singolarità della matrice di collocazione, che è stata studiata, tra gli altri, da Hon e Schaback in [5].

Vi è una controparte simmetrica, che viene proposta da Fasshauer in [2] e che trae ispirazione dall'interpolazione generalizzata di Hermite. Il fatto che la matrice di collocazione sia simmetrica sembra costituire un vantaggio, ma vedremo come in alcuni casi - specialmente per quanto riguarda sistemi non-lineari - non sia affatto semplice applicare questa tipologia di metodi.

Seguendo l'idea di Fedoseyev, Friedman e Kansa (si veda [4]) apporteremo una modifica ai metodi, imponendo che i punti di collocazione di bordo soddisfino la PDE. Una prima analisi dei risultati derivanti da questa modifica, messa a confronto con i metodi già noti, è riportata in [7] ad opera di Larsson e Fonberg.

Risolveremo con i metodi sopra citati alcuni problemi ellittici in due dimensioni variando diversi aspetti del metodo risolutivo: dalla collocazione dei centri al bordo alla diversa distribuzione dei punti interni, dal parametro di forma alla funzione di base radiale utilizzata. In base ai risultati che otterremo, potremo trarre conclusioni riguardo l'efficienza dei due approcci. Tali problemi e parte della discussione teorica prendono spunto dal libro [3] di Fasshauer e dal documento [1].

La trattazione è organizzata secondo il seguente schema. Nel primo capitolo vengono definite le funzioni di base radiali insieme ad alcune loro proprietà di definitezza e viene trattata l'interpolazione generalizzata di Hermite. Il secondo capitolo si concentra sui metodi: descrive dapprima il metodo non-simmetrico, poi quello simmetrico e infine si sofferma sui problemi che entrambi i metodi presentano al bordo del dominio di collocazione. Il terzo capitolo è interamente dedicato ai risultati numerici: tramite l'utilizzo di tabelle e grafici si può analizzare la performance dei metodi rispetto a diversi problemi ellittici. Chiudono la trattazione le conclusioni, seguite da un'Appendice contenente i codici Matlab utilizzati nei test numerici.

Capitolo 1

Preliminari

Cominciamo la trattazione con alcuni cenni teorici utili per la comprensione dei capitoli seguenti. Ci concentriamo soprattutto sulle funzioni di base radiali (trattate in modo dettagliato in [8]) e sull'interpolazione generalizzata di Hermite, che riprenderemo quando parleremo della collocazione simmetrica.

1.1 Funzioni di base radiali RBF

Definizione 1.1. Sia ϕ una funzione definita su $\mathbb{R}^s$. Essa si dice *funzione radiale* se il suo valore in ogni punto dipende solo dalla distanza tra quel punto e il punto centrale della funzione:

$$\phi(x) = \varphi(r), \quad x = (x_1, \ldots, x_s) \in \mathbb{R}^s, \quad r = \sqrt{x_1^2 + \cdots + x_s^2}. \tag{1.1}$$

Nell'approssimazione con funzioni di base radiali (RBF), prima di tutto scegliamo un insieme di punti (che saranno i nostri nodi) oppure usiamo un insieme di nodi in cui siano disponibili i dati; procediamo centrando in ogni nodo un traslato di una specifica funzione radiale, che solitamente è la stessa per tutti i nodi (per praticità). Usiamo quindi questi traslati come funzioni di base radiali e utilizziamo come approssimante una loro combinazione lineare

$$\mathcal{P}_f(x, \varepsilon) = \sum_{j=1}^{N} c_j \varphi(\|x - \xi_j\|, \varepsilon), \tag{1.2}$$

con i coefficienti $c_1, \ldots, c_N$ per le N funzioni di base centrate nei nodi $x_1, \ldots, x_N$ e con il parametro ε che permette di ottimizzare l'accuratezza (è tuttavia difficile sapere quale parametro scegliere a priori).

Se ci vengono forniti dei dati $f = (f_1, \ldots, f_N)^T$ e vogliamo interpolarli nei nodi $(x_1, \ldots, x_N)^T$, poniamo il valore dell'approssimante uguale al corrispondente valore del

dato in ogni nodo, cioè

$$\mathcal{P}_f(x_i, \varepsilon) = f_i, \quad i = 1, \ldots, N, \tag{1.3}$$

che possiamo esprimere anche in forma matriciale come

$$Ac = f, \tag{1.4}$$

dove

$$A_{ij} = \varphi(\|x_i - \xi_j\|, \varepsilon),$$
$$f_i = f(x_i).$$

Una volta risolto il sistema (1.4) per i coefficienti c_j, possiamo valutare l'approssimante (1.2) in ogni punto del dominio.

Parliamo di *approssimazione* quando consideriamo equazioni differenziali, mentre la tecnica corrispondente per le equazioni differenziali alle derivate parziali (PDE) è chiamata *collocazione*. Parliamo, altresì, di *collocazione globale* quando approssimiamo la soluzione contemporaneamente su tutto il dominio, invece di costruire una soluzione globale partendo da una combinazione di approssimanti locali.

I metodi RBF sono piuttosto semplici da implementare, dal momento che non richiedono una mesh, e sono caratterizzati da una certa flessibilità sul dominio, poiché non è richiesto che i nodi seguano strutture ben precise. Inoltre, i metodi che si appoggiano su RBF lisce presentano, solitamente, una convergenza esponenziale, tipicamente nei casi in cui il numero di nodi cresca e il valore del parametro di forma ε diminuisca. Bisogna però fare attenzione, in quanto una distanza eccessivamente piccola tra due nodi si riflette in un cattivo condizionamento della matrice di interpolazione/collocazione.

Ci sono diversi tipi di RBF; elenchiamo di seguito i principali: Gaussiane $\varphi(r) = e^{-(\varepsilon r)^2}$, Multiquadratiche (MQ) $\varphi(r) = \sqrt{1 + (\varepsilon r)^2}$, Quadratiche Inverse $\varphi(r) = \frac{1}{1+(\varepsilon r)^2}$, Multiquadratiche Inverse (IMQ) $\varphi(r) = \frac{1}{\sqrt{1+(\varepsilon r)^2}}$.

Definizione 1.2. Una funzione continua a valori reali ϕ è *definita positiva* su $\mathbb{R}^s$ se è pari, cioè se $\phi(x) = \phi(-x)$, e se

$$\sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j \phi(x_i - x_j) \geq 0, \tag{1.5}$$

per ogni $x_i, x_j \in \mathbb{R}^s$ a due a due distinti, $\lambda = (\lambda_1, \ldots, \lambda_N)^T \in \mathbb{R}^N$. La funzione ϕ è *strettamente definita positiva* su $\mathbb{R}^s$ se la forma quadratica (1.5) è zero solo per $\lambda \equiv 0$.

Tra le RBF elencate in precedenza, le Gaussiane, le Quadratiche Inverse e le Multi-

quadratiche Inverse sono strettamente definite positive. Questa proprietà garantisce che la matrice di interpolazione A in (1.4) è definita positiva e quindi invertibile.

Definizione 1.3. Una funzione continua pari a valori reali ϕ è *condizionatamente definita positiva di ordine* m su $\mathbb{R}^s$ se

$$\sum_{i=1}^{N}\sum_{j=1}^{N}\lambda_i\lambda_j\phi(x_i - x_j) \geq 0, \tag{1.6}$$

per ogni $x_i, x_j \in \mathbb{R}^s$ a due a due distinti, e $\lambda = (\lambda_1, \ldots, \lambda_N)^T \in \mathbb{R}^N$ soddisfa

$$\sum_{i=1}^{N}\lambda_i p(x_i) = 0, \tag{1.7}$$

per ogni polinomio p a valori reali di grado al più $m-1$. La funzione ϕ è *strettamente condizionatamente definita positiva di ordine* m su $\mathbb{R}^s$ se la forma quadratica (1.6) è zero solo per $\lambda \equiv 0$.

Le MQ sono un esempio di RBF strettamente condizionatamente definite positive. Purtroppo per questo insieme di funzioni di base radiali non c'è la certezza che la matrice A in (1.4) sia invertibile, a meno che non si tratti di alcune speciali funzioni strettamente condizionatamente definite positive di ordine uno, come ad esempio le Multiquadratiche generalizzate $\phi(r, \varepsilon) = (1 + (\varepsilon r)^2)^\beta$ con $0 < \beta < 1$.

1.2 Interpolazione generalizzata di Hermite

Al fine di comprendere al meglio il metodo di collocazione simmetrica di cui parleremo nel prossimo capitolo, è necessario sapere in cosa consiste l'interpolazione generalizzata di Hermite.

Supponiamo di essere in possesso dei dati sparsi $(x_i, \lambda_i f)$, $i = 1, \ldots, N$, $x_i \in \mathbb{R}^s$, dove $\Lambda = \{\lambda_1, \ldots, \lambda_N\}$ sono funzionali linearmente indipendenti e f è una funzione liscia assegnata. La scelta dei λ_i è piuttosto libera e, una volta che la si è effettuata, si cerca un'interpolante del tipo

$$\mathcal{P}_f(x) = \sum_{j=1}^{N} c_j\psi(\|x\|), \quad x \in \mathbb{R}^s, \tag{1.8}$$

dove le ψ_j sono funzioni di base appropriate.
Imponiamo le condizioni di interpolazione generalizzata:

$$\lambda_i\mathcal{P}_f = \lambda_i f, \quad i = 1, \ldots, N. \tag{1.9}$$

Indichiamo i centri delle funzioni di base radiali con $\Xi = \{\xi_1, \ldots, \xi_N\}$ e con $\mathcal{X} = \{x_1, \ldots, x_N\}$ i *data sites*; come vedremo successivamente, spesso Ξ e $\mathcal{X}$ coincidono. A questo punto si comprende perché sia stato necessario introdurre la nozione di RBF, dal momento che porremo

$$\psi_j(\|x\|) = \lambda_j^{\xi}\varphi(\|x - \xi\|), \tag{1.10}$$

dove φ è una funzione di base radiale e λ^{ξ} è il funzionale λ che agisce sul secondo argomento di φ.
Inserendo ora (1.10) in (1.8), si ottiene un'espressione più esplicita dell'interpolante generalizzata

$$\mathcal{P}_f(x) = \sum_{j=1}^{N} c_j \lambda_j^{\xi}\varphi(\|x - \xi\|), \quad x \in \mathbb{R}^s, \tag{1.11}$$

tale che soddisfi le condizioni di interpolazione generalizzata (1.9).
Possiamo scrivere il sistema lineare associato $Ac = f_\lambda$, in cui le entrate della matrice A sono date da

$$A_{ij} = \lambda_i \lambda_j^{\xi}\varphi, \quad i, j = 1, \ldots, N,$$

e il termine noto è dato da $f_\lambda = [\lambda_1 f, \ldots, \lambda_N f]^T$.

Potremmo utilizzare un'interpolante più semplice del tipo

$$\mathcal{P}_f(x) = \sum_{j=1}^{N} c_j \varphi(\|x - \xi\|), \quad x \in \mathbb{R}^s, \tag{1.12}$$

ma in questo modo si perde la simmetria della matrice, e con essa tante proprietà.

Un'altra osservazione importante riguarda la non-singolarità della matrice A. Un risultato teorico riportato in [9] afferma che, nel caso di funzioni definite positive (con la possibilità di estensione anche per le funzioni condizionatamente definite positive), se i funzionali λ_j sono linearmente indipendenti sullo spazio nativo $\mathcal{N}_\phi(\Omega)$, si può mostrare che A è simmetrica e definita positiva e una conseguenza è che A è non-singolare per la classe di RBF utilizzate per l'interpolazione di dati sparsi.

Capitolo 2

Metodi di collocazione globale con RBF

Vediamo ora i principali metodi di collocazione globale con RBF. Per farlo consideriamo tre casi: la collocazione non simmetrica, l'approccio simmetrico basato su Hermite e l'imposizione della PDE sul bordo.

2.1 Collocazione non-simmetrica

L'approccio della collocazione non-simmetrica è stato proposto per la prima volta da Kansa in [6] negli anni '90. Prevede l'utilizzo di funzioni di base radiali Multiquadratiche Inverse (IMQ), motivo per cui questo metodo viene ricordato anche col nome di metodo di Kansa o metodo delle Multiquadratiche.

Definizione 2.1. Una equazione differenziale alle derivate parziali si dice *ellittica* se i coefficienti delle derivate di grado massimo sono positivi.

Consideriamo, dato un dominio $\Omega \subset \mathbb{R}^s$, un'equazione differenziale alle derivate parziali ellittica della forma

$$\mathcal{L}u(x) = f(x), \quad x \in \Omega \tag{2.1}$$

con condizioni al bordo di Dirichlet

$$u(x) = g(x), \quad x \in \partial\Omega. \tag{2.2}$$

Per applicare il metodo di collocazione di Kansa, rappresentiamo la soluzione ap-

prossimata $\hat{u}$ tramite un'espansione di funzioni di base radiali analoga a quella usata per l'interpolazione di dati sparsi, cioè

$$\hat{u}(x) = \sum_{j=1}^{N} c_j \varphi(\|x - \xi_j\|), \tag{2.3}$$

dove l'insieme $\Xi = \{\xi_1, \ldots, \xi_N\}$ indica i *centri*, mentre l'insieme $\mathcal{X} = \{x_1, \ldots, x_n\}$ indica i *punti di collocazione*.

I punti di collocazione, nella maggior parte dei casi, sono scelti in modo che coincidano con i centri. A volte, tuttavia, è utile distinguere i due insiemi per migliorare l'accuratezza del metodo nei pressi del bordo. Per la trattazione che segue si assume che coincidano.
Imponendo le condizioni di collocazione

$$\begin{aligned} \mathcal{L}\hat{u}(x_i) &= f(x_i), \quad x_i \in \mathcal{I}, \\ \hat{u}(x_i) &= g(x_i), \quad x_i \in \mathcal{B}, \end{aligned}$$

si può quindi scrivere la matrice di collocazione A

$$A = \begin{bmatrix} \tilde{A}_{\mathcal{L}} \\ \tilde{A} \end{bmatrix}, \tag{2.4}$$

dove

$$\begin{aligned} (\tilde{A}_{\mathcal{L}})_{ij} &= \mathcal{L}\varphi(\|x_i - \xi_j\|), && x_i \in \mathcal{I}, \xi_j \in \Xi, \\ \tilde{A}_{ij} &= \varphi(\|x_i - \xi_j\|), && x_i \in \mathcal{B}, \xi_j \in \Xi \end{aligned}$$

dove gli insiemi $\mathcal{I}$ e $\mathcal{B}$ sono due sottoinsiemi di $\mathcal{X}$, che indicano rispettivamente i punti interni e i punti del bordo.

Scriviamo il sistema lineare $Ac = y$, in cui y è il vettore di entrate $f(x_i)$, $x_i \in \mathcal{I}$, seguite da $g(x_i)$, $x_i \in \mathcal{B}$. Diciamo che il problema è ben posto se questo sistema lineare ammette un'unica soluzione.

Sorge tuttavia un problema: nulla garantisce che la matrice di collocazione A sia non-singolare, anzi è un quesito ancora aperto quello di trovare condizioni sufficienti sui centri affinché la matrice sia invertibile. Si è cercato più volte in passato di dimostrare la non-singolarità: una delle più importanti dimostrazioni fu quella fornita dallo stesso Kansa, che però fu successivamente smentito da Hon e Schaback grazie ad un esperimento numerico nell'articolo [5]. Nello specifico, l'esperimento considera l'equazione di Poisson sul quadrato $\Omega = [-1, 1] \times [-1, 1]$, fissa otto punti equispaziati sul bordo e altri nove punti di collocazione all'interno del dominio in modo casuale; utilizza poi funzioni IMQ e costruisce la matrice di collocazione non-simmetrica che poi si rivela singolare.

Si può in parte porre rimedio alla questione scegliendo particolari centri, all'interno di un più grande insieme di centri candidati, in maniera adattativa; ciò assicura l'invertibilità della matrice di collocazione, attraverso un algoritmo iterativo.

Sebbene nella descrizione del metodo sia stata considerata una generica RBF, Kansa nel suo articolo si focalizzò sull'utilizzo delle Multiquadratiche Inverse, da cui il metodo prende il nome. Uno studio risalente alla fine degli anni '90 ad opera di Moridis e Kansa mostrò che tale metodo può essere applicato anche a PDE ellittiche non lineari, sistemi di PDE ellittiche, PDE paraboliche o iperboliche dipendenti dal tempo e problemi ellittici con coefficienti variabili.

2.2 Collocazione simmetrica basata sull'interpolazione generalizzata di Hermite

Consideriamo l'equazione differenziale alle derivate parziali ellittica (2.1), descritta a proposito del metodo di Kansa, con le condizioni al bordo (2.2). Il vantaggio di basarsi sull'interpolazione generalizzata di Hermite è che la matrice di collocazione risulta non-singolare. A differenza di quanto visto per il metodo di Kansa, per la soluzione utilizziamo un'approssimante della seguente forma

$$\hat{u}(x) = \sum_{j=1}^{N_{\mathcal{I}}} c_j \mathcal{L}^{\xi} \varphi(\|x - \xi_j\|) + \sum_{j=N_{\mathcal{I}}+1}^{N} c_j \varphi(\|x - \xi_j\|), \tag{2.5}$$

dove $N_{\mathcal{I}}$ indica il numero di nodi interni ad Ω e N indica il numero totale di nodi; $\mathcal{L}^{\xi}$ è l'operatore differenziale dell'equazione (2.1), che agisce su φ vista come funzione del secondo argomento. Continuiamo a considerare il caso in cui $\mathcal{X} = \Xi$, ossia punti di collocazione e centri coincidono.
Imponendo le condizioni di collocazione

$$\begin{aligned} \mathcal{L}\hat{u}(x_i) &= f(x_i), \quad x_i \in \mathcal{I}, \\ \hat{u}(x_i) &= g(x_i), \quad x_i \in \mathcal{B}, \end{aligned}$$

si può quindi scrivere la matrice di collocazione A

$$A = \begin{bmatrix} \hat{A}_{\mathcal{L}\mathcal{L}^{\xi}} & \hat{A}_{\mathcal{L}} \\ \hat{A}_{\mathcal{L}^{\xi}} & \hat{A} \end{bmatrix}, \tag{2.6}$$

dove

$$
\begin{aligned}
(\hat{A}_{\mathcal{L}\mathcal{L}^\xi})_{ij} &= \mathcal{L}\mathcal{L}^\xi\varphi(\|x_i - \xi_j\|), && x_i, \xi_j \in \mathcal{I} \\
(\hat{A}_{\mathcal{L}})_{ij} &= \mathcal{L}\varphi(\|x_i - \xi_j\|), && x_i \in \mathcal{I}, \xi_j \in \mathcal{B} \\
(\hat{A}_{\mathcal{L}^\xi})_{ij} &= \mathcal{L}^\xi\varphi(\|x_i - \xi_j\|), && x_i \in \mathcal{B}, \xi_j \in \mathcal{I} \\
(\hat{A})_{ij} &= \varphi(\|x_i - \xi_j\|), && x_i, \xi_j \in \mathcal{B}.
\end{aligned}
$$

La matrice (2.6) che si ottiene con questo tipo di collocazione è non-singolare, dal momento che è dello stesso tipo delle matrici di interpolazione generalizzata di Hermite. Vi è da aggiungere un altro aspetto fondamentale: la matrice (2.6) è simmetrica. Questa proprietà presenta pro e contro: il principale vantaggio risiede nel fatto che la simmetria comporta un'implementazione più efficiente; risulta, tuttavia, più complicata da assemblare, poiché richiede l'utilizzo di funzioni di base più regolari rispetto a quelle utilizzate nel metodo di Kansa e ciò si riflette in una soluzione non soddisfacente di problemi non lineari.

2.3 Collocazione di bordo delle PDE

Notoriamente l'accuratezza dei metodi di interpolazione e collocazione con RBF tende a diminuire nelle vicinanze del bordo, cioè al bordo l'errore che si compie è maggiore. Per questa ragione, nell'articolo [4] redatto da Fedoseyev, Friedman e dallo stesso Kansa, essi teorizzarono che, al fine di ottenere approssimazioni più accurate, occorrerebbe che i punti di collocazione sul bordo soddisfacessero la PDE.

Poiché vogliamo imporre più condizioni sul bordo, dobbiamo aggiungere un corrispondente numero di centri in modo da avere tante variabili quante sono le equazioni del sistema; questi centri dovrebbero essere localizzati all'esterno del dominio Ω in modo tale da creare delle funzioni di base addizionali e mantenere la stessa distanza media tra i nodi al fine di preservare il condizionamento. I centri che non si trovano in prossimità del bordo continuano a coincidere con i punti di collocazione (come visto nei precedenti metodi):

$$
\xi_j = \begin{cases} x_j & \xi_j \in \mathcal{I} \cup \mathcal{B}, \\ \text{un punto fuori da } \Omega & \xi_j \in \Xi \setminus (\mathcal{I} \cup \mathcal{B}). \end{cases}
$$

Aggiungendo condizioni (e quindi centri) aumenta la grandezza del sistema da risolvere. Al fine di poter confrontare in modo attendibile questo approccio con gli altri metodi di collocazione, dobbiamo accertarci che il numero dei nodi sia lo stesso per tutti i metodi.

Larsson e Fornberg hanno riportato in [7] un confronto tra metodo di collocazione di Kansa originario, metodo di Kansa modificato nel modo appena descritto e approc-

cio simmetrico basato sull'interpolazione di Hermite. La conclusione a cui sono giunti implementando i metodi utilizzando le IMQ è che il metodo simmetrico è il più accurato fra i tre descritti, seguito dal metodo non-simmetrico con collocazione al bordo. Il motivo di questa conclusione risiede nel fatto che il sistema simmetrico è caratterizzato, in generale, da un migliore condizionamento.

Capitolo 3

Esperimenti numerici

Consideriamo ora diversi problemi standard a cui applicheremo i metodi RBF e per i quali ci esprimeremo in termini di convergenza, accuratezza e mal-condizionamento. Gli esempi sono tratti da [3].

3.1 Collocazione non-simmetrica

Vediamo come agisce nella pratica il metodo non-simmetrico di Kansa. Per farlo consideriamo nello specifico tre problemi ellittici un due dimensioni la cui soluzione è nota e, perciò, facilmente verificabile.

Problema 3.1. Consideriamo il seguente problema di Poisson con condizioni al bordo di Dirichlet:

$$\begin{aligned} \nabla^2 u(x,y) &= -\frac{5}{4}\pi^2 \sin(\pi x)\cos\left(\frac{\pi y}{2}\right), && (x,y) \in \Omega = [0,1]^2, \\ u(x,y) &= \sin(\pi x), && (x,y) \in \Gamma_1, \\ u(x,y) &= 0, && (x,y) \in \Gamma_2, \end{aligned}$$

dove $\Gamma_1 = \{(x,y) : 0 \leq x \leq 1, y = 0\}$ e $\Gamma_2 = \partial\Omega \setminus \Gamma_1$.
La soluzione esatta è data da $u(x,y) = \sin(\pi x)\cos(\frac{\pi y}{2})$.

Per trovare e discutere la soluzione del Problema 3.1 utilizziamo il metodo di collocazione non-simmetrico implementato nel Codice 3.1 (riportato in Appendice A: Codici Matlab) e confrontiamo le performance di due diverse RBF: Gaussiane e Multiquadratiche Inverse (IMQ). Ci serviamo anche di due diversi data sites: una distribuzione uniforme di punti e punti di Halton. Per quanto riguarda i punti interni, i centri delle funzioni di base coincidono con i punti di collocazione, come trattato nella parte teorica.

Alla fine del capitolo precedente, nella sezione 2.3, ci siamo soffermati sui problemi

Tabella 3.1: Soluzione con collocazione non-simmetrica del Problema 3.1 con IMQ, $\varepsilon = 3$ e punti interni di Halton.

	Centri sul bordo		Centri esterni	
N(interni)	RMS-error	cond(A)	RMS-error	cond(A)
9	5.642192e-02	5.276474e+02	6.029293e-02	4.399608e+02
25	1.039322e-02	3.418858e+03	4.187975e-03	2.259698e+03
81	2.386062e-03	1.726995e+06	4.895870e-04	3.650369e+05
289	4.904715e-05	1.706884e+10	2.668524e-05	5.328110e+09
1089	3.553856e-08	6.308637e+17	2.012396e-08	3.815741e+17

Tabella 3.2: Soluzione con collocazione non-simmetrica del Problema 3.1 con Gaussiane e IMQ, $\varepsilon = 3$, punti interni uniformi e centri di bordo esterni al dominio.

	Gaussiane		IMQ	
N(interni)	RMS-error	cond(A)	RMS-error	cond(A)
3×3	1.981675e-01	1.258837e+03	1.526456e-01	2.794516e+02
5×5	7.199931e-03	4.136193e+03	6.096534e-03	2.409431e+03
9×9	1.947108e-04	2.529708e+10	8.071271e-04	8.771630e+05
17×17	5.252432e-07	3.459168e+20	3.219110e-05	5.981238e+10
33×33	3.389598e-06	3.227540e+22	1.541740e-07	4.358316e+19

che i metodi di collocazione presentano nei pressi del bordo; a tal proposito imponiamo ora che i punti di collocazione e i centri soddisfino le condizioni al bordo; per queste ultime sono quindi necessari punti di collocazione addizionali, che prenderemo equispaziati lungo tutto il bordo (che ordiniamo in senso antiorario partendo dall'origine). Possiamo prendere i centri sul bordo in due modi diversi:

- Possiamo mantenere la coincidenza tra centri di bordo e punti di collocazione al bordo; è tuttavia un approccio sconsigliato nel caso si scelgano punti interni uniformi, in quanto essi contengono già punti sul bordo e quindi verrebbero generate colonne duplicate con conseguente matrice di collocazione singolare. Quando però lavoriamo con i punti di Halton non ci sono problemi;

- Possiamo, in alternativa, creare centri di bordo fuori dal dominio. Questo approccio sembra fornire una soluzione più accurata, ma in questo caso i centri differiscono dai data sites, e ciò genera dubbi sull'invertibilità della matrice e l'errore al bordo. Ancora, non è ben chiaro quale sia la migliore collocazione per i centri di bordo; nel nostro caso è una buona idea prenderli ad una piccola distanza perpendicolare ai punti di collocazione di bordo.

Tabella 3.3: Soluzione con collocazione non-simmetrica del Problema 3.1 con IMQ, $\varepsilon = 2$ e $\varepsilon = 4$, punti interni uniformi e centri di bordo esterni al dominio.

	IMQ, $\varepsilon = 2$		IMQ, $\varepsilon = 4$	
N(interni)	RMS-error	cond(A)	RMS-error	cond(A)
3×3	6.324969e-02	2.018876e+02	2.178337e-01	5.392389e+02
5×5	2.294189e-03	1.304853e+04	3.626603e-02	1.389566e+03
9×9	2.129792e-04	6.332293e+07	1.404654e-03	1.105283e+05
17×17	2.770412e-06	7.005273e+14	1.105312e-04	5.416301e+08
33×33	3.301061e-07	1.918461e+20	1.557749e-06	7.380245e+15

Riportiamo in Figura 3.1 i punti di collocazione e i centri (uniformi e di Halton) nel caso di $N = 289$ punti interni. Queste distribuzioni di punti ricorreranno anche negli esempi successivi.

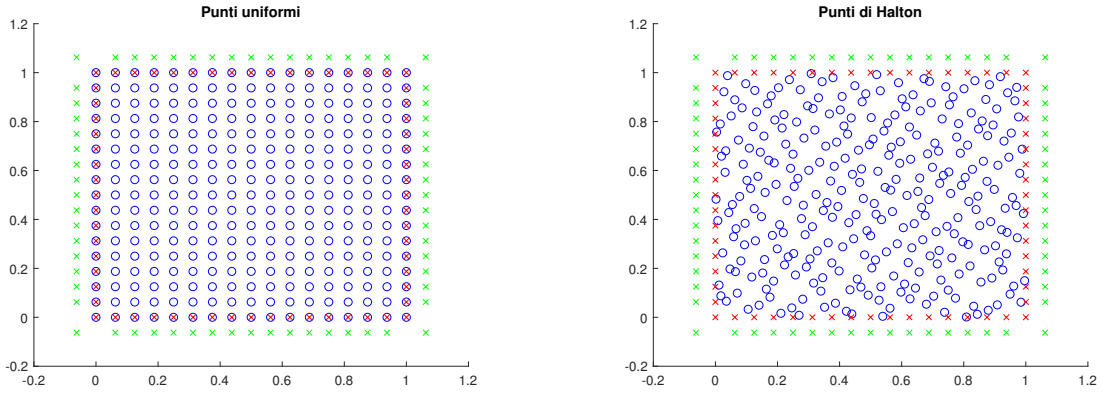

Figura 3.1: A sinistra: Punti di collocazione uniformi (interni cerchiati in blu, al bordo crociati in rosso) e centri (interni cerchiati in blu, al bordo crociati in verde). A destra: Punti di collocazione di Halton (interni cerchiati in blu, al bordo crociati in rosso) e centri (interni cerchiati in blu, al bordo crociati in verde).

Nelle Tabelle 3.1, 3.2 e 3.3 sono riportati il $\texttt{RMS} - \texttt{error}$, che rappresenta la radice dell'errore quadratico medio, e il numero di condizionamento della matrice di collocazione per la soluzione della PDE (Problema 3.1) ottenuta con metodo di collocazione non-simmetrico. Nello specifico, nella Tabella 3.1 e nella parte destra della Tabella 3.2 presentiamo alcuni risultati per il metodo di collocazione con le RBF Multiquadratiche Inverse con un parametro di forma $\varepsilon = 3$, mentre nella Tabella 3.3 confrontiamo i risultati ottenuti utilizzando altri due parametri, ossia $\varepsilon = 2$ e $\varepsilon = 4$. Nella Tabella 3.1 i punti interni sono punti di Halton spaziati non regolarmente, per i quali possiamo confrontare l'effetto di porre i centri sul bordo (coincidenti con i punti di collocazione di bordo) con il prenderli fuori dal dominio. Per i risultati delle Tabelle 3.2 e 3.3, invece, abbiamo utilizzato punti interni uniformemente spaziati e centri di bordo esterni al dominio. Nella Tabella 3.2 confrontiamo l'utilizzo delle IMQ con le Gaussiane (mante-

nendo il parametro di forma $\varepsilon = 3$).

Consideriamo ora le tabelle e traiamo alcune conclusioni in merito all'accuratezza dei metodi utilizzati. Confrontando le parti destre delle Tabelle 3.1 e 3.2, si evince che l'utilizzo dei punti di Halton è preferibile a quello dei punti uniformi, poiché sia gli errori sia i numeri di condizionamento risultano minori. Esaminando la Tabella 3.1, osserviamo che sembra essere più vantaggioso porre i centri di bordo fuori dal dominio, in quanto, anche in questo caso, errori e numeri di condizionamento decrescono. Confrontando la performance delle Gaussiane con quella delle IMQ (Tabella 3.2) notiamo come le prime siano caratterizzate da un condizionamento peggiore rispetto alle IMQ; notiamo inoltre che il $\texttt{RMS} - \texttt{error}$ per le Gaussiane con 1089 punti interni uniformi e centri di bordo esterni al dominio è peggiore rispetto al caso di 289 punti, il che significa che non necessariamente l'utilizzo di un maggior numero di punti implica una soluzione più accurata. Infine ci concentriamo sui parametri di forma (confronto tra Tabella 3.3 e parte destra della Tabella 3.2). Gli errori ottenuti utilizzando il metodo con parametro di forma $\varepsilon = 2$ sono i più piccoli; osserviamo, tuttavia, che i numeri di condizionamento tendono a crescere molto velocemente, a differenza di quanto non accada per $\varepsilon = 4$, che presenta errori maggiori ma un miglior condizionamento. Alla luce di questi test numerici quindi il parametro $\varepsilon = 3$ sembra il compromesso migliore per ottenere soluzioni piuttosto accurate con una matrice di collocazione non eccessivamente mal-condizionata.

In Figura 3.2 mostriamo la soluzione nel caso particolare $N = 81$ punti interni uniformi e $\varepsilon = 3$ utilizzando prima le IMQ e poi le Gaussiane; la colorazione dell'immagine è dovuta all'errore massimo compiuto in ogni punto.

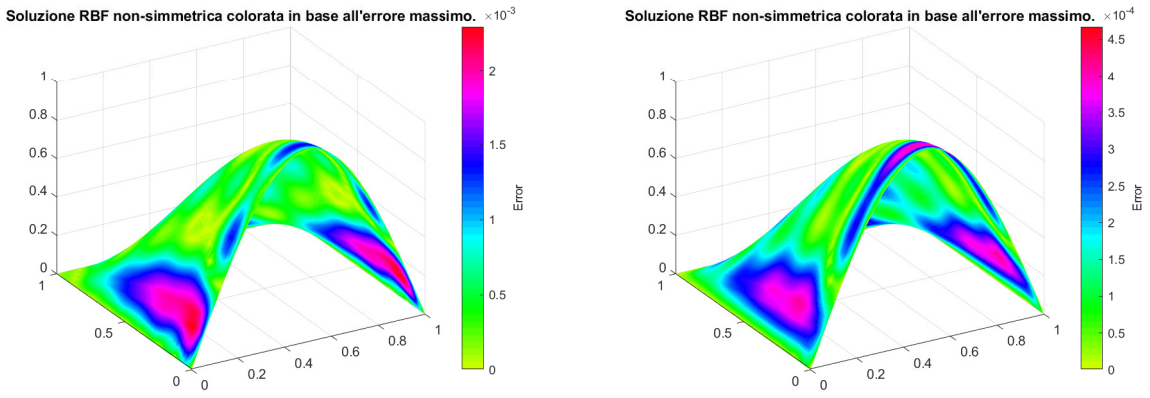

Figura 3.2: A sinistra: Soluzione ottenuta tramite collocazione non-simmetrica usando IMQ con $\varepsilon = 3$ e $N = 81$ punti interni uniformi; l'errore è proporzionale a 10^{-3}. A destra: Soluzione ottenuta tramite collocazione non-simmetrica usando Gaussiana con $\varepsilon = 3$ e $N = 81$ punti interni uniformi; l'errore è proporzionale a 10^{-4}.

Tabella 3.4: Soluzione con collocazione non-simmetrica del Problema 3.2 con Gaussiane e IMQ, $\varepsilon = 3$, punti interni di Halton e centri di bordo esterni al dominio.

	Gaussiane		IMQ	
N(interni)	RMS-error	cond(A)	RMS-error	cond(A)
9	6.852103e-02	8.874341e+03	1.123770e-01	6.954910e+02
25	1.091888e-02	4.898291e+03	1.123575e-02	3.302471e+03
81	1.854386e-04	1.286993e+09	1.370992e-03	4.992219e+05
289	4.290096e-07	3.850307e+19	8.105108e-05	7.527456e+09
1089	2.114008e-06	4.440726e+20	6.960421e-08	6.685182e+17

Problema 3.2. Consideriamo la seguente equazione ellittica con coefficienti variabili e condizioni al bordo di Dirichlet omogenee:

$$\frac{\partial}{\partial x}\left(a(x,y)\frac{\partial}{\partial x}u(x,y)\right) + \frac{\partial}{\partial y}\left(a(x,y)\frac{\partial}{\partial y}u(x,y)\right) = f(x,y), \qquad (x,y) \in \Omega = [0,1]^2,$$
$$u(x,y) = 0, \qquad (x,y) \in \Gamma = \partial\Omega,$$

dove

$$f(x,y) = -16x(1-x)(3-2y)e^{x-y} + 32(1-y)(3x^2+y^2-x-2)$$

e i coefficienti sono dati da

$$a(x,y) = 2 - x^2 - y^2,$$
$$b(x,y) = e^{x-y}.$$

La soluzione esatta è data da $u(x,y) = 16x(1-x)y(1-y)$.

Per trovare e discutere la soluzione del Problema 3.2 utilizziamo il metodo di collocazione non-simmetrico implementato nel Codice 3.2. Otteniamo risultati analoghi a quelli trovati risolvendo il Problema 3.1. Riportiamo la Tabella 3.4, in cui confrontiamo la soluzione ottenuta con le Gaussiane e con le Multiquadratiche Inverse basandoci su punti interni di Halton e centri esterni al dominio. Anche in questo caso la soluzione con IMQ è meglio condizionata.

La Figura 3.3 contiene la soluzione approssimata e l'errore massimo per la soluzione del Problema 3.2, ricavata tramite Multiquadratiche Inverse su $N = 289$ punti interni di Halton e 64 punti di bordo. Notiamo che, sebbene il problema abbia una soluzione simmetrica, la soluzione approssimata non è abbastanza simmetrica, come si può vedere dal plot dell'errore.

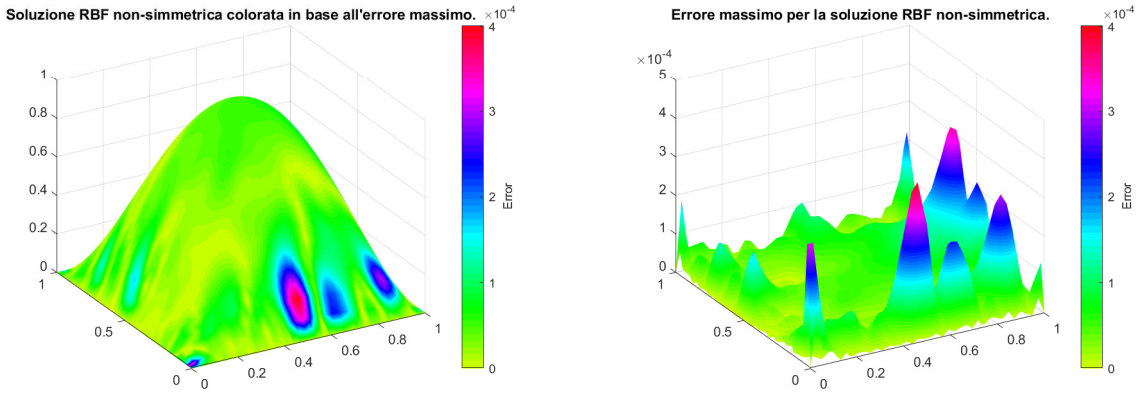

Figura 3.3: A sinistra: Soluzione ottenuta tramite collocazione non-simmetrica. A destra: Plot dell'errore. In entrambi i casi è stata utilizzata una IMQ con $\varepsilon = 3$ e $N = 289$ punti interni di Halton.

Problema 3.3. Consideriamo il seguente problema di Poisson con condizioni miste al bordo:

$$\begin{aligned} \nabla^2 u(x,y) &= -5.4x, && (x,y) \in \Omega = [0,1]^2, \\ \frac{\partial}{\partial n} u(x,y) &= 0, && (x,y) \in \Gamma_1 \cup \Gamma_3, \\ u(x,y) &= 0.1, && (x,y) \in \Gamma_2, \\ u(x,y) &= 1, && (x,y) \in \Gamma_4, \end{aligned}$$

dove

$$\begin{aligned} \Gamma_1 &= \{(x,y) : 0 \leq x \leq 1, y = 0\}, \\ \Gamma_2 &= \{(x,y) : x = 1, 0 \leq y \leq 1\}, \\ \Gamma_3 &= \{(x,y) : 0 \leq x \leq 1, y = 1\}, \\ \Gamma_4 &= \{(x,y) : x = 0, 0 \leq y \leq 1\}. \end{aligned}$$

La soluzione esatta è data da $u(x,y) = 1 - 0.9x^3$.

Con la notazione $\frac{\partial}{\partial n}$ si intende la derivata normale, che sul lato Γ_1 e Γ_3 è data da $\frac{\partial}{\partial y}$ e da $-\frac{\partial}{\partial y}$. Ciò significa che nel codice MATLAB non basta scrivere la funzione di base e il suo Laplaciano, ma occorre esplicitare anche la derivata parziale rispetto a y.

Per trovare e discutere la soluzione del Problema 3.3 utilizziamo il metodo di collocazione non-simmetrico implementato nel Codice 3.3. Nella Tabella 3.5 mettiamo di nuovo in parallelo la performance del metodo di collocazione che utilizza RBF Gaussia-

Tabella 3.5: Soluzione con collocazione non-simmetrica del Problema 3.3 con Gaussiane e IMQ, $\varepsilon = 3$, punti interni di Halton e centri di bordo esterni al dominio.

	Gaussiane		IMQ	
N(interni)	RMS-error	cond(A)	RMS-error	cond(A)
9	3.423330e-01	5.430073e+03	7.937403e-02	2.782348e+02
25	1.065826e-02	1.605086e+03	5.605445e-03	1.680888e+03
81	5.382387e-04	3.684159e+08	1.487160e-03	2.611650e+05
289	1.803971e-06	3.235718e+19	1.822077e-04	3.775455e+09
1089	3.194618e-05	2.460756e+21	1.914906e-07	2.825336e+17

ne con quello che utilizza Multiquadratiche Inverse. I punti interni continuano ad essere quelli di Halton e i centri di bordo sono presi esternamente al dominio. Ancora una volta abbiamo la conferma, a parità di parametro di forma, della peggiore stabilità della soluzione Gaussiana.

Per quanto riguarda la Figura 3.4 possiamo fare le stesse considerazioni che sono state fatte a proposito delle Figura 3.3.

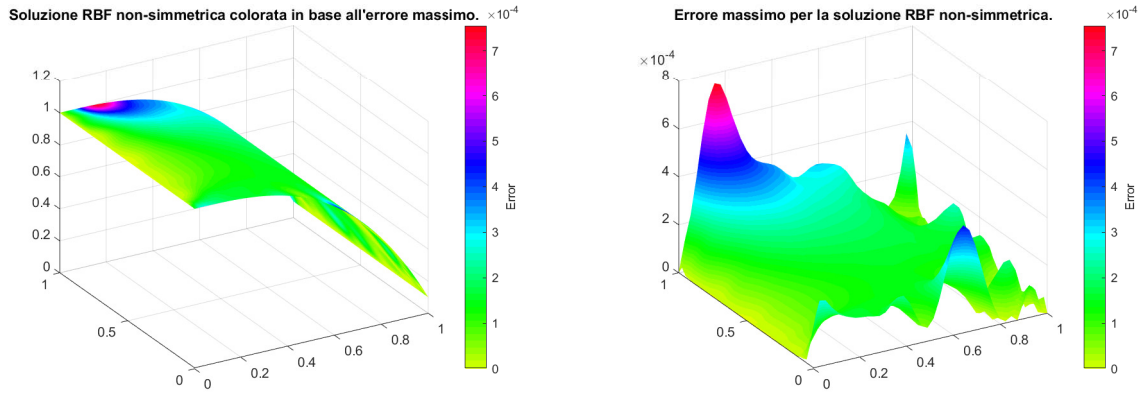

Figura 3.4: A sinistra: Soluzione ottenuta tramite collocazione non-simmetrica. A destra: Plot dell'errore. In entrambi i casi è stata utilizzata una IMQ con $\varepsilon = 3$ e $N = 289$ punti interni di Halton.

3.2 Collocazione simmetrica

In questa sezione concentriamo la nostra attenzione sul metodo implementativo simmetrico basato sull'interpolazione generalizzata di Hermite. Al fine di rendere più immediato il paragone tra i due metodi, risolveremo con questo metodo uno stesso problema della sezione precedente. Completeremo con un problema ancora ellittico in due dimensioni, la cui soluzione tuttavia non sarà nota.

All'interno dei codici che risolvono i problemi che andremo a presentare è necessario inserire anche il Laplaciano iterato della RBF utilizzata. Introduciamo perciò l'operatore differenziale

$$\begin{aligned}\nabla_\xi^2\nabla^2 &= \left(\frac{\partial^2}{\partial\xi^2}+\frac{\partial^2}{\partial\eta^2}\right)\left(\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}\right)\\ &= \left(\frac{\partial^2}{\partial\xi^2}\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial\eta^2}\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial\xi^2}\frac{\partial^2}{\partial y^2}+\frac{\partial^2}{\partial\eta^2}\frac{\partial^2}{\partial y^2}\right)\\ &= \left(\frac{\partial^4}{\partial x^4}+2\frac{\partial^4}{\partial x^2y^2}+\frac{\partial^4}{\partial y^4}\right),\end{aligned}$$

dove l'ultimo passaggio è dovuto al fatto che abbiamo a che fare con ordini di derivate pari e funzioni della forma $\varphi = \varphi(\|x-\xi\|)$. Se ad esempio definiamo $r = \|x-\xi\|$, otteniamo i seguenti Laplaciani iterati (riportiamo quelli relativi a Gaussiane e IMQ, in quanto ci serviamo di quelle specifiche funzioni per l'approssimazione della soluzione):

$$\nabla_\xi^2\nabla^2 e^{-(\varepsilon r)^2} = 16\varepsilon^4\left(2-4(\varepsilon r)^2+(\varepsilon r)^4\right)e^{-(\varepsilon r)^2}, \quad \text{Gaussiana,} \tag{3.1}$$

$$\nabla_\xi^2\nabla^2\frac{1}{\sqrt{1+(\varepsilon r)^2}} = \frac{3\varepsilon^4\left(3(\varepsilon r)^4-24(\varepsilon r)^2+8\right)}{\left(1+(\varepsilon r)^2\right)^{9/2}}, \quad \text{IMQ.} \tag{3.2}$$

Problema 3.4. Utilizziamo la stessa equazione alle derivate parziali e condizioni al bordo incontrata nel Problema 3.1.

Utilizziamo il metodo di collocazione simmetrico implementato nel Codice 3.4. Lungo la frontiera continuiamo a scegliere punti di collocazione e centri diversi, di conseguenza la matrice di collocazione non è simmetrica; potremmo renderla tale modificando il codice. Quando abbiamo introdotto questa stessa imposizione a proposito del metodo di Kansa, abbiamo sottolineato il fatto che quell'approccio fosse efficiente qualora i punti scelti fossero di Halton; nel caso di punti uniformi, però, restituiva una matrice di collocazione singolare. Nel contesto della collocazione simmetrica, invece, non ci sono problemi in nessun caso.

Raccogliamo i dati in tabelle analoghe a quelle riportate per il Problema 3.1, applicando però il metodo di Hermite. Possiamo trarre conclusioni molto simili a quelle tratte per il metodo di Kansa: la scelta migliore (per quanto riguarda errore e condizionamento) è quella di utilizzare Multiquadratiche Inverse con punti interni di Halton e centri di bordo esterni al dominio.

Tabella 3.6: Soluzione con collocazione simmetrica del Problema 3.4 con IMQ, $\varepsilon = 3$ e punti interni di Halton.

	Centri sul bordo		Centri esterni	
N(interni)	RMS-error	cond(A)	RMS-error	cond(A)
9	1.869505e-01	9.055720e+03	2.438041e-01	3.549895e+04
25	7.698471e-02	8.506782e+04	9.429580e-02	1.162027e+05
81	4.839682e-03	1.338599e+07	5.070833e-03	1.017388e+07
289	4.480250e-05	9.991615e+10	3.448546e-05	7.180249e+10
1089	2.481618e-08	2.821204e+18	1.907614e-08	2.268722e+18

Tabella 3.7: Soluzione con collocazione simmetrica del Problema 3.4 con Gaussiane e IMQ, $\varepsilon = 3$, punti interni uniformi e centri di bordo esterni al dominio.

	Gaussiane		IMQ	
N(interni)	RMS-error	cond(A)	RMS-error	cond(A)
3×3	4.088188e-01	1.196486e+05	2.806897e-01	3.105155e+04
5×5	7.704584e-03	1.359899e+05	1.583948e-01	1.216534e+05
9×9	2.272289e-04	2.453107e+10	8.650782e-04	2.016503e+07
17×17	5.961016e-08	1.847201e+21	3.962654e-05	6.051588e+11
33×33	1.767756e-07	2.185824e+22	1.867921e-07	3.133590e+20

Naturalmente ci si chiede quale metodo sia da preferire tra collocazione non-simmetrica e collocazione simmetrica; ebbene, confrontando le Tabelle 3.1, 3.2, 3.6, 3.7 notiamo che la differenza di performance tra i due approcci è piccola. Un'ulteriore conferma di questo fatto deriva da un confronto dell'errore compiuto dai due metodi utilizzando gli stessi parametri (Figura 3.5). Soprattutto con questo problema si nota che la convergenza è molto veloce utilizzando funzioni IMQ lisce su un problema che ha una soluzione liscia.

Per avere una visione d'insieme ancora più chiara riguardo i dati delle tabelle sopra citate, realizziamo un grafico di confronto degli errori (Figura 3.6). Notiamo innanzitutto come le curve relative alle Gaussiane si discostino da quelle relative alle IMQ: esse hanno un andamento interessante - persino migliore rispetto alle altre - fino al caso di $N = 289$ punti interni, ma quando questo numero cresce, l'accuratezza peggiora decisamente, indice del fatto che queste particolari RBF sono più indicate per domini con un numero di punti interni non troppo elevato. La collocazione di Hermite è da preferire a partire da $N = 81$, ma per quanto riguarda i punti precedenti, le curve degli errori dei due metodi sono quasi perfettamente sovrapponibili. Per quanto riguarda le Multiquadratiche Inverse, invece, notiamo come la crescita del numero di punti interni comporti una decrescita dell'errore, specialmente quando vengono utilizzati punti interni di Halton. In generale, con le IMQ, sembra che il metodo di Kansa abbia una performance

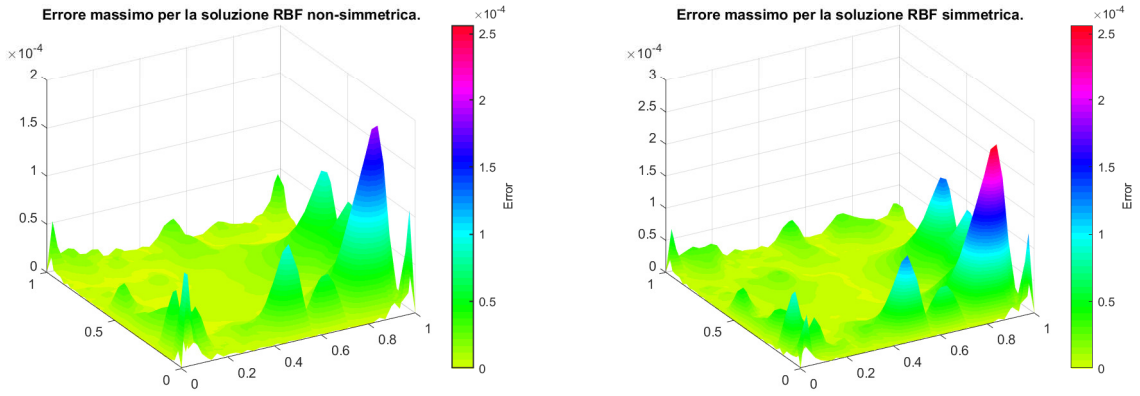

Figura 3.5: A sinistra: Plot dell'errore con collocazione non-simmetrica. A destra: Plot dell'errore con collocazione simmetrica. In entrambi i casi è stata utilizzata una IMQ con $\varepsilon = 3$, $N = 289$ punti interni di Halton e centri di bordo esterni al dominio.

migliore, soprattutto per domini con pochi punti interni; tuttavia più crescono i punti e più le linee dei due metodi si avvicinano fino quasi a coincidere.

Vorremmo applicare il metodo di Hermite al Problema 3.2, ma la presenza di coefficienti variabili complica notevolmente il tutto. L'operatore differenziale $\mathcal{L}$ è dato da

$$\mathcal{L} = \frac{\partial}{\partial x}\left(a(x,y)\frac{\partial}{\partial x}\right) + \frac{\partial}{\partial y}\left(b(x,y)\frac{\partial}{\partial y}\right),$$

mentre l'espansione basica della soluzione è

$$\sum_{j=1}^{N_{\mathcal{B}}} c_j \varphi(\|x-\xi_j\|) + \sum_{j=N_{\mathcal{B}}+1}^{N} c_j \mathcal{L}^{\xi}\varphi(\|x-\xi_j\|),$$

dove

$$\mathcal{L}^{\xi} = \frac{\partial}{\partial \xi}\left(a(\xi,\eta)\frac{\partial}{\partial \xi}\right) + \frac{\partial}{\partial \eta}\left(b(\xi,\eta)\frac{\partial}{\partial \eta}\right).$$

All'interno del codice, quindi, bisognerebbe modificare la computazione del blocco $\hat{A}_{\mathcal{L}\mathcal{L}^{\xi}}$ della matrice simmetrica di collocazione sviluppando il seguente calcolo:

$$\mathcal{L}\mathcal{L}^{\xi} = \left[\frac{\partial}{\partial x}\left(a(x,y)\frac{\partial}{\partial x}\right) + \frac{\partial}{\partial y}\left(b(x,y)\frac{\partial}{\partial y}\right)\right]\left[\frac{\partial}{\partial \xi}\left(a(\xi,\eta)\frac{\partial}{\partial \xi}\right) + \frac{\partial}{\partial \eta}\left(b(\xi,\eta)\frac{\partial}{\partial \eta}\right)\right],$$

che è molto laborioso e rende l'implementazione piuttosto pesante.

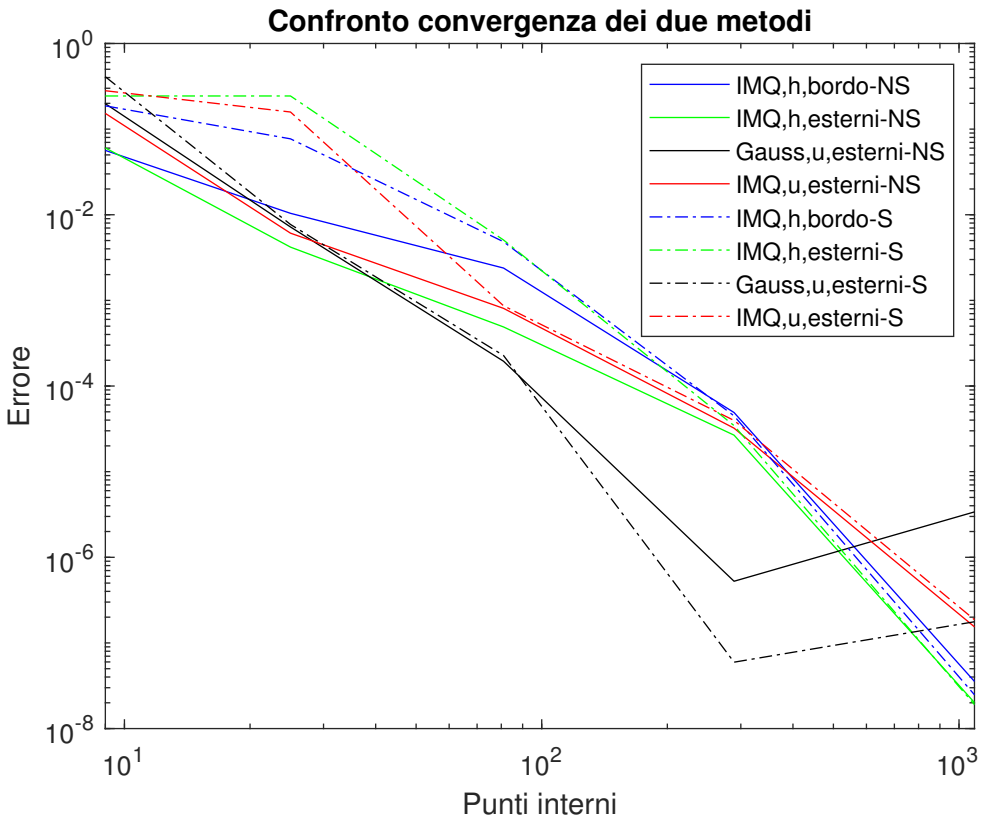

Figura 3.6: Confronto convergenza dei due metodi. Le curve di linea continua indicano l'utilizzo del metodo non-simmetrico `NS`, mentre quelle tratteggiate derivano dall'utilizzo dell'approccio simmetrico `S`; le altre diciture in legenda si riferiscono rispettivamente alla tipologia di RBF impiegata (`IMQ` : $\varphi(r) = \frac{1}{\sqrt{1+(\varepsilon r)^2}}$, `Gauss` : $\varphi(r) = e^{-(\varepsilon r)^2}$), alla distribuzione di punti interni scelta (`h` indica che sono stati scelti punti interni di Halton e `u` indica una distribuzione uniforme), alla posizione dei centri di bordo (`bordo` significa che i centri sono stati presi esattamente sul bordo e `esterni` vuol dire che sono stati presi esternamente al dominio di collocazione). Per una migliore comprensione della disposizione dei centri di bordo si veda Figura 3.7.

Consideriamo in alternativa un problema leggermente diverso da quelli visti in precedenza:

Problema 3.5. Consideriamo la seguente equazione con condizione al bordo definita a tratti:

$$\begin{aligned}
\nabla^2 u(x,y) &= 0, && (x,y) \in \Omega = (-1,1)^2,\\
u(x,y) &= 0, && (x,y) \in \Gamma_1 \cup \Gamma_3 \cup \Gamma_5,\\
u(x,y) &= \frac{1}{5}\sin(3\pi y), && (x,y) \in \Gamma_2,\\
u(x,y) &= \sin^4(\pi x), && (x,y) \in \Gamma_4,
\end{aligned}$$

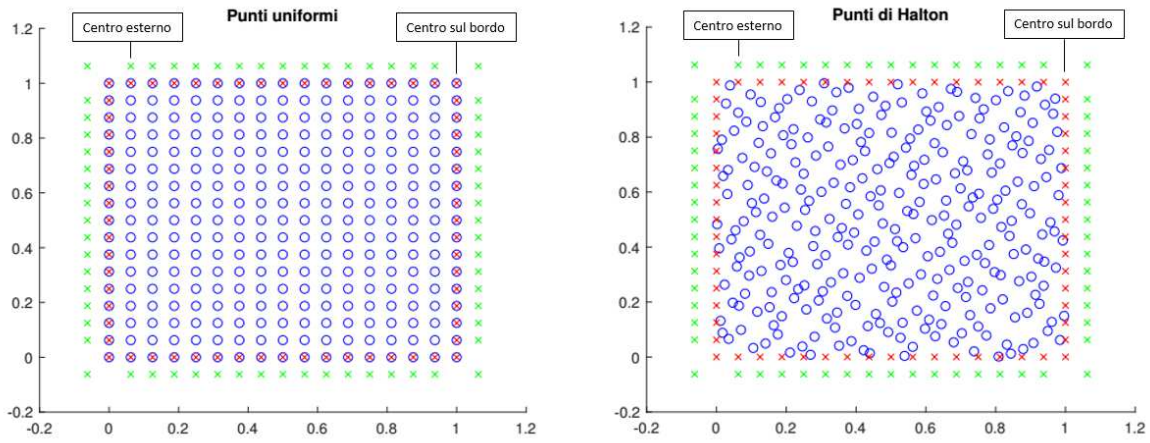

Figura 3.7: A sinistra: rappresentazione di un centro esterno e di un centro sul bordo nel caso di distribuzione di punti uniformi. A destra: rappresentazione di un centro esterno e di un centro sul bordo nel caso di distribuzione di punti di Halton.

dove

$$\Gamma_1 = \{(x, y) : \ -1 \leq x \leq 1, \ y = -1\},$$
$$\Gamma_2 = \{(x, y) : \ x = 1, \ -1 \leq y \leq 1\},$$
$$\Gamma_3 = \{(x, y) : \ 0 \leq x \leq 1, \ y = 1\},$$
$$\Gamma_4 = \{(x, y) : \ -1 \leq x \leq 0, \ y = 1\},$$
$$\Gamma_5 = \{(x, y) : \ x = -1, \ 0 \leq y \leq 1\}.$$

Utilizziamo il metodo di collocazione simmetrico implementato nel Codice 3.5. A differenza dei problemi considerati in precedenza, in questo caso non abbiamo a disposizione una soluzione esatta, quindi utilizziamo la soluzione pseudo-spettrale, che supponiamo nota.

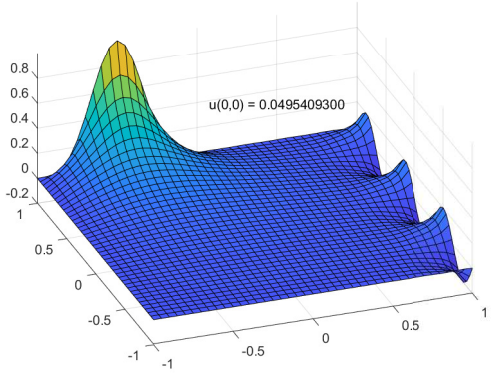

Figura 3.8: Soluzione ottenuta tramite collocazione simmetrica usando IMQ con $\varepsilon = 3$, $N = 289$ punti interni uniformi e 64 punti supplementari fuori dal dominio.

Si differenzia dai problemi precedenti anche in base al dominio utilizzato: negli altri casi era il quadrato $[0,1]^2$, mentre in questo esempio è il quadrato $[-1,1]^2$, quindi bisogna apportare alcune modifiche nel codice specialmente per quanto riguarda i punti di collocazione.

In Figura 3.8 vediamo il grafico della soluzione ottenuta applicando il metodo di Hermite; la qualità della soluzione è molto simile alla soluzione trovata con il metodo pseudo-spettrale. Nella figura abbiamo indicato il valore $u(0,0) = 0.0495409300$ ottenuto applicando questo metodo; con la soluzione pseudo-spettrale abbiamo ottenuto un valore pari a $u(0,0) = 0.0495946063$; notiamo che vi è davvero poca differenza.

Conclusioni

Alla luce degli esperimenti numerici che abbiamo realizzato, possiamo trarre delle conclusioni riguardo i metodi di collocazione simmetrico e non-simmetrico. Non possiamo esprimere in maniera netta la preferenza per l'uno o l'altro metodo, in quanto, come spesso accade, la preferenza è influenzata dal problema che dobbiamo risolvere.

In linea generale, il metodo di Kansa, a parità di parametro di forma ε, sembra essere migliore del metodo di Hermite sia riguardo all'errore sia riguardo al numero di condizionamento; si tratta comunque di differenze molto piccole. D'altra parte, il metodo simmetrico genera una matrice di collocazione simmetrica nel caso in cui tutti i centri coincidano con i punti di collocazione, perciò se si utilizza un risolutore per sistemi simmetrici, occorreranno molte meno computazioni.

Il metodo di Kansa necessita di meno derivate della funzione di base radiale, a differenza del metodo di Hermite, che richiede il calcolo del Laplaciano iterato; questo fa sì che il primo metodo sia più semplice da implementare e che si possa applicare facilmente a problemi con soluzioni di classi più deboli. C'è un'altra classe di problemi a cui il metodo non-simmetrico si applica facilmente: i problemi con coefficienti variabili; abbiamo visto infatti quanto sia immediata l'implementazione di questo metodo per questi problemi a differenza di quanto non lo sia quello di Hermite, che richiede calcoli laboriosi. In generale possiamo dire che non è ben chiaro come utilizzare il metodo simmetrico per problemi non-lineari.

Possiamo riassumere le osservazioni fatte dicendo che si tratta di due ottimi metodi, che vengono implementati e usati in molte applicazioni; da un lato il metodo di Hermite è meno versatile, ma veramente vantaggioso nel caso di matrici simmetriche; dall'altro lato il metodo di Kansa potrebbe richiedere un maggior numero di computazioni, ma si presta alla risoluzione di molte tipologie di problemi e ha la capacità di mantenere bassi sia gli errori sia i numeri di condizionamento.

Appendice A: Codici Matlab

Riportiamo di seguito i codici utilizzati nella risoluzione dei problemi del Capitolo 3, tratti dal libro [3]. Per ognuno di essi specificheremo come realizzare le varianti che abbiamo considerato nella nostra analisi.

Codice 3.1: KansaLaplace_2D

```
% KansaLaplace_2D
% Script that performs Kansa collocation for 2D Laplace
   equation
% Calls on: DistanceMatrix
% IMQ RBF and its Laplacian
rbf = @(e,r) 1./sqrt(1+(e*r).^2); ep = 3;
Lrbf = @(e,r) e^2*((e*r).^2-2)./(1+(e*r).^2).^(5/2);
% Gaussian RBF and its derivatives
% rbf = @(e,r) exp(-(e*r).^2); ep = 3;
% Lrbf = @(e,r) 4*e^2*exp(-(e*r).^2).*((e*r).^2-1);
% Exact solution and its Laplacian for test problem
u = @(x,y) sin(pi*x).*cos(pi*y/2);
Lu = @(x,y) -1.25*pi^2*sin(pi*x).*cos(pi*y/2);
% Number and type of collocation points
N = 289; gridtype = 'h';
neval = 40;
% Load (interior) collocation points
name = sprintf('Data2D_%d%s',N,gridtype); load(name);
intdata = dsites;
% Additional (equally spaced) boundary collocation
   points
sn = sqrt(N); bdylin = linspace(0,1,sn)';
bdy0 = zeros(sn-1,1); bdy1 = ones(sn-1,1);
bdydata = [bdylin(1:end-1) bdy0; bdy1 bdylin(1:end-1)
   ;...
     flipud(bdylin(2:end)) bdy1; bdy0 flipud(bdylin(2:
        end))];
% Create additional boundary centers OUTSIDE the domain
h = 1/(sn-1); bdylin = (h:h:1-h)';
bdy0 = -h*ones(sn-2,1); bdy1 = (1+h)*ones(sn-2,1);
% bdyctrs = bdydata;
```

```
bdyctrs = [-h -h; bdylin bdy0; 1+h -h; bdy1 bdylin;...
    1+h 1+h; flipud(bdylin) bdy1; -h 1+h; bdy0 flipud(
       bdylin)];
ctrs = [intdata; bdyctrs];
% Create neval-by-neval equally spaced evaluation
   locations
% in the unit square
grid = linspace(0,1,neval); [xe,ye] = meshgrid(grid);
epoints = [xe(:) ye(:)];
% Compute evaluation matrix
DM_eval = DistanceMatrix(epoints,ctrs);
EM = rbf(ep,DM_eval);
exact = u(epoints(:,1),epoints(:,2));
% Compute blocks for collocation matrix
DM_intdata = DistanceMatrix(intdata,ctrs);
LCM = Lrbf(ep,DM_intdata);
DM_bdydata = DistanceMatrix(bdydata,ctrs);
BCM = rbf(ep,DM_bdydata);
CM = [LCM; BCM];
% Create right-hand side
rhs = [Lu(intdata(:,1),intdata(:,2)); ...
     u(bdydata(:,1),bdydata(:,2))];
% Compute RBF solution
Pf = EM * (CM\rhs);
% Compute maximum error on evaluation grid
maxerr = norm(Pf-exact,inf);
rms_err = norm(Pf-exact)/neval;
fprintf('RMS error:     %e\n', rms_err)
fprintf('Maximum error: %e\n', maxerr)
fprintf('Cond:          %e\n', cond(CM))
% Plot collocation points and centers
hold on; plot(intdata(:,1),intdata(:,2),'bo');
plot(bdydata(:,1),bdydata(:,2),'rx');
plot(bdyctrs(:,1),bdyctrs(:,2),'gx'); hold off
fview = [-30,30];  % viewing angles for plot
caption = ['Soluzione RBF non-simmetrica '...
    'colorata in base all''errore massimo.'];
PlotSurf(xe,ye,Pf,neval,exact,maxerr,fview,caption);
caption = 'Errore massimo per la soluzione RBF non-
   simmetrica.';
PlotError2D(xe,ye,Pf,exact,maxerr,neval,fview,caption)
```

Abbiamo analizzato il diverso comportamento del metodo nel caso in cui i centri siano scelti sul bordo o esternamente al bordo: il codice riportato realizza il secondo scenario, ma se volessimo realizzare il primo dovremmo togliere il commento alla riga che riporta `bdyctrs = bdydata` ed eliminare la successiva. Si ha possibilità di scelta

anche per quanto concerne i punti interni (sia per quanto riguardo il numero sia per la loro distribuzione): basta modificare il valore di `N` e la lettera associata a `gridtype`, che è `h` nel caso in cui i punti interni siano punti di Halton e `u` nel caso in cui siano uniformi. Molto semplice è anche la modifica del parametro di forma dell'equazione di base radiale considerata, poiché viene modificato solo il valore di `ep`. Infine, abbiamo mostrato la performance delle IMQ confrontata con quella delle Gaussiane; il codice attuale risolve l'equazione servendosi delle IMQ, ma per utilizzare altre funzioni non bisogna fare altro che scrivere in `rbf` la funzione stessa e in `Lrbf` il suo Laplaciano.
Le modifiche scritte per questo codice sono valide anche per i codici successivi.

Codice 3.2: KansaEllipticVC_2D

```
% KansaEllipticVC_2D
% Script that performs Kansa collocation for 2D elliptic
    PDE
% with variable coefficients
% Calls on: DistanceMatrix, Differencematrix
% IMQ RBF and its derivatives
rbf = @(e,r) 1./sqrt(1+(e*r).^2); ep = 3;
dxrbf = @(e,r,dx) -dx*e^2./(1+(e*r).^2).^(3/2);
dyrbf = @(e,r,dy) -dy*e^2./(1+(e*r).^2).^(3/2);
dxxrbf = @(e,r,dx) e^2*(3*(e*dx).^2-1-(e*r).^2)./...
                        (1+(e*r).^2).^(5/2);
dyyrbf = @(e,r,dy) e^2*(3*(e*dy).^2-1-(e*r).^2)./...
                        (1+(e*r).^2).^(5/2);
% Gaussian RBF and its derivatives
% rbf = @(e,r) exp(-(e*r).^2); ep = 3;
% dxrbf = @(e,r,dx) -2*dx*e^2.*exp(-(e*r).^2);
% dyrbf = @(e,r,dy) -2*dy*e^2.*exp(-(e*r).^2);
% dxxrbf = @(e,r,dx) 2*e^2*(2*(e*dx).^2-1).*exp(-(e*r)
    .^2);
% dyyrbf = @(e,r,dy) 2*e^2*(2*(e*dy).^2-1).*exp(-(e*r)
    .^2);
% Test problem input (right-hand side, coefficients)
u = @(x,y) 16*x.*(1-x).*y.*(1-y);
Lu = @(x,y) -16*x.*exp(x-y).*(1-x).*(3-2*y)+...
            32*y.*(1-y).*(3*x.^2+y.^2-x-2);
a = @(x,y) 2-x.^2-y.^2; ax = @(x,y) -2*x;
b = @(x,y) exp(x-y); by = @(x,y)-exp(x-y);
N = 289; gridtype = 'h';
neval = 40;
% Load (interior) collocation points
name = sprintf('Data2D_%d%s',N,gridtype); load(name);
intdata = dsites;
```

```
% Additional boundary collocation points
sn = sqrt(N); bdylin = linspace(0,1,sn)';
bdy0 = zeros(sn-1,1); bdy1 = ones(sn-1,1);
bdydata = [bdylin(1:end-1) bdy0; bdy1 bdylin(1:end-1)
   ;...
     flipud(bdylin(2:end)) bdy1; bdy0 flipud(bdylin(2:
        end))];
% Create additional boundary centers OUTSIDE the domain
h = 1/(sn-1); bdylin = (h:h:1-h)';
bdy0 = -h*ones(sn-2,1); bdy1 = (1+h)*ones(sn-2,1);
bdyctrs = [-h -h; bdylin bdy0; 1+h -h; bdy1 bdylin;...
    1+h 1+h; flipud(bdylin) bdy1; -h 1+h; bdy0 flipud(
       bdylin)];
ctrs = [intdata; bdyctrs];
% Create neval-by-neval equally spaced evaluation
   locations
% in the unit square
grid = linspace(0,1,neval); [xe,ye] = meshgrid(grid);
epoints = [xe(:) ye(:)];
% Compute evaluation matrix
DM_eval = DistanceMatrix(epoints,ctrs);
EM = rbf(ep,DM_eval);
exact = u(epoints(:,1),epoints(:,2));
% Compute blocks for collocation matrix
DM_intdata = DistanceMatrix(intdata,ctrs);
DM_bdydata = DistanceMatrix(bdydata,ctrs);
dx_intdata = Differencematrix(intdata(:,1),ctrs(:,1));
dy_intdata = Differencematrix(intdata(:,2),ctrs(:,2));
LCM = diag(ax(intdata(:,1))) * ...
      dxrbf(ep,DM_intdata,dx_intdata) + ...
      diag(a(intdata(:,1),intdata(:,2))) * ...
      dxxrbf(ep,DM_intdata,dx_intdata) + ...
      diag(by(intdata(:,1),intdata(:,2))) * ...
      dyrbf(ep,DM_intdata,dy_intdata) + ...
      diag(b(intdata(:,1),intdata(:,2))) * ...
      dyyrbf(ep,DM_intdata,dy_intdata);
BCM = rbf(ep,DM_bdydata);
CM = [LCM; BCM];
% Create right-hand side
rhs = [Lu(intdata(:,1),intdata(:,2)); zeros(4*(sn-1),1)
   ];
% RBF solution
Pf = EM * (CM\rhs);
% Compute maximum error on evaluation grid
maxerr = norm(Pf-exact,inf);
rms_err = norm(Pf-exact)/neval;
fprintf('RMS error:     %e\n', rms_err)
```

```
fprintf('Maximum error: %e\n', maxerr)
fprintf('Cond:          %e\n', cond(CM))
% Plot collocation points and centers
hold on; plot(intdata(:,1),intdata(:,2),'bo');
plot(bdydata(:,1),bdydata(:,2),'rx');
plot(bdyctrs(:,1),bdyctrs(:,2),'gx'); hold off
fview = [-30,30];  % viewing angles for plot
caption = ['Soluzione RBF non-simmetrica '...
    'colorata in base all''errore massimo.'];
PlotSurf(xe,ye,Pf,neval,exact,maxerr,fview,caption);
caption = 'Errore massimo per la soluzione RBF non-
   simmetrica.';
PlotError2D(xe,ye,Pf,exact,maxerr,neval,fview,caption)
```

L'unica differenza con le modifiche descritte in precedenza risiede nel fatto che occorre lavorare con le derivate parziali prime e seconde della funzione di base scelta, quindi qualsiasi sia la RBF che scegliamo di utilizzare, di essa bisogna fornire funzione (`rbf`), derivate prime (`dxrbf` e `dyrbf`) e derivate seconde (`dxxrbf` e `dyyrbf`). Notiamo inoltre che vi è molta somiglianza con il codice precedente, fatta eccezione per l'assemblaggio della matrice di collocazione, che è più complicato perché dobbiamo applicare l'operatore differenziale alle funzioni di base.

Codice 3.3: KansaLaplaceMixedBC_2D

```
% KansaLaplaceMixedBC_2D
% Script that performs Kansa collocation for 2D Laplace
   equation
% with mixed BCs
% Calls on: DistanceMatrix, Differencematrix
% IMQ RBF and its Laplacian
rbf = @(e,r) 1./sqrt(1+(e*r).^2); ep = 3;
dyrbf = @(e,r,dy) -dy*e^2./(1+(e*r).^2).^(3/2);
Lrbf = @(e,r) e^2*((e*r).^2-2)./(1+(e*r).^2).^(5/2);
% Gaussian RBF and its derivatives
% rbf = @(e,r) exp(-(e*r).^2); ep = 3;
% dyrbf = @(e,r,dy) -2*dy*e^2.*exp(-(e*r).^2);
% Lrbf = @(e,r) 4*e^2*exp(-(e*r).^2).*((e*r).^2-1);
% Exact solution and its Laplacian for test problem
u = @(x,y) 1-0.9*x.^3+0*y;
Lu = @(x,y) -5.4*x+0*y;
% Number and type of collocation points
N = 289; gridtype = 'h';
neval = 40;
% Load (interior) collocation points
name = sprintf('Data2D_%d%s',N,gridtype); load(name);
```

```
intdata = dsites;
% Additional boundary collocation points
sn = sqrt(N); bdylin = linspace(0,1,sn)';
bdy0 = zeros(sn-1,1); bdy1 = ones(sn-1,1);
bdydata = [bdylin(1:end-1) bdy0; bdy1 bdylin(1:end-1);
   ...
     flipud(bdylin(2:end)) bdy1; bdy0 flipud(bdylin(2:
        end))];
% Create additional boundary centers OUTSIDE the domain
h = 1/(sn-1); bdylin = (h:h:1-h)';
bdy0 = -h*ones(sn-2,1); bdy1 = (1+h)*ones(sn-2,1);
bdyctrs = [-h -h; bdylin bdy0; 1+h -h; bdy1 bdylin; ...
    1+h 1+h; flipud(bdylin) bdy1; -h 1+h; bdy0 flipud(
       bdylin)];
ctrs = [intdata; bdyctrs];
% Create neval-by-neval equally spaced evaluation
   locations
% in the unit square
grid = linspace(0,1,neval); [xe,ye] = meshgrid(grid);
epoints = [xe(:) ye(:)];
% Compute evaluation matrix
DM_eval = DistanceMatrix(epoints,ctrs);
EM = rbf(ep,DM_eval);
exact = u(epoints(:,1),epoints(:,2));
% Compute blocks for collocation matrix
DM_intdata = DistanceMatrix(intdata,ctrs);
DM_bdydata = DistanceMatrix(bdydata,ctrs);
dy_bdydata = Differencematrix(bdydata(:,2),ctrs(:,2));
LCM = Lrbf(ep,DM_intdata);
BCM1 = -dyrbf(ep,DM_bdydata(1:sn-1,:),dy_bdydata(1:sn
   -1,:));
BCM2 = rbf(ep,DM_bdydata(sn:2*sn-2,:));
BCM3 = dyrbf(ep,DM_bdydata(2*sn-1:3*sn-3,:),...
             dy_bdydata(2*sn-1:3*sn-3,:));
BCM4 = rbf(ep,DM_bdydata(3*sn-2:end,:));
CM = [LCM; BCM1; BCM2; BCM3; BCM4];
% Create right-hand side
rhs = [Lu(intdata(:,1),intdata(:,2)); zeros(sn-1,1); ...
       0.1*ones(sn-1,1); zeros(sn-1,1); ones(sn-1,1)];
% RBF solution
Pf = EM * (CM\rhs);
% Compute maximum error on evaluation grid
maxerr = norm(Pf-exact,inf);
rms_err = norm(Pf-exact)/neval;
fprintf('RMS error:     %e\n', rms_err)
fprintf('Maximum error: %e\n', maxerr)
fprintf('Cond:          %e\n', cond(CM))
```

```
% Plot collocation points and centers
hold on; plot(intdata(:,1),intdata(:,2),'bo');
plot(bdydata(:,1),bdydata(:,2),'rx');
plot(bdyctrs(:,1),bdyctrs(:,2),'gx'); hold off
fview = [-30,30];  % viewing angles for plot
caption = ['Soluzione RBF non-simmetrica '...
    'colorata in base all''errore massimo.'];
PlotSurf(xe,ye,Pf,neval,exact,maxerr,fview,caption);
caption = 'Errore massimo per la soluzione RBF non-
   simmetrica.';
PlotError2D(xe,ye,Pf,exact,maxerr,neval,fview,caption)
```

Nel Problema 3.3 compare la derivata normale, che è funzione della derivata parziale rispetto a y, quindi nel caso si scelgano diverse RBF bisogna fornire per ciascuna la funzione stessa, la derivata parziale rispetto a y e il Laplaciano. Notiamo anche in questo caso la differenza nell'assemblaggio della matrice di collocazione.

Codice 3.4: HermiteLaplace_2D

```
% HermiteLaplace_2D
% Script that performs Hermite collocation for 2D
   Laplace equation
% Calls on: DistanceMatrix
% IMQ RBF and its Laplacian and double Laplacian
rbf = @(e,r) 1./sqrt(1+(e*r).^2); ep = 3;
Lrbf = @(e,r) e^2*((e*r).^2-2)./(1+(e*r).^2).^(5/2);
L2rbf = @(e,r) 3*e^4*(3*(e*r).^4-24*(e*r).^2+8)./...
                    (1+(e*r).^2).^(9/2);
% Gaussian RBF and its Laplacian and double Laplacian
% rbf = @(e,r) exp(-(e*r).^2); ep = 3;
% Lrbf = @(e,r) 4*e^2*exp(-(e*r).^2).*((e*r).^2-1);
% L2rbf = @(e,r) 16*e^4.*(2-4.*(e*r).^2+(e*r).^4).*exp
   (-(e*r).^2);
% Exact solution and its Laplacian for test problem
u = @(x,y) sin(pi*x).*cos(pi*y/2);
Lu = @(x,y) -1.25*pi^2*sin(pi*x).*cos(pi*y/2);
% Number and type of collocation points
N = 289; gridtype = 'h';
neval = 40;
% Load (interior) collocation points
name = sprintf('Data2D_%d%s',N,gridtype); load(name);
intdata = dsites;
% Additional (equally spaced) boundary collocation
   points
sn = sqrt(N); bdylin = linspace(0,1,sn)';
```

```
bdy0 = zeros(sn-1,1); bdy1 = ones(sn-1,1);
bdydata = [bdylin(1:end-1) bdy0; bdy1 bdylin(1:end-1);
   ...
     flipud(bdylin(2:end)) bdy1; bdy0 flipud(bdylin(2:
        end))];
% Create additional boundary centers OUTSIDE the domain
h = 1/(sn-1); bdylin = (h:h:1-h)';
bdy0 = -h*ones(sn-2,1); bdy1 = (1+h)*ones(sn-2,1);
bdyctrs = [-h -h; bdylin bdy0; 1+h -h; bdy1 bdylin; ...
     1+h 1+h; flipud(bdylin) bdy1; -h 1+h; bdy0 flipud(
        bdylin)];
% bdyctrs = bdydata;
intctrs = intdata;
% Create neval-by-neval equally spaced evaluation
   locations
% in the unit square
grid = linspace(0,1,neval); [xe,ye] = meshgrid(grid);
epoints = [xe(:) ye(:)];
% Compute evaluation matrix
DM_inteval = DistanceMatrix(epoints,intctrs);
LEM = Lrbf(ep,DM_inteval);
DM_bdyeval = DistanceMatrix(epoints,bdyctrs);
BEM = rbf(ep,DM_bdyeval);
EM = [LEM BEM];
exact = u(epoints(:,1),epoints(:,2));
% Compute blocks for collocation matrix
DM_IIdata = DistanceMatrix(intdata,intctrs);
LLCM = L2rbf(ep,DM_IIdata);
DM_IBdata = DistanceMatrix(intdata,bdyctrs);
LBCM = Lrbf(ep,DM_IBdata);
DM_BIdata = DistanceMatrix(bdydata,intctrs);
BLCM = Lrbf(ep,DM_BIdata);
DM_BBdata = DistanceMatrix(bdydata,bdyctrs);
BBCM = rbf(ep,DM_BBdata);
CM = [LLCM LBCM; BLCM BBCM];
% Create right-hand side
rhs = [Lu(intdata(:,1),intdata(:,2)); ...
       sin(pi*bdydata(1:sn-1,1)); zeros(3*(sn-1),1)];
% Compute RBF solution
Pf = EM * (CM\rhs);
% Compute maximum error on evaluation grid
maxerr = norm(Pf-exact,inf);
rms_err = norm(Pf-exact)/neval;
fprintf('RMS error:     %e\n', rms_err)
fprintf('Maximum error: %e\n', maxerr)
fprintf('Cond:          %e\n', cond(CM))
% Plot collocation points and centers
```

```
hold on; plot(intdata(:,1),intdata(:,2),'bo');
plot(bdydata(:,1),bdydata(:,2),'rx');
plot(bdyctrs(:,1),bdyctrs(:,2),'gx'); hold off
fview = [-30,30];  % viewing angles for plot
caption = ['Soluzione RBF simmetrica '...
    'colorata in base all''errore massimo.'];
PlotSurf(xe,ye,Pf,neval,exact,maxerr,fview,caption);
caption = 'Errore massimo per la soluzione RBF
   simmetrica.';
PlotError2D(xe,ye,Pf,exact,maxerr,neval,fview,caption)
```

Come accennato al momento della risoluzione del Problema 3.4, nel codice va inserito il Laplaciano iterato (`L2rbf`) della RBF. In generale l'implementazione è più complicata di quelle viste in precedenza in quanto la matrice di valutazione `EM` è composta da due pezzi (come la matrice di collocazione nel caso non-simmetrico) e di conseguenza la matrice di collocazione `CM` è formata da quattro pezzi.

Codice 3.5: HermiteLaplaceMixedBCTref_2D

```
% HermiteLaplaceMixedBCTref_2D
% Script that performs Hermite collocation for 2D
   Laplace equation
% Note: Prog 36 in Trefethen (2000), exact solution not
   provided
% Calls on: DistanceMatrix
% IMQ RBF and its Laplacian
rbf = @(e,r) 1./sqrt(1+(e*r).^2); ep = 3;
Lrbf = @(e,r) e^2*((e*r).^2-2)./(1+(e*r).^2).^(5/2);
L2rbf = @(e,r) 3*e^4*(3*(e*r).^4-24*(e*r).^2+8)./...
                    (1+(e*r).^2).^(9/2);
%  Laplacian for test problem
Lu = @(x,y) zeros(size(x));
% Number and type of collocation points
N = 289; gridtype = 'u';
neval = 41;
% Load (interior) collocation points
name = sprintf('Data2D_%d%s',N,gridtype); load(name);
intdata = 2*dsites-1;
% Additional boundary collocation points
sn = sqrt(N); bdylin = linspace(-1,1,sn)';
bdy1 = ones(sn-1,1);
bdydata = [bdylin(1:end-1) -bdy1; bdy1 bdylin(1:end-1);
   ...
     flipud(bdylin(2:end)) bdy1; -bdy1 flipud(bdylin(2:
        end))];
```

```
% Create additional boundary centers OUTSIDE the domain
h = 2/(sn-1); bdylin = (-1+h:h:1-h)';
bdy0 = repmat(-1-h,sn-2,1); bdy1 = repmat(1+h,sn-2,1);
bdyctrs = [-1-h -1-h; bdylin bdy0; 1+h -1-h; bdy1 bdylin
   ; ...
     1+h 1+h; flipud(bdylin) bdy1; -1-h 1+h; bdy0 flipud
        (bdylin)];
intctrs = intdata;
% Create neval-by-neval equally spaced evaluation
   locations
% in the unit square
grid = linspace(-1,1,neval); [xe,ye] = meshgrid(grid);
epoints = [xe(:) ye(:)];
% Compute evaluation matrix
DM_inteval = DistanceMatrix(epoints,intctrs);
LEM = Lrbf(ep,DM_inteval);
DM_bdyeval = DistanceMatrix(epoints,bdyctrs);
BEM = rbf(ep,DM_bdyeval);
EM = [LEM BEM];
% Compute blocks for collocation matrix
DM_IIdata = DistanceMatrix(intdata,intctrs);
LLCM = L2rbf(ep,DM_IIdata);
DM_IBdata = DistanceMatrix(intdata,bdyctrs);
LBCM = Lrbf(ep,DM_IBdata);
DM_BIdata = DistanceMatrix(bdydata,intctrs);
BLCM = Lrbf(ep,DM_BIdata);
DM_BBdata = DistanceMatrix(bdydata,bdyctrs);
BBCM = rbf(ep,DM_BBdata);
CM = [LLCM LBCM; BLCM BBCM];
% Create right-hand side
rhs = [Lu(intdata(:,1),intdata(:,2)); zeros(sn-1,1); ...
       0.2*sin(3*pi*bdydata(sn:2*sn-2,2)); zeros((sn-1)
          /2,1);...
       sin(pi*bdydata((5*sn-3)/2:3*sn-3,1)).^4; zeros(sn
          -1,1)];
% Compute RBF solution
Pf = EM * (CM\rhs);
surf(xe,ye,reshape(Pf,neval,neval));
view(-20,45),  axis([-1 1 -1 1 -.2 1]);
text(0,.8,.5,sprintf('u(0,0) = %12.10f',Pf(841)))
```

Per questo ultimo codice, le modifiche sono dovute principalmente al fatto che nel Problema 3.5 il dominio è il quadrato $[-1,1]^2$, mentre negli altri casi era il quadrato $[0,1]^2$, quindi ad esempio i punti di collocazione che carichiamo dal file vengono trasformati imponendo $\texttt{intdata} = 2 * \texttt{dsites} - 1$.

Bibliografia

[1] Davide Boscaini, Antoine Glorieux, and Simone Parisotto. *Solving PDEs with RBF collocation*. Università degli Studi di Verona, 2012. `http://profs.sci.univr.it/~bos/DatiSparsi/AA2011/rbfcollocation.pdf`.

[2] Gregory E Fasshauer. Solving partial differential equations by collocation with radial basis functions. In *Proceedings of Chamonix*, volume 1997, pages 1–8. Vanderbilt University Press Nashville, TN, 1996.

[3] Gregory E Fasshauer. *Meshfree approximation methods with MATLAB*, volume 6. World Scientific, 2007.

[4] AI Fedoseyev, MJ Friedman, and EJ Kansa. Improved multiquadric method for elliptic partial differential equations via PDE collocation on the boundary. *Computers & Mathematics with Applications*, 43(3-5):439–455, 2002.

[5] YC Hon and Robert Schaback. On unsymmetric collocation by radial basis functions. *Applied Mathematics and Computation*, 119(2-3):177–186, 2001.

[6] Edward J Kansa. Multiquadrics - A scattered data approximation scheme with applications to computational fluid-dynamics - II solutions to parabolic, hyperbolic and elliptic partial differential equations. *Computers & mathematics with applications*, 19(8-9):147–161, 1990.

[7] Elisabeth Larsson and Bengt Fornberg. A numerical study of some radial basis function based solution methods for elliptic PDEs. *Computers and Mathematics with Applications*, 46(5-6):891–902, 2003.

[8] Ulrika Pettersson. *Global radial basis function collocation methods for PDEs*. PhD thesis, Uppsala University, 2020.

[9] Holger Wendland. *Scattered data approximation*, volume 17. Cambridge university press, 2004.