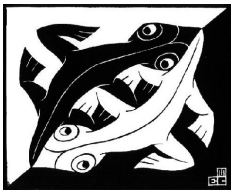
Solving PDEs with RBF collocation

Davide Boscaini, Antoine Glorieux e Simone Parisotto

Le presenti note, oggetto di presentazione in data 7 Maggio 2012, sono tratte principalmente da [9, cap. 16] e [1, cap. 36-41]. In questo seminario si propongono metodi simmetrici e non simmetrici per risolvere Equazioni a Derivate Parziali (PDEs). Invitiamo chiunque trovasse eventuali errori di segnalarli scrivendo ad uno qualsiasi degli indirizzi davide.boscaini@studenti.univr.it, antoine.c.glorieux@gmail.com o simone.parisotto@studenti.univr.it.



Colophon

Questo lavoro è stato realizzato con LaTeX usando la classe `scrreprt`, una rielaborazione della più nota classe `report`, reperibile presso http://www.ctan.org/tex-archive/macros/latex/contrib/koma-script.

La copertina riproduce l'incisione su legno *Plane filling motif with reptiles* di Maurits Cornelis Escher, tratta da http://www.mcescher.com. Volevamo rappresentare l'idea che i metodi di collocazione sono una perfetta sintesi tra la teoria dell'interpolazione generalizzata su dati sparsi, tipicamente numerica, e i metodi risolutivi per PDE, prevalentemente analitici.

Indice

1 Interpolazione generalizzata di Hermite **4**
1.1 Un po' di teoria 4
1.2 Verso la formulazione simmetrica 5

2 Metodi di Collocazione con RBF **8**
2.1 L'approccio non-simmetrico di Kansa 8
2.2 L'approccio simmetrico basato su Hermite 10
2.3 Stime dell'errore per la collocazione simmetrica 11
2.4 Altri problemi 12

3 Alcune applicazioni **14**
3.1 Collocazione non-simmetrica 14
3.2 Collocazione simmetrica 24
3.3 Collocazione con RBF a supporto compatto 32

4 Esempio di matrice di collocazione singolare **38**

Riferimenti bibliografici **44**

1 Interpolazione generalizzata di Hermite

Per interpolazione *generalizzata* di Hermite si intende la possibilità di interpolare non solo valori presi da qualche esperimento o da campionamenti discreti di funzioni note ma anche di dati astratti contenenti altre informazioni sulle funzioni. Come vedremo più avanti c'è un legame stretto tra l'interpolazione generalizzata e il metodo simmetrico di collocazione per PDE ellittiche.

1.1 Un po' di teoria

Consideriamo i dati $\{\boldsymbol{x}_i, \lambda_i f\}$, $i = 1, \ldots, N$, $\boldsymbol{x}_i \in \mathbb{R}^s$, dove $\Lambda = \{\lambda_1, \ldots, \lambda_N\}$ sono dei funzionali lineari linearmente indipendenti e f è una qualche funzione liscia assegnata. Se, ad esempio, scegliamo come λ_i le valutazioni puntuali sui punti $\boldsymbol{x}_i$, si ritorna ad una interpolazione nel senso classico. Un'altra possibile scelta è quella di considerare come λ_i le derivate nei punti $\boldsymbol{x}_i$, oppure ancora i λ_i possono rappresentare degli integrali locali. Fatta una particolare scelta dei λ_i cerchiamo un'interpolante della forma

$$\mathcal{P}_f(\boldsymbol{x}) = \sum_{j=1}^{N} c_j \psi_j(\|\boldsymbol{x}\|), \quad \boldsymbol{x} \in \mathbb{R}^s, \tag{1.1}$$

dove le ψ_j sono appropriate funzioni di base e imponiamo la condizione di interpolazione generalizzata

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

Per mantenere la scrittura il più chiara possibile indicheremo con $\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_N$ i centri delle funzioni radiali di base e con $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ i data sites, anche se spesso questi due insiemi li considereremo coincidenti. Come potremmo vedere nel prossimo capitolo, sarà naturale porre $\psi_j(\|\boldsymbol{x}\|) = \lambda_j^{\boldsymbol{\xi}} \varphi(\|\boldsymbol{x} - \boldsymbol{\xi}\|)$ dove φ una funzione radiale di base di quelle solitamente usate. Con la notazione $\lambda^{\boldsymbol{\xi}}$ indichiamo che il funzionale λ ora agisce sul secondo argomento di φ. Assumeremo quindi che l'interpolante generalizzata di Hermite sia della forma

$$\mathcal{P}_f(\boldsymbol{x}) = \sum_{j=1}^{N} c_j \lambda_j^{\boldsymbol{\xi}} \varphi(\|\boldsymbol{x} - \boldsymbol{\xi}\|), \quad \boldsymbol{x} \in \mathbb{R}^s \tag{1.2}$$

e che soddisfi la condizione "di interpolazione"

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

Il sistema lineare $A\boldsymbol{c} = \boldsymbol{f}_\lambda$ che ne deriva ha matrice

$$A_{ij} = \lambda_i \lambda_j^{\boldsymbol{\xi}} \varphi, \quad i, j = 1, \ldots, N, \tag{1.3}$$

e termine noto $\boldsymbol{f}_\lambda = [\lambda_1 \boldsymbol{f}, \ldots, \lambda_N \boldsymbol{f}]^T$.

Si osservi che, nell'assemblaggio della matrice, i funzionali λ agiscono su φ sia come funzione della prima variabile che come funzione della seconda variabile, questo implica che dovremo usare funzioni C^{2k} per interpolare dati C^k: questo è il prezzo da pagare per assicurare l'invertibilità di A.

Si può anche usare un'espansione RBF più semplice del tipo

$$\mathcal{P}_f(\boldsymbol{x}) = \sum_{j=1}^{N} c_j \varphi(\|\boldsymbol{x} - \boldsymbol{\xi}\|), \quad \boldsymbol{x} \in \mathbb{R}^s, \tag{1.4}$$

tuttavia in questo caso la matrice non sarà più simmetrica e, quindi, si perdono molte buone proprietà. Come vedremo nel prossimo capitolo l'approccio appena considerato è spesso usato nella soluzione di PDE ellittiche e, come approfondiremo più in dettaglio nell'ultimo capitolo, per certe configurazioni di punti la matrice del sistema lineare potrà essere singolare.

Infine in [9, pag. 292] troviamo il seguente teorema

Teorema 1.1. *Supponiamo che $\Phi \in L_1(\mathbb{R}^s) \cap C^{2k}(\mathbb{R}^s)$ sia un nucleo strettamente definito positivo. Se i funzionali $\lambda_j = \delta_{\boldsymbol{x}_j} \circ D^{\boldsymbol{\alpha}^{(j)}}$, $j = 1, \ldots, N$, con multi-indici $|\boldsymbol{\alpha}^{(j)}| \leq k$, sono a due a due distinti, con ciò intendendo che $\boldsymbol{\alpha}^{(j)} \neq \boldsymbol{\alpha}^{(\ell)}$ se $\boldsymbol{x}_j = \boldsymbol{x}_\ell$ per $j \neq \ell$, allora essi sono linearmente indipendenti sullo spazio nativo $\mathcal{N}_\Phi(\mathbb{R}^s)$.*

In particolare con $\delta_{\boldsymbol{x}_j}$ si indica la valutazione nel punto $\boldsymbol{x}_j$ e il nucleo $\Phi(\boldsymbol{x}, \boldsymbol{\xi})$ è legato a φ nel solito modo, cioè $\Phi(\boldsymbol{x}, \boldsymbol{\xi}) = \varphi(\|\boldsymbol{x} - \boldsymbol{\xi}\|)$. Come molti dei risultati sulle funzioni definite positive, questo teorema può ovviamente essere generalizzato anche al caso si funzioni *condizionatamente* definite positive.

Se poi i funzionali λ_j sono linearmente indipendenti sullo spazio nativo $\mathcal{N}_\Phi(\Omega)$, si può mostrare che una tale A è simmetrica e definita positiva, perché vale

$$\lambda_i \lambda_j^{\boldsymbol{\xi}} \Phi(x, y) = \langle \lambda_i \Phi(\cdot, x), \lambda_j^{\boldsymbol{\xi}} \Phi(\cdot, y) \rangle_{\mathcal{N}_\Phi(\Omega)}.$$

Un corollario di questo fatto implica che A è non singolare per la classe delle RBF solitamente usate per l'interpolazione di dati sparsi. Per i dettagli consultare [9, par. 16.2–16.3].

1.2 Verso la formulazione simmetrica

In questo paragrafo illustreremo perché la formulazione (1.2) è naturale per il problema dell'interpolazione di Hermite. Mostreremo inoltre che, se scegliamo le funzioni di base come in (1.2), la matrice associata all'interpolazione di Hermite può essere vista come limite di matrici di interpolazione di Lagrange classica. Vale infatti il seguente

Teorema 1.2. *In dimensione uno, sia M_L la matrice di Lagrange corrispondente alle funzioni di base centrate in ξ_j e in $\xi_j + \Delta\xi$ ed interpolante i valori x_i e $x_i + \Delta x$,*

$$M_L = \begin{bmatrix} \varphi(|x_i - \xi_j|) & \varphi(|x_i - (\xi_j + \Delta\xi)|) \\ \varphi(|(x_i + \Delta x) - \xi_j|) & \varphi(|(x_i + \Delta x) - (\xi_j + \Delta\xi)|) \end{bmatrix}$$

ed M_H il blocco di Hermite associato

$$M_H = \begin{bmatrix} \varphi(|x_i - \xi_j|) & -\frac{\partial}{\partial \xi}\varphi(|x_i - \xi_j|) \\ \frac{\partial}{\partial x}\varphi(|x_i - \xi_j|) & -\frac{\partial^2}{\partial x \partial \xi}\varphi(|x_i - \xi_j|) \end{bmatrix},$$

allora vale

$$\frac{\det M_L}{\Delta x \Delta \xi} = \det M_H + \mathcal{O}(\Delta x) + \mathcal{O}(\Delta \xi).$$

Illustriamo ora l'approccio dell'interpolazione di Hermite con un semplice esempio in dimensione due usando come funzionali delle derivate del primo ordine.

Esempio 1.1. Consideriamo i dati $\{\boldsymbol{x}_i, f(\boldsymbol{x}_i)\}_{i=1}^n$ e $\{\boldsymbol{x}_i, \frac{\partial f}{\partial x}(\boldsymbol{x}_i)\}_{i=n+1}^N$ con $\boldsymbol{x} = (x, y) \in \mathbb{R}^2$ fissato, $\boldsymbol{\xi} = (\xi, \eta) \in \mathbb{R}^2$ e

$$\lambda_i = \begin{cases} \delta_{\boldsymbol{x}_i}, & i = 1, \dots, n, \\ \delta_{\boldsymbol{x}_i} \circ \frac{\partial}{\partial x}, & i = n+1, \dots, N. \end{cases}$$

Allora

$$\begin{aligned} \mathcal{P}_f(\boldsymbol{x}) &= \sum_{j=1}^N c_j \lambda_j^{\boldsymbol{\xi}} \varphi(\|\boldsymbol{x} - \boldsymbol{\xi}\|) \\ &= \sum_{j=1}^n c_j \varphi(\|\boldsymbol{x} - \boldsymbol{\xi}_j\|) + \sum_{j=n+1}^N c_j \frac{\partial \varphi}{\partial \xi}(\|\boldsymbol{x} - \boldsymbol{\xi}_j\|) \\ &= \sum_{j=1}^n c_j \varphi(\|\boldsymbol{x} - \boldsymbol{\xi}_j\|) - \sum_{j=n+1}^N c_j \frac{\partial \varphi}{\partial x}(\|\boldsymbol{x} - \boldsymbol{\xi}_j\|). \end{aligned}$$

Dopo aver imposto le condizioni di interpolazione la matrice del sistema lineare diventa

$$A = \begin{bmatrix} \widetilde{A} & \widetilde{A}_\xi \\ \widetilde{A}_x & \widetilde{A}_{x\xi} \end{bmatrix}$$

dove

$$\begin{aligned} \widetilde{A}_{ij} &= \varphi(\|\boldsymbol{x}_i - \boldsymbol{\xi}_j\|) \quad i, j = 1, \dots, n, \\ (\widetilde{A}_\xi)_{ij} &= \frac{\partial \varphi}{\partial \xi}(\|\boldsymbol{x}_i - \boldsymbol{\xi}_j\|) = -\frac{\partial \varphi}{\partial x}(\|\boldsymbol{x}_i - \boldsymbol{\xi}_j\|) \quad i = 1, \dots, n,\ j = n+1, \dots, N, \\ (\widetilde{A}_x)_{ij} &= \frac{\partial \varphi}{\partial x}(\|\boldsymbol{x}_i - \boldsymbol{\xi}_j\|) \quad i = n+1, \dots, N,\ j = 1, \dots, n, \\ (\widetilde{A}_{x\xi})_{ij} &= \frac{\partial^2 \varphi}{\partial x^2}(\|\boldsymbol{x}_i - \boldsymbol{\xi}_j\|) \quad i, j = n+1, \dots, N. \end{aligned}$$

Si osservi che, la derivata parziale di φ rispetto alla coordinata x conterrà sempre un fattore lineare in x, infatti posto $\varphi(\|\boldsymbol{x}\|) = \varphi(r) = \varphi(\sqrt{x^2 + y^2})$, per la regola della catena

$$\begin{aligned} \frac{\partial}{\partial x}\varphi(\|\boldsymbol{x}\|) &= \frac{d}{dr}\varphi(r)\frac{d}{dx}r(x, y) \\ &= \frac{d}{dr}\varphi(r)\frac{x}{\sqrt{x^2 + y^2}} \\ &= \frac{d}{dr}\varphi(r)\frac{x}{r}. \end{aligned}$$

Gli stessi calcoli possono essere generalizzati per una qualsiasi derivata di ordine dispari. Di conseguenza il cambio di segno dovuto alla differenziazione rispetto alla seconda variabile in $\widetilde{A}_\xi$ può essere cancellato scambiando il ruolo delle variabili mute $\boldsymbol{x}_i$ e $\boldsymbol{\xi}_j$, di conseguenza i due blocchi $\widetilde{A}_\xi$ e $\widetilde{A}_x$ sono identici.

Osserviamo infine che la matrice A è simmetrica anche nel caso di derivate di ordine pari, infatti

$$\frac{\partial^2}{\partial^2 x}\varphi(\|\boldsymbol{x}\|) = \frac{1}{r^2}\left(x^2\frac{d^2}{d^2 r}\varphi(r) + \frac{y^2}{r}\frac{d}{dr}\varphi(r)\right),$$

e ancora una volta se scambiamo $\boldsymbol{x}_i$ con $\boldsymbol{\xi}_j$ questo non causerà un cambio di segno. Ovviamente lo stesso accade anche se deriviamo due volte φ rispetto alla seconda variabile $\boldsymbol{\xi}$.

2 Metodi di Collocazione con RBF

In questo capitolo discuteremo come le tecniche basate sull'interpolazione di Hermite possano essere applicate alla soluzione numerica di equazioni differenziali parziali ellittiche. Il metodo numerico risultante avrà un *approccio di collocazione* basato sulle funzioni radiali di base. Nella letteratura delle PDE, in questo contesto ci si riferisce alla *soluzione in forma forte*. Per rendere la nostra discussione trasparente inizieremo a focalizzarci sul caso di una equazione differenziale parziale ellittica indipendente dal tempo, in $\mathbb{R}^2$.

2.1 L'approccio non-simmetrico di Kansa

Il *metodo non-simmetrico* per trovare la soluzione di PDEs ellittiche, oggi molto popolare, è stato proposto in [7] da Kansa. Per chiarire le differenze tra il metodo di Kansa ed il metodo simmetrico, proposto in [2] da Fasshauer, richiamiamo alcune delle proprietà di base dell'interpolazione di dati sparsi con funzioni radiali di base in $\mathbb{R}^s$.

Nel contesto di interpolazione di dati sparsi, ci vengono forniti dei dati $\{\boldsymbol{x}_i, f_i\}$, $i = 1, \ldots, N$, $\boldsymbol{x}_i \in \mathbb{R}^s$, e dove supponiamo che i valori f_i provengano da un campionamento della funzione $f : \mathbb{R}^s \to \mathbb{R}$. Lo scopo è quello di trovare una interpolante della forma

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

tale che

$$\mathcal{P}_f(\boldsymbol{x}_i) = f_i, \quad i = 1, \ldots, N.$$

La soluzione di questo problema porta al sistema lineare $A\boldsymbol{c} = \boldsymbol{f}$, con le entrate della matrice A date da

$$A_{ij} = \varphi(\|\boldsymbol{x}_i - \boldsymbol{x}_j\|), \quad i, j = 1, \ldots, N.$$

Come discusso in precedenza, la matrice A è non singolare per una larga classe di funzioni radiali, incluse le Multiquadriche Inverse, le Gaussiane e le funzioni strettamente definite positive a supporto compatto di Wendland, Wu, Gneiting e Buhmann. Nel caso delle funzioni strettamente condizionatamente definite positive come le splines poliarmoniche, bisogna aumentare lo spazio del problema con dei polinomi.

Passiamo alla soluzione di equazioni differenziali parziali ottenuta per collocazione. Assumiamo che venga dato un dominio $\Omega \subset \mathbb{R}^s$, e un'equazione differenziale parziale ellittica della forma

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

con (per semplicità della descrizione) condizioni al bordo di Dirichlet

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

Per il metodo di collocazione di Kansa, scegliamo di rappresentare la soluzione approssimata $\hat{u}$ tramite una espansione di funzioni radiali di base analoga a quella usata per l'interpolazione di dati sparsi, cioè

$$\hat{u}(\boldsymbol{x}) = \sum_{j=1}^{N} c_j \varphi(\|\boldsymbol{x} - \boldsymbol{\xi}_j\|). \tag{2.3}$$

Come nel capitolo precedente, basato sull'interpolazione di Hermite, in questo contesto distingueremo formalmente nella nostra notazione i *centri* $\Xi = \{\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_N\}$ e i *punti di collocazione* $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\} \subset \Omega$. Formalmente differenti, questi punti spesso coincideranno fisicamente. L'ipotesi in cui $\Xi \neq \mathcal{X}$ sarà esplorata nei capitoli successivi. Per i ragionamenti che seguono, assumeremo l'ambientazione più semplice possibile, cioè $\Xi = \mathcal{X}$ e nessun termine polinomiale sarà aggiunto all'equazione (2.3).

La matrice di collocazione che deriva dalla equazione differenziale (2.1) e le condizioni di bordo (2.2) nei punti di collocazione $\mathcal{X}$ sarà della forma

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

dove i due blocchi sono generati nel modo seguente:

$$\begin{aligned} (\tilde{A}_{\mathcal{L}})_{ij} &= \mathcal{L}\varphi(\|\boldsymbol{x} - \boldsymbol{\xi}_j\|)|_{\boldsymbol{x}=\boldsymbol{x}_i}, \quad \boldsymbol{x}_i \in \mathcal{I},\, \boldsymbol{\xi}_j \in \Xi, \\ \tilde{A}_{ij} &= \varphi(\|\boldsymbol{x}_i - \boldsymbol{\xi}_j\|), \quad \boldsymbol{x}_i \in \mathcal{B},\, \boldsymbol{\xi}_j \in \Xi. \end{aligned}$$

Qui, l'insieme dei punti di collocazione $\mathcal{X}$ è diviso in due insiemi: $\mathcal{I}$, insieme dei punti interni, e $\mathcal{B}$ insieme dei punti di bordo. Il problema è ben posto se il sistema lineare $A\boldsymbol{c} = \boldsymbol{y}$, con $\boldsymbol{y}$ un vettore di entrate $f(\boldsymbol{x}_i)$, $\boldsymbol{x}_i \in \mathcal{I}$, seguito da $g(\boldsymbol{x}_i)$, $\boldsymbol{x}_i \in \mathcal{B}$, ha un'unica soluzione.

Notiamo che un cambio delle condizioni di bordo (2.2) si riflette in un semplice cambio di poche righe della matrice A di (2.4) così come in un cambio del vettore dei termini noti $\boldsymbol{y}$.

Sottolineiamo che mentre questo è una descrizione piuttosto generale del metodo numerico, con nessuna RBF esplicitata, in [7] Kansa specificò esplicitamente l'uso delle Multiquadriche Inverse nell'equazione (2.3) e di conseguenza questo approccio di collocazione non simmetrica spesso appare nella letteratura sotto il nome di *metodo delle Multiquadratiche*. Kansa descrisse tre gruppi di esperimenti usando il metodo delle Multiquadratiche e concluse che le Multiquadratiche offrivano una prestazione maggiore in termini di complessità computazionale ed accuratezza, confrontate al metodo degli elementi finiti. Inoltre, Kansa suggerì di variare il parametro di forma ε_j, $j = 1, \ldots, , N$.

Mentre l'analisi teorica del metodo risultante non è maneggevole, Kansa mostrò che questa tecnica migliora l'accuratezza e la stabilità del metodo, confrontato all'uso di un solo parametro di forma ε. Solo un lavoro di Bozzini, Lenarduzzi e Schaback, nel

solo contesto dell'interpolazione, ha trattato gli aspetti teorici del variare il parametro di forma.

Uno dei problemi del metodo proposto da Kansa, per un parametro ε fissato, è che la matrice A potrebbe essere singolare per una certa configurazione dei centri $\boldsymbol{\xi}_j$. In origine, Kansa assunse che la non singolarità della matrice derivasse dal Teorema di Micchelli per le matrici di interpolazione. Ma ciò non è vero come dimostra l'esperimento numerico di Hon e Schaback nell'articolo [6] del 2001, proposto nel Capitolo 4. Ciò non deve sorprendere, infatti la matrice per il problema di collocazione è composta da righe che sono costruite da funzioni *differenti* che, dipendendo anche dall'operatore differenziale $\mathcal{L}$, potrebbero non essere radiali. I risultati della non singolarità di A si basano sul fatto che A è generata da una *singola* funzione φ. Poiché l'esperimento numerico di Hon e Schaback ha mostrato che il metodo di Kansa non può essere ben posto per un arbitrario insieme di centri di collocazione, è un problema aperto quello di trovare condizioni sufficienti sui centri affinché la matrice di Kansa sia invertibile. Un possibile approccio, basato sull'idea di algoritmi Greedy, è di selezionare *buoni* centri in maniera adattativa in un più grande insieme di centri candidati. Seguendo questo procedimento, è possibile assicurare l'invertibilità della matrice di collocazione, attraverso un algoritmo iterativo.

Il metodo proposto da Kansa può essere comunque applicato ad altri tipo di equazioni differenziali parziali, come PDEs ellittiche non lineari, sistemi di PDEs ellittiche e PDEs paraboliche o iperboliche dipendenti dal tempo (si veda ad esempio un lavoro del 1994 di Moridis e Kansa).

2.2 L'approccio simmetrico basato su Hermite

Il seguente metodo di collocazione simmetrico è basato sul metodo di interpolazione generalizzato di Hermite (Capitolo 1). Assumiamo che ci venga dato la stessa PDE ellittica lineare (2.1), con condizioni al bordo di Dirichlet (2.2), vista nel capitolo che descrive il metodo di Kansa. Per applicare i risultati dell'interpolazione generalizzata di Hermite che assicurano la non singolarità della matrice di collocazione, proponiamo la seguente espansione per la funzione sconosciuta u:

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

Qui $N_{\mathcal{I}}$ denota il numero di nodi interni ad Ω e $\mathcal{L}^{\boldsymbol{\xi}}$ è l'operatore differenziale usato nell'equazione differenziale (2.1), ma agente su φ vista come funzione del secondo argomento, cioè $\mathcal{L}\varphi$ è uguale a $\mathcal{L}^{\boldsymbol{\xi}}\varphi$ a meno di una differenza di segno. Così i funzionali lineari λ in (1.2) sono dati da

$$\lambda_j = \begin{cases} \delta_{\boldsymbol{\xi}_j} \circ \mathcal{L}, & j = 1, \dots, N_{\mathcal{I}}, \\ \delta_{\boldsymbol{\xi}_j}, & j = N_{\mathcal{I}} + 1, \dots, N. \end{cases}$$

Dopo aver forzato le condizioni di collocazione

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

concludiamo che la matrice di collocazione è della forma

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

Qui i quattro blocchi sono generati nel modo seguente:

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

Notare che abbiamo identificato i due insiemi $\mathcal{X} = \mathcal{I} \cup \mathcal{B}$ dei punti di collocazione e Ξ dei centri.

La matrice (2.6) è dello stesso tipo delle matrici di interpolazione generalizzata di Hermite (1.3) e perciò, per quanto spiegato nel capitolo 1, è non singolare.

Così, vedendo $\hat{u}$ espansa come in (2.5), il metodo di collocazione è ben posto. Un altro punto a favore dell'approccio basato sul metodo di Hermite è che la matrice (2.6) è simmetrica, contrariamente alla matrice del metodo non simmetrico (2.4). Questa proprietà è un punto di forza di questo metodo quando sarà da implementare. Notiamo anche che A è composta da quattro blocchi, ma ancora di dimensione $N \times N$, come la matrice di collocazione non-simmetrica (2.4) del metodo di Kansa.

Comunque la matrice di collocazione simmetrica è più complicata da assemblare perché richiede l'uso di funzioni di base più regolari del metodo non-simmetrico di Kansa, e perciò non si presta molto bene alla soluzione di problemi non lineari.

2.3 Stime dell'errore per la collocazione simmetrica

Un'analisi di convergenza per il metodo di collocazione simmetrica è stata data nel 1998 da Franke e Shaback in [5] e [4]. Le stime dell'errore proposte in questi documenti richiedono che la soluzione della PDE sia molto regolare. Perciò si dovrebbe essere in grado di usare le tecniche *meshfree* di collocazione di funzioni radiali di base specialmente per problemi PDE in alte dimensioni con una soluzione regolare, possibilmente su domini irregolari. A causa del controesempio del metodo non-simmetrico, prima mostrato, deve ancora essere fornita una analisi di convergenza per quel metodo. Tuttavia, per una versione adattativa del metodo non-simmetrico è stata data recentemente una analisi di convergenza.

In [9, pag. 304] si trova il seguente risultato di convergenza:

Teorema 2.1. *Sia $\Omega \subseteq \mathbb{R}^s$ una regione poligonale e aperta. Sia $\mathcal{L} \neq 0$ un operatore differenziale ellittico lineare con coefficienti in $C^{k-2}(\bar{\Omega})$ tale che o siano nulli su $(\bar{\Omega})$ o che non abbiano alcuno zero lì. Supponiamo che $\Phi \in C^{2k}(\mathbb{R}^s)$ sia una funzione strettamente definita positiva. Supponiamo inoltre che il problema con valori al bordo*

$$\begin{aligned}
\mathcal{L}u &= f, && \text{su } \Omega, \\
u &= g, && \text{su } \partial\Omega,
\end{aligned}$$

abbia un'unica soluzione $u \in \mathcal{N}_\Phi(\Omega)$ *per una data* $f \in C(\Omega)$ *e* $g \in C(\partial\Omega)$. *Sia* $\hat{u}$ *la soluzione approssimata ottenuta tramite collocazione della forma* (2.5) *basata su* $\Phi = \varphi(\|\cdot\|)$. *Allora*

$$\|u - \hat{u}\|_{L_\infty(\Omega)} \leq ch^{k-2}\|u\|_{\mathcal{N}_\Phi(\Omega)}$$

per tutti gli h *sufficientemente piccoli, dove* h *è la più grande fill distance sia all'interno del dominio che sul bordo di* Ω.

La dimostrazione, qui omessa, si basa sullo *splitting theorem* che permette di dividere l'errore in un errore di bordo e in un errore interno al dominio. Come conseguenza della dimostrazione, Wendland suggerisce che i punti di collocazione e i centri siano scelti tali che la fill distance sul bordo sia più piccola di quella interna poiché gli ordini di approssimazione differiscono di un fattore ℓ (per operatori differenziali di ordine ℓ). Più precisamente, viene suggerito di distribuire i punti in maniera tale che

$$h_{\mathcal{I},\Omega}^{k-\ell} \approx h_{\mathcal{B},\partial\Omega}^{k}.$$

2.4 Altri problemi

Poiché i metodi descritti sopra furono originariamente usati con funzioni radiali di base a supporto globale, gli stessi si riferiscono alla stabilità e all'efficienza numerica, degli argomenti relativi, per i problemi di interpolazione.

Nel 2002, Fedoseyev, assieme ad altri autori, propose, in [3], che i punti di collocazione sul bordo dovrebbero essere in grado di soddisfare la PDE. La motivazione di questa modifica è il fatto, ben conosciuto, che sia nell'interpolazione sia nella collocazione con funzioni radiali di base, l'errore è maggiore al bordo. Per evitare che la matrice di collocazione diventi banalmente singolare (usando colonne duplicate, cioè funzioni di base) è stato suggerito che i centri corrispondenti sarebbero dovuti essere localizzati esternamente al dominio Ω (creando così delle funzioni di base addizionali). In molti esperimenti numerici, questa strategia ha mostrato di migliorare l'accuratezza del metodo di Kansa non-simmetrico: sarà proprio questo l'approccio implementato nelle simulazioni dei capitoli successivi. Comunque, dovrebbe essere notato che ci sono altre due motivazioni teoriche per questa modifica sia per il metodo non-simmetrico sia per quello simmetrico.

Larsson e Fornberg in [8] hanno confrontato il metodo di collocazione di Kansa originario, la modifica appena descritta e l'approccio simmetrico basato sul metodo di Hermite. Usando le funzioni Multiquadratiche di base nell'implementazione standard, conclusero che il metodo simmetrico è il più accurato, seguito dal metodo non-simmetrico con collocazione al bordo. La ragione sta nel miglior condizionamento del sistema simmetrico. Discussero anche di una implementazione dei metodi appena descritti usando il metodo di integrazione complessa di Contour-Padè: con questa tecnica, i problemi di stabilità sono stati superati e viene mostrato che sia il metodo simmetrico che quello non-simmetrico si comportano con un'accuratezza confrontabile. La collocazione di bordo delle PDE porta ad un miglioramento solo se queste condizioni sono usate come equazioni addizionali, cioè incrementino la grandezza del problema. Dovrebbe anche essere notato che spesso i risultati più accurati sono stati raggiunti con valori del parametro di forma ε che avrebbero potuto portare a un grande

malcondizionamento, usando una implementazione standard, e perciò questi risultati possono solo essere ottenuti usando il metodo di integrazione complessa. Inoltre, sempre in [8], sono state usate delle funzioni radiali di base che dovevano soddisfare un grado di accuratezza molto superiore a quello standard delle differenze finite del secondo ordine, o al metodo di decomposizione spettrale di Fourier-Chebyshev.

3 Alcune applicazioni

In questo e nei prossimi due capitoli presentiamo alcune implementazioni MATLAB per problemi standard di Laplace/Poisson, problemi con coefficienti variabili e problemi con condizioni definite al bordo miste e a tratti.

3.1 Collocazione non-simmetrica

In questo paragrafo discuteremo il metodo non-simmetrico di Kansa. Verrà proposta una presentazione dettagliata poiché il codice MATLAB cambia significativamente da un problema all'altro.

Ci restringeremo ai problemi 2-dimensionali ellittici la cui soluzione analitica è nota e perciò facilmente verificabile. Ci riferiremo ai punti $\boldsymbol{x}$ in $\mathbb{R}^2$ come (x, y).

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

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

dove $\Gamma_1 = \{(x,y) : 0 \le x \le 1, y = 0\}$ e $\Gamma_2 = \partial\Omega \setminus \Gamma_1$. La soluzione esatta è data da

$$u(x,y) = \sin(\pi x)\cos\left(\frac{\pi y}{2}\right).$$

Il Programma 3.1 presenta un codice MATLAB per trovare la soluzione del Problema 3.1 usando un metodo di collocazione non-simmetrico e Multiquadratiche inverse. In questo programma non verrà solo richiesta la definizione della funzione di base ma anche del suo Laplaciano per questo problema test. Notiamo che quando definiamo il termine noto del problema (`rhs`), invece di spezzare la condizione di bordo nei due pezzi dati dalla definizione del problema, valuteremo semplicemente la soluzione nota sul bordo: questo non sarà certamente possibile nel caso generale in cui la soluzione non sia nota. In quel caso si userà una nuova variabile `rhs`, leggermente più complicata:

```
rhs = [Lu(intdata(:,1),intdata(:,2)); ...
    sin(pi*bdydata(1:sn-1,1);zeros(3*(sn-1),1))];
```

Il programma proposto carica nella variabile `dsites` i punti di collocazione (interni) da un file già disponibile: `Data2D_289u`, dove `289` sta per il numero di punti voluti e `u` per la loro distribuzione in modo uniforme. È possibile caricare altri data sites

di punti, detti di Halton e di Chebyshev, indicati rispettivamente con `h` e `c`, in dimensioni spaziali diversi. Come sempre, i centri delle funzioni di base associati ai punti interni sono identici ai punti di collocazione (i.e. ai `dsites`).

Comunque, come espresso nel capitolo precedente, richiediamo ora che i punti di collocazione e i centri soddisfino le condizioni al bordo. Ci sono diversi approcci per raggiungere lo scopo:

- Potremmo usare quei punti di collocazione, letti dal file, che stanno sul bordo come punti di collocazione di bordo (e centri). Questo significherebbe identificare quei punti nell'array `dsites`. Questo approccio sarebbe il più vicino possibile allo spirito della teoria discussa nel capitolo precedente. In MATLAB uno potrebbe facilmente codificare questo con i comandi

  ```
  indx = find(dsites(:,1)==0 | dsites(:,1)==1 | ...
              distes(:,2)==0 | dsites(:,2)==1);
  bdydata = dsites(indx,:);
  intdata = dsites(setdiff([1:N],indx),:);
  bdyctrs = bdydata;
  ```

  In ogni caso in queste note non seguiremo questo approccio.

- Possiamo creare punti di collocazione addizionali per le condizioni al bordo. Questi punti possono stare ovunque sul bordo. Li prenderemo equispaziati. Notare che ordiniamo il bordo in senso antiorario partendo dall'origine. Ora possiamo prendere i centri sul bordo in diverse maniere:
  - Possiamo far coincidere i centri di bordo con i punti di collocazione al bordo. Comunque, questo approccio porterà ad una matrice di collocazione singolare per una scelta di punti interni uniformi (poiché quelli già impostati contengono punti sul bordo e e perciò verrebbero create delle colonne duplicate). Notiamo, comunque, che questo approccio lavora bene se noi prendessimo come punti di collocazione interna i punti di Halton (poiché non starebbero sul bordo del dominio scelto). Questo approccio può essere realizzato imponendo

    ```
    bdyctrs = bdydata;
    ```

  - Possiamo creare dei centri di bordo *fuori* dal dominio. Seguiremo questo approccio in questo contesto poiché sembra fornire una soluzione più accurata. Mettere centri di bordo esternamente al bordo è stato raccomandato recentemente da un numero crescente di autori. Notiamo che questo approccio ci porta nel reame dei metodi RBF per i quali i centri differiscono dai data sites (o punti di collocazione), e abbiamo già affermato che la teoria non è molto sviluppata riguardo l'invertibilità della matrice e gli errori al bordo di questo ultimo caso: è un problema aperto trovare la miglior locazione per i centri di bordo. Li prenderemo ad una piccola distanza perpendicolare ai punti di collocazione di bordo (Figura 3.1).

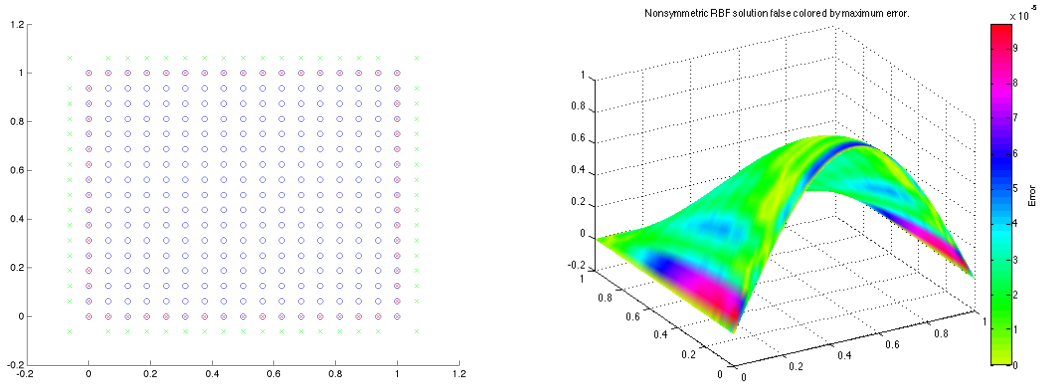


Figura 3.1: A sinistra: Punti di collocazione (interni cerchiati blu, al bordo crociati in rosso) e centri (interni cerchiati in blu, al bordo crociati in verde). A destra: soluzione ottenuta tramite collocazione non-simmetrica usando IMQs con $\varepsilon = 3$ e $N = 289$ punti interni.

Programma 3.1: Implementazione del metodo di Kansa per risolvere un'equazione di Laplace in 2D. Il programma `KansaLaplace_2D.m` è stato tratto da [1, pag. 355].

```
clear all
close all
clc
% 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);
% 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 = 'u';
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)];
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)
% 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
print -dpng ../figure/pb1externalcollocationpoints;
fview = [-30,30];  % viewing angles for plot
caption = ['Nonsymmetric RBF solution '...
    'false colored by maximum error.'];
PlotSurf(xe,ye,Pf,neval,exact,maxerr,fview,caption);
% set(gca,'FontSize',18)
caption = 'Maximum error for nonsymmetric RBF solution.';
% print -dpng ../figure/pb1solution289u;
PlotError2D(xe,ye,Pf,exact,maxerr,neval,fview,caption)
```

Nelle Tabelle 3.1 e 3.2 riportiamo il `RMS-error` e il numero di condizionamento per la soluzione della PDE (problema 3.1) ottenuta con metodo di collocazione non-simmetrico. Nella Tabella 3.1 e nella parte destra della Tabella 3.2 presentiamo alcuni risultati per il metodo di collocazione con le RBF Multiquadriche Inverse usando un parametro di forma $\varepsilon = 3$. Nella Tabella 3.1 i punti interni sono punti di Halton spaziati non regolarmente, mentre nella Tabella 3.2 usiamo punti interni uniformemente spaziati. I centri di bordo sono stati presi esternamente al dominio per i risultati della Tabella 3.2. Nella Tabella 3.1 confrontiamo l'effetto di porre i centri di bordo direttamente sul bordo (coincidenti con i punti di collocazione di bordo) con il prenderli fuori dal dominio come in Figura 3.1.

La parte sinistra della Tabella 3.1 confronta l'uso di Gaussiane (con lo stesso parametro di forma $\varepsilon = 3$) con le Multiquadriche Inverse. Per utilizzare le Gaussiane, basta ridefinire il parametro `rbf` nel Programma 3.1

```
rbf = @(e,r) exp(-(e*r).^2); ep = 3;
Lrbf = @(e,r) 4*e^2*exp(-(e*r).^2).*((e*r).^2-1);
```

Molte osservazioni possono essere fatte guardando alle Tabelle 3.1 e 3.2. L'uso dei

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

	Centri sul bordo		Centri esterni	
N (interni)	RMS-error	cond(A)	RMS-error	cond(A)
9	5.642192e-002	5.276474e+002	6.029293e-002	4.399608e+002
25	1.039322e-002	3.418858e+003	4.187975e-003	2.259698e+003
81	2.386062e-003	1.726995e+006	4.895870e-004	3.650369e+005
289	4.904715e-005	1.706884e+010	2.668524e-005	5.328110e+009
1089	3.676576e-008	1.446865e+018	1.946954e-008	5.015917e+017

Tabella 3.2: Soluzione con collocazione non-simmetrica del Problema 3.1 con Gaussiane e IMQs, $\varepsilon = 3$ e 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-001	1.258837e+003	1.526456e-001	2.794516e+002
5×5	7.199931e-003	4.136193e+003	6.096534e-003	2.409431e+003
9×9	1.947108e-004	2.529708e+010	8.071271e-004	8.771630e+005
17×17	4.174290e-008	5.335000e+019	3.219110e-005	5.981238e+010
33×33	1.408750e-005	7.106505e+020	1.552047e-007	1.706638e+020

punti di Halton invece dei punti uniformi sembra apportare dei benefici poiché gli errori e i numeri di condizionamento sono più piccoli (basta confrontare la parte destra della Tabella 3.1 con la parte destra della Tabella 3.2). Porre i centri di bordo fuori dal dominio sembra essere vantaggioso poiché, ancora una volta, gli errori e i numeri di condizionamento decrescono (vedi Tabella 3.1). Inoltre, l'ultima riga della Tabella 3.2 sembra indicare che le Gaussiane sono più propense ad essere mal-condizionate rispetto alle Multiquadriche. Infine salta subito all'occhio che il `RMS-error` per le Gaussiane con 1089 punti interni uniformi e centri di bordo esterni al dominio è peggiore rispetto al caso di 289 punti.

Certamente, queste sono osservazioni superficiali basate solo su pochi esperimenti numerici. Per molti di questi non esiste alcuna fondazione teorica, e ci sarebbe bisogno di molti altri esperimenti per fare delle affermazioni conclusive (per esempio non è stato fatto alcun tentativo per trovare la miglior approssimante, cioè trovare il valore del parametro di forma, ε, ottimale. Inoltre Kansa e Carlson nel 1992 hanno anche proposto di provare con diversi parametri di forma per l'interno del dominio e per il bordo.

I punti di collocazione e i centri usati qui (e nella maggior parte degli esempi successivi) sono mostrati nel plot di sinistra della Figura 3.1, mentre il plot di destra contiene una soluzione per $N = 289$ punti interni di Halton corrispondenti alla riga 4 nella parte di destra della Tabella 3.2.

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

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

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

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

È facile vedere che la soluzione esatta del problema è data da

$$u(x,y)=16x(1-x)y(1-y).$$

Il codice MATLAB corrispondente è proposto nel Programma 3.2. La sezione di definizione del problema di questo codice è più lunga del codice precedente perché dobbiamo lavorare con la derivata parziale prima e seconda della funzione di base scelta. Inoltre, vengono richiesti anche i coefficienti a e b, con le loro derivate parziali.

Mentre a maggior parte delle linee del codice sono identiche a quello precedente, l'assemblaggio della matrice di collocazione è più complicato poiché dobbiamo applicare l'operatore differenziale alle funzioni di base (notiamo che il calcolo di `LCM`, che corrisponde al blocco $\tilde{A}_{\mathcal{L}}$ nella nostra precedente discussione (2.4).

Programma 3.2: Implementazione del metodo di Kansa per risolvere un'equazione ellittica a coefficienti variabili in 2D. Il programma `KansaEllipticVC_2D.m` è stato tratto da [1, pag. 358].

```
% 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)
```

```
% 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 = ['Nonsymmetric RBF solution '...
    'false colored by maximum error.'];
PlotSurf(xe,ye,Pf,neval,exact,maxerr,fview,caption);
% print -dpng ../figure/pb2solution289h;
caption = 'Maximum error for nonsymmetric RBF solution.';
PlotError2D(xe,ye,Pf,exact,maxerr,neval,fview,caption);
% print -dpng ../figure/pb2error289h;
cond(CM)
```

Nella Tabella 3.3 confrontiamo la soluzione ottenuta con le Gaussiane e con le Multiquadriche Inverse basandoci su punti interni di Halton. I centri di bordo sono presi esternamente al dominio come in Figura 3.1 a sinistra. Di nuovo, la soluzione con Multiquadriche Inverse è leggermente meglio condizionata. Per confrontarla con la soluzione ottenuta usando Gaussiane basta sovrascrivere il seguente codice:

```
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);
```

La Figura 3.2 a sinistra contiene la soluzione approssimata e l'errore massimo per la soluzione ricavata tramite Inverse Multiquadriche su $N = 289$ punti interni e 64 punti di bordo.

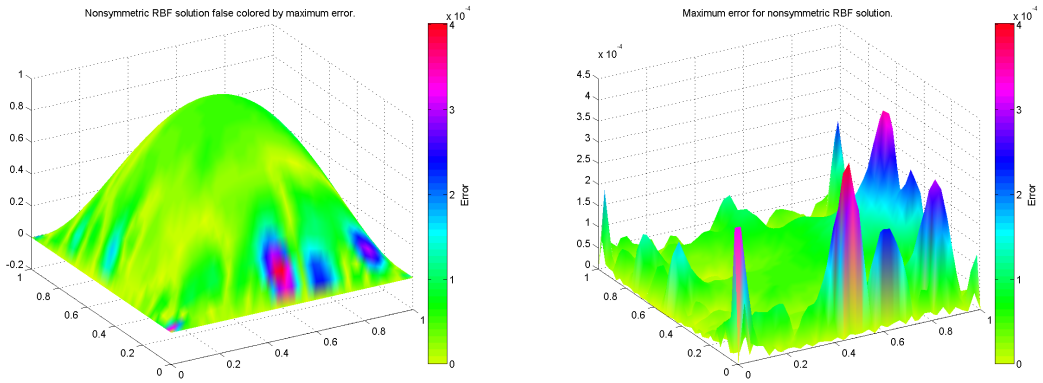


Figura 3.2: A sinistra: soluzione ottenuta tramite collocazione non-simmetrica. A destra: plot dell'errore. Risultati ottenuti usando IMQs con $\varepsilon = 3$ e $N = 289$ punti interni di Halton.

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

Tabella 3.3: Soluzione con collocazione non-simmetrica del Problema 3.1 con Gaussiane e IMQs, $\varepsilon = 3$ e punti interni uniformi e centri di bordo esterni al dominio.

	Gaussiane		IMQ	
N (interni)	RMS-error	cond(A)	RMS-error	cond(A)
9	6.852103e-002	8.874341e+003	1.123770e-001	6.954910e+002
25	1.091888e-002	4.898291e+003	1.123575e-002	3.302471e+003
81	1.854386e-004	1.286993e+009	1.370992e-003	4.992219e+005
289	8.445637e-007	7.031011e+019	8.105109e-005	7.527456e+009
1089	2.559824e-005	4.553162e+020	7.041415e-008	7.785955e+017

al bordo:

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

dove

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

Per questo problema la soluzione esatta è data da

$$u(x,y) = 1 - 0.9x^3.$$

Notiamo che la derivata normale sul lato Γ_1 e Γ_3 è data da $\frac{\partial}{\partial y}$ e $-\frac{\partial}{\partial y}$. Perciò nel codice MATLAB richiederemo che la derivata parziale rispetto a y della funzione di base in aggiunta alla sua definizione e al suo Laplaciano. Di nuovo, la principale differenza nel codice è nell'assemblaggio della matrice di collocazione. Notiamo che questa volta abbiamo bisogno di trattare attentamente le condizioni al bordo e il termine noto. È importante che l'orientamento del punti di bordo sia consistente.

Programma 3.3: Implementazione del metodo di Kansa per risolvere un'equazione di Laplace in 2D con condizioni di bordo miste. Il programma `KansaLaplaceMixedBC_2D.m` è stato tratto da [1, pag. 361].

```
% 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);
  % 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)
  % 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 = ['Nonsymmetric RBF solution '...
    'false colored by maximum error.'];
PlotSurf(xe,ye,Pf,neval,exact,maxerr,fview,caption);
print -dpng ../figure/pb3solution289h;
caption = 'Maximum error for nonsymmetric RBF solution.';
PlotError2D(xe,ye,Pf,exact,maxerr,neval,fview,caption)
print -dpng ../figure/pb3error289h;
```

Nella Tabella 3.4 confrontiamo di nuovo l'uso delle Gaussiane e delle Multiquadriche Inverse su un insieme di $N = 9, 25, 81, 289$ e 1089 punti interni di Halton (con centri di bordo addizionali presi esternamente al dominio). Come negli esperimenti precedenti, la soluzione Gaussiana è leggermente peggiore in termini di stabilità per lo stesso valore del parametro di forma $\varepsilon = 3$.

Tabella 3.4: Soluzione con collocazione non-simmetrica del Problema 3.1 con Gaussiane e IMQs, $\varepsilon = 3$ e punti interni uniformi e centri di bordo esterni al dominio.

	Gaussiane		IMQ	
N (interni)	RMS-error	cond(A)	RMS-error	cond(A)
9	3.423330e-001	5.430073e+003	7.937403e-002	2.782348e+002
25	1.065826e-002	1.605086e+003	5.605445e-003	1.680888e+003
81	5.382387e-004	3.684159e+008	1.487160e-003	2.611650e+005
289	6.181855e-006	1.452124e+019	1.822077e-004	3.775455e+009
1089	2.060470e-006	1.628262e+021	1.822221e-007	3.155751e+017

Nella Figura 3.3, mostriamo la soluzione ottenuta per collocazione con le Multiquadriche Inverse per $N = 289$ punti interni con il suo errore massimo. Notiamo che (sebbene il problema abbia una soluzione simmetrica), la soluzione approssimata non è abbastanza simmetrica (come dimostrato dal plot dell'errore) ed è lo stesso risultato che si può concludere dalla Figura 3.2).

Recentemente, alcuni autori, come Li, hanno riportato che la soluzione ottenuta tramite collocazione non-simmetrica per questo problema con le RBF Multiquadriche Inverse è molto più accurata, in magnitudine, rispetto ad una soluzione ottenuta con elementi finiti lineari a tratti usando lo stesso numero di nodi.

3.2 Collocazione simmetrica

In questo paragrafo discuteremo il metodo implementativo simmetrico basato sull'interpolazione di Hermite con il corrispondente codice MATLAB. Come fatto nel precedente paragrafo, ci ristringeremo ai problemi 2-dimensionali ellittici la cui soluzione analitica è nota per una facilità di verifica dei risultati numerici trovati. Ci riferiremo ai punti $\boldsymbol{x}$ in $\mathbb{R}^2$ come (x, y).

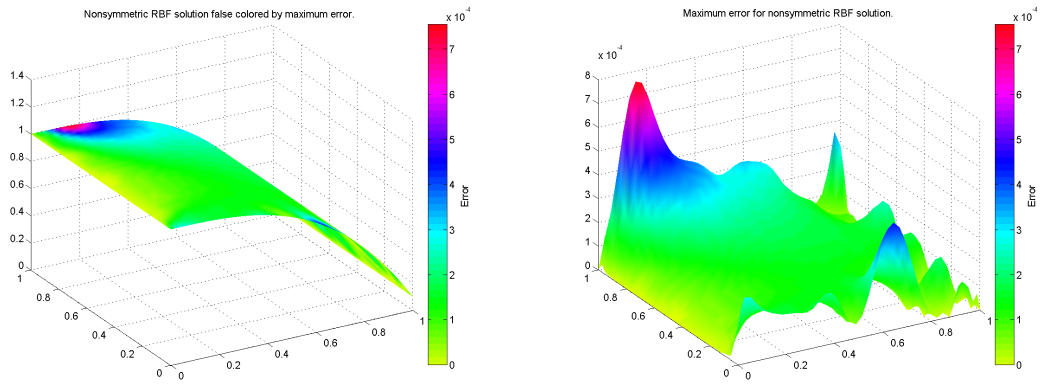


Figura 3.3: A sinistra: soluzione ottenuta tramite collocazione non-simmetrica. A destra: plot dell'errore. Risultati ottenuti usando IMQs con $\varepsilon = 3$ e $N = 289$ punti interni di Halton.

Per i problemi con Laplaciano, adesso ci vuole l'operatore differenziale

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

dove la semplificazione dell'ultima linea viene giustificata dal fatto che lavoriamo con ordini di derivate pari e funzioni della forma $\varphi = \varphi(\|\boldsymbol{x} - \boldsymbol{\xi}\|)$. Per esempio, con $r = \|\boldsymbol{x} - \boldsymbol{\xi}\|$, otteniamo per diverse RBF in $\mathbb{R}^2$:

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

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

$$\Delta_{\boldsymbol{\xi}}\Delta\sqrt{1+(\varepsilon r)^2} = \frac{\varepsilon^4\left((\varepsilon r)^4 + 8(\varepsilon r)^2 - 8\right)}{(1+(\varepsilon r)^2)^{9/2}}, \text{ MQ,} \tag{3.3}$$

Problema 3.4. Usiamo la stessa equazione alle derivate parziali e condizione al bordo che nel problema 3.1. Il Programma 3.4 implementato in MATLAB la collocazione simmetrica di Hermite. Si nota che questo programma è un po' più complicato che quello corrispondente al metodo non-simetrico (c.f. Programma 3.1). Adesso la matrice di valutazione `EM` è fatta di due pezzi (come la matrice di collocazione per il caso non-simmetrico, si vedano le linee 34-39), di conseguenza la matrice di collocazione risulta essere formata da quattro pezzi (c.f. linee 41-49). Inoltre, è necessario calcolare uno dei Laplaciani iterati del tipo calcolato in (3.1)-(3.3).

Programma 3.4: Implementazione del metodo simmetrico per risolvere un'equazione di Laplace in 2D. Il programma `HermiteLaplace_2D.m` è stato tratto da [1, pag. 366].

```
% 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);
  % 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 = 'u';
  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)];
  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);
```

```
57  rms_err = norm(Pf-exact)/neval;
58  fprintf('RMS error:     %e\n', rms_err)
59  fprintf('Maximum error: %e\n', maxerr)
60  % Plot collocation points and centers
61  hold on; plot(intdata(:,1),intdata(:,2),'bo');
62  plot(bdydata(:,1),bdydata(:,2),'rx');
63  plot(bdyctrs(:,1),bdyctrs(:,2),'gx'); hold off
64  fview = [-30,30];  % viewing angles for plot
65  caption = ['Symmetric RBF solution '...
66      'false colored by maximum error.'];
67  PlotSurf(xe,ye,Pf,neval,exact,maxerr,fview,caption);
68  caption = 'Maximum error for symmetric RBF solution.';
69  PlotError2D(xe,ye,Pf,exact,maxerr,neval,fview,caption)
```

Come prima, trattiamo la frontiera permettendo l'uso di punti di collocazione e centri diversi lungo la frontiera. Da questo risulta che la matrice di collocazione non è simmetrica, e quindi, non vale più la fondazione teorica vista nella Sezione 2.2. Per usare una matrice simmetrica (e quindi invertibile), bisogna sostituire le linee 24-27 con

```
bdyctrs = bdydata;
```

In questo caso, è anche possibile semplificare l'assemblaggio della matrice di collocazione. Si può togliere le linee 45-46 e mettere al posto della linea 49

```
CM = [LLCM LBCM; LBCM' BBCM];
```

Si nota che, contrariamente all'approccio non-simmetrico di Kansa, quello funziona sia per punti interni uniformi che non-uniformi.

Gli stessi esperimenti che per il metodo non-simmetrico di Kansa sono esposti nelle tabelle 3.5 e 3.6 per il metodo di Hermite.

Tabella 3.5: Soluzione con collocazione simmetrica del Problema 3.4 con IMQs, $\varepsilon = 3$ e punti interni di Halton.

	Centri sul bordo		Centri esterni	
N (interni)	RMS-error	cond(A)	RMS-error	cond(A)
9	1.869505e-001	9.055720e+003	2.438041e-001	3.549895e+004
25	7.698471e-002	8.506782e+004	9.429580e-002	1.162027e+005
81	4.839682e-003	1.338599e+007	5.070833e-003	1.017388e+007
289	4.480250e-005	9.991615e+010	3.448546e-005	7.180249e+010
1089	2.481407e-008	2.820823e+018	1.907000e-008	2.262777e+018

Si nota che, come nel caso della collocazione non-simmetrica, le multiquadratiche inverse con punti interni di Halton e centri alla frontiera fuori del dominio sembrano essere una scelta un po' meglio che le altre (i.e., Gaussiane, punti interni uniformi o centri alla frontiera sulla frontiera).

E comunque importante notare come la differenza di performance tra gli approcci simmetrici e non-simmetrici è piccola. Questo si può concludere comparando le tabelle

Tabella 3.6: Soluzione con collocazione simmetrica del Problema 3.4 con Gaussiane e IMQs, $\varepsilon = 3$ e punti interni uniformi centri alla frontiera fuori del dominio.

	Gaussiane		IMQ	
N (interni)	RMS-error	cond(A)	RMS-error	cond(A)
9	4.088188e-001	1.196486e+005	2.806897e-001	3.105155e+004
25	7.704584e-003	1.359899e+005	1.583948e-001	1.216534e+005
81	2.272289e-004	2.453107e+010	8.650782e-004	2.016503e+007
289	5.271776e-008	4.338406e+021	3.962654e-005	6.051588e+011
1089	5.805757e-007	1.438258e+022	1.870210e-007	2.324115e+020

del problema 3.1 con quelle del problema 3.4. Questo si vede anche nella figura 3.4 che rappresenta l'errore dei due metodi usando gli stessi parametri.

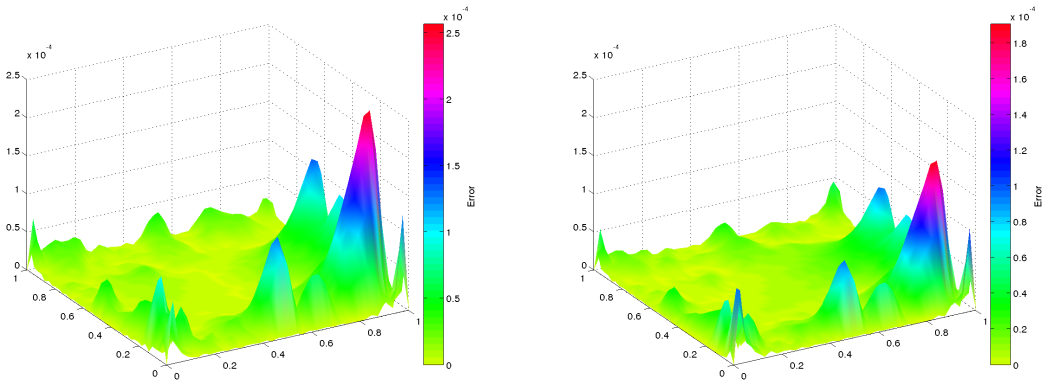


Figura 3.4: Grafici dell'errore delle soluzioni con la collocazione non-simmetrica di Kansa (a sinistra) e simmetrica di Hermite (a destra) usando IMQs , $\varepsilon = 3$, $N = 289$ punti interni uniformi e 64 centri alla frontiera fuori del dominio.

Questo esempio di problema mostra una convergenza molto veloce quando si usano Multiquadratiche Inverse lisce su un problema che ha una soluzione liscia come previsto nell'introduzione per il Teorema 2.1.

Problema 3.5. Un'implementazione MATLAB del problema 3.2 con coefficienti variabili, anche se teoricamente possibile, sarebbe troppo pesante ed ingombrante. Per esempio, siccome l'operatore differenziale $\mathcal{L}$ è dato per

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

l'espansione basica della soluzione è

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

dove

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

Questo implica che i componenti del blocco $\hat{A}_{\mathcal{L}\mathcal{L}^{\boldsymbol{\xi}}}$ della matrice simmetrica di collocazione vengono computati con l'operatore differenziale

$$\begin{aligned}
\mathcal{L}\mathcal{L}^{\boldsymbol{\xi}} =& \left[\frac{\partial}{\partial x}\left(a(x,y)\frac{\partial}{\partial x}\right) + \frac{\partial}{\partial y}\left(b(x,y)\frac{\partial}{\partial y}\right)\right]\left[\frac{\partial}{\partial \xi}\left(a(\xi,\eta)\frac{\partial}{\partial \xi}\right) + \frac{\partial}{\partial \eta}\left(b(\xi,\eta)\frac{\partial}{\partial \eta}\right)\right] \\
=& a(x,y)a(\xi,\eta)\frac{\partial^4}{\partial x^4} + (a(x,y)b(\xi,\eta) + b(x,y)a(\xi,\eta))\frac{\partial^4}{\partial x^2 \partial y^2} \\
&+ b(x,y)b(\xi,\eta)\frac{\partial^4}{\partial x^4} + \left(\frac{\partial a(x,y)}{\partial x}a(\xi,\eta) - a(x,y)\frac{\partial a(\xi,\eta)}{\partial \xi}\right)\frac{\partial^3}{\partial x^3} \\
&+ \left(\frac{\partial b(x,y)}{\partial y}a(\xi,\eta) - a(x,y)\frac{\partial b(\xi,\eta)}{\partial \eta}\right)\frac{\partial^3}{\partial x^2 \partial y} \\
&+ \left(\frac{\partial a(x,y)}{\partial x}b(\xi,\eta) - b(x,y)\frac{\partial a(\xi,\eta)}{\partial \xi}\right)\frac{\partial^3}{\partial x \partial y^2} \\
&+ \left(\frac{\partial b(x,y)}{\partial y}b(\xi,\eta) - b(x,y)\frac{\partial b(\xi,\eta)}{\partial \eta}\right)\frac{\partial^3}{\partial y^3} + \frac{\partial a(x,y)}{\partial x}\frac{\partial a(\xi,\eta)}{\partial \xi}\frac{\partial^2}{\partial x^2} \\
&- \left(\frac{\partial a(x,y)}{\partial x}\frac{\partial b(\xi,\eta)}{\partial \eta} + \frac{\partial b(x,y)}{\partial y}\frac{\partial a(\xi,\eta)}{\partial \xi}\right)\frac{\partial^2}{\partial x \partial y} + \frac{\partial b(x,y)}{\partial y}\frac{\partial b(\xi,\eta)}{\partial \eta}\frac{\partial^2}{\partial y^2}.
\end{aligned}$$

Abbiamo espresso le derivate rispetto alla seconda variabile $\boldsymbol{\xi} = (\xi,\eta)$ della funzione di base in termini di quelle rispetto alla prima variabile $\boldsymbol{x} = (x,y)$ avendo in mente che ogni differenziazione introduce un cambio di segno.

Problema 3.6. Invece di ripeter i calcoli del Problema 3.3, presentiamo un problema diverso con condizione al bordo definita a tratti.

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

dove

$$\begin{aligned}
\Gamma_1 &= \{(x,y):\ -1 \le x \le 1,\ y = -1\}, \\
\Gamma_2 &= \{(x,y):\ x = 1,\ -1 \le y \le 1\}, \\
\Gamma_3 &= \{(x,y):\ 0 \le x \le 1,\ y = 1\}, \\
\Gamma_4 &= \{(x,y):\ -1 \le x \le 0,\ y = 1\}, \\
\Gamma_5 &= \{(x,y):\ x = -1,\ 0 \le y \le 1\}.
\end{aligned}$$

Per questo problema, non abbiamo una soluzione esatta disponibile, quindi useremo la soluzione pseudo-spettrale per fare il paragone con la nostra.

Programma 3.5: Implementazione del metodo di collocazione simmetrico per risolvere un'equazione di Laplace con condizioni al bordo miste in 2D e confronto con una soluzione pseudo-spettrale di riferimento. Il programma `HermiteLaplaceMixedBCTref_2D.m` è stato tratto da [1, pag. 355].

```
% 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); ...
```

```
51          0.2*sin(3*pi*bdydata(sn:2*sn-2,2)); zeros((sn-1)/2,1);...
52          sin(pi*bdydata((5*sn-3)/2:3*sn-3,1)).^4; zeros(sn-1,1)];
53  % Compute RBF solution
54  Pf = EM * (CM\rhs);
55  surf(xe,ye,reshape(Pf,neval,neval));
56  view(-20,45),  axis([-1 1 -1 1 -.2 1]);
57  text(0,.8,.5,sprintf('u(0,0) = %12.10f',Pf(841)))
```

La definizione delle condizioni al bordo del programma 3.5 è simile a quella del programma 3.3. Ma questo volta, il dominio è il quadrato $[-1,1]^2$ e non $[0,1]^2$, e quindi ci vogliono alcune rettifiche. Per esempio, i punti di collocazione carichiamo dal file sono trasformati in linea 17. Della stessa maniera, i centri al bordo si mettono fuori di una frontiera diversa (si vedano le linee 24-27).

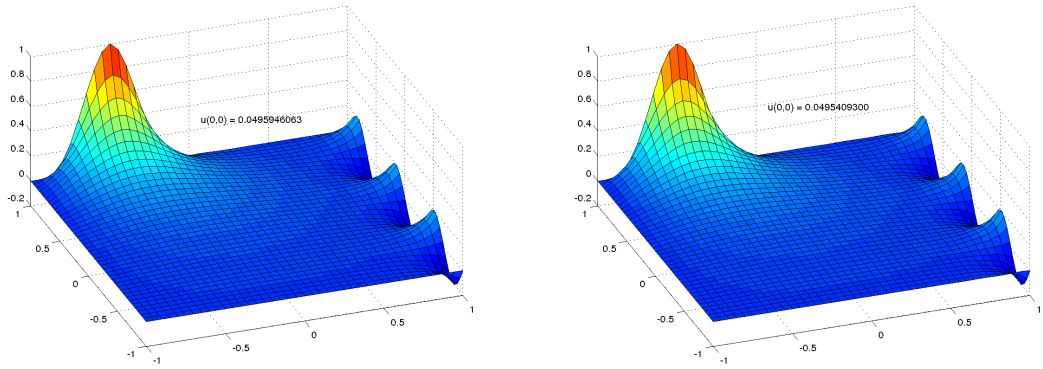


Figura 3.5: Grafici delle soluzioni per il problema 3.6 usando il metodo pseudo-spettrale con 361 punti (a sinistra) e la collocazione simmetrica con IMQs, $\varepsilon = 3$, $N = 289$ punti interni uniformi e 64 punti supplementari fuori del dominio.

Si nota che la qualità dei due soluzioni rappresentate nella figura 3.5 è molto simile. Il metodo pseudo-spettrale usa un totale di 361 punti allora che la soluzione RBF ne usa un totale di 352 (289 interni più 64 al bordo).

Apportiamo ora alcune osservazioni conclusive sui metodi di collocazione simmetrico e non-simmetrico. Tutto sommato, il metodo non-simmetrico (Kansa) sembra di dare dei risultati un po' meglio di quelli del metodo non-simmetrico (Hermite) (si comparino le tabelle 3.1 e 3.2 alle tabelle 3.5 e 3.6. Per un valore del shape parameter ε dato, sia l'errore che il condizionamento sono un po' più piccoli.

Il vantaggio dell'approccio di Hermite su quello di Kansa è che, se tutti i centri coincidono con i punti di collocazione, la matrice di collocazione risultante dal metodo di Hermite è simmetrica. Ne risulta che il numero di computazioni necessari può essere considerevolmente ridotto usando un risolutore per sistemi simmetrici. Siccome il metodo d Kansa necessita meno derivate della funzione di base, è più semplice da implementare e applicabile ai problemi con soluzioni di classi piu deboli. Inoltre, come abbiamo visto nei problemi 3.2 e 3.5, il metodo non-simmetrico è molto più semplice per i problemi con coefficienti non-costanti. Infatti, non è chiaro come usare il metodo simmetrico per problemi non-lineari. Entrambe i metodi descritti vengono implementati e usati in molti applicazioni.

3.3 Collocazione con RBF a supporto compatto

In questo terzo capitolo descriveremo l'implementazione in MATLAB del metodo di collocazione con RBF a supporto compatto per problemi ellittici in dimensione due.

Sebbene originariamente Kansa propose il metodo di collocazione non-simmetrico usando solo le multiquadratiche, tale metodo è più generale e può quindi essere applicato ad ogni tipo di RBF, comprese quelle a supporto compatto. Lo stesso vale ovviamente anche per il metodo simmetrico. In particolare presenteremo il codice MATLAB per il metodo simmetrico e la funzione C^6 di Wendland

$$\varphi_{3,3}(r) = (1-r)_+^8(32r^3 + 25r^2 + 8r + 1).$$

Il suo Laplaciano e le sue derivate bi-armoniche sono, rispettivamente,

$$\begin{aligned}\Delta\varphi_{3,3}(r) &= 44(1-r)_+^6(88r^3 + 3r^2 - 6r - 1),\\ \Delta^2\varphi_{3,3}(r) &= 1056(1-r)_+^4(297r^3 - 212r^2 + 16r + 4).\end{aligned}$$

La prima cosa di cui abbiamo bisogno è un programma che costruisca la matrice delle distanze per la nostra scelta delle RBF di base. Per questo abbiamo usato la routine `DistanceMatrixCSRBF.m` scritta da Fasshauer e riportata qui di seguito. Come è facile vedere dalle prime righe del programma 3.6, tale routine sfrutta la struttura dati *kd*-tree tramite il tool `kdtree` di Guy Shechter, reperibile presso la pagina web http://www.mathworks.com/matlabcentral/fileexchange/4586-k-d-tree.

Programma 3.6: Calcolo della matrice delle distanze per funzioni di base a supporto compatto sfruttando strutture dati *kd*-tree. Programma tratto da [1, pag. 96]

```
% DM = DistanceMatrixCSRBF(dsites,ctrs,ep)
% Forms the distance matrix of two sets of points in R^s
% for compactly supported radial basis functions, i.e.,
%      DM(i,j) = || datasite_i - center_j ||_2.
% The CSRBF used with this code must be given in shifted form
% rbf2(u) = rbf(r), u=1-e*r.
% For example, the Wendland C2
% rbf = @(e,r) max(1-e*r,0).^4.*(4*e*r+1);
% becomes
% rbf2 = @(u) u.^4.*(4*u+5);
% Input
%   dsites: Nxs matrix representing a set of N data sites
%              in R^s (i.e., each row contains one
%              s-dimensional point)
%   ctrs:   Mxs matrix representing a set of M centers for
%              RBFs in R^s (also one center per row)
%   ep:        determines size of support of basis function.
%              Small ep yields wide function,
%              i.e., supportsize = 1/ep
% Output
%   DM:     NxM SPARSE matrix that contains the Euclidean
%              u-distance (u=1-e*r) between the i-th data
%              site and the j-th center in the i,j position
% Uses:     k-D tree package by Guy Shechter from
```

```
25 %                MATLAB Central File Exchange
26   function DM = DistanceMatrixCSRBF(dsites,ctrs,ep)
27   N = size(dsites,1);  M = size(ctrs,1);
28   % Build k-D tree for data sites
29   % For each center (basis function), find the data sites
30   % in its support along with u-distance
31   support = 1/ep;
32   nzmax = 25*N; rowidx = zeros(1,nzmax); colidx = zeros(1,nzmax);
33   validx = zeros(1,nzmax); istart = 1; iend = 0;
34   if M > N  % faster if more centers than data sites
35      [tmp,tmp,Tree] = kdtree(ctrs,[]);
36      for i = 1:N
37         [pts,dist,idx] = kdrangequery(Tree,dsites(i,:),support);
38         newentries = length(idx);
39         iend = iend + newentries;
40         rowidx(istart:iend) = repmat(i,1,newentries);
41         colidx(istart:iend) = idx';
42         validx(istart:iend) = 1-ep*dist';
43         istart = istart + newentries;
44      end
45   else
46      [tmp,tmp,Tree] = kdtree(dsites,[]);
47      for j = 1:M
48         [pts,dist,idx] = kdrangequery(Tree,ctrs(j,:),support);
49         newentries = length(idx);
50         iend = iend + newentries;
51         rowidx(istart:iend) = idx';
52         colidx(istart:iend) = repmat(j,1,newentries);
53         validx(istart:iend) = 1-ep*dist';
54         istart = istart + newentries;
55      end
56   end
57   idx = find(rowidx);
58   DM = sparse(rowidx(idx),colidx(idx),validx(idx),N,M);
59   % Free the k-D Tree from memory.
60   kdtree([],[],Tree);
```

Per prima cosa osserviamo che, per poter sfruttare il precedente codice MATLAB dobbiamo rappresentare le funzioni di base e le loro derivate nella forma shiftata

$$\begin{aligned}\widetilde{\varphi}_{3,3}(r) &= r^8(66 - 154r + 121r^2 - 32r^3),\\ \Delta\widetilde{\varphi}_{3,3}(r) &= 44r^6(84 - 264r + 267r^2 - 88r^3),\\ \Delta^2\widetilde{\varphi}_{3,3}(r) &= 1056r^4(105 - 483r + 679r^2 - 297r^3),\end{aligned}$$

come implementato nelle righe $6-9$ del programma 3.7. Osserviamo poi che, anche se non è necessario introdurre un fattore di scala ε nel codice MATLAB per le funzioni di base, dal momento che le righe $7-9$ del programma 3.6 tengono già conto della dimensione del supporto, ne abbiamo comunque bisogno per le sue derivate per una conseguenza della regola della catena. Infine nel caso a supporto compatto dobbiamo fare particolare attenzione nella disposizione dei centri sul bordo, infatti se per le funzioni a supporto globale abbiamo beneficiato della disposizione di alcuni centri all'esterno del dominio, questo non ha più senso nel contesto delle funzioni a supporto compatto. D'altra parte è anche chiaro che ogni funzione di base il cui supporto ha raggio inferiore della distanza del suo centro dal bordo del dominio non contribuirà

alla soluzione del problema. Useremo quindi dei punti interni di Halton e dei punti di bordo equispaziati sia come centri che come punti di collocazione. Questa scelta è riflessa nelle righe $26 - 27$ del programma 3.7. Per il resto, il programma 3.7 è del tutto identico ai precedenti. Riportiamo comunque l'intero codice per facilitare il lettore.

Programma 3.7: Risoluzione di una PDE ellittica tramite un metodo di collocazione simmetrico. Programma tratto da [1, pag. 376].

```
1  % HermiteLaplace_2D_CSRBF
2  % Script that performs Hermite collocation for 2D Laplace equation
3  % with sparse matrices
4  % Calls on: DistanceMatrixCSRBF
5    % Wendland C6 RBF, its Laplacian and double Laplacian
6    rbf = @(e,r) r.^8.*(66*spones(r)-154*r+121*r.^2-32*r.^3);
7    Lrbf = @(e,r) 44*e^2*r.^6.*(84*spones(r)-264*r+267*r.^2-88*r.^3);
8    L2rbf = @(e,r) 1056*e^4*r.^4.*...
9                   (105*spones(r)-483*r+679*r.^2-297*r.^3);
10   ep = 0.25;
11   % Exact solution and its Laplacian for test problem
12   u = @(x,y) sin(pi*x).*cos(pi*y/2);
13   Lu = @(x,y) -1.25*pi^2*sin(pi*x).*cos(pi*y/2);
14   % Number and type of collocation points
15   N = 289; gridtype = 'h';
16   neval = 40;
17   % Load (interior) collocation points
18   name = sprintf('Data2D_%d%s',N,gridtype); load(name);
19   intdata = dsites;
20   % Additional (equally spaced) boundary collocation points
21   sn = sqrt(N); bdylin = linspace(0,1,sn)';
22   bdy0 = zeros(sn-1,1); bdy1 = ones(sn-1,1);
23   bdydata = [bdylin(1:end-1) bdy0; bdy1 bdylin(1:end-1); ...
24        flipud(bdylin(2:end)) bdy1; bdy0 flipud(bdylin(2:end))];
25   % Let centers coincide with ALL data sites
26   bdyctrs = bdydata;
27   ctrs = [intdata; bdyctrs];
28   % Create neval-by-neval equally spaced evaluation locations
29   % in the unit square
30   grid = linspace(0,1,neval); [xe,ye] = meshgrid(grid);
31   epoints = [xe(:) ye(:)];
32   % Compute evaluation matrix
33   DM_inteval = DistanceMatrixCSRBF(epoints,intdata,ep);
34   DM_bdyeval = DistanceMatrixCSRBF(epoints,bdyctrs,ep);
35   LEM = Lrbf(ep,DM_inteval);
36   BEM = rbf(ep,DM_bdyeval);
37   EM = [LEM BEM];
38   exact = u(epoints(:,1),epoints(:,2));
39   % Compute blocks for collocation matrix
40   DM_IIdata = DistanceMatrixCSRBF(intdata,intdata,ep);
41   DM_IBdata = DistanceMatrixCSRBF(intdata,bdyctrs,ep);
42   DM_BIdata = DistanceMatrixCSRBF(bdydata,intdata,ep);
43   DM_BBdata = DistanceMatrixCSRBF(bdydata,bdyctrs,ep);
44   LLCM = L2rbf(ep,DM_IIdata);
45   LBCM = Lrbf(ep,DM_IBdata);
46   BLCM = Lrbf(ep,DM_BIdata);
```

```
47  BBCM = rbf(ep,DM_BBdata);
48  CM = [LLCM LBCM; BLCM BBCM];
49  % Create right-hand side
50  rhs = [Lu(intdata(:,1),intdata(:,2)); ...
51         sin(pi*bdydata(1:sn-1,1)); zeros(3*(sn-1),1)];
52  % Compute RBF solution
53  Pf = EM * (CM\rhs);
54  % Compute maximum error on evaluation grid
55  maxerr = norm(Pf-exact,inf);
56  rms_err = norm(Pf-exact)/neval;
57  fprintf('RMS error:     %e\n', rms_err)
58  fprintf('Maximum error: %e\n', maxerr)
59  % Plot collocation points and centers
60  hold on; plot(intdata(:,1),intdata(:,2),'bo');
61  plot(bdydata(:,1),bdydata(:,2),'rx');
62  plot(bdyctrs(:,1),bdyctrs(:,2),'gx'); hold off
63  fview = [-30,30];  % viewing angles for plot
64  caption = ['Symmetric CSRBF solution '...
65      'false colored by maximum error.'];
66  PlotSurf(xe,ye,Pf,neval,exact,maxerr,fview,caption);
67  caption = 'Maximum error for symmetric CSRBF solution.';
68  PlotError2D(xe,ye,Pf,exact,maxerr,neval,fview,caption)
```

Se vogliamo rimpiazzare il metodo di collocazione simmetrico con uno non-simmetrico dobbiamo sostituire le linee $33 - 48$ del precedente programma con

```
% Compute evaluation matrix
DM_eval = DistanceMatrixCSRBF(epoints, ctrs, ep);
EM = rbf(ep, DM_eval);
exact = u(epoints(:,1),epoints(:,2));
% Compute blocks for collocation matrix
DM_intdata = DistanceMatrixCSRBF(intdata, ctrs, ep);
DM_bdydata = DistanceMatrixCSRBF(bdydata, ctrs, ep);
LCM = Lrbf(ep,DM_intdata);
BCM = rbf(ep,DM_bdydata);
CM = [LCM; BCM];
```

Esempio 3.1. Consideriamo il problema test dei capitoli precedenti, ma in questo caso compariamo uno schema di approssimazione stazionario con uno schema non-stazionario sia per il metodo di collocazione simmetrico che per quello non-simmetrico. Nel contesto stazionario prendiamo un valore iniziale di $\varepsilon = 0.25$ e cioè una funzione di base con un supporto abbastanza esteso e duplichiamo tale valore per ogni esperimento successivo. Questo perché vogliamo che la larghezza della banda della matrice di collocazione rimanga fissa e di conseguenza il metodo di approssimazione numerico sia più efficiente. In ogni caso, analogamente a quanto succedeva per l'applicazione all'interpolazione di dati scarsi, nel contesto stazionario non ci dobbiamo aspettare convergenza per $h \to 0$, come si può vedere chiaramente nella parte sinistra delle tabelle 3.7 e 3.8. Nel caso specifico del metodo simmetrico le cose vanno particolarmente male: non solo non c'è convergenza ma addirittura l'errore cresce.

Osservando la parte destra delle tabelle 3.7 e 3.8 notiamo che, nel caso non-stazionario il tasso di convergenza è pressapoco lo stesso per entrambi gli approcci. Inoltre le matrici di collocazione sono completamente dense e questo, di fatto, sconsiglia l'uso di funzioni a supporto compatto. È interessante però notare che l'accuratezza

Tabella 3.7: Confronto tra il caso stazionario (ε di partenza pari a 0.25) di collocazione non-simmetrica dell'esempio 3.1 con CSRBFs e il caso non stazionario per N punti interni di Halton, $4(\sqrt{N}-1)$ centri equispaziati sul bordo coincidenti con i relativi punti di collocazione sul bordo.

	Stazionario		Non stazionario	
N punti interni	RMS-error	cond(A)	RMS-error	cond(A)
3×3	6.077025e-003	1.495127e+005	6.077025e-003	1.495127e+005
5×5	2.352498e-003	6.274058e+005	3.810928e-004	2.720833e+007
9×9	1.947271e-003	6.192333e+006	4.430301e-005	2.851716e+010
17×17	1.326745e-003	6.113189e+007	2.200286e-006	2.293235e+013
33×33	5.703309e-003	1.824487e+008	9.986944e-008	8.537419e+015

Tabella 3.8: Confronto tra il caso stazionario (ε di partenza pari a 0.25) di collocazione simmetrica dell'esempio 3.1 con CSRBFs e il caso non stazionario per N punti interni di Halton, $4(\sqrt{N}-1)$ centri equispaziati sul bordo coincidenti con i relativi punti di collocazione sul bordo.

	Stazionario		Non stazionario	
N punti interni	RMS-error	cond(A)	RMS-error	cond(A)
3×3	5.866837e-003	4.448249e+004	5.866837e-003	4.448249e+004
5×5	3.190108e-003	2.454493e+006	4.757992e-004	3.582872e+007
9×9	8.381144e-003	8.693440e+007	3.825086e-005	2.360746e+010
17×17	1.115179e-001	2.162078e+009	2.099321e-006	1.392838e+013
33×33	3.696962e-001	3.164425e+010	8.680882e-008	6.127122e+015

ottenuta usando la funzione C^6 di Wendland è simile a quella raggiunta dalle funzioni a supporto globale gaussiana e multiquadratica inversa usate nell'esempio 3.1, ad indicare che la soluzione della PDE non vive nello spazio nativo delle gaussiane o delle multiquadratiche inverse.

Nel contesto stazionario si verifica una crescita degli errori perché, per mantenere il metodo efficiente le funzioni di base sono scelte troppo locali e di conseguenza non viene tenuto conto delle condizioni al bordo per il calcolo della soluzione. In figura 3.6 si può trovare la soluzione del nostro esempio per $\varepsilon = 2$ (supporto piccolo) e per $\varepsilon = 0.25$ (supporto molto esteso), considerando $N = 33 \times 33 = 289$ punti interni di Halton e 64 punti equispaziati sul bordo. Infine, come indicato in precedenza, i punti di collocazione sono scelti in coincidenza con i centri. Per gli errori di approssimazione e il numero di condizionamento della matrice di collocazione risultante consultare la quarta riga delle tabelle 3.7 e 3.8.

Osserviamo ancora che, nel caso $\varepsilon = 2$, la matrice di collocazione è sparsa e ha

- solo l'11% di componenti non nulle, se si usa il metodo di collocazione simmetrico;
- il 44% se si usa quello non-simmetrico.

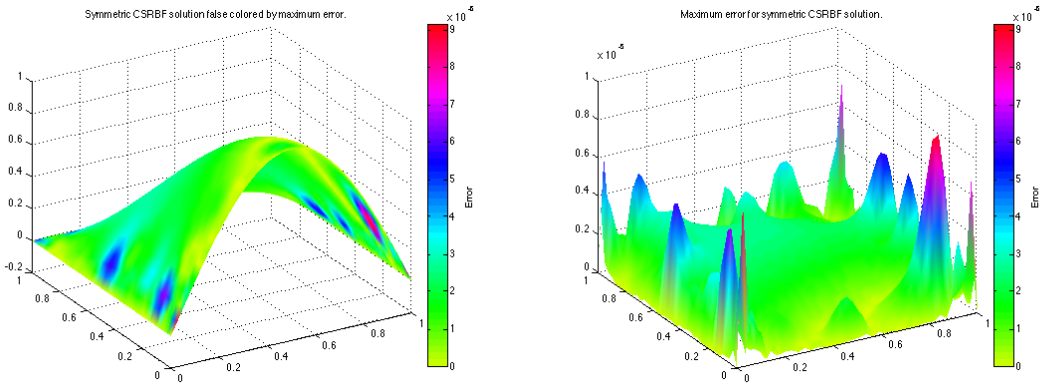


Figura 3.6: A sinistra: soluzione dell'esempio 3.1 con un metodo di collocazione simmetrico usando CSRBFs con $\varepsilon = 2$. A destra: errore rispetto alla soluzione analitica.

La grande differenza tra queste percentuali è dovuta al fatto che gli elementi della matrice simmetrica sono calcolati sfruttando derivate delle funzioni di base di ordine maggiore rispetto a quanto si fa per il caso non-simmetrico e, mentre le derivate di φ teoricamente hanno tutte lo stesso supporto della funzione di partenza φ, numericamente la dimensione del supporto aumenta all'aumentare dell'ordine di differenziazione. Di conseguenza l'approssimazione nel caso stazionario simmetrico è meno accurata perché le funzioni di base e le loro derivate sono troppo locali, quindi non si tiene conto adeguatamente dell'informazione sul bordo.

Per finire osserviamo che:

- Fasshauer in un paper del 1999 propose l'uso di un pre-condizionatore diagonale di Jacobi per accelerare la convergenza del metodo del gradiente coniugato per risolvere il sistema lineare che coinvolge la matrice di collocazione, tuttavia, dal momento che questa tecnica non apportava miglioramenti sull'accuratezza della soluzione, quest'idea non è stata sviluppata ulteriormente.

- Gli esperimenti precedenti mostrano che le stime d'errore del teorema 2.1 sono troppo pessimistiche. Infatti per problemi ellittici e funzioni di base C^6 il teorema prevede un errore $\mathcal{O}(h)$, mentre gli esperimenti numerici in tabella 3.8 per il metodo non-stazionario suggeriscono un errore $\mathcal{O}(h^3)$.

4 Esempio di matrice di collocazione singolare

In questo quarto capitolo descriviamo un esempio di matrice di collocazione singolare ottenuta da Hon e Schaback in [6] usando il metodo non-simmetrico.

L'articolo citato si apre con "il metodo proposto da Kansa nel 1990 è stato applicato efficacemente ad una grande varietà di applicazioni diverse, ma non è stata ancora provata la non-singolarità della matrice del sistema lineare coinvolto. Questo articolo mostra che, in effetti, una tale dimostrazione non esiste". Esso infatti esibisce un esempio in cui la matrice di collocazione è singolare. Troviamo che valga la pena capire come gli autori siano arrivati a tale risultato anche perché sfruttano un'interessante applicazione del teorema degli zeri.

Consideriamo l'equazione di Poisson sul quadrato $\Omega = [-1,1] \times [-1,1]$ e fissiamo otto punti equispaziati sul bordo e altri nove punti Y di collocazione all'interno del dominio in modo casuale. Scegliamo poi come funzioni di base le multiquadratiche inverse. Con questi punti costruiamo la matrice di collocazione non-simmetrica 17×17, che indicheremo con A. Chiediamo poi che tale interpolazione generalizzata riproduca esattamente gli elementi di $\mathbb{P}_m^d$, ovvero i polinomi in d-variabili di grado $m+1$. Vogliamo cioè che la soluzione del metodo di collocazione soddisfi anche la condizione aggiuntiva

$$\sum_{j=1}^{N} \alpha_j p_j(x_j) = 0 \quad \text{per ogni } p \in \mathbb{P}_m^d. \tag{4.1}$$

Dalla teoria sappiamo che, se siamo interessati ad un problema di interpolazione e se m è scelto grande abbastanza, il sistema risultante è non-singolare. Sempre dalla teoria sappiamo poi che per le multiquadratiche basta prendere $m \geq 1$. Scelto quindi $m = 1$, aggiungiamo una funzione costante e quindi una sola condizione di tipo (4.1), ottenendo una matrice 18×18 che chiameremo B . Abbiamo così ottenuto una matrice $B(Y)$ i cui elementi sono funzioni lisce dei nove punti interni $Y = \{x_9, \ldots, x_{17}\} \subset [-1,1]^{18}$ e il cui determinante $\det(B(Y))$ è una funzione liscia su $[-1,1]^{18}$. Per trovare quindi il controesempio cercato basterà trovare due insiemi non degeneri $Y_1, Y_2 \subset [-1,1]^{18}$ con

$$\det(A(Y_1)) \cdot \det(A(Y_2)) < 0, \tag{4.2}$$

in tal caso infatti grazie al teorema degli zeri sappiamo che su un qualsiasi percorso congiungente Y_1 ad Y_2 il determinante deve annullarsi almeno una volta.

Per trovare due insiemi Y_1 ed Y_2 non degeneri che soddisfino (4.2) gli autori hanno eseguito un gran numero di test considerando di volta in volta nuovi punti distribuiti casualmente all'interno di Ω, scartando in casi in cui la distanza di due dei nove

punti era inferiore a 0.1 per evitare che due punti collassassero e quindi evitare zeri banali del determinante. Hon e Schaback hanno scartato anche casi in cui il numero di condizionamento della matrice di collocamento era troppo alto in relazione al valore assoluto del determinante, ovvero che fosse rispettata la condizione

$$\frac{\mathrm{cond}(A)}{\det(A)} < 10^{10}. \tag{4.3}$$

Nello stesso articolo gli autori spiegano che, dopo 7846 esperimenti con multiquadratiche di parametro $c = 0.5$, sono riusciti a trovare la speciale configurazione di punti

$$\begin{aligned}
x_9 &= (0.8273563011676801, 0.1802418079135202),\\
x_{10} &= (0.2394614691098507, -0.3252848579200380),\\
x_{11} &= (0.9794477615409753, -0.9277954753152073),\\
x_{12} &= (-0.9372443072205616, -0.2431062149084667),\\
x_{13} &= (0.8888265033666167, -0.5002654378769293),\\
x_{14} &= (0.1633663452997728, 0.4030965172700103),\\
x_{15} &= (-0.9984683087088486, 0.0582202193598357 9),\\
x_{16} &= (-0.07520452098697639, -0.3715893613973583),\\
x_{17} &= (0.3489601022326201, 0.1330205365703536),
\end{aligned}$$

rappresentati in figura 4.1.

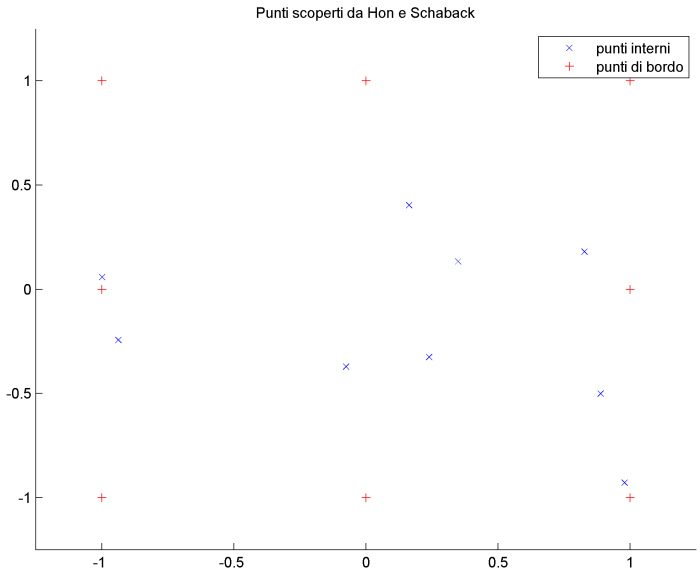


Figura 4.1: In blu: disposizione dei punti $x_9, \ldots, x_{17}$ individuati da Hon e Schaback. In rosso: disposizione dei punti equispaziati sul bordo $x_1, \ldots, x_8$.

Ebbene, questa particolare configurazione di punti porta ad una matrice di collocazione con determinante negativo. Riportiamo di seguito sia il codice MATLAB che i risultati numerici che abbiamo ottenuto.

Programma 4.1: Calcolo della matrice di collocazione con il metodo non-simmetrico usando come centri e punti di collocazione i punti di Hon e Schaback

```
function B = collocationMatrix(varargin)
% authors: Davide Boscaini, Antoine Glorieux, Simone Parisotto
% date: 2012/04/20
% function: compute the collocation matrix with the particular choice
%           of the firsts 8 interior points of Y.C. Hon and R. Schaback
%           in "On unsymmetric collocation by radial basis functions",
%           Appl. Math. Comput., 119, pagg. 177-186, 2001,
%           and with P as last point.

% IMQ RBF and its Laplacian
c = 0.5;
rbf = @(r) c./sqrt(c^2 + r.^2);
Lrbf = @(r) (c*(r.^2 - 2*c^2))./((c^2+r.^2).^(5/2));

% centers, interior points
interior_points = [ 8.273563011676801e-01, 1.802418079135202e-01;
     2.394614691098507e-01, -3.252848579200380e-01;
     9.794477615409753e-01, -9.277954753152073e-01;
    -9.372443072205616e-01, -2.431062149084667e-01;
     8.888265033666167e-01, -5.002654378769293e-01;
     1.633663452997728e-01,  4.030965172700103e-01;
    -9.984683087088486e-01,  5.822021935983579e-02;
    -7.520452098697639e-02, -3.715893613973583e-01;
     3.489601022326201e-01,  1.330205365703536e-01 ];

if length(varargin)>0
    % replace the last point
    P = varargin{1};
    interior_points(end,:) =  P;
end

% centers, equispaced boundary points
sn = floor(sqrt(length(interior_points)));
bdylin = linspace(-1,1,sn)';
bdy0 = -1*ones(sn-1,1);
bdy1 = ones(sn-1,1);
boundary_points = [bdylin(1:end-1) bdy0; bdy1 bdylin(1:end-1);...
    flipud(bdylin(2:end)) bdy1; bdy0 flipud(bdylin(2:end))];

% centers
centers = [interior_points; boundary_points];

% number of centers
N = length(centers);
N_interior = length(interior_points);
N_boundary = length(boundary_points);

% collocation points
x = centers;

% initialize collocation matrix
A = zeros(N,N);

% compute interior collocation matrix
```

```
for i = 1:N_interior
    for j = 1:N
       A(i,j) = Lrbf(norm(x(i,:)-centers(j,:)));
    end
end

% compute boundary collocation matrix
for i = N_interior+1:N
    for j = 1:N
       A(i,j) = rbf(norm(x(i,:)-centers(j,:)));
    end
end

% additional condition: we are forcing exact reproduction of costant
% polynomials
B = zeros(N+1,N+1);
B(1:N,1:N) = A;
B(N+1,1:N) = ones(N,1)';
B(1:N,N+1) = ones(N,1);

% additional condition: we are forcing exact reproduction of linear
% polynomials
C = zeros(N+3,N+3);
C(1:N,1:N) = A;
C(N+1,1:N) = ones(N,1)';
C(N+2,1:N) = centers(:,1)';
C(N+3,1:N) = centers(:,2)';
C(1:N,N+1) = ones(N,1);
C(1:N,N+2) = centers(:,1);
C(1:N,N+3) = centers(:,2);
```

Se lanciamo le seguenti righe da terminale

```
> B = collocationMatrix();
> fprintf('Determinant:     %e\n', det(B));
> fprintf('Condition number:     %e\n', cond(B));
```

troviamo

```
> Determinant:     -6.412006e+004
> Condition number:     1.167376e+003
```

a conferma del fatto che questa matrice ha determinante negativo e che il numero condizionamento non è troppo grande. Per accertarci che i risultati trovati non siano troppo affetti da errori numerici e ricordando la disuguaglianza

$$\|\text{errore}\| \leq \|\text{residuo}\| \cdot \text{cond}(B), \tag{4.4}$$

facciamo un test sul residuo, cioè

```
> b = rand(length(B),1);
> x = B\b;
> residuo = norm(B*x-b);
> errore = residuo*cond(B);
> fprintf('Error:     %e\n', errore);
```

trovando

```
> Error:     1.461007e-011.
```

Per il teorema degli zeri e quanto spiegato precedentemente possiamo affermare che esiste una configurazione di punti tale per cui il determinante della matrice B sia nullo (si veda la figura 4.2). Questo risultato di esistenza ci spinge ad indagare ulteriormente per trovare la particolare configurazione per cui il determinante è nullo. Per far questo scegliamo di muovere l'ultimo punto all'interno di Ω, e quindi considerare il determinante della matrice risultante come funzione del punto P che muoviamo, il cui grafico è stato riportato in figura 4.3.

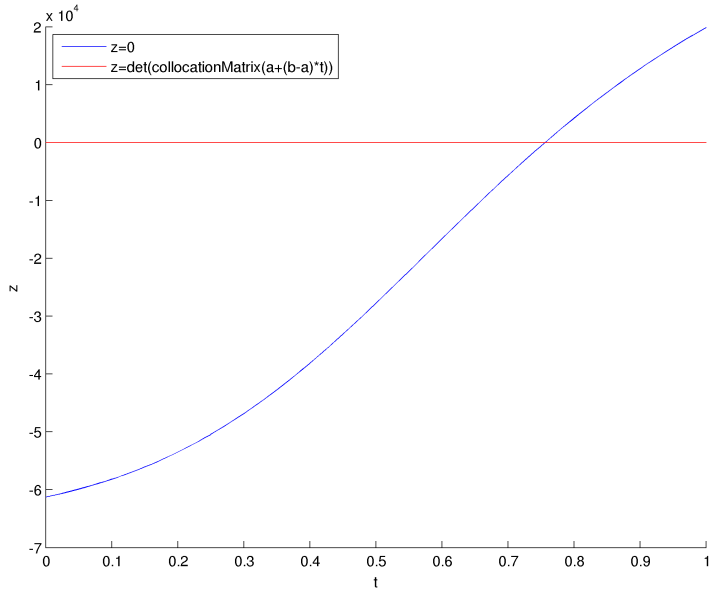


Figura 4.2: Esistenza di uno zero della funzione determinante.

Come si può vedere in figura 4.3, i punti del dominio disposti lungo la linea arancione hanno determinante nullo. A questo punto, fissati i due punti di partenza, $a = (-0.6, 0)$ e $b = (-0.92, 0.22)$, uno nella parte di dominio che corrisponde a valori negativi del determinante e l'altro nella regione a determinante positivo, per trovare il punto cercato, lanciamo un algoritmo di bisezione con tolleranza 10^{-15} e numero massimo di iterazioni 100. Il codice MATLAB che implementa il tutto è riportato di seguito.

Programma 4.2: dangerousPoint

```
% authors: Davide Boscaini, Antoine Glorieux, Simone Parisotto
% date: 2012/04/23
% script: research of a zero of the determinant function with a
%         naive bisection method.
clear all;
close all;
clc;

```

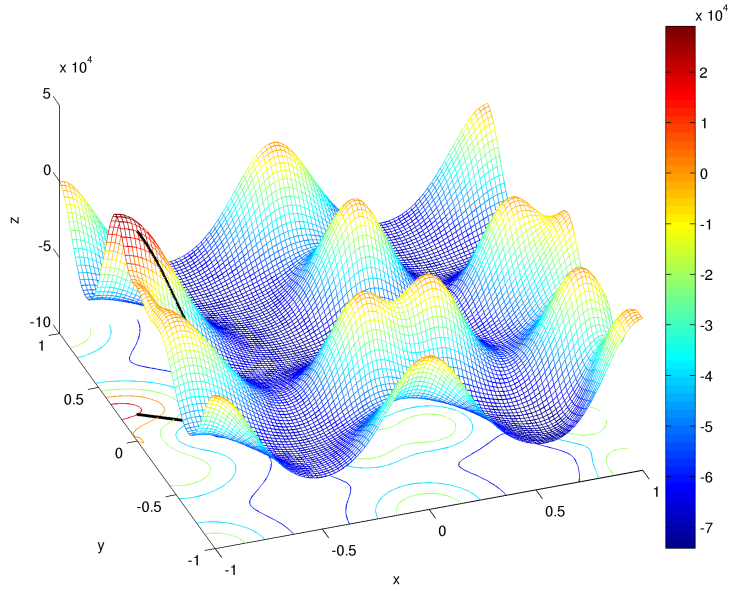


Figura 4.3: Grafico del determinante visto come funzione di un punto.

```
% starting point
a = [ -0.6,  0];
% last point
b = [-0.92, 0.22];

fun = @(P) det(collocationMatrix(P));
% tollerance
tol = 1e-15;
% maximum number of iterations
maxit = 100;
% bisection method
[x,iter,Δ] = bisezione(fun,a,b,tol,maxit);

% dangerous point
fprintf('Dangerous point:     %e\t %e\n', x);

% collocation matrix
A = collocationMatrix(x);

% determinant of the quasi-singular matrix
fprintf('Determinant:     %e\n', det(A));

% condition number of the resulting matrix
fprintf('Condition number:     %e\n', cond(A));

% residual safe-control
R = A*pinv(A)-eye(length(A));
fprintf('Residual:     %e\n', norm(sum(R,2)));

b = rand(length(A),1);
y = inv(R)*A\(inv(R)*b);
residuo = norm(A*y-b);
```

```
errore = residuo*cond(A);
fprintf('Error:     %e\n', errore);
```

Lanciando lo script precedente si trova

```
> Dangerous point:     -8.419254e-001  1.663237e-001
> Determinant:     2.171973e-010
> Condition number:     3.749949e+016
```

quindi il determinante si annulla in corrispondenza della configurazione precedente a cui è sostituito come ultimo punto $P = (-0.8476210563544359, 0.1702394762436744)$.

I punti sono ancora ben separati, quindi la configurazione finale non è degenere, il determinante è decisamente piccolo e il numero di condizionamento, di conseguenza, è molto grande. A questo punto gli autori notano che la norma della somma sulle righe della matrice dei residui è molto bassa (nel nostro caso 0.009431651) e usano questo fatto per provare che il risultato non è affetto da errori numerici troppo grandi. Non possiamo fare a meno di notare che questo passaggio non è particolarmente chiaro: non si definisce cosa si intende per matrice dei residui e perché la somma sulle righe (o colonne) dia l'informazione cercata. Infine la stima (4.4) non è per niente incoraggiante: se si lanciano le righe

```
> b = rand(length(A),1);
> y = A\b;
> residuo = norm(A*y-b);
> fprintf('Residuo:     %e\n', residuo);
> errore = residuo*cond(A);
> fprintf('Error:     %e\n', errore);
```

oltre ad un warning si ottiene

```
> Residuo:     0.0391
> Error:     5.430934e+016
```

Se nel programma 4.1 si ritorna la matrice C al posto della matrice B, si può ripetere lo stesso esempio ma ora la nostra interpolante generalizzata interpolerà esattamente i polinomi lineari e non più solo quelli costanti.

Quali conclusione possono quindi essere tratte da questo esempio? La conseguenza ovvia è che non può esistere una dimostrazione di non singolarità della matrice di collocazione non-simmetrica, tuttavia, come dicono gli stessi autori, i controesempi sono "rare birds". Nelle applicazioni, quindi, è a tutti gli effetti preferibile usare il metodo di Kansa per le sue migliori performance: l'esistenza di casi singolari così patologici e rari non è infatti un'obiezione ad un metodo numerico valido.

Bibliografia

[1] G. E. Fasshauer. *Meshfree Approximation Methods with Matlab*. U.S.A.: World Scientific, 2007 (cit. alle pp. 1, 16, 19, 22, 25, 30, 32, 34).

[2] G. E. Fasshauer. "Solving partial differential equations by collocation with radial basis functions". In: *Surface Fitting and Multiresolution Methods* (1997), pp. 131–138 (cit. a p. 8).

[3] A. I. Fedoseyev, M. J. Friedman e E. J. Kansa. "Improve multiquadric methods for elliptic partial differential equations via PDE collocation on the boundary". In: *Comput. Math. Applic.* 43 (2002), pp. 439–455 (cit. a p. 12).

[4] C. Franke e R. Schaback. "Convergence order of meshless collocation methods using radial basis functions". In: *Adv. in Comput. Math.* 8 (1998), pp. 381–399 (cit. a p. 11).

[5] C. Franke e R. Schaback. "Solving partial differential equations by collocation with radial basis functions". In: *Appl. Math. Comput.* 93 (1998), pp. 73–82 (cit. a p. 11).

[6] Y. C. Hon e R. Schaback. "On Unsymmetric Collocation by Radial Basis Functions". In: *Appl. Math. Comput.* 119 (2001), pp. 177–186 (cit. alle pp. 10, 38).

[7] E. J. Kansa. "Multiquadrics – A scattered data approximation scheme with applications with computational fluid-dinamics – II: Solutions to parabolic, hyperbolic and elliptic partial differential equations". In: *Comput. Math. Appl.* 19 (1990), pp. 147–161 (cit. alle pp. 8, 9).

[8] E. Larsson e B. Fornberg. "A numerical study of some radial basis function based solution methods for elliptic PDEs". In: *Comput. Math. Appl.* 46 (2003), pp. 891–902 (cit. alle pp. 12, 13).

[9] H. Wendland. *Scattered Data Approximation*. Cambridge: Cambridge University Press, 2005 (cit. alle pp. 1, 5, 11).