Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Corso di Laurea in Ingegneria dell'Automazione

Progettazione Sistemi di Controllo

Ottimizzazione distribuita per Mapbuilding in reti robotiche

Carlo Bonato
Federico Antonio Pancino
Marco Tognon

Docente:
Prof. Luca Schenato

A.A. 2012/2013

Indice

Abstract

Questa relazione presenta l'ottimizzazione di un algoritmo di stima distribuita finalizzato alla realizzazione di mappe di livello in ambienti sconosciuti.
La prima parte del progetto consiste nell'analisi della stima del campo, in maniera centralizzata, attraverso due tipologie differenti di basi a funzione radiale (RBF), ma comunque con posizioni fissate. Da tale analisi si vuole capire qual'è la miglior disposizione delle funzioni e il miglior numero di parametri per stimare il campo.
Osservati i limiti di questo metodo di stima, si è cercato un algoritmo per posizionare le funzioni radiali in modo ottimale. Questa seconda parte del progetto è articolata in varie sezioni di complessità crescente; inizialmente viene effettuata una stima centralizzata attraverso un algoritmo Quasi-Newton supponendo di avere già a disposizione un set di misure.
Tale soluzione, però, è poco pratica, in quanto si deve aspettare che vengano prelevate tutte le informazioni dal campo, perciò, viene sviluppato poi un algoritmo di stima iterativo nelle misure, in cui la base di stima evolve ad ogni misura acquisita. Infine, vengono adattati tali procedimenti ad una stima di tipo distribuito. In tal modo i vari agenti condividono le informazioni apprese dall'area assegnata loro, ottenendo, in un numero finito di iterazioni, la convergenza ad una rappresentazione comune del campo.

Introduzione

Recentemente i sistemi multirobot sono stati oggetto di varie ricerche da parte della comunità scientifica, data la loro applicabilità a differenti ambiti come nei servizi, nel settore militare e nella ricerca. L'interesse nell'uso di sistemi multirobot è dovuto alle loro caratteristiche uniche (come affidabilità e flessibilità nel portare a termine i compiti), le quali fanno si che il sistema sia tollerante al mal funzionamento di singoli agenti, rendono possibili: alcune applicazioni altrimenti impraticabili, la riduzione dei tempi di lavoro e l'aumento delle performance tramite l'esecuzione parallela del compito assegnato. Molte ricerche sono centrate sullo studio di sistemi ecologici, videosorveglianza, monitoraggio di ambienti ([19], [10] e [5]) e soprattutto esplorazione e mapping di aree sconosciute. La tipologia di sistemi in questione è molto importante, in quanto andrà a sostituire l'uomo in applicazioni pericolose, come l'individuazione e la ricognizione di zone a rischio, o addirittura l'esplorazione di zone impraticabili. Inoltre l'aspetto distribuito si rivela necessario in tutti quei settori dove l'area di interesse è eccessivamente vasta e la quantità di dati da analizzare va oltre la capacità di calcolo di un unico agente.

Applicazioni in cui la soluzione distribuita si rivela efficace, se non necessaria, sono ad esempio: il controllo degli incendi attraverso droni dotati di sensori di temperatura e telecamere termiche, il monitoraggio di segnali radio in zone a rischio (zone di guerra) e lo scandaglio di fondali marini attraverso agenti dotati di sonar. Un altro esempio applicativo del problema di mapping, in cui l'uso di sistemi multiagente risulta essere rilevante, si trova nei settori ambientale [23], ecologico e metereologico [7]. In tutte queste applicazioni lo scopo è quello di stimare la mappa di livello di una generica grandezza fisica. Ad esempio, nel controllo degli incendi, andando a misurare la temperatura, si vogliono determinare le zone in cui l'incendio è maggiormente concentrato, nonché la sua distribuzione sull'intera area per gestire al meglio le operazioni di estinzione.

Al fine di interpretare e stimare le misure del campo, uno degli strumenti più utilizzati è dato dai modelli neurali [1] e, nello specifico, quelli a funzioni RBF [9]. Essi infatti si adattano molto bene a descrivere superfici curve nello spazio e, di conseguenza, i campi di nostro interesse. Nella maggior parte delle tecniche di stima si utilizza un modello lineare nel quale l'unico parametro è l'ampiezza delle funzioni RBF, mantenendo fissate a priori la loro forma e distribuzione nello spazio. Tali tecniche sono in genere implementate con algoritmi basati sui minimi quadrati. In letteratura si trovano però algoritmi di tuning del modello per posizionare in maniera ottimale le funzioni; in [4] viene utilizzata la tecnica dei minimi quadrati su modello non lineare, mentre in [3] viene implementato un algoritmo con albero di regressione. A livello distribuito, in [16] si utilizza sempre un algoritmo ai minimi quadrati su modello lineare, in cui le misure vengono acquisite dai robot ma la stima vera e propria viene fatta a livello centralizzato, una volta raccolti tutti i dati. Invece in [20] si effettua realmente una stima distribuita M.Q., convergendo ad un risultato comune tramite scambio di informazione fra i vari agenti, utilizzando tecniche di consensus in media [21].

Dalle nostre ricerche in letteratura si evince l'esistenza di buoni algoritmi di stima centralizzata che utilizzano classi di modelli in cui i parametri non sono unicamente le ampiezze delle funzioni ma anche la loro posizione e forma. Questo tipo di algoritmi presentano però importanti difetti per l'applicazione ai nostri scopi: la loro natura non iterativa nelle misure non ne consente un'applicazione in ambiti real-time, infatti per ottenere il risultato della stima occorre prima effettuare la rilevazione di un insieme considerevole di dati. Un'ulteriore problematica è quella di rendere distribuiti, su più agenti, questa tipologia di algoritmi. Infatti quelli già esistenti, in ambito multiagente, si basano su semplici modelli RBF stimando unicamente il parametro ampiezza; così facendo si ottiene un semplice problema di stima lineare le cui tecniche di soluzione sono facilmente implementabili in modo distribuito.

Ci siamo così prefissi di implementare un algoritmo di stima non lineare, in maniera distribuita ed iterativa nelle misure, cercando di soddisfare al meglio le specifiche di tempo reale. Per raggiungere tale scopo si è quindi suddiviso il progetto nei tre punti chiave di cui sopra. Dopo un'accurata analisi simulativa atta a decidere la miglior struttura del modello, si è pensato di implementare un algoritmo di stima non lineare con direzione di ricerca di Newton. Questo tecnica dovrebbe comportare la possibilità di ottenere una stima più accurata con modelli di complessità minore. In seconda istanza si è proceduto con l'implementazione di una sua versione iterativa per poterlo rendere adatto alle specifiche real-time; scelta che comporta peraltro un'onerosità computazionale minore. Di contro, però, queste tecniche soffrono di instabilità e quindi hanno bisogno di un settaggio dei parametri molto raffinato. In conclusione si è passati alla realizzazione della versione distribuita dell'algoritmo, che consente tempi di stima nettamente inferiori, richiedendo d'altro canto di porre una certa attenzione sugli algoritmi di comunicazione.

Parte I

Stima centralizzata con basi statiche

Introduzione

Nella prima parte del progetto si è preso in considerazione il problema della stima centralizzata del campo tramite un modello di misura lineare (sezione 1.2) basato sul metodo dei minimi quadrati. Tale stima viene effettuata rappresentando il campo attraverso una combinazione lineare di Radial Basis Function (RBF, capitolo 1).
Le due tipologie di basi utilizzate in questa prima parte del progetto sono statiche, ovvero i centri e le varianze delle funzioni, di cui sono composte sono fissati a priori, mentre solo le ampiezze possono variare.
Verranno effettuate quindi delle simulazioni su varie tipologie di campi, in modo da poter verificare la bontà del metodo in varie situazioni, equiparabili a quelle che si possono incontrare nelle applicazioni reali. Tali prove saranno diversificate per tipologia di base utilizzata e soprattutto per numero di funzioni utilizzate.
Questa prima parte del progetto ha quindi come scopo finale quello di trovare la miglior disposizione statica delle funzioni di stima e soprattutto la densità ottimale per rappresentare i vari campi di misura.

Capitolo 1

Reti RBF e stima centralizzata

1.1 Reti neurali

In questo progetto vengono utilizzate, per costruire il modello di stima, le Radial Basis Function (RBF), ovvero basi a funzione radiale che sono alla base delle reti neurali. Le reti neurali sono un utile strumento per risolvere vari problemi, ad esempio di pattern recognition, di analisi dei dati e di controllo.
Tra tutti i tipi di reti neurali che sono state studiate, le più importanti e utilizzate, anche in applicazioni pratiche, sono le reti multilayer perception e, appunto, le radial basis function, che rientrano nella classificazione di reti feed-forward, ovvero reti in cui gli output possono essere espressi come funzioni deterministiche degli input.
Le reti neurali RBF sono costituite complessivamente da tre strati (fig. 1.1).

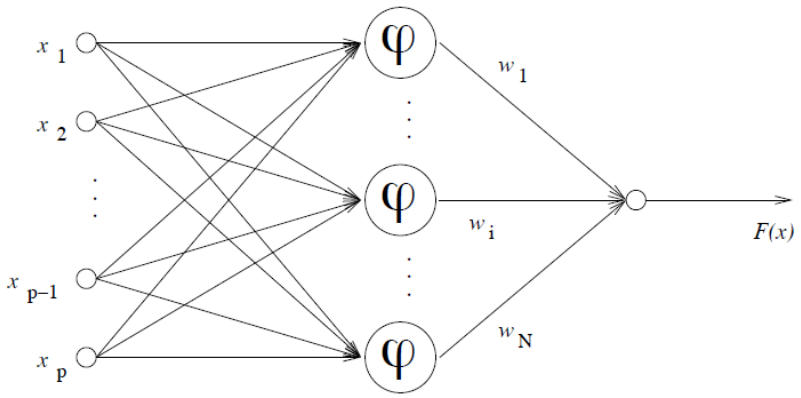


Figura 1.1: Struttura di una rete neurale.

Essi ricevono in ingresso un vettore p-dimensionale, effettuano una trasformazione lineare o meno e generano un'uscita scalare.
In dettaglio:

- il primo strato (*Input*), costituito da p "neuroni", viene utilizzato per prelevare il vettore $\mathbf{x}$ posto in ingresso; esso viene quindi scomposto nelle sue componenti scalari, che vengono inviate a tutti i neuroni del secondo strato;

- nel secondo strato (chiamato anche strato nascosto o *Hidden*), costituito da N neuroni, vengono definite le funzioni a base radiale (da qui il nome di radial basis functions) e si effettua la trasformazione (generalmente non-lineare) richiesta dall'algoritmo. I parametri di quest'ultimo vengono usualmente ottenuti durante una fase preliminare di addestramento (training);

- il terzo ed ultimo strato (*Output*), costituito da un unico neurone, combina linearmente (moltiplicandole per opportuni coefficienti w_i, detti pesi) le uscite dello strato precedente. Nel nostro caso viene effettuata la stima dei parametri del modello, generando così l'uscita complessiva della rete.

Poiché convenzionalmente il numero di strati di una rete è riferito soltanto a quelli nascosti, le reti RBF vengono anche chiamate a singolo strato.
Si può quindi dire che l'architettura RBF è una tipologia di rete che si basa sull'idea di una funzione y($\mathbf{x}$) che può essere ottenuta dalla combinazione di lineare di varie funzioni base.
In genere si hanno N funzioni base, corrispondenti ognuna a un punto dell'insieme di dati; la funzione in questione sarà della forma $\phi(\|x - x_q\|)$, la norma è solitamente una distanza euclidea tra x, la posizione dove si va a prendere il dato e x_q, il punto dove è "centrata" la funzione e ha il ruolo di traslarla nello spazio.
L'output della mappa sarà quindi, come detto, una combinazione lineare delle suddette funzioni

$$y(x, w) = \sum_{k=1}^{N} w_k \phi_k(\|x - x_k\|) \tag{1.1}$$

La funzione $\phi(x)$ si può scegliere in molti modi:

- Gaussiana:
$$\phi_k(x) = exp\Big(-\frac{\|x - x_k\|^2}{2\sigma_k^2}\Big) \tag{1.2}$$
- Multiquadratica:
$$\phi_k(x) = \sqrt{1 + \|x - x_k\|^2} \tag{1.3}$$
- Quadratica Inversa :
$$\phi_k(x) = \frac{1}{1 + \|x - x_k\|^2} \tag{1.4}$$
- Polyharmonic Spline:
$$\phi_k(x) = \|x - x_k\|^j, \qquad j = 1, 3, 5, \ldots \tag{1.5}$$
$$\phi_k(x) = \|x - x_k\|^j \ln(\|x - x_k\|), \qquad j = 2, 4, 6, \ldots \tag{1.6}$$

In questo progetto viene utilizzata la funzione Gaussiana in quanto in letteratura è la più comune e soprattutto perchè si è visto che con le altre funzioni non si riuscivano ad ottenere dei campi con un andamento continuo, ovvero che non avessero "punti angolosi" o "cuspidi", che difficilmente si trovano in applicazioni reali.
Una funzione radiale gaussiana è mostrata in fig. 1.2.

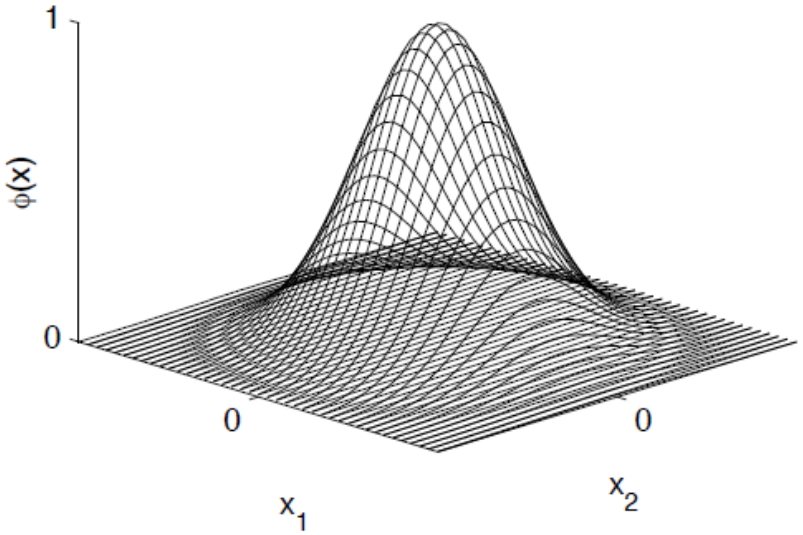


Figura 1.2: RBF Gaussiana bidimensionale centrata nell'origine.

1.2 Stima centralizzata

La stima del campo di interesse può essere vista come un'operazione d'interpolazione. Si supponga di avere un insieme S di N punti $c_t \in \mathcal{R}^2$; a ciascun punto sia associato uno scalare $h_t \in \mathcal{R}$. Si può immaginare che la corrispondenza $c_t \longrightarrow h_t$ sia regolata da una relazione prefissata, incognita.
Il problema dell'interpolazione è quello di determinare una funzione $f(c, \theta)$ tale che:

$$f(c_t, \theta) = h_t, \qquad t = 1, 2, ..., N \tag{1.7}$$

Imponendo che la soluzione sia espressa mediante una rete neurale RBF, l'interpolatore deve essere della forma (fig. 1.1), ovvero una combinazione convessa della base di funzioni a simmetria radiale $[\phi_1(c_t) \dots \phi_p(c_t)]$:

$$f(c_t, \theta) = \sum_{k=1}^{p} \phi_k(c_t)\theta_k \tag{1.8}$$

dove gli N vettori c_t siccome sono punti nel piano, saranno composti da (x_i, y_i).
Applicare la condizione di interpolazione, significa imporre che la funzione $f(c, \theta)$ definita nell'equazione (1.8) rispetti la condizione (1.7) per tutte le coppie (c_t, h_t).
Si definisce quindi il modello lineare:

$$\mathcal{M} = \{M(\theta) : \quad \mathbf{h} = \mathbf{F} + \mathbf{w} = \mathbf{C}\theta + \mathbf{w}\} \tag{1.9}$$

dove:

- $\mathbf{h} = [h_1 \quad h_2 \quad \cdots \quad h_N]^T$ misura rilevata dall'agente, $\mathbf{h} \in \mathbb{R}^N$;
- $\mathbf{F} = [f(c_1, \theta) \quad f(c_2, \theta) \quad \cdots \quad f(c_N, \theta)]^T \in \mathbb{R}^N$;
- $\mathbf{C} = \begin{bmatrix} \phi_1(c_1) & \cdots & \phi_p(c_1) \\ \phi_1(c_2) & \cdots & \phi_p(c_2) \\ \vdots & \ddots & \vdots \\ \phi_1(c_N) & \cdots & \phi_p(c_N) \end{bmatrix}$, matrice i cui elementi sono i valori delle funzioni RBF di stima nei punti di misura, $\mathbf{C} \in \mathbb{R}^{N \times p}$;
- $\theta = [\alpha_1 \quad \alpha_2 \quad \cdots \quad \alpha_p]^T \in \mathbb{R}^p$,
  è la cifra di merito ovvero i parametri del campo, che in questo caso corrispondono alle ampiezze delle RBF gaussiane;
- $\mathbf{w} = [w_1 \quad w_2 \quad \cdots \quad w_N]^T$vettore degli errori di misura, $\mathbf{w} \in \mathbb{R}^N$
- N = numero di misure;
- p = numero di funzioni radiali.

in cui le RBF gaussiane utilizzate sono definite come:

$$\phi_i = exp\Big(-\frac{(x - x_i)^2 + (y - y_i)^2}{2{\sigma_i}^2}\Big) \tag{1.10}$$

centrate in (x_i, y_i) con varianza ${\sigma_i}^2$.
Si ricava quindi, come indicato in [18], minimizzando la norma

$$\mathbf{V}(\theta) = \|\mathbf{h} - \mathbf{C}\theta\| \tag{1.11}$$

lo *stimatore ai Minimi Quadrati*

$$\hat{\theta}(\mathbf{h}) = \left[\mathbf{C}^T\mathbf{C}\right]^{-1}\mathbf{C}^T\mathbf{h} \tag{1.12}$$

di varianza

$$Var(\hat{\theta}(\mathbf{h})) = \left[\mathbf{C}^T\mathbf{C}\right]^{-1}\frac{\sigma_{\mathbf{w}}^2}{N} \tag{1.13}$$

da cui si vede che lo stimatore è funzione lineare delle misure.

La stima del campo si definisce quindi come

$$\hat{\mathbf{h}} = \mathbf{C}\hat{\theta} \tag{1.14}$$

e viene poi valutata attraverso l'errore quadratico medio

$$MSe = \sum_{t=1}^{N} \frac{(h_t - \hat{h_t})^2}{N} \tag{1.15}$$

Osserviamo inoltre che tale stima inoltre è anche di *Massima Verosimiglianza* nel caso in cui l'errore di misura **w** sia effettivamente gaussiano.

Capitolo 2

Basi RBF di stima

La stima del campo è stata effettuata, come detto precedentemente, con delle basi a distribuzione costante su tutta la mappa, in cui si è variato anche il numero di funzioni.
Le tipologie utilizzate sono le seguenti:

1. Base *uniforme*;
2. Base *centroidale*;
3. Base *di creazione*;

2.1 Base RBF Uniforme

La base RBF uniforme prevede, appunto, una distribuzione uniforme dei centri e quindi delle funzioni RBF su tutta la mappa. In particolare, l'algoritmo di creazione implementato valuta le dimensioni del campo (altezza e lunghezza), ed in base al livello di risoluzione scelto dispone le funzioni coprendo interamente il campo di misura.
Aumentando la risoluzione aumenta il numero di funzioni e quindi aumenta la precisione della stima (finché non si arriva all'*overfitting*, capitolo 3).
Un esempio di posizionamento uniforme dei centri delle RBF è quello di fig. 2.1

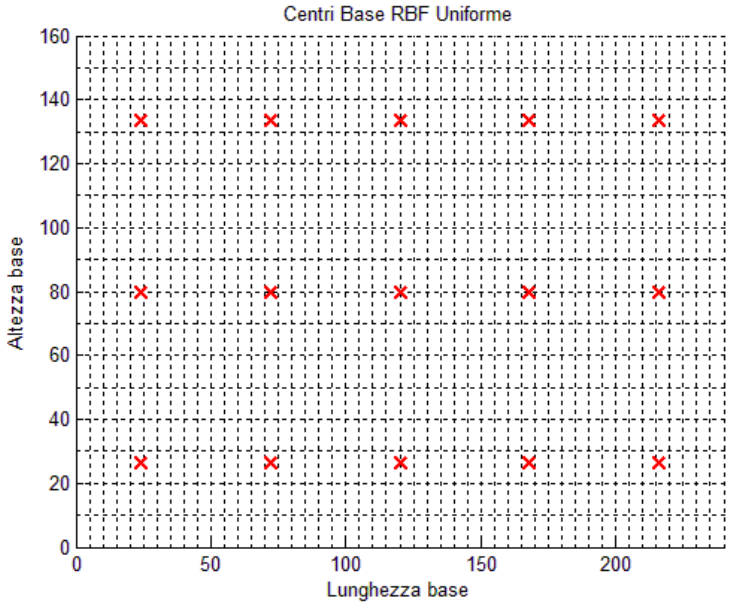
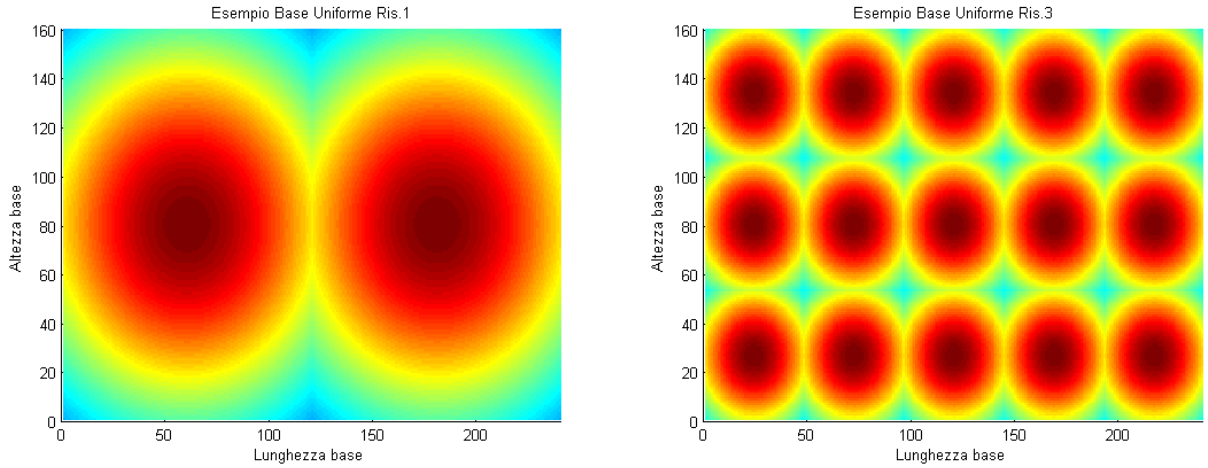


Figura 2.1: Centri Base RBF Uniforme Risoluzione 3.

Per quanto riguarda la varianza (σ_i^2), delle gaussiane che compongono le basi RBF di stima, si è posto che $3\sigma_i$ corrisponda alla distanza minima del centro della funzione dal perimetro del campo.
Esempi di basi RBF uniformi in cui si può vedere l'adattamento della varianza sono quelli di fig. 2.2



(a) Risoluzione 1

(b) Risoluzione 3

Figura 2.2: Esempi Base RBF Uniforme Risoluzione 1 e 3.

2.2 Base RBF Centroidale

L'algoritmo implementato per la creazione della base RBF centroidale prevede che, data una mappa (quadrata o rettangolare), a seconda del numero di livelli desiderati, i centri delle gaussiane che costituiscono la base siano posti uno al centro della mappa, ovvero a metà tra lunghezza e altezza, suddividendo la stessa in 4 quadranti, dei quali viene calcolato il centro, in ognuno viene posta una funzione, suddividendo così la mappa in ulteriori 16 quadranti, di cui si può calcolare il centro e così via.
In pratica, il numero di funzioni (n) cresce secondo la legge

$$n = \sum_{k=0}^{L-1} 4^k \tag{2.1}$$

dove L è il livello di suddivisione massimo che si vuole ottenere.
Un esempio di posizionamento centroidale delle RBF è quello di fig. 2.3

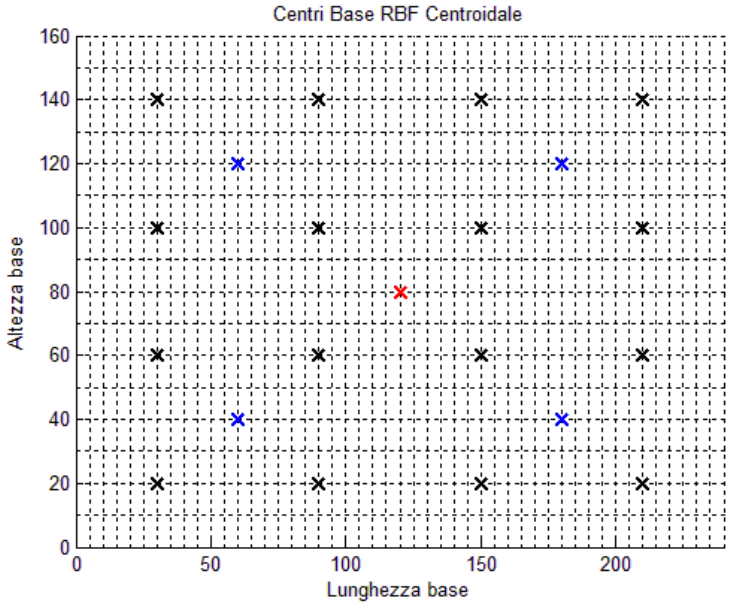


Figura 2.3: Centri Base RBF Centroidale Livello max 3.

in cui si possono vedere in *rosso* il centro del livello 1, in *blu* i centri del livello 2 ed in *nero* i centri del livello 3.

Per quanto concerne la varianza delle gaussiane che compongono la base RBF di stima, come in precedenza, si è posto che $3\sigma_i$ corrisponde alla distanza minima del centro dall'ipotetico perimetro del quadrante; si ha quindi che la varianza di un livello successivo ad un altro è sempre la metà rispetto al livello precedente (all'aumentare dei livelli, le gaussiane hanno varianza sempre minore).
Esempio di basi RBF centroidale in cui si può vedere l'adattamento della varianza è quello di fig. 2.4.

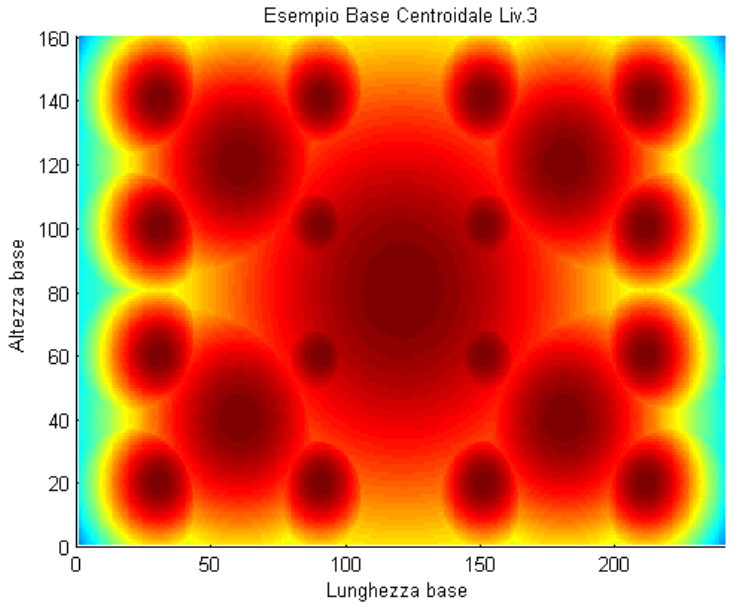


Figura 2.4: Esempio Base RBF Centroidale Livello max 3.

Si è scelta questa particolare configurazione per la base di stima in quanto la gaussiana centrale ha lo scopo di rappresentare la media del campo, mentre le funzioni di contorno servono a migliorare l'accuratezza dello stesso.

2.3 Base RBF di creazione

Nei casi precedenti, con basi uniforme e centroidale, la stima che si otterrà non sarà sicuramente consistente, ovvero non si arriverà ad avere un errore di stima che corrisponde al solo rumore dovuto alle misure; perciò per verificare la bontà delle stime si è scelto di utilizzare come base per il modello di stima la stessa del campo vero come vedremo nel capitolo 3 per confrontare gli errori.

Capitolo 3

Simulazioni

I campi presi in considerazione sono costituiti da una base con le seguenti dimensioni:

- Altezza = 160 unità pixel;
- Lunghezza = 240 unità pixel;

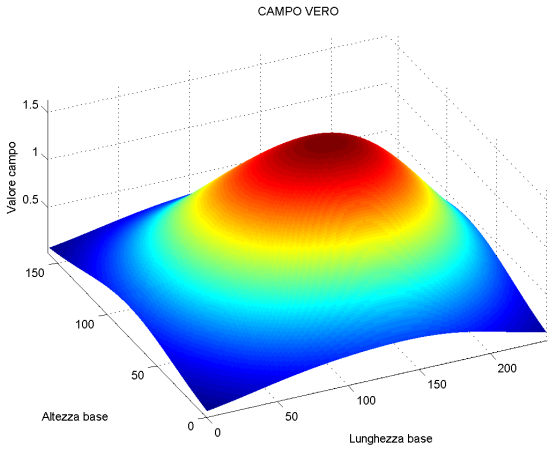
Nelle simulazioni si è effettuata la stima di campi, creati come combinazione lineare di RBF, di tipologia diversa per variabilità, localizzazione e concentrazione, ovvero:

1. campo poco variabile (fig. 3.1a);
2. campo concentrato in due vertici della mappa (fig. 3.1b);
3. campo variabile (fig. 3.1c);
4. campo concentrato al centro della mappa (fig. 3.1d);
5. campo concentrato su un lato della mappa (fig. 3.1e).

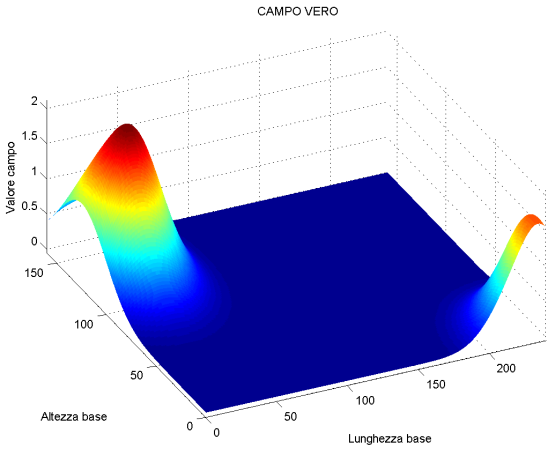
Ciò si è fatto per capire quali sono le prestazioni delle stime attraverso le basi RBF uniforme e centroidale a seconda della tipologia del campo da stimare in relazione ad applicazioni reali.
In questo modo, se si avessero a disposizione delle informazioni a priori sul tipo di campo da stimare, si potrebbe scegliere adeguatamente la base RBF per il modello di stima.
Nelle simulazioni presentate vengono effettuate 10000 misure, in maniera casuale, ad ognuna delle quali viene aggiunta una componente di rumore con distribuzione gaussiana di varianza $\sigma^2_{\mathbf{w}} = 0.2$, in modo da simulare il rumore di misura.

Nota 1 Siccome il numero di RBF della base centroidale varia con la funzione (2.1), con 7 livelli il numero di funzioni è 5461 e quindi per motivi computazionali si è scelta questa come soglia massima di stima. Come si noterà successivamente tale livello è anche il primo in cui si inizia ad osservare l'effetto di ***overfitting*** in maniera evidente, cioè l'errore di stima aumenta in quanto ci sono più funzioni radiali di quante ne servirebbero, ovvero si ha una stima peggiore rispetto ai precedenti livelli (con un numero di funzioni della base RBF minore).

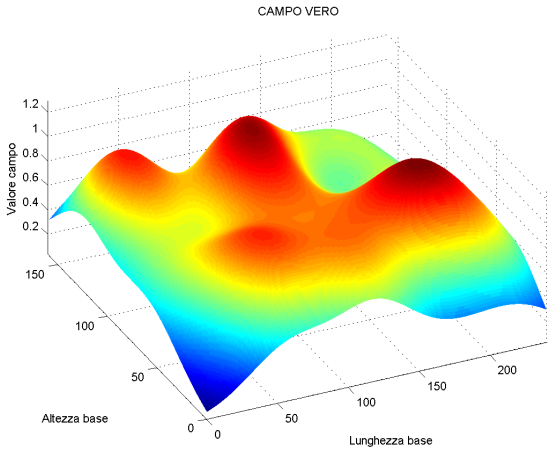
Nota 2 Come indicato in [18] per il calcolo dell'errore di predizione, vengono presi due set di misure. Sul primo set (di *identificazione*), più numeroso (10000 misure), viene calcolata la stima dei parametri e del campo, mentre col secondo set (di *validazione*), meno numeroso (1000 misure), viene calcolato l'errore quadratico medio come indicato nell'equazione (1.15).



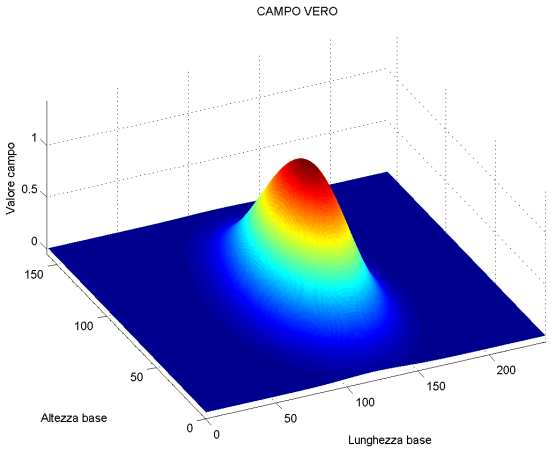
(a) Campo 1



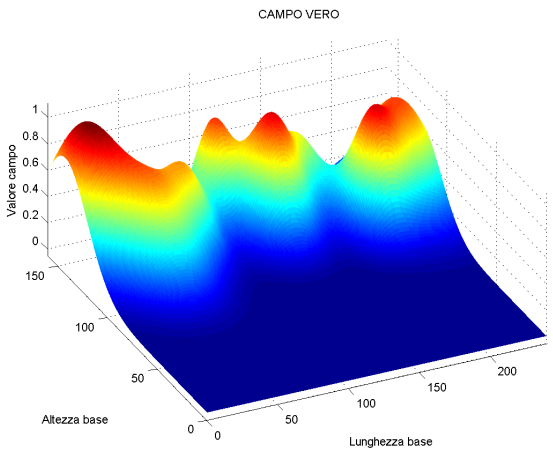
(b) Campo 2



(c) Campo 3



(d) Campo 4



(e) Campo 5

Figura 3.1: Campi di stima

3.1 Campo poco variabile

Il primo campo stimato è quello poco variabile (fig. 3.2):

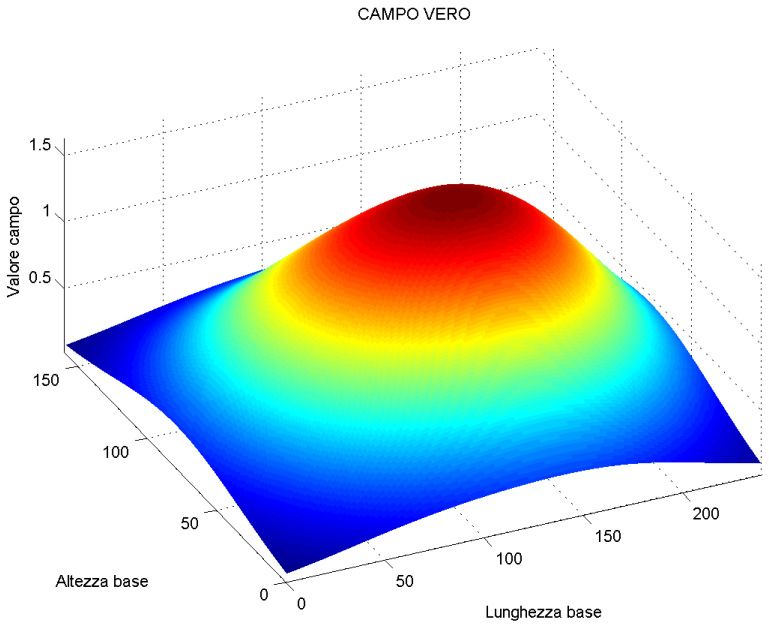


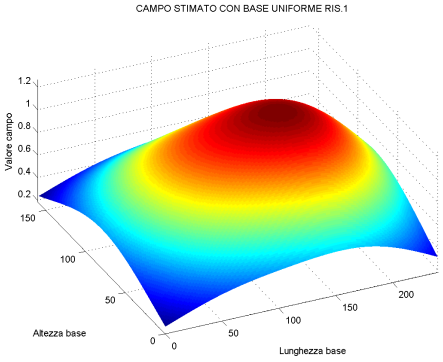
Figura 3.2: Campo da stimare

3.1.1 Stima con Base RBF Uniforme

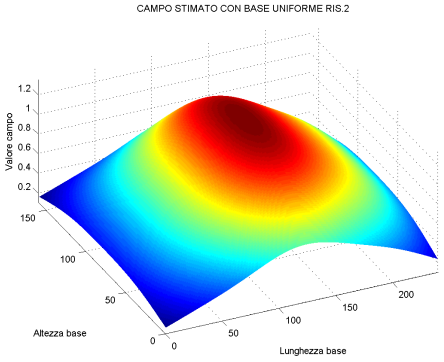
L'evoluzione della stima con base RBF uniforme, in funzione della risoluzione, è presentata in fig. 3.3 e fig. 3.4.
Sia graficamente che dall'evoluzione dell'errore di stima di fig. 3.6, si nota che al crescere del livello di risoluzione la stima del campo migliora. In particolare si ha un netto miglioramento (abbassamento dell'errore di stima) tra Ris.1 e Ris.3, dove la curva dell'errore stesso è molto ripida, poi l'andamento diventa quasi piatto fino al valore minimo di Ris.4 e successivamente risale per fenomeno di overfitting. I valori degli errori di stima sono presentati in tab. 3.1.

Risoluzione	Errore di stima	Numero RBF	Varianza Max	Varianza Min	Varianza Media
1	0.0731	2	8.9689e-009	8.8351e-009	8.9020e-009
2	0.0718	6	2.9757e-008	2.5573e-008	2.7254e-008
3	0.0410	15	8.2083e-008	6.3617e-008	7.1790e-008
4	***0.0390***	***24***	1.4024e-007	9.8460e-008	1.2327e-007
5	0.0393	40	2.4020e-007	1.5522e-007	2.0460e-007
6	0.0396	54	3.5077e-007	2.1459e-007	2.9378e-007
7	0.0397	77	/	/	/
17	0.0408	442	/	/	/
27	0.0449	1107	/	/	/
37	0.0514	2072	/	/	/
47	0.0690	3337	/	/	/
57	0.1957	4902	/	/	/

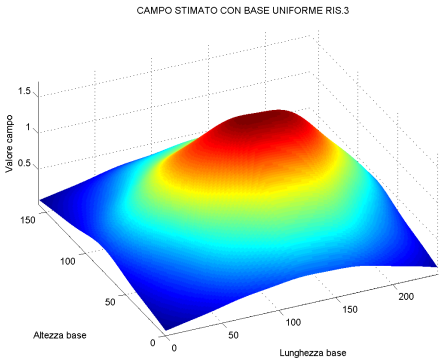
Tabella 3.1: Errori di stima Base Uniforme - Varianze parametri



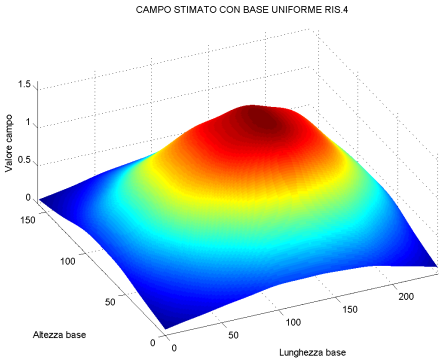
(a) Risoluzione 1



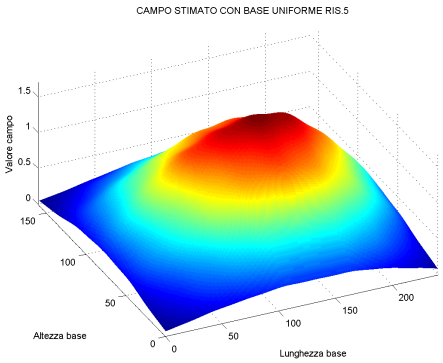
(b) Risoluzione 2



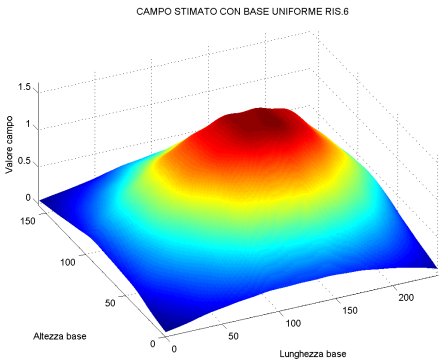
(c) Risoluzione 3



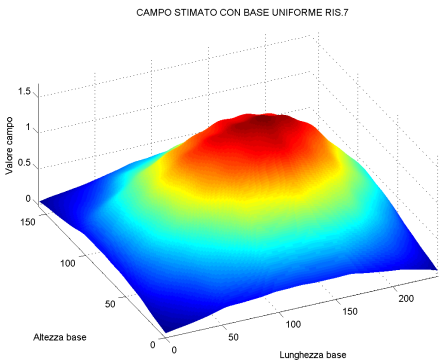
(d) Risoluzione 4



(e) Risoluzione 5

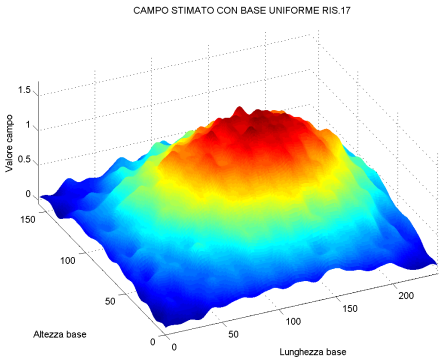


(f) Risoluzione 6

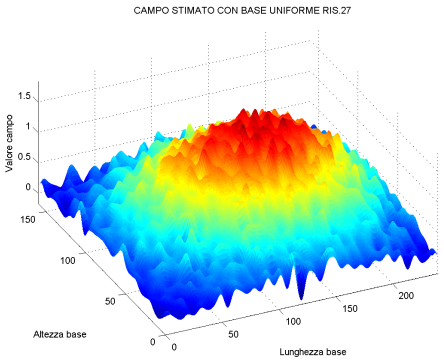


(g) Risoluzione 7

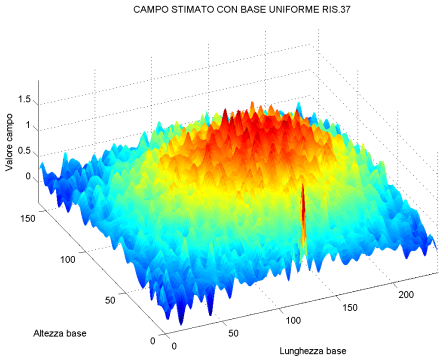
Figura 3.3: Evoluzione stima Base Uniforme Risoluzione max 7



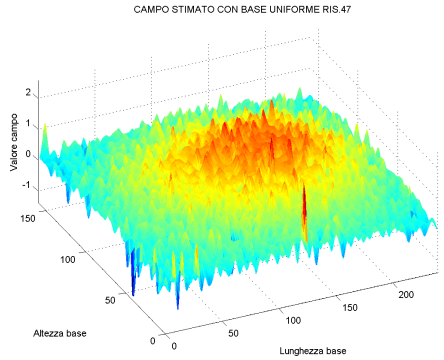
(a) Risoluzione 17



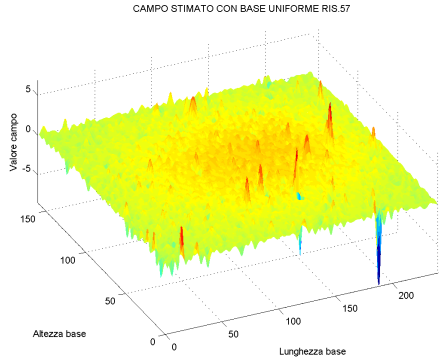
(b) Risoluzione 27



(c) Risoluzione 37



(d) Risoluzione 47



(e) Risoluzione 57

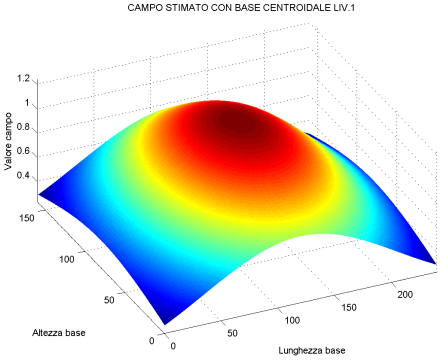
Figura 3.4: Evoluzione stima Base Uniforme Risoluzione max 57

3.1.2 Stima con Base RBF Centroidale

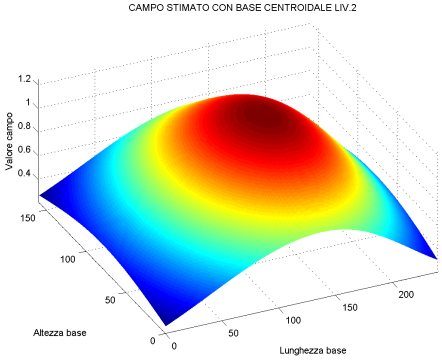
L'evoluzione della stima con base RBF centroidale su numero di livelli è presentata in fig. 3.5.
Dall'evoluzione dell'errore di stima di fig. 3.6, si nota che all'aumentare dei livelli della base, la stima del campo migliora. Si ha un forte miglioramento tra Liv.1 e Liv.3, poi l'andamento diventa quasi piatto fino al valore minimo di Liv.4 e successivamente risale per overfitting.
I valori degli errori di stima sono presentati in tab. 3.2.

Livello	Errore di stima	Numero RBF	Varianza Max	Varianza Min	Varianza Media
1	0.0871	1	4.3275e-009	4.3275e-009	4.3275e-009
2	0.0818	5	8.8749e-008	6.2997e-008	6.9005e-008
3	0.0412	21	4.4197e-007	2.2924e-007	3.2236e-007
4	***0.0396***	***85***	2.5403e-006	9.1187e-007	1.4116e-006
5	0.0402	341	1.5926e-005	3.0529e-006	6.0888e-006
6	0.0472	1365	1.0282e-004	1.1269e-005	2.8615e-005
7	0.3230	5461	/	/	/

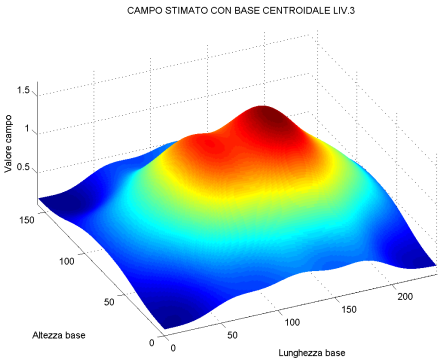
Tabella 3.2: Errori di stima Base Centroidale - Varianze parametri



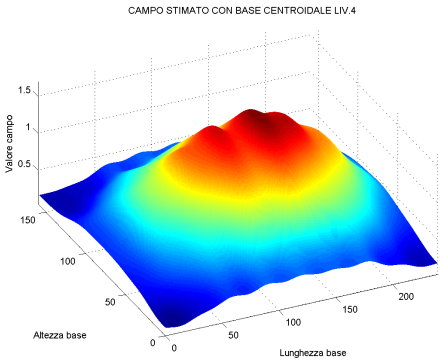
(a) Livello 1



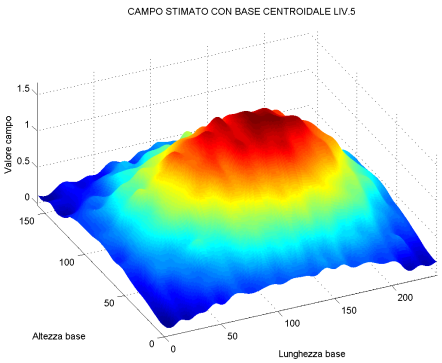
(b) Livello 2



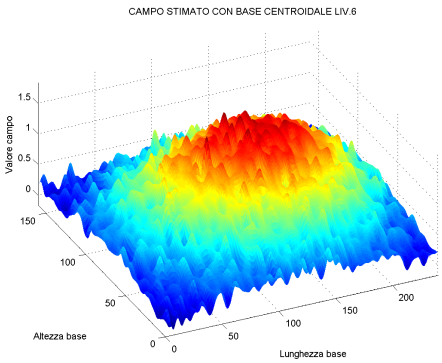
(c) Livello 3



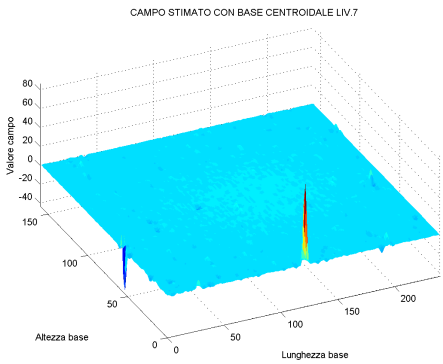
(d) Livello 4



(e) Livello 5



(f) Livello 6



(g) Livello 7

Figura 3.5: Evoluzione stima Base Centroidale

3.1.3 Confronto stime

Dopo aver effettuato le stime con base uniforme e centroidale, si sono confrontate tra di loro e con la stima consistente, fatta sulla base di creazione del campo, tramite l'errore di stima.
I confronti sono stati eseguiti in funzione del numero di funzioni radiali utilizzate nelle basi, come mostrato in fig. 3.6.

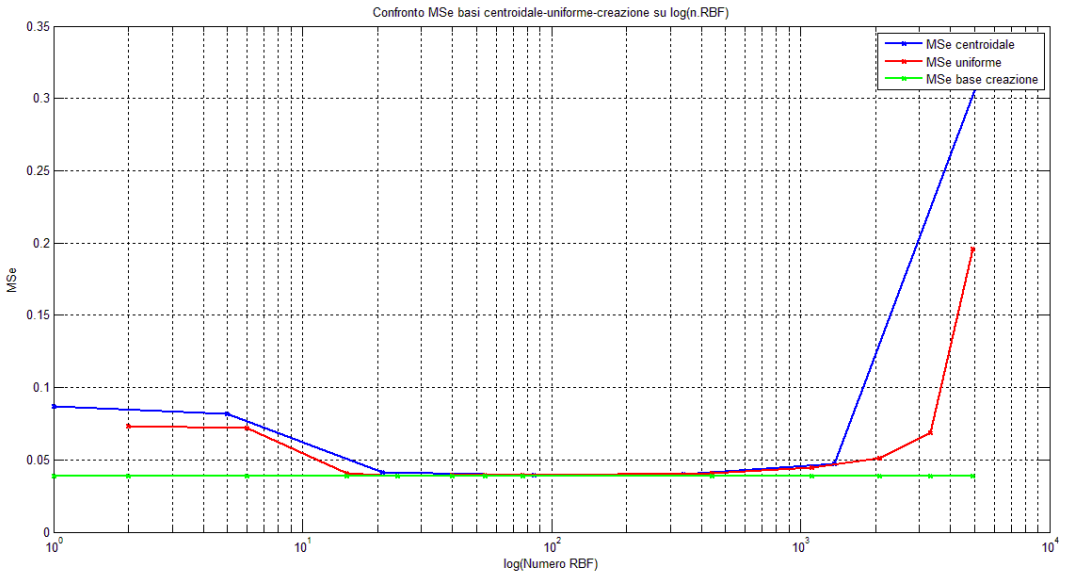


Figura 3.6: Confronto errori di stima

Le due basi hanno errore di stima minimo:

Uniforme: n. funzioni $= 24 \Longrightarrow$ MSe $= 0.0390$;

Centroidale: n. funzioni $= 85 \Longrightarrow$ MSe $= 0.0396$.

Si può notare, per un *campo poco variabile*, che la stima uniforme è migliore rispetto a quella centroidale in quanto utilizza un numero di basi inferiore per raggiungere lo stesso errore di stima. L'errore (*costante*) della stima consistente, riportato in figura per vari numeri di funzioni radiali per chiarezza grafica, ha valore:

$$MSe_{consistente} = 0.0389$$

Si vede quindi che con entrambe le basi si arriva molto vicini alla stima ottima del campo.

3.2 Campo ai vertici

Il secondo campo stimato è quello concentrato in due vertici della mappa (fig. 3.7):

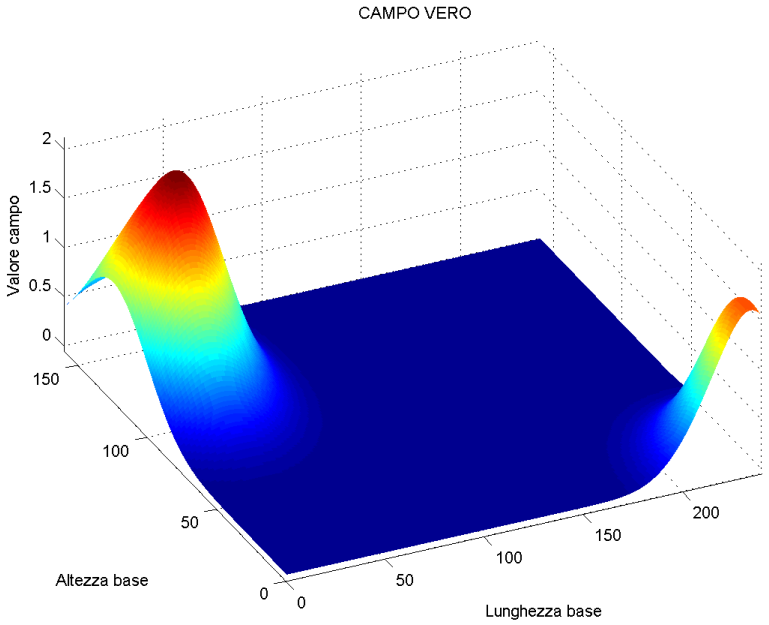


Figura 3.7: Campo da stimare

3.2.1 Stima con Base RBF Uniforme

L'evoluzione della stima con base RBF uniforme è presentata in fig. A.1 e fig. A.2.
I valori degli errori di stima sono in tab. 3.3 e la loro evoluzione è in fig. 3.8.

Livello	Errore di stima	Numero RBF	Varianza Max	Varianza Min	Varianza Media
1	0.2197	2	8.9799e-009	8.8703e-009	8.9251e-009
2	0.1098	6	2.9557e-008	2.5797e-008	2.7214e-008
3	0.0674	15	7.8484e-008	6.4742e-008	7.1986e-008
4	0.0582	24	1.4566e-007	1.0408e-007	1.2498e-007
5	0.0470	40	2.4291e-007	1.6859e-007	2.0292e-007
6	0.0433	54	3.4712e-007	2.2590e-007	2.9324e-007
7	0.0427	77	5.0794e-007	3.2186e-007	4.1707e-007
17	***0.0421***	***442***	4.9224e-006	1.4552e-006	2.6197e-006
27	0.0461	1107	1.8660e-005	3.3304e-006	7.3353e-006
37	0.0525	2072	/	/	/
47	0.0704	3337	/	/	/
57	0.1906	4902	/	/	/

Tabella 3.3: Errori di stima Base Uniforme - Varianze parametri

3.2.2 Stima con Base RBF Centroidale

L'evoluzione della stima con base RBF centroidale è presentata in fig. A.3.
In fig. 3.8 è presentato l'errore di stima i cui valori sono riportati in tab. 3.4.

Livello	Errore di stima	Numero RBF	Varianza Max	Varianza Min	Varianza Media
1	0.2383	1	4.2996e-009	4.2996e-009	4.2996e-009
2	0.1579	5	8.6467e-008	6.2744e-008	6.7728e-008
3	0.0677	21	4.6155e-007	2.3258e-007	3.2274e-007
4	0.0468	85	2.5816e-006	9.7151e-007	1.4128e-006
5	***0.0435***	***341***	1.6100e-005	3.5139e-006	6.1148e-006
6	0.0489	1365	1.1307e-004	1.1171e-005	2.8913e-005
7	0.3190	5461	/	/	/

Tabella 3.4: Errori di stima Base Centroidale - Varianze parametri

3.2.3 Confronto stime

I confronti sono stati eseguiti in funzione del numero di funzioni utilizzate nelle basi (fig. 3.8).

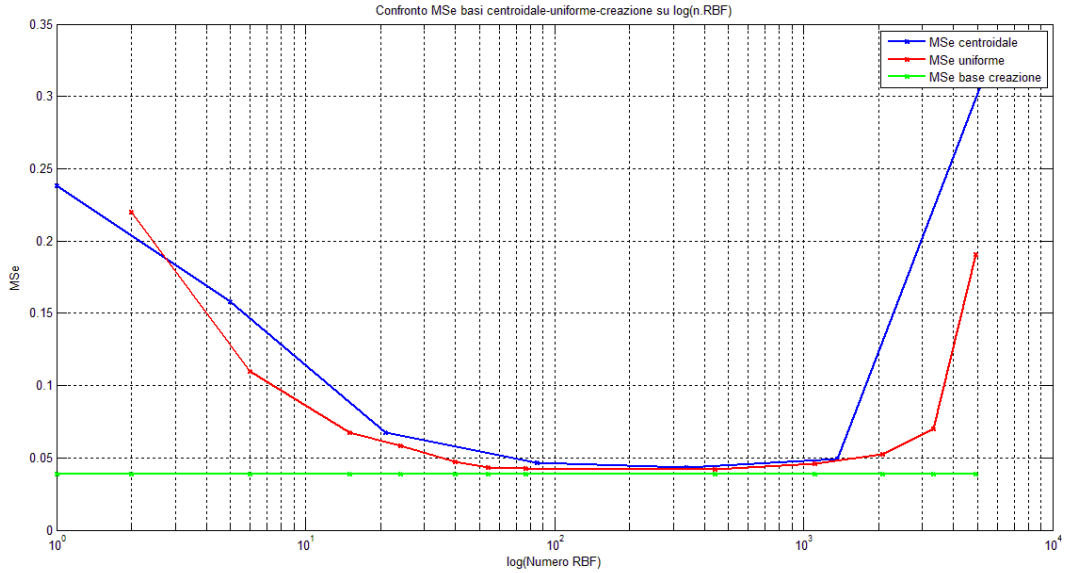


Figura 3.8: Confronto errori di stima

Si può notare, per un campo *concentrato in due vertici* della mappa, che la stima uniforme è migliore di quella centroidale per ogni concentrazione di funzioni delle basi. Ciò si può anche intuire dal fatto che una distribuzione uniforme delle funzioni, riesce a coprire meglio tutti i punti della mappa rispetto a quella centroidale, in particolare negli angoli della mappa, anche se il valore minimo dell'errore di stima è abbastanza lontano da quello ottimo. Una plausibile spiegazione del fenomeno è che ci sono molte funzioni, quelle posizionate al centro della mappa, che danno un contributo, seppur minimo, alla stima e quindi sono fonte di errore.
L'errore (*costante*) della stima consistente ha valore:

$$MSe_{consistente} = 0.0390$$

Le due basi hanno errore di stima minimo:

Uniforme: n. funzioni $= 442 \Longrightarrow$ MSe $= 0.0421$;

Centroidale: n. funzioni $= 341 \Longrightarrow$ MSe $= 0.0435$.

In particolare si vede che la base uniforme utilizza 100 funzioni in più della base centroidale ma ha un errore di stima più piccolo. A questo punto la scelta della base migliore va fatta, favorendo la base (uniforme) che produce un minor errore di stima, o favorendo la base (centroidale) che dal punto di vista computazionale è meno onerosa.

3.3 Campo variabile

Il terzo campo stimato è quello variabile (fig. 3.9):

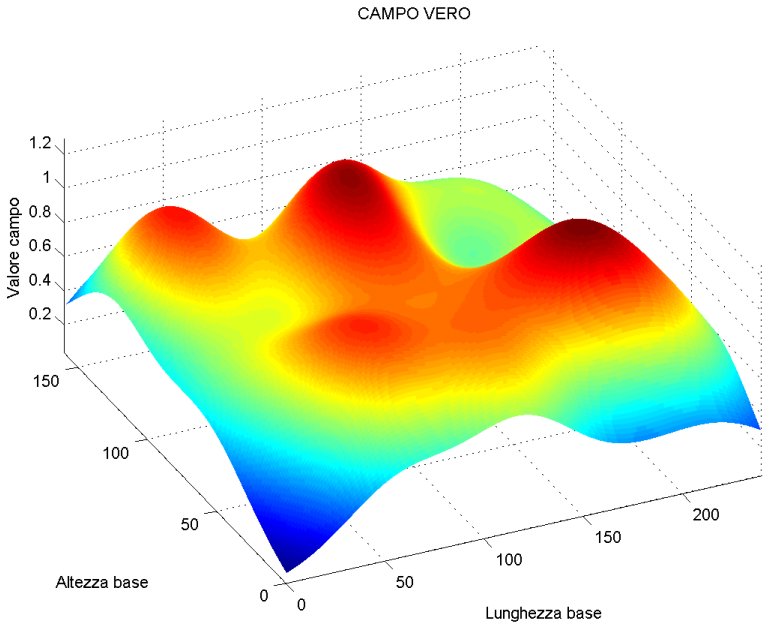


Figura 3.9: Campo da stimare

3.3.1 Stima con Base RBF Uniforme

L'evoluzione della stima con base RBF uniforme è presentata in fig. A.4 e fig. A.5.
In fig. 3.10 è riportata l'evoluzione dell'errore di stima i cui valori sono in tab. 3.5.

Livello	Errore di stima	Numero RBF	Varianza Max	Varianza Min	Varianza Media
1	0.0685	2	8.9999e-009	8.9544e-009	8.9772e-009
2	0.0540	6	2.9517e-008	2.5642e-008	2.7111e-008
3	0.0476	15	7.7370e-008	6.2989e-008	7.2111e-008
4	0.0415	24	1.4780e-007	1.0188e-007	1.2514e-007
5	***0.0394***	***40***	2.3176e-007	1.5669e-007	2.0471e-007
6	0.0401	54	3.4982e-007	2.1381e-007	2.9227e-007
7	0.0402	77	5.3228e-007	2.8800e-007	4.1169e-007
17	0.0412	442	/	/	/
27	0.0453	1107	/	/	/
37	0.0517	2072	/	/	/
47	0.0693	3337	/	/	/
57	0.1948	4902	/	/	/

Tabella 3.5: Errori di stima Base Uniforme - Varianze parametri

3.3.2 Stima con Base RBF Centroidale

L'evoluzione della stima con base RBF centroidale è presentata in fig. A.6. L'evoluzione dell'errore di stima è in fig. 3.10 e i valori sono riportati in tab. 3.6.

Livello	Errore di stima	Numero RBF	Varianza Max	Varianza Min	Varianza Media
1	0.0700	1	4.3234e-009	4.3234e-009	4.3234e-009
2	0.0578	5	8.5895e-008	6.2123e-008	6.7403e-008
3	0.0440	21	4.4302e-007	2.3936e-007	3.2564e-007
4	***0.0399***	***85***	2.5138e-006	9.0006e-007	1.4257e-006
5	0.0407	341	1.6563e-005	3.0702e-006	6.2046e-006
6	0.0474	1365	1.1334e-004	9.3749e-006	2.8904e-005
7	0.3196	5461	/	/	/

Tabella 3.6: Errori di stima Base Centroidale - Varianze parametri

3.3.3 Confronto stime

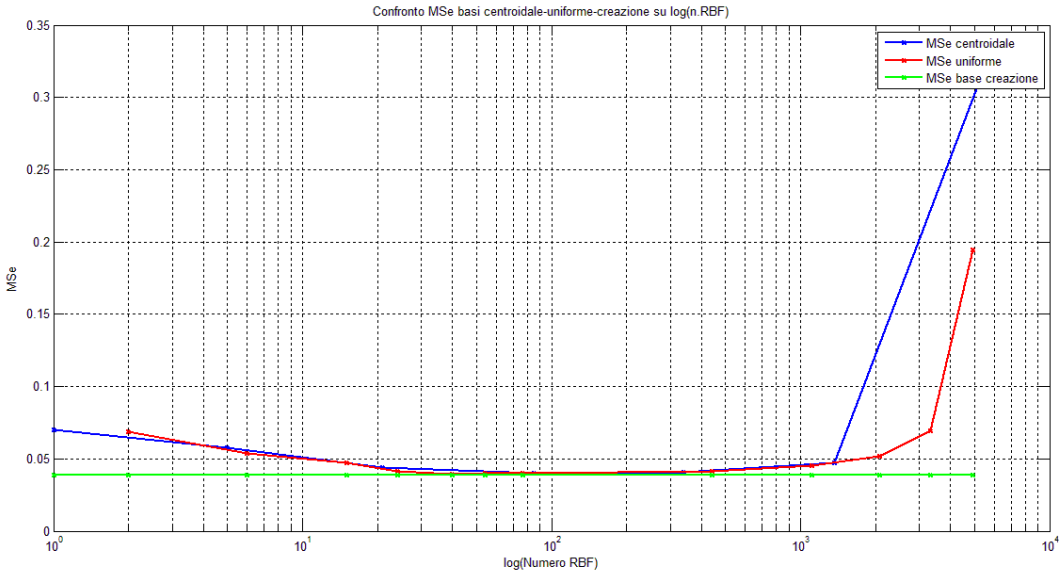


Figura 3.10: Confronto errori di stima

Dai confronti di fig. 3.10 si può notare che, per un campo *variabile* i cui valori di picco sono concentrati nell'area centrale, la stima uniforme è migliore di quella centroidale nei primi gradi di risoluzione (con poche funzioni radiali), in quanto la base centroidale ha una distribuzione di funzioni più dispersiva all'interno della mappa, mentre per i livelli successivi, le stime si equivalgono.
L'errore (*costante*) della stima consistente è:

$$MSe_{consistente} = 0.0390$$

I valori di errore minimi, riportati in seguito, sono molto vicini all'ottimo, ciò è intuibile dal fatto che, essendo il campo omogeneo nella mappa, entrambe le basi hanno una distribuzione che copre bene l'area e quindi producono una buona stima.

Uniforme: n. funzioni $= 40 \Longrightarrow$ MSe $= 0.0394$;

Centroidale: n. funzioni $= 85 \Longrightarrow$ MSe $= 0.0399$.

Si può quindi dire che la base uniforme è migliore di quella centroidale in quanto utilizza un minor numero di funzioni per avere la stessa stima.

3.4 Campo al centro della mappa

Il quarto campo stimato è quello concentrato al centro della mappa (fig. 3.11):

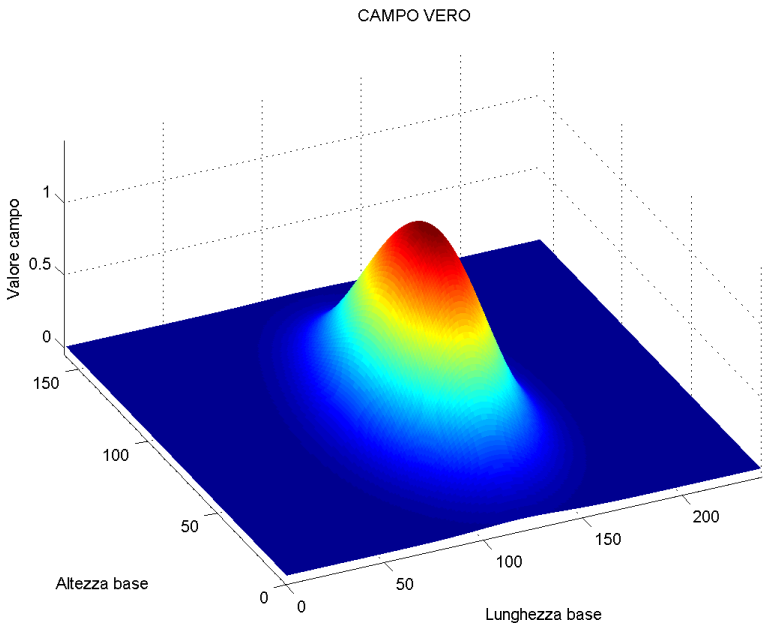


Figura 3.11: Campo da stimare

3.4.1 Stima con Base RBF Uniforme

L'evoluzione della stima con base RBF uniforme è presentata in fig. A.7 e fig. A.8.
L'evoluzione dell'errore di stima di fig. 3.12 ha miglioramento tra Ris.1 e Ris.3, dove la curva dell'errore è molto ripida, poi la stima peggiora in Ris.4, in quanto le funzioni della base sono ancora poche e hanno una distribuzione molto distante dal centro della mappa; l'errore di stima successivamente torna a decrescere fino al valore minimo di Ris.7 (il numero di funzioni cresce e la disposizione favorisce la stima) e successivamente risale per fenomeno di overfitting.
I valori degli errori di stima sono presentati in tab. 3.7.

Livello	Errore di stima	Numero RBF	Varianza Max	Varianza Min	Varianza Media
1	0.1003	2	8.9213e-009	8.8889e-009	8.9051e-009
2	0.0670	6	2.9505e-008	2.5107e-008	2.7193e-008
3	0.0415	15	7.9783e-008	6.3532e-008	7.2688e-008
4	0.0440	24	1.4109e-007	1.0421e-007	1.2458e-007
5	0.0400	40	2.3138e-007	1.7610e-007	2.0464e-007
6	0.0395	54	3.4927e-007	2.3722e-007	2.9214e-007
7	***0.0392***	***77***	4.8714e-007	3.2711e-007	4.0843e-007
17	0.0406	442	5.2134e-006	1.3873e-006	2.6099e-006
27	0.0448	1107	1.9435e-005	2.6021e-006	7.3077e-006
37	0.0513	2072	/	/	/
47	0.0692	3337	/	/	/
57	0.1940	4902	/	/	/

Tabella 3.7: Errori di stima Base Uniforme - Varianze parametri

3.4.2 Stima con Base RBF Centroidale

L'evoluzione della stima con base RBF centroidale è in fig. A.12. L'evoluzione dell'errore di stima di fig. 3.12 ha valori riportati in tab. 3.8.

Livello	Errore di stima	Numero RBF	Varianza Max	Varianza Min	Varianza Media
1	0.0972	1	4.3273e-009	4.3273e-009	4.3273e-009
2	0.0730	5	8.7222e-008	6.1884e-008	6.7951e-008
3	0.0533	21	4.4829e-007	2.2285e-007	3.2378e-007
4	0.0411	85	2.6114e-006	8.8584e-007	1.4172e-006
5	***0.0403***	***341***	1.6539e-005	3.2733e-006	6.1131e-006
6	0.0472	1365	1.1383e-004	1.1951e-005	2.8618e-005
7	0.3230	5461	/	/	/

Tabella 3.8: Errori di stima Base Centroidale - Varianze parametri

3.4.3 Confronto stime

I confronti degli errori di stima, eseguiti in corrispondenza del numero di funzioni radiali utilizzate nelle basi, è mostrato in fig. 3.12.

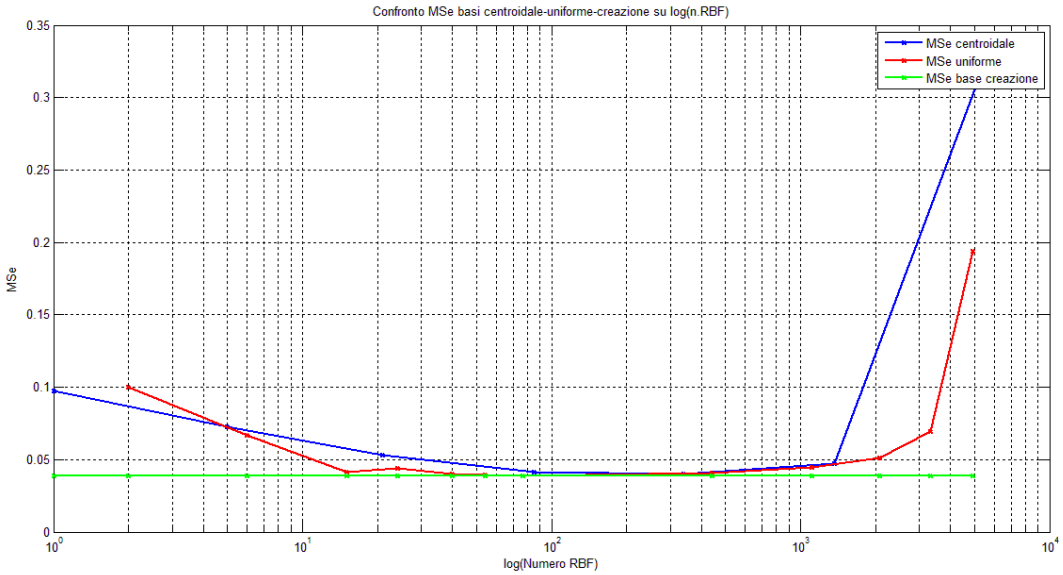


Figura 3.12: Confronto errori di stima

Si può notare, per un campo *concentrato al centro* della mappa, che la base centroidale è migliore di quella uniforme nelle prime stime, in quanto ha una gaussiana centrata proprio in mezzo alla mappa, mentre l'uniforme nel primo grado di risoluzione presenta due funzioni poste simmetricamente rispetto al centro del campo.
L'errore (*costante*) della stima consistente, riportato in figura per vari numeri di funzioni radiali per chiarezza grafica, ha valore:

$$MSe_{consistente} = 0.0390$$

Si nota inoltre che le due basi hanno errore di stima minimo per un numero di basi molto differente:

Uniforme: n. funzioni $= 77 \Longrightarrow$ MSe $= 0.0392$;

Centroidale: n. funzioni $= 341 \Longrightarrow$ MSe $= 0.0403$.

Ciò si traduce nel fatto che la stima con base uniforme è migliore di quella centroidale in quanto ha errore di stima più piccolo, con un numero di funzioni radiali minore.

3.5 Campo su un lato della mappa

Il quinto campo stimato è quello concentrato su un lato della mappa, in questo caso è posizionato sul lato lungo (fig. 3.13):

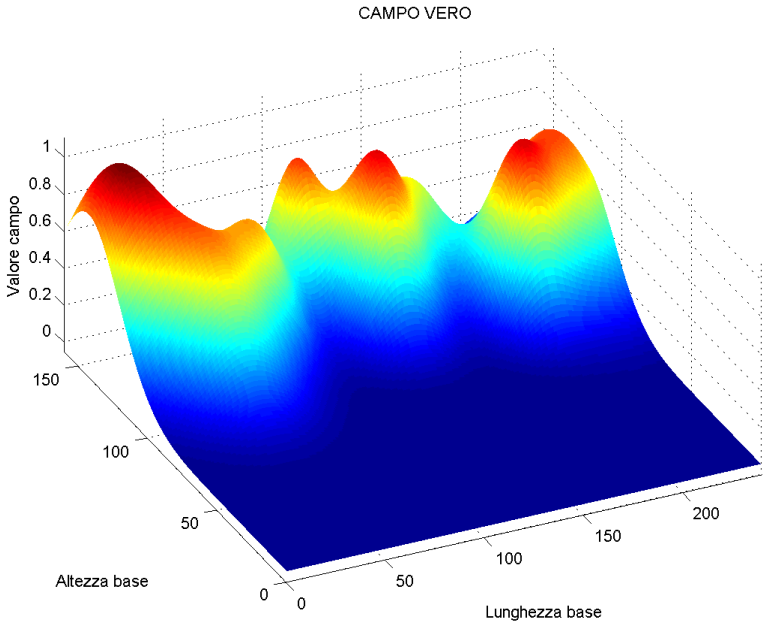


Figura 3.13: Campo da stimare

3.5.1 Stima con Base RBF Uniforme

L'evoluzione della stima con base RBF uniforme è presentata in fig. A.10 e fig. A.11.
L'errore di stima è quello di fig. 3.14. I valori di tale errore sono presentati in tab. 3.9.

Livello	Errore di stima	Numero RBF	Varianza Max	Varianza Min	Varianza Media
1	0.1281	2	9.0516e-009	8.8796e-009	8.9656e-009
2	0.0680	6	2.8843e-008	2.5995e-008	2.6973e-008
3	0.0503	15	7.9479e-008	6.4429e-008	7.1258e-008
4	0.0472	24	1.4602e-007	1.0407e-007	1.2449e-007
5	0.0406	40	2.3284e-007	1.6994e-007	2.0709e-007
6	0.0406	54	3.6256e-007	2.3518e-007	2.9543e-007
7	***0.0397***	***77***	4.9107e-007	3.3420e-007	4.1070e-007
17	0.0407	442	4.5807e-006	1.5759e-006	2.6271e-006
27	0.0448	1107	/	/	/
37	0.0513	2072	/	/	/
47	0.0692	3337	/	/	/
57	0.1890	4902	/	/	/

Tabella 3.9: Errori di stima Base Uniforme - Varianze parametri

3.5.2 Stima con Base RBF Centroidale

L'evoluzione della stima con base RBF centroidale è in fig. A.9. L'evoluzione dell'errore di stima è in fig. 3.14 e i suoi valori sono riportati in tab. 3.10.

Livello	Errore di stima	Numero RBF	Varianza Max	Varianza Min	Varianza Media
1	0.1276	1	4.2930e-009	4.2930e-009	4.2930e-009
2	0.0755	5	8.5337e-008	6.2410e-008	6.7374e-008
3	0.0509	21	4.5046e-007	2.3124e-007	3.2165e-007
4	***0.0407***	***85***	2.6538e-006	8.8898e-007	1.4189e-006
5	0.0408	341	1.6742e-005	3.5762e-006	6.1642e-006
6	0.0473	1365	/	/	/
7	0.3282	5461	/	/	/

Tabella 3.10: Errori di stima Base Centroidale - Varianze parametri

3.5.3 Confronto stime

I confronti degli errori di stima sono mostrati in fig. 3.14.

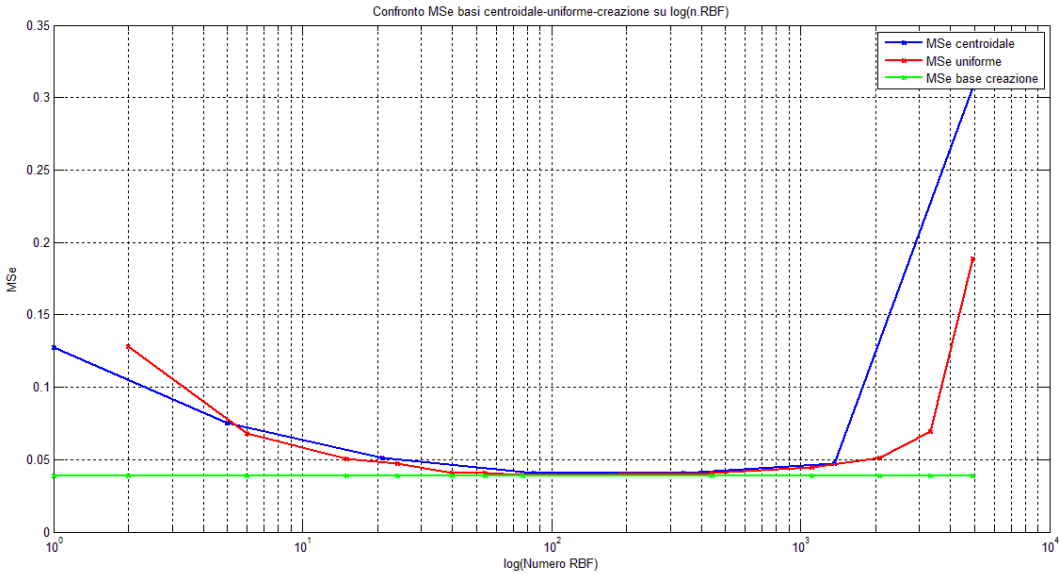


Figura 3.14: Confronto errori di stima

Si può notare, per un campo *concentrato su un lato* della mappa, che la stima centroidale e quella uniforme si equivalgono a meno di piccole differenze dell'errore di stima.
L'errore (*costante*) della stima consistente, riportato in figura per vari numeri di funzioni radiali per chiarezza grafica, ha valore:

$$MSe_{consistente} = 0.0391$$

Si nota che le due basi hanno errori di stima minimi adiacenti, per un numero di basi molto vicino:

Uniforme: n. funzioni $= 77 \Longrightarrow$ MSe $= 0.0397$;

Centroidale: n. funzioni $= 85 \Longrightarrow$ MSe $= 0.0407$.

Tale considerazione porta a dire che in questo specifico caso le due basi di stima tendono ad equivalersi.

3.6 Confronto stima dei campi

	Errori minimi di Stima		
Campo	Base RBF Uniforme (#RBF)	Base RBF Centroidale (#RBF)	Base RBF Creazione
1	0.0390 (24)	0.0396 (85)	0.0389
2	0.0421 (442)	0.0435 (341)	0.0390
3	0.0394 (40)	0.0399 (85)	0.0390
4	0.0392 (77)	0.0403 (341)	0.0390
5	0.0397 (77)	0.0407 (85)	0.0391

Tabella 3.11: Errori minimi di stima

Come si può vedere dai dati riassuntivi riportati in tab. 3.11, per campi distribuiti in maniera quasi uniforme sull'intera area, le basi utilizzate restituiscono una stima valida, con errore prossimo a quello ottimo.
Per campi concentrati unicamente in zone limitate, invece, la stima risulta essere peggiore o comunque a parità d'errore richiede una quantità RBF maggiore. Questo si può spiegare in quanto un gran numero di funzioni, quelle corrispondenti alle zone nulle del campo, risultano essere "sprecate".
Si evince, inoltre, che per avvicinarsi al caso consistente, occorrono un numero di funzioni elevato. Questo, come si può notare nelle tabelle, da 3.1 a 3.10, comporta una varianza dei parametri elevata, infatti all'aumentare del numero di parametri, ovvero il numero di RBF, aumenta anche la varianza calcolata con (1.13) degli stessi.
Il miglioramento della stima, quindi, ha lo svantaggio di una maggior incertezza nella determinazione dei parametri portando ad ottenere un modello poco utilizzabile come predittore.
In conclusone, in fase di scelta del numero di funzioni con cui creare la base di stima, occorre tener presente questo fatto e arrivare ad un compromesso.

Parte II

Stima centralizzata con basi mobili

Capitolo 4

Stima PEM con direzione di Newton

Introduzione

Nella prima parte del progetto sono state analizzate le prestazioni di uno stimatore dei soli pesi delle funzioni di base del modello neurale utilizzato. Mantenendo costante la base, ovvero mantenendo fissi centri e varianze delle funzioni RBF e stimando unicamente l'ampiezza di quest'ultime, si ottengono dei buoni risultati ma limitati dalla struttura "statica" del modello. Se si riuscisse a modificare maggiormente la sua struttura, si potrebbero ottenere risultati migliori, anche con un numero di funzioni inferiore. Potendo infatti "muovere" le basi sulla mappa, potremmo posizionare le funzioni in modo realmente ottimale così da ottenere delle stime più precise.

In precedenza, lo spazio dei parametri comprendeva unicamente le ampiezze α_k, ora andremo a stimare anche tutti i fattori caratteristici delle funzioni RBF quali ampiezze, centri e varianze. Con questa nuova parametrizzazione, il modello non risulta essere più lineare in θ e non è più possibile applicare le tecniche di stima viste in precedenza. Da una stima ai minimi quadrati siamo dunque passati ad utilizzare la nota tecnica di minimizzazione dell'errore di predizione PEM. Non potendo comunque ricavare una forma esatta per lo stimatore, per via della non linearità del modello, abbiamo utilizzato l'algoritmo iterativo di Newton adattato per la ricerca del minimo della funzione costo.
In questo capitolo analizzeremo l'implementazione e le proprietà di tale algoritmo applicato a modelli RBF. Inizialmente si supporrà di avere già a disposizione un certo numero di misure in vari punti del campo con le quali sarà effettuata la stima. Sotto tali ipotesi studieremo le prestazioni dell'algoritmo in termini di velocità di convergenza e qualità della stima in funzione del numero di iterazioni e del tipo di condizioni iniziali imposte.

4.1 Modello RBF

Nella parte precedente la classe parametrica di modelli era:

$$\mathcal{M} = \{M(\theta) : h(t) = f(t,\theta) + w(t); \theta = [\alpha_1, ..., \alpha_p]^T \in \Theta\} \tag{4.1}$$

In cui $f(t,\theta)$ è del tipo 1.8, $w(t)$ è rumore bianco[1] e $\Theta \subseteq \mathbb{R}^p$. In tale modello l'unico parametro libero che si andava a stimare era il vettore dei combinatori delle funzioni RBF, in altre parole le relative ampiezze. Questo limita di molto la struttura del modello e di conseguenza anche le prestazioni della stima. Per questo motivo abbiamo pensato di allargare la classe parametrica $\mathcal{M}$ prendendo come parametri liberi anche i centri e le varianze delle RBF.

$$\Theta = \{\theta = [\alpha_1..\alpha_p, x_1..x_p, y_1..y_p, \sigma_1..\sigma_p]^T; \theta \in \mathbb{R}^{4p}\} \tag{4.2}$$

In questo modo, andando a cambiare la "struttura" del modello è possibile ottenere un gran numero di campi di struttura diversa, anche diminuendo il numero di funzioni.
Così facendo, supponendo che il campo vero sia esso stesso combinazione lineare di funzioni RBF gaussiane (il campo vero $\mathcal{C}$ appartiene alla classe parametrica $\mathcal{M}$ allora $\exists\ \theta_0$ tale che $M(\theta_0) = \mathcal{C}$), sarebbe possibile, almeno teoricamente, stimare esattamente il campo, ottenendo uno stimatore consistente (Teorema.1). Nelle condizioni della parte I, per avere lo stesso risultato avremmo dovuto usare un numero di funzioni molto più elevato.

Nella classe parametrica descritta da (4.2), l'unico parametro che dovrà essere "imposto a priori" è il numero di basi, ovvero di funzioni RBF. Questa scelta è stata dettata dal fatto che i metodi più comuni di stima dell'ordine del modello necessitano di un set di dati già a disposizione. Ad esempio, le tecniche basate sull'analisi preliminare sui dati, il confronto fra vari modelli (FPE,AIC,MDL), la validazione di un modello [14], o i metodi che utilizzano alberi di ricorsione [3], hanno bisogno di un insieme di dati su cui lavorare. Questi strumenti sono quindi incompatibili con l'obbiettivo finale che ci prefiggiamo, ovvero la stima ricorsiva sulle misure. Per questo motivo il numero di funzioni p viene scelto a priori. Supponendo di avere qualche informazione sulla natura del campo da stimare, in termini della sua variabilità, possiamo scegliere il p ottimo in base alle considerazioni ed ai risultati mostrati nella sezione precedente, tenendo comunque ben presente che all'aumentare del numero di parametri, o più in generale della complessità del modello, la varianza della stima risulta maggiore ottenendo dunque risultati meno precisi in termini di predizione.

4.2 Stimatore PEM: algoritmo Quasi-Newton

Il metodo è basato sulla minimizzazione dell'errore quadratico medio di predizione (PEM). Data quindi una classe parametrica di modelli $\mathcal{M} = \{M(\theta); \theta \in \Theta\}$ con Θ del tipo (4.2), e una serie di dati rumorosi $h^N := \{h(c(t)); t = 1, 2, ..., N\}$[2] in cui $c(t) = (x(t), y(t))$ corrisponde al punto della mappa in cui viene eseguita la misura, si cerca il valore di θ che minimizza la funzione:

$$V_N(\theta) := \frac{1}{N}\sum_{t=1}^{N} \varepsilon_\theta^2(t) = \frac{1}{N}\sum_{t=1}^{N}(h(t) - \hat{h}_\theta(t))^2 \tag{4.3}$$

$$\hat{\theta} = \underset{\theta\in\Theta}{\text{Argmin}} V_N(\theta) \tag{4.4}$$

dove, per un generico valore di θ, $\hat{h}_\theta(t)$ è il predittore a minima varianza all'istante $t-1$ dell'uscita successiva, $h(t)$. Per la classe di modelli utilizzata, applicando il principio delle proiezioni ortogonali di Wiener [17], notando che $w \perp y^t\ \forall\ t$, il predittore è semplicemente:

$$\hat{h}_\theta(t) = f(t,\theta) = \sum_{i=1}^{p} \alpha_k \cdot \phi_k(t, x_i, y_i, \sigma_i) = \sum_{i=1}^{p} \alpha_i exp\Big(-\frac{(x(t)-x_i)^2 + (y(t)-y_i)^2}{2{\sigma_i}^2}\Big) \tag{4.5}$$

in cui $(x(t), y(t))$ corrispondono alle coordinate sulla mappa del punto in cui viene eseguita la misura $h(t)$. Possiamo osservare in (4.5) che il predittore dipende unicamente da θ e dal punto $c(t)$ in cui la misura $h(t)$ che si vuole predire è calcolata.

[1] Nella parte precedente, per le varie simulazioni, avevamo settato la deviazione standard dell'errore di misura a 0,2. In questa sezione invece, per ottenere risultati più chiari, si è preferito abbassarla a 0,01.

[2] Il numero di misure acquisite nelle simulazioni successive sarà pari a 1000.

4.2.1 Consistenza asintotica

L'obbiettivo del processo di identificazione è quello di ottenere, dai dati misurati, i valori "veri" del modello che descrivono i dati. E' noto da [18] e [14] che, se il modello vero $\mathcal{C}$ è lineare nei parametri e appartiene alla classe di modelli $\mathcal{M}$, si ha convergenza, per N tendente all'infinito, della stima al parametro vero:

$$\lim_{N\to\infty} \hat{\theta} = \theta_0 \quad \text{con probabilità uno} \tag{4.6}$$

Teorema 1 (Consistenza logica)**.** *Si assuma che*

1. *I dati $h^N := \{h(t); t = 1, 2, ..., N\}$ siano generati da un processo stazionario, ergodico del secondo ordine;*
2. *Il modello parametrico $M(\theta)$ dipenda dal parametro in modo differenziabile;*
3. *La cifra di merito sia una funzione quadratica dell'errore di predizione, ad esempio del tipo (4.3);*
4. *Il processo "vero" che genera i dati sia descritto da un modello che appartiene alla classe parametrica $\mathcal{M}$ dei modelli scelti per l'identificazione, ovvero esiste $\theta_0 \in \Theta$ tale che $\mathbf{h} \sim M(\theta_0) \in \mathcal{M}$;*
5. *L'errore di predizione $\varepsilon_\theta(t)$ sia calcolato mediante il predittore lineare a minima varianza $\hat{h}_\theta(t)$.*

Allora, se la classe parametrica dei modelli $\mathcal{M} = \{M(\theta); \theta \in \Theta\}$ è identificabile localmente in $\theta = \theta_0$, si ha

$$\lim_{N\to\infty} \hat{\theta} = \theta_0$$

con probabilità uno. In altri termini, lo stimatore PEM è ***consistente****.*

Nel nostro caso il modello non è lineare nei parametri ed inoltre, dato che stiamo facendo un'operazione di stima del campo misurato non possiamo sapere se il modello vero che lo rappresenta appartenga o meno alla classe di modelli che noi usiamo. Ciò nonostante c'è un teorema che ci consente di dire che il metodo PEM fornisce, asintoticamente, i parametri del modello $M(\hat{\theta})$ che meglio approssima il campo vero [13].

Teorema 2. *Sia dato un certo modello $\mathcal{S} := h(t) = f_{\mathcal{S}}(t, h^{t-1}) + w(t)$, dove $w(t)$ è rumore bianco, un insieme di misure $\mathcal{X} := \{h^t\}$, ed una certa classe parametrica $\mathcal{M} = \{M(\theta); \theta \in \Theta\}$ differenziabile rispetto θ per ogni $\theta \in \Theta$ insieme compatto. Sia inoltre $V_N(\theta)$ del tipo (4.3) ed esista il limite $\lim_{N\to\infty} V_N(\theta)$.*
Allora *la stima $\hat{\theta}$ ottenuta dal metodo PEM converge con probabilità uno all'insieme $\mathcal{M}*$ dei modelli che meglio approssimano il sistema $\mathcal{S}$ sotto la condizione sperimentale $\mathcal{X}$.*

Questo teorema ci garantisce che, pur non essendo nelle usuali condizioni di consistenza asintotica, ovvero modello lineare e modello vero appartenente alla classe parametrica, di ottenere la stima migliore possibile per la specifica classe parametrica utilizzata. La scelta di $\mathcal{M}$ diventa quindi centrale al fine di ottenere una buona stima.
La scelta del modello RBF per la rappresentazione del campo è stata fatta proprio a tal fine: utilizzare una classe parametrica che si adattasse bene al tipo di campo oggetto della stima.

4.2.2 Calcolo dello stimatore PEM: algoritmo di Quasi-Newton

Il metodo PEM presenta però anche degli svantaggi, infatti la minimizzazione di (4.3) si può fare esplicitamente solo nel caso di modelli a predittore lineare nei parametri. Nella maggior parte dei casi, come nel nostro, non avendo a disposizione un modello lineare, si è costretti a ricorre ad algoritmi iterativi di ottimizzazione [14], [2]. Tali metodi aggiornano ricorsivamente la stima del punto di minimo di $V_N(\theta)$ secondo una "direzione" $f(k-1)$ dipendente dal valore di $V_N(\theta)$ al passo precedente.

$$\theta_k = \theta_{k-1} + \alpha f(k-1) \tag{4.7}$$

dove α è una costante tale da ottenere un adeguato decremento di $V_N(\theta)$ (questo argomento verrà ripreso nella sezione 4.2.6).
La funzione $f(k)$ può essere di vario tipo:

- Una funzione unicamente di k;
- Una funzione dipendente da $V_N(\theta)$ o dal suo gradiente;
- Una funzione dipendente dal gradiente e dall'Hessiana (derivata seconda) di $V_N(\theta)$.

Una delle soluzioni più utilizzate ed efficienti è la terza, ed in particolare *l'algoritmo di Newton* [18]. In quest'ultimo, la funzione $f(k)$, chiamata *passo di Newton*, risulta essere:

$$f(k) = -\left[V_N''(\theta_k)\right]^{-1} V_N'(\theta_k) \tag{4.8}$$

Utilizzando come funzione costo la (4.3), il suo gradiente e l'Hessiana sono:

$$V_N'(\theta) = -\frac{1}{N}\sum_{t=1}^{N} \psi_\theta(t)\varepsilon_\theta(t) \tag{4.9}$$

$$V_N''(\theta) = \frac{1}{N}\sum_{t=1}^{N} \psi_\theta(t)\psi_\theta^T(t) - \psi_\theta'(t)\varepsilon_\theta(t) \tag{4.10}$$

dove $\psi_\theta(t)$ corrisponde al gradiente del predittore $\hat{h}_\theta(t)$:

$$\psi_\theta(t) = \frac{\partial \hat{h}_\theta(t)}{\partial \theta} \tag{4.11}$$

Da quanto detto, l'iterazione di ricerca del minimo, al generico passo k, è espressa da:

$$\theta_{k+1} = \theta_k - \left[V_N''(\theta_k)\right]^{-1} V_N'(\theta_k) \tag{4.12}$$

Il calcolo di $\psi_\theta'(t)$ (e quindi dell'Hessiana) può risultare molto arduodifficile e costoso in termini di complessità computazionale. Inoltre, se si è lontani da un minimo, $V_N''(\theta)$ potrebbe risultare indefinita o addirittura definita negativa, portando così θ verso un punto di crescita della funzione $V_N(\theta)$. Ciò che si fa è quindi utilizzare una *stima* della derivata seconda di $V_N(\theta)$ introducendo delle approssimazioni che la rendano almeno sempre semidefinita positiva. Tali algoritmi vengono chiamati di *Quasi-Newton* [18], [14]. Ipotizzando θ vicino ad un punto di minimo è possibile trascurare il termine con le derivate seconde di $\varepsilon_\theta(t)$, ottenendo così la *Quasi Hessiana*:

$$H_\theta := \frac{1}{N}\sum_{t=1}^{N} \psi_\theta(t)\psi_\theta^T(t) \simeq V_N''(\theta) \tag{4.13}$$

Osserviamo che tale matrice, essendo somma di termini maggiori di zero, è sempre almeno semidefinita positiva il che garantisce la convergenza ad un punto stazionario.
Riassumendo il tutto si ottiene il seguente algoritmo:

Algoritmo 1 (Quasi-Newton)**.** *Data una stringa di dati h^N, la classe di modelli $M(\theta)$, un valore iniziale θ_0 e la stima θ_k alla k-esima iterazione,*

1. *Si calcolano la stringa degli errori di predizione $\varepsilon_{\theta_k} = [\varepsilon_{\theta_k}(1)...\varepsilon_{\theta_k}(N)]^T$*

   $$\varepsilon_{\theta_k}(t) = h(t) - \hat{h}_{\theta_k}(t) \quad ; \quad t = 1, ..., N \tag{4.14}$$

   prendendo condizioni iniziali arbitrarie (successivamente riprenderemo questo problema nella sezione 4.2.5).

2. *Si calcola la stringa dei gradienti $\Psi_{\theta_k} = [\psi_{\theta_k}(1)...\psi_{\theta_k}(N)] \in \mathbb{R}^{4p\times N}$ del predittore per il modello $M(\theta_k)$.*

3. *Si calcola la matrice pseudo-Hessiana*

   $$H_{\theta_k} := \sum_{t=1}^{N} \psi_{\theta_k}(t)\psi_{\theta_k}(t)^T = \Psi_{\theta_k}\Psi_{\theta_k}^T \tag{4.15}$$

4. *Si aggiorna θ_k mediante la,*

   $$\theta_k = \theta_{k-1} - H_{\theta_k}^{-1}\Psi_{\theta_k}\varepsilon_{\theta_k} \tag{4.16}$$

5. *si torna al passo 1) ponendo $\theta_k = \theta_{k+1}$*

Contestualizziamo ora al modello RBF.

4.2.3 Gradiente dell'errore di predizione

Calcoliamo il gradiente del predittore (4.5).

$$\begin{aligned}\psi_\theta(t) &= \frac{\partial \hat{h}_\theta(t)}{\partial \theta} \\ &= \left[\frac{\partial \hat{h}_\theta(t)}{\partial \alpha_1}...\frac{\partial \hat{h}_\theta(t)}{\partial \alpha_p}\frac{\partial \hat{h}_\theta(t)}{\partial x_1}...\frac{\partial \hat{h}_\theta(t)}{\partial x_p}\frac{\partial \hat{h}_\theta(t)}{\partial y_1}...\frac{\partial \hat{h}_\theta(t)}{\partial y_p}\frac{\partial \hat{h}_\theta(t)}{\partial \sigma_1}...\frac{\partial \hat{h}_\theta(t)}{\partial \sigma_p}\right]^T \end{aligned} \tag{4.17}$$

$$\frac{\partial \hat{h}_\theta(t)}{\partial \alpha_i} = exp\left\{-\frac{1}{2\sigma_i^2}\left[(x_t - x_i)^2 + (y_t - y_i)^2\right]\right\} \tag{4.18}$$

$$\frac{\partial \hat{h}_\theta(t)}{\partial x_i} = \alpha_i exp\left\{-\frac{1}{2\sigma_i^2}\left[(x_t - x_i)^2 + (y_t - y_i)^2\right]\right\}\left(\frac{x_t - x_i}{\sigma_i^2}\right) \tag{4.19}$$

$$\frac{\partial \hat{h}_\theta(t)}{\partial y_i} = \alpha_i exp\left\{-\frac{1}{2\sigma_i^2}\left[(x_t - x_i)^2 + (y_t - y_i)^2\right]\right\}\left(\frac{y_t - y_i}{\sigma_i^2}\right) \tag{4.20}$$

$$\frac{\partial \hat{h}_\theta(t)}{\partial \sigma_i} = \alpha_i exp\left\{-\frac{1}{2\sigma_i^2}\left[(x_t - x_i)^2 + (y_t - y_i)^2\right]\right\}\left(\frac{(x_t - x_i)^2 + (y_t - y_i)^2}{2\sigma_i^4}\right) \tag{4.21}$$

4.2.4 Pseudo Hessiana regolarizzata

Affinché $f(k)$ (4.8) sia una direzione di discesa per $V_N(\theta)$, deve essere tale che da θ_k corrente a θ_{k+1}, secondo tale direzione, si verifichi $V_N(\theta_{k+1}) < V_N(\theta_k)$. Matematicamente, $f(k)$ è una *direzione discendente* da θ_k, se la derivata direzionale di $V_N(\theta_k)$ lungo $f(k)$ è negativa:

$$V_N'(k)^T f(k) < 0 \tag{4.22}$$

Considerando la direzione data dal passo di Newton in (4.8) si ottiene la condizione

$$-V_N'(k)^T \left[V_N''(\theta_k)\right]^{-1} V_N'(\theta_k) < 0 \tag{4.23}$$

che è vera se $[V_N''(\theta_k)]^{-1}$ o, equivalentemente, se $H(\theta_k)$ è definita positiva. Risulta quindi fondamentale renderla tale anche quando non lo è.
Inoltre, nell'Algoritmo 1 è necessario invertire la matrice pseudo-Hessiana H_{θ_k} e, affinché tale inversione sia possibile, Ψ_{θ_k} deve avere rango pieno. Specialmente durante le prime iterazioni la H_{θ_k} può risultare singolare o mal condizionata portando ad un overflow nel calcolo dell'inversa [22]. Ciò che si fa è quindi usare un fattore di regolarizzazione in modo che la matrice risulti sempre definita positiva e quindi anche invertibile:

$$P_{\theta_k} = [H_{\theta_k} + \delta(k)I]^{-1} \tag{4.24}$$

L'introduzione di tale fattore comporta un ulteriore "errore" di approssimazione nel calcolo dell'inversa dell'Hessiana. Per minimizzare tale errore esistono vari metodi di regolarizzazione [22], [8], noi abbiamo preso $\delta(k)$ pari ad una funzione che tendesse a zero con l'aumentare del numero di iterazioni.

$$\delta(k) = 0.001k^{-2} \tag{4.25}$$

Come viene suggerito in [6], $\delta(k)$ dev'essere tale che $\delta(k) = 0$ se H_{θ_k} è definita positiva, altrimenti $\delta(k) > 0$ sufficientemente grande affinché H_{θ_k} sia definita positiva. Il fattore "decrescente" (4.25) (fig. 4.1), fa sì che dopo un certo numero di iterazioni, quando l'Hessiana è singolare con meno probabilità, il fattore di regolarizzazione sia più basso, cosicché l'errore di approssimazione sia minore.

4.2.5 Problema del minimo locale e condizioni iniziali

Ogni algoritmo di minimizzazione iterativo di questo tipo, sfortunatamente, assicura unicamente la convergenza ad un *minimo locale* [6], mentre a noi interessa il *minimo globale* della funzione $V_N(\theta)$.

Teorema 3. *Sia data $V_N'(\theta)$ continuamente differenziabile in $\Theta \subset \mathbb{R}^p$. Assumendo che esistano dei punti $\theta_* \in \mathbb{R}^p$ (minimi locali) e $r, \beta > 0$, tali che:*

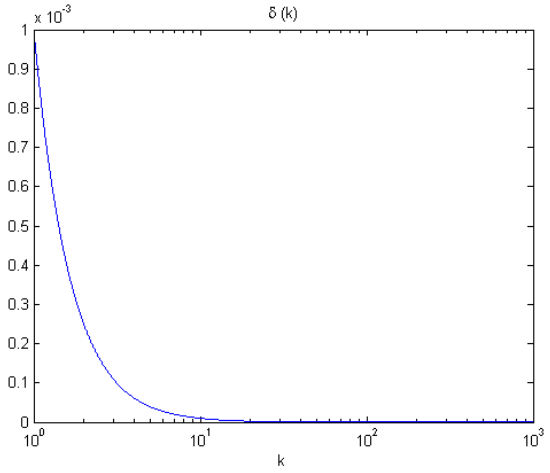


Figura 4.1: $\delta(k)$

- *l'intorno di* θ_* *di raggio* r, $N(\theta_*, r) = \{\bar{\theta} \in \mathbb{R}^p : \|\bar{\theta} - \theta\| < r\} \subset \Theta$;
- $V'_N(\theta_*) = 0$;
- $H(\bar{\theta})^{-1}$ *esiste e* $\|H(\bar{\theta})^{-1}\| \leqslant \beta$

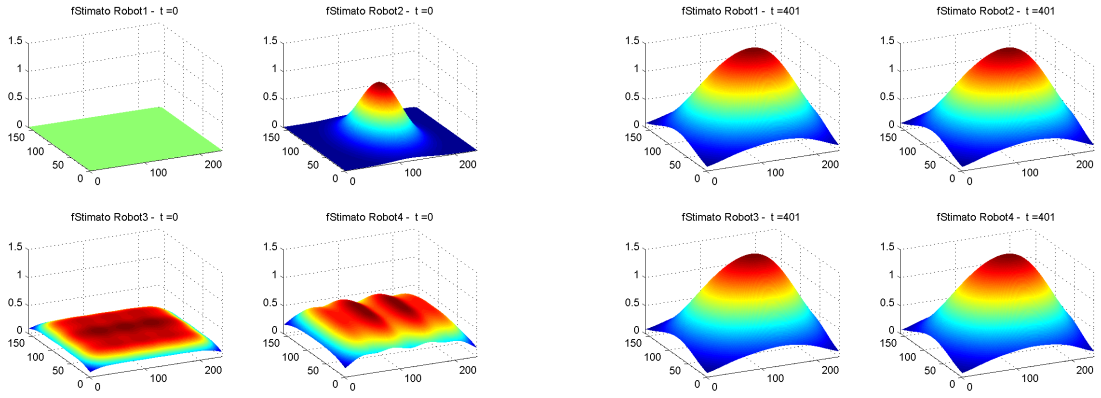
Allora esiste un $\varepsilon > 0$ *per cui per ogni* $\theta_0 \in N(\theta_*, \varepsilon)$ *la sequenza* $\theta_1, \theta_2, ...$ *generata da (4.12) per* $k = 0, 1, ..., N$, *è ben definita, converge a* θ_* *e soddisfa*

$$\|\theta_{k+1} - \theta_*\| \leqslant \beta \|\theta_k - \theta_*\|^2, \quad k = 0, 1, ..., N \tag{4.26}$$

Dall'analisi di quest'ultimo teorema possiamo trarre due fondamentali risultati. Il primo è che il rate di convergenza dell'algoritmo di Newton è *q-quadratico*, il secondo è che la convergenza vale solo in un determinato intorno del punto di minimo.
Questo porta al punto critico della scelta delle condizioni iniziali dell'algoritmo. Se si prendono delle condizioni iniziali molto distanti dal minimo, o addirittura in un massimo di $V_N(\theta)$, in cui l'Hessiana risulta definita negativa, il passo di Newton (4.7) potrebbe essere completamente errato, portando ad un risultato fuorviante.
Sfortunatamente nell'algoritmo non c'è tutela contro il rischio di convergere ad un minimo locale. Questo perché il passo di Newton fa sì che il parametro si avvicini ad un punto critico dell'errore quadratico medio attuale, sia esso minimo locale o globale. Ciò che si può fare è confrontare i risultati della minimizzazione iterativa per varie condizioni iniziali, scegliendo poi il migliore. Un ulteriore possibilità è quella di eseguire una stima preliminare per ottenere delle condizioni iniziali "vicine" al minimo globale [14].
Ciò che abbiamo fatto noi è stato far ciclare l'algoritmo partendo da condizioni iniziali diverse, confrontando poi i risultati ottenuti. Abbiamo per lo scopo considerato quattro valori iniziali dei parametri relativi a quattro configurazioni diverse del campo (fig. 4.2a).

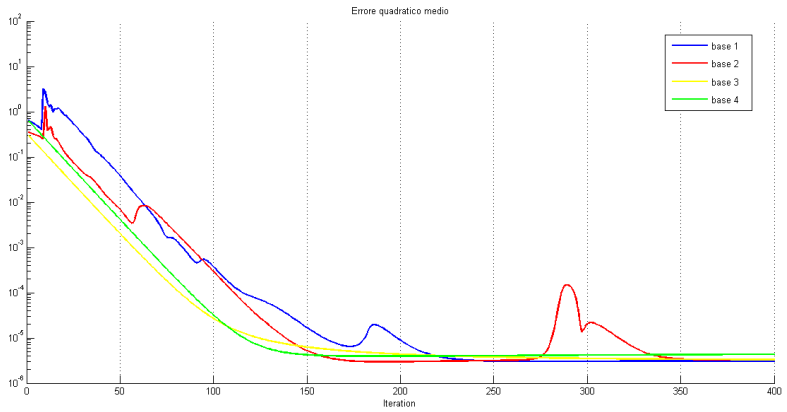
1. basi posizionate al centro della mappa con ampiezza nulla;
2. basi posizionate al centro della mappa con ampiezza diversa da 0;
3. base uniforme (2.1);
4. base centroidale (2.2).

E' immediato constatare la presenza di minimi locali. In fig. 4.2c si può chiaramente osservare che il valore minimo della funzione costo $V_N(\theta)$ converge a valori diversi a seconda della base iniziale, anche se tali valori sono molto vicini tra loro. Inoltre si può notare una diversa velocità e regolarità nella convergenza al minimo. Infatti, in tutte le simulazioni, notiamo che per un campo iniziale "simile" al campo da stimare, la convergenza al minimo richiede un numero di iterazioni inferiore. Questo è il caso della stima con base iniziale uniforme e centroidale. Queste due configurazioni possono quindi essere delle buone soluzioni da scegliere come condizioni iniziali.

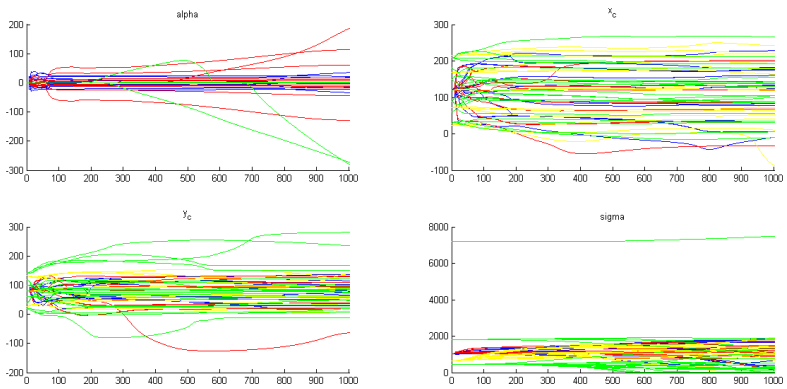


(a) Base iniziale.

(b) Stima finale.



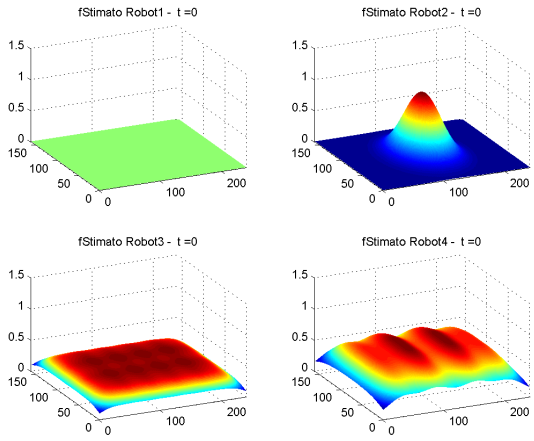
(c) Errore quadratico medio.



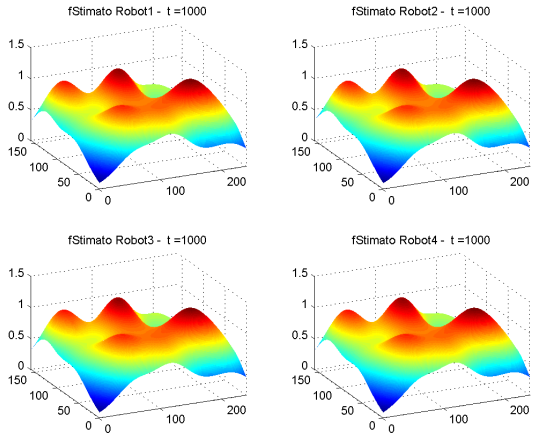
(d) Evoluzione dei parametri per diverse condizioni iniziali.

Figura 4.2: Simulazione di stima su campo poco variabile 3.1a con condizioni iniziali diverse.

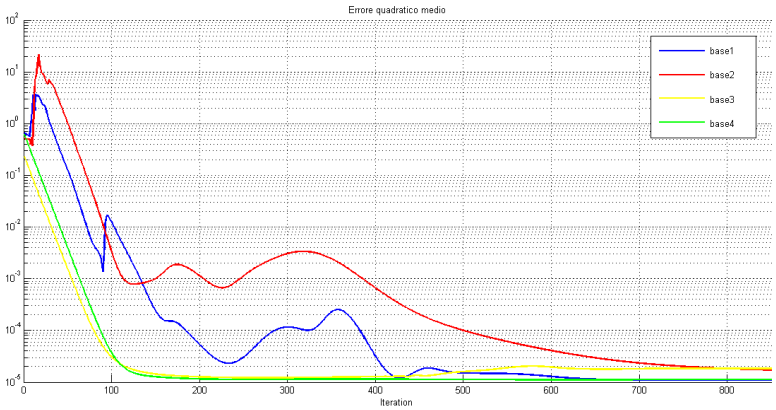
Altro fattore da osservare è il valore finale del parametro θ nei quattro casi. Infatti possiamo vedere in fig. 4.2d (anche se con qualche difficoltà date dall'elevato numero di RBF impiegate per la stima (14)) che i parametri, nei quattro casi, tendono a valori finali diversi. Questo ci dice che l'argomento che minimizza $V_N(\theta)$ è un insieme contenente più di un possibile valore.



(a) Base iniziale.

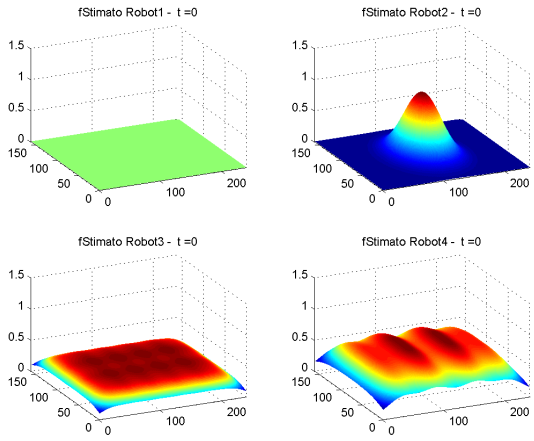


(b) Stima finale.

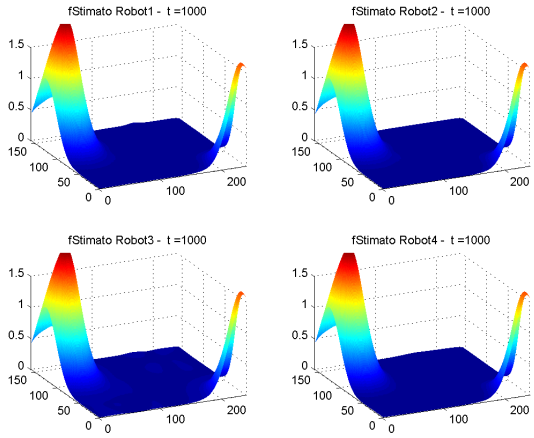


(c) Errore quadratico medio.

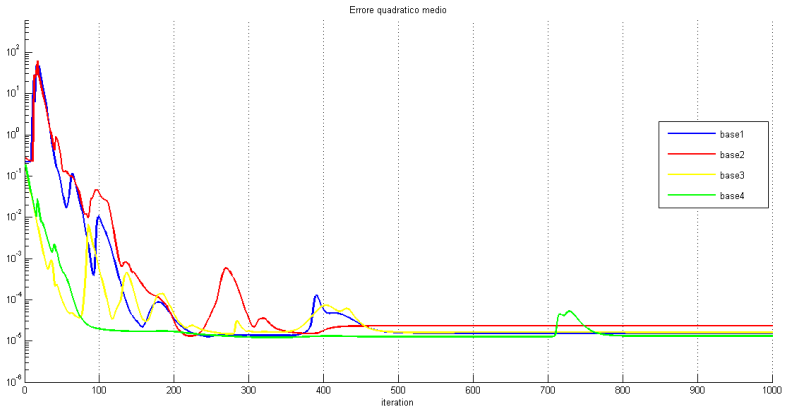
Figura 4.3: Simulazione di stima su campo molto variabile 3.9 con condizioni iniziali diverse.



(a) Base iniziale.



(b) Stima finale.



(c) Errore quadratico medio.

Figura 4.4: Simulazione di stima su campo con vertici ai lati 3.7 con condizioni iniziali diverse.

4.2.6 Passo di Newton

Como possiamo vedere in fig. 4.2c l'andamento dell'errore quadratico medio $V_N(\theta)$ non è regolare per quelle basi "distanti dal campo vero". Si evidenzia addirittura che in alcuni istanti l'errore aumenta invece che diminuire come dovrebbe. Questo è dovuto al fatto che per qualche iterazione k, specialmente quando θ_k è distante dal minimo, o meglio non appartiene a $N(\theta_*, r)$ definito nel Teorema 3, la direzione $f(k)$ (4.8) non è tale da garantire la diminuzione dell'errore.
Una soluzione per risolvere il problema di stima partendo da punti fuori dall'intervallo di convergenza di Newton è quella di introdurre un peso λ_k al passo di aggiornamento

$$\theta_{k+1} = \theta_k - \lambda_k \left[V_N''(\theta_k)\right]^{-1} V_N'(\theta_k) \tag{4.27}$$

tale che:

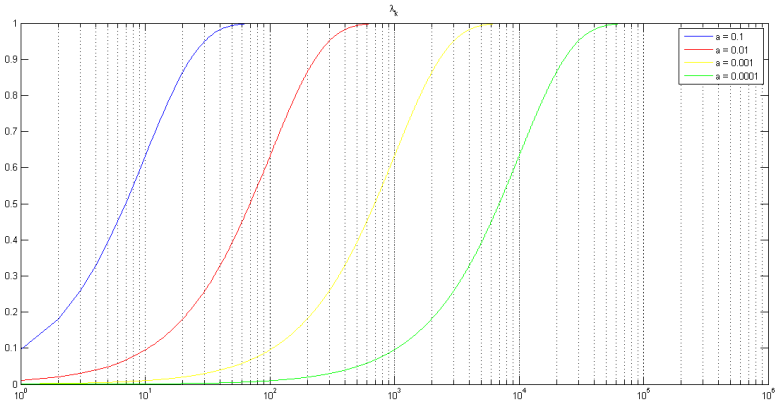
$$V_N(\theta_{k+1}) < V_N(\theta_k) \tag{4.28}$$

In [6] viene proposto il metodo del *line searches*. Tale algoritmo prevede che ad ogni iterazione λ_k sia impostato a 1 per poi venir ridotto secondo la relazione $\lambda_k = \rho\lambda_k$ con $\rho \in [0, 1]$ finché la relazione (4.28) risulti soddisfatta.
Questo algoritmo richiede però un elevato costo computazionale in quanto occorre calcolare θ_{k+1} e $V_N(\theta_{k+1})$ per ogni scelta di λ_k fino ad ottenere quella ottima.
L'idea di fondo è che se θ_k è distante dal minimo il peso λ_k dovrà essere piccolo, mentre, vicino al minimo, λ_k dovrà essere circa 1. Secondo tale linea di principio, siccome durante le prime iterazioni dell'algoritmo di Newton θ_k è distante dal minimo e va via via avvicinandosi, abbiamo preso λ_k pari a una funzione crescente esponenzialmente con asintoto orizzontale in 1, in modo da pesare poco i primi aggiornamenti, andando poi ad aumentare il peso con l'avvicinarsi di θ_k al minimo.

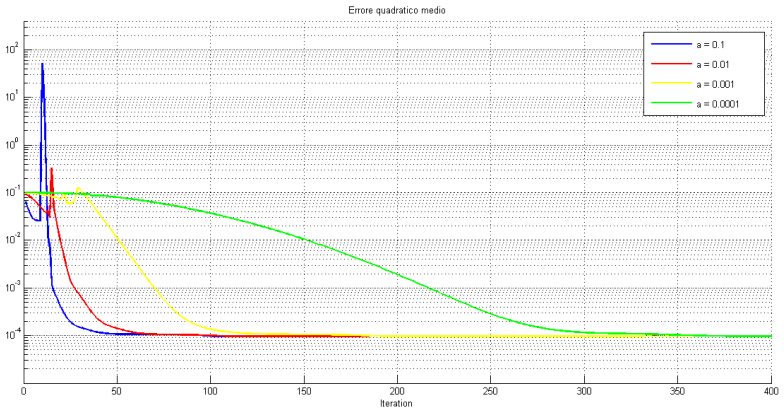
$$\lambda_k = 1 - e^{-ak} \tag{4.29}$$

La variabile a, ovvero il rate di crescenza della funzione, è stata impostata facendo alcune simulazioni (fig. 4.5).

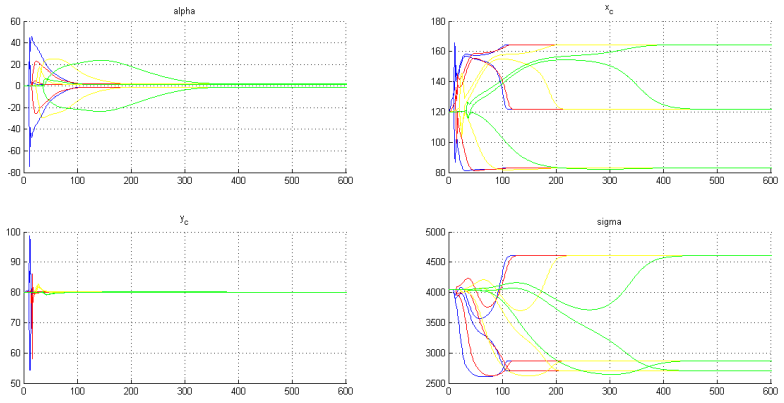
Con un rate alto si ottiene una convergenza al minimo di $V_N(\theta)$ molto velocemente ma con molte irregolarità e fasi di aumento del errore di predizione medio. Viceversa, diminuendo il rate di crescenza, la velocità di convergenza si riduce ma acquisisce una dinamica più regolare. In particolare, l'errore di predizione risulta avere un andamento monotono decrescente. Anche dall'evoluzione dei parametri (fig. 4.5c) si può ben vedere l'aumento della regolarità col diminuire del valore di a.
Siccome la regolarità per questi tipi di algoritmi è sinonimo di stabilità, è preferibile impostare a in modo da ottenere una dinamica dei parametri più continua anche se più lenta.



(a) λ_k per alcuni valori di a.



(b) Errore quadratico medio.

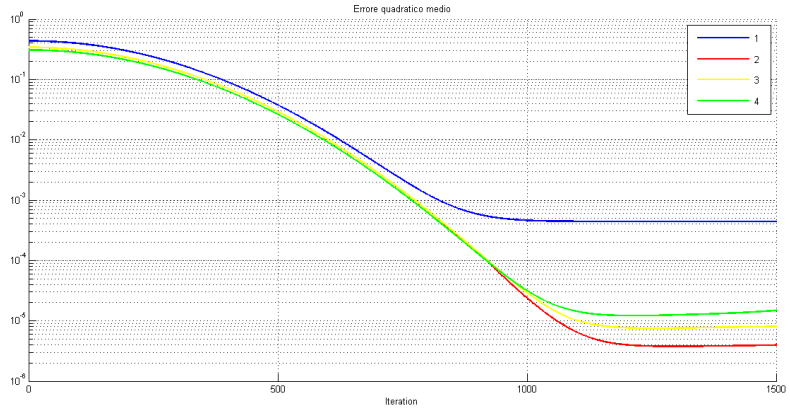


(c) Evoluzione dei parametri.

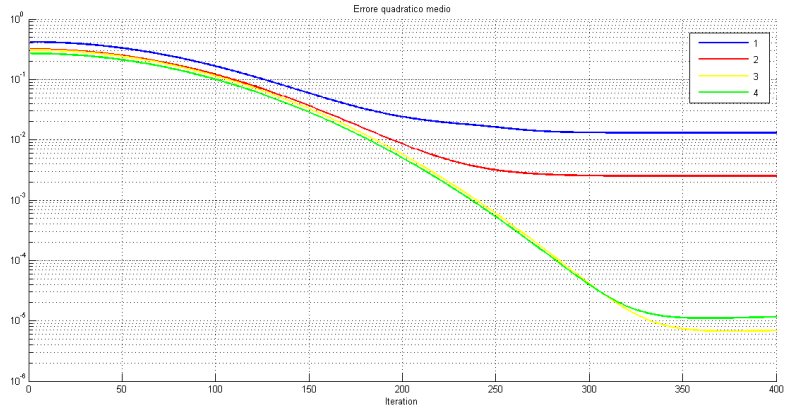
Figura 4.5: Simulazione di stima su campo poco variabile 3.1a con diversi valori di $a = 0.1, 0.01, 0.001, 0.0001$. Configurazione iniziale corrispondente a 3 basi posizionate al centro della mappa con ampiezza diversa da 0.

4.2.7 Alcune simulazioni

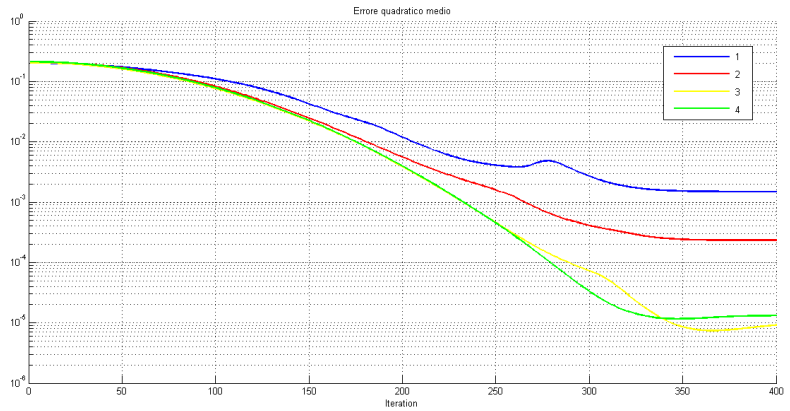
Dopo aver settato in modo ottimale tutti i parametri dell'algoritmo visti precedentemente, abbiamo fatto alcune simulazioni sui campi già studiati in (Parte I), quali poco variabile, molto variabile e con campo ai vertici. Per ognuno di questi abbiamo stimato il campo partendo da una base uniforme, per un numero crescente di basi.
Com'era prevedibile, si vede che aumentando la risoluzione, e quindi il numero di basi, l'errore finale diminuisce. Come osservato però nel capitolo 3, con un numero troppo elevato di basi si verifica il fenomeno dell'*overfitting*. Infatti si può notare dai grafici, che la stima con risoluzione quattro ha un errore finale sempre più elevato di quella con risoluzione tre.



(a) Campo poco variabile fig. 3.1a.



(b) Campo molto variabile fig. 3.9.



(c) Campo ai vertici fig. 3.7.

Figura 4.6: Errore quadratico medio su vari tipi di campi per un numero crescente di basi. Stima iniziale corrispondente a base uniforme.

	Risoluzione	numero di basi	Errore basi fisse	Errore basi mobili
Campo poco variabile	1	2	0.034	4.4e-4
	2	6	0.034	3.8e-6
	3	15	0.0021	7.6e-6
	4	24	3.1e-4	1.3e-5
Campo molto variabile	1	2	0.028	0.013
	2	6	0.015	0.0025
	3	15	0.0074	6.9e-6
	4	24	0.0026	1.1e-5
Campo ai vertici	1	2	0.18	0.0015
	2	6	0.067	0.00023
	3	15	0.024	8.6e-6
	4	24	0.0185	1.3e-5

Tabella 4.1: Errore quadratico medio.

Nella tabella 4.1 abbiamo riportato il valore finale dell'errore nei vari casi mettendo a confronto le prestazioni della stima con basi fisse e quella con basi mobili. Dall'analisi si può dedurre che a parità di errore finale, la stima a basi mobili permette di utilizzare un numero molto inferiore di funzioni RBF. Il caso più evidente è quello del campo ai vertici. Con le basi mobili, già con sole due funzioni, si ottiene un errore inferiore di un ordine di grandezza rispetto la stima con 24 basi fisse. Infatti, in linea di principio basterebbero proprio due basi posizionate agli angoli per stimare bene il campo. Questo è possibile con le basi mobili mentre, per ottenere lo stesso risultato con la versione a basi fisse, è necessario aumentare il numero di funzioni, molte delle quali però, in particolare quelle al centro della mappa, rimangono praticamente inutilizzate.
Il fatto di poter usare un numero inferiore di basi consente, oltre che di ridurre la complessità computazionale dell'algoritmo, anche di ottenere stime dei parametri più precise e con varianze minori.

4.3 Riassumendo

In questo capitolo abbiamo visto come l'algoritmo di Newton si presti molto bene al problema di stima di campi tramite l'utilizzo di reti neurali con funzioni RBF. Tale algoritmo ci ha consentito di stimare, oltre che i combinatori α_k delle funzioni RBF, anche le loro caratteristiche principali quali centro e varianza.
Così facendo abbiamo potuto, a parità di errore finale di predizione, ridurre il numero di basi, il che comporta una diminuzione della varianza dei parametri, ottenendo quindi stime più precise.
Tutto questo è stato pagato però con l'aumento della complessità computazionale e del tempo di convergenza alla soluzione. Essendo infatti un procedimento iterativo non esatto, occorre - come si è visto - aspettare un certo numero di iterazioni prima di ottenere la soluzione. Inoltre ogni iterazione richiede il calcolo del gradiente e l'inversa dell'Hessiana di $V_N(\theta_k)$ che può risultare oneroso con un numero di basi e di misure elevato.
Occorre ricordare inoltre il problema dei minimi locali: esso implica la necessità di una certa quantità di informazione a priori sul campo da stimare per ottenere dei risultati soddisfacenti in tempi ragionevoli.

Capitolo 5

Stima PEM ricorsiva nelle misure

Introduzione

Nelle sezioni precedenti abbiamo descritto gli algoritmi impiegati per poter ottenere una stima ottima del campo con distribuzione adattiva di posizione, ampiezza e varianza del set di funzioni radiali di base. A questo scopo si è dovuto passare, come descritto nel capitolo 4, da uno stimatore lineare ai minimi quadrati ad uno stimatore non lineare realizzato con algoritmo ricorsivo Quasi-Newton. Con l'algoritmo descritto siamo riusciti ad ottenere ottime prestazioni e stime di qualità elevata.
Purtroppo però in molte situazioni operative, come nel nostro caso, non è possibile assumere di avere a disposizioni l'intero insieme di misure sin dall'inizio, ma si rende invece necessario elaborare i dati man man che essi vengono raccolti. Nel caso in esame, ad esempio, mentre il campo viene stimato, gli agenti robotici continuano a muoversi nell'ambiente e, ad ogni passo, perseverano nel raccogliere misure della grandezza da stimare. Si rende perciò auspicabile l'utilizzo di un algoritmo che sfrutti i dati non appena essi vengono acquisiti mantenendo contemporaneamente traccia dell'evoluzione passata, il tutto occupando un tempo di calcolo ragionevole per l'applicazione.
Infatti l'idea più banale per raggiungere lo scopo desiderato è quella di ricalcolare, ad ogni nuova misura acquisita, le matrici di aggiornamento dei metodi in maniera ricorsiva sfruttando di volta in volta tutto l'insieme di dati a disposizione. Si osserva però che questo tipo di algoritmo richiede una quantità di tempo di calcolo decisamente esagerata e che diventerebbe perciò la limitazione principale alla rapidità nella stima del campo d'interesse.
Si è così tentato di applicare un algoritmo Quasi-Newton con implementazione ricorsiva nei dati: allo scopo di calcolare le quantità d'interesse si sfrutta solo l'ultima misura acquisita e il vettore dei parametri ottenuto al passo precedente. Si ottiene così un algoritmo snello ed efficiente, coniugando rapidità, qualità delle stime e ricorsività.
In questa sezione andiamo perciò ad illustrare l'implementazione di questo algoritmo ricorsivo in maniera centralizzata (singolo agente), valutandone le caratteristiche di precisione delle stime parametriche, onerosità computazionale e stabilità.

5.1 Formato degli algoritmi ricorsivi

Per l'implementazione del metodo di stima iterativa si è seguita, adattata e applicata principalmente la teoria proposta da Ljung e Soderstrom in [14] e [11].
Un metodo d'identificazione generico è sostanzialmente una mappa dall'insieme dei dati Z^t allo spazio dei parametri Θ_t:

$$\hat{\theta}_t = F(t, Z^t) \tag{5.1}$$

dove la funzione F può essere definita implicitamente (e.g., come la minimizzazione di una funzione costo). Un'espressione di questo tipo non può però essere utilizzata in un algoritmo ricorsivo, infatti l'estrazione di F potrebbe richiedere enorme costo computazionale, e quindi un tempo superiore a quello che intercorre tra due istanti di campionamento successivi.
Prendiamo invece algoritmi ricorsivi del seguente formato:

$$\begin{aligned} X(t) &= H(t, X(t-1), y(t), u(t)) \\ \hat{\theta}_t &= h(X(t)) \end{aligned} \tag{5.2}$$

Qui $X(t)$ è un vettore di dimensione fissata che rappresenta tutta l'informazione passata che viene sfruttata per il calcolo della nuova stima. Le funzioni H e h sono invece espressioni esplicite che possono essere ricavate ad ogni passo con un costo computazionale fisso e noto a priori, indipendente quindi dall'aumentare del numero dei dati in possesso. In questo modo ci possiamo assicurare che $\hat{\theta}_t$ sia computabile durante un intervallo di campionamento.
Siccome l'informazione contenuta nell'ultima coppia di misure, $y(t), u(t)$, è solitamente piccola rispetto a quella già accumulata con le misure precedenti, l'algoritmo (5.2) assume la forma:

$$\begin{aligned} \hat{\theta}_t &= \hat{\theta}_{t-1} + \gamma_t Q_\theta(X(t), y(t), u(t)) \\ X(t) &= X(t-1) + \mu_t Q_X(X(t-1), y(t), u(t)) \end{aligned} \tag{5.3}$$

dove γ e μ sono valori piccoli che rappresentano il valore relativo d'informazione contenuto nell'ultima misura.
Questa famiglia generale di algoritmi si particolarizza poi in base all'applicazione specifica e al criterio di minimizzazione prescelto.

5.2 RPEM: Recursive Prediction Error Methods

Per il nostro fine abbiamo implementato un algoritmo di stima basato sulla **minimizzazione dell'errore di predizione**, continuando la strada intrapresa nella versione non ricorsiva trattata nel capitolo precedente. Ovviamente in questo caso ne abbiamo realizzato la versione **ricorsiva**.
Abbiamo preso una funzione costo di tipo quadratico nell'errore di predizione:

$$V_t(\theta, Z^t) = \gamma(t)\frac{1}{2}\sum_{k=1}^{t}\beta(t,k)\varepsilon^2(k,\theta) \tag{5.4}$$

La funzione utilizzata risulta dello stesso tipo della (4.3), con l'introduzione di **sequenze peso** che modificano la rilevanza dei dati in funzione del tempo, rendendo sempre meno influenti le variazioni del gradiente della funzione costo con l'aumentare del numero di campioni. La presenza di funzioni peso di questo tipo potrebbe apparire un fattore secondario per il funzionamento dell'algoritmo o comunque una raffinatezza tesa a migliorare la qualità delle stime. La realtà è totalmente diversa: infatti nel corso dell'implementazione dell'algoritmo ci siamo resi conto della reale necessita di funzioni peso particolari per rendere l'algoritmo corretto e funzionante. Questo argomento verrà affrontato diffusamente nel paragrafo (5.4), per il momento invece elenchiamo solo alcune proprietà fondamentali per la comprensione delle derivazioni successive.

Sequenze peso

La **sequenza peso** (β) ha le seguenti proprietà:

$$\begin{aligned} \beta(t,k) &= \lambda(t)\beta(t-1,k), \qquad 1 \le k \le t-1 \\ \beta(t,t) &= 1 \end{aligned} \tag{5.5}$$

Perciò:

$$\beta(t,k) = \prod_{j=k+1}^{t} \lambda(j) \tag{5.6}$$

dove $\lambda(j)$ è chiamato **fattore d'oblio**.
Da quest'ultimo si deriva il **guadagno normalizzato** ($\gamma(t)$):

$$\gamma(t) = \left[\sum_{k=1}^{t} \beta(t,k)\right]^{-1} \tag{5.7}$$

secondo la relazione:

$$\frac{1}{\gamma(t)} = \frac{\lambda(t)}{\gamma(t-1)} + 1 \tag{5.8}$$

Caratteristiche e significati di queste funzioni verranno delineati più chiaramente nel paragrafo 5.4.

Gradiente

Calcoliamo il gradiente della funzione costo:

$$\begin{aligned} V_t'(\theta, Z^t) &= -\gamma(t) \sum_{k=1}^{t} \beta(t,k)\psi(k,\theta)\epsilon(k,\theta) \\ &= \gamma(t)\left[\lambda(t)\frac{1}{\gamma(t-1)} V_{t-1}'(\theta, Z^{t-1}) - \psi(t,\theta)\varepsilon(t,\theta)\right] \\ &= V_{t-1}'(\theta, Z^{t-1}) + \gamma(t)\left[-\psi(t,\theta)\varepsilon(t,\theta) - V_{t-1}'(\theta, Z^{t-1})\right] \end{aligned} \tag{5.9}$$

Come per il caso non iterativo sappiamo che una famiglia di direzioni di ricerca valide per questo algoritmo è del tipo:

$$\hat{\theta}_N^{(i+1)} = \hat{\theta}_N^{(i)} - \mu_N^{(i)} \left[R_N^{(i)}\right]^{-1} V_N'(\hat{\theta}_N^{(i)}, Z^N) \tag{5.10}$$

dove $\hat{\theta}_N^{(i)}$ indica la stima del parametro alla i-esma iterazione. $R_N^{(i)}$ è una matrice quadrata che modifica la direzione di ricerca e $\mu_N^{(i)}$ è la *dimensione del passo*, cioè il parametro che regola l'entità della variazione da applicare al passo i-esimo.
Nell'algoritmo (5.10), il pedice t indica la stima al tempo t, cioè basata sull'insieme di dati Z^t, mentre l'apice (i) denota la i-esima iterazione del processo di minimizzazione.
Assumendo che, ad ogni iterazione i, venga acquisita una nuova misura, allora varrà sempre $i = t$ e quindi l'algoritmo si può scrivere:

$$\hat{\theta}(t) = \hat{\theta}(t-1) - \mu(t)\left[R(t)\right]^{-1} V_t'(\hat{\theta}(t-1), Z^t) \tag{5.11}$$

Assumendo ora per ipotesi induttiva che $\hat{\theta}(t-1)$ minimizzi realmente $V_{t-1}(\theta, Z^t)$ si ha

$$V_{t-1}'(\hat{\theta}(t-1), Z^{t-1}) = 0 \tag{5.12}$$

(questa è un'approssimazione). Quindi per il calcolo del gradiente al passo t, noti solo i dati in $t-1$, si ottiene l'equazione (5.13), che d'ora in poi chiameremo $g(t)$:

$$V_t'(\hat{\theta}(t-1), Z^t) = -\gamma(t)\psi(t, \hat{\theta}(t-1))\varepsilon(t, \hat{\theta}(t-1)) \stackrel{\triangle}{=} g(t) \tag{5.13}$$

con questa approssimazione e prendendo $\mu(t) = 1$, cioè lasciando a $\gamma(t)$ la definizione dell'ampiezza del passo, come suggerito dalla teoria in [6], arriviamo all'algoritmo finale:

$$\hat{\theta}(t) = \hat{\theta}(t-1) + \gamma(t)R^{-1}(t)\psi(t, \hat{\theta}(t-1))\varepsilon(t, \hat{\theta}(t-1)) = \hat{\theta}(t-1) + \gamma(t)R^{-1}(t)g(t) \tag{5.14}$$

Per quanto riguarda la scelta della matrice $R(t)$, secondo il metodo Gauss-Newton, come per il caso non iterativo, si sceglie la *Pseudo Hessiana* regolarizzata:

$$R(t) \equiv H(t) \stackrel{\triangle}{=} \gamma(t) \sum_{k=1}^{t} \beta(t,k)\psi(t)\varepsilon(t) + \delta(t)I_N \tag{5.15}$$

Usando la (5.15) in (5.14), si ottiene infine l'algoritmo ricorsivo:

Algoritmo PEM Ricorsivo

$$\begin{aligned} \varepsilon(t) &= h(t) - \hat{h}(t) \\ \hat{\theta}(t) &= \hat{\theta}(t-1) + \gamma(t) H^{-1}(t)\psi(t)\varepsilon(t) = \hat{\theta}(t-1) + H^{-1}(t)g(t) \\ H(t) &= H(t-1) + \gamma(t)\left[\psi(t)\psi^T(t) - H(t-1)\right] \end{aligned} \tag{5.16}$$

5.3 Proprietà asistotiche

Il metodo RPEM (5.16) è progettato per fare aggiornamenti del parametro in una direzione che "in media" è un gradiente negativo modificato di

$$\bar{V}(\theta) = \frac{1}{2}\bar{E}\varepsilon^2(t,\theta)$$

cioè:

$$\frac{d}{d\theta}\bar{V}(\theta) = -\bar{E}\psi(t,\theta)\varepsilon(t,\theta)$$

In questo caso $\hat{\theta}(t)$ converge a un minimo locale di $\bar{V}(\theta)$, sotto opportune condizioni di regolarità (cfr: [12]). Oltre a questo però, per un RPEM di tipo Gauss-Newton, con $\gamma(t) = 1/t$, si può dimostrare che $\hat{\theta}(t)$ ha distribuzione asintotica normale, e coincidente con la corrispondente stima non iterativa (4.2). Si ha infatti:

- se $H(t)$ è invertibile (i.e. regolarizzata) e $\gamma(t) \to 0$ per $t \to \infty$, allora con probabilità 1, $\hat{\theta}(t)$ converge a un minimo locale di $\bar{V}$;
- assumendo che lo stimatore sia consistente (il modello vero appartenga alla famiglia parametrica) e che quindi $\hat{\theta}(t) \to \theta_0$; utilizzando il RPEM Gauss-Newton (5.16) con $\gamma(t) = 1/t$ vale il Teorema del Limite Centrale e asintoticamente si ottiene:

$$\begin{aligned} \sqrt{t}(\hat{\theta}(t) - \theta_0) &\to \mathcal{N}(0, P_\theta) \\ P_\theta &= \lambda_0 \left[\bar{E}\psi(t,\theta_0)\psi^T(t,\theta_0)\right]^{-1} \end{aligned} \tag{5.17}$$

Vedere [14, pp. 329-335] per la dimostrazione rigorosa.

5.4 Fattore d'oblio

Nell'implementazione di algoritmi ricorsivi l'utilizzo di sequenze di pesi nella funzione costo risulta essere una componente fondamentale per garantire la convergenza e la stabilità. I metodi di progetto possibili consistono nello scegliere una sequenza di pesi $\beta(t,k)$, oppure il fattore d'oblio $\lambda(t)$ o ancora il guadagno $\gamma(t)$. I tre approcci sono in realtà equivalenti date le relazioni (cfr. (5.5) e successive):

$$\begin{aligned} \beta(t,k) &= \prod_{j=k+1}^{t} \lambda(j) = \frac{\gamma(k)}{\gamma(t)} \prod_{j=k+1}^{t} (1-\gamma(j)) \\ \lambda(t) &= \frac{\gamma(t-1)}{\gamma(t)}(1-\gamma(t)) \\ \gamma(t) &= \frac{1}{\frac{\lambda(t)}{\gamma(t-1)} + 1} \end{aligned} \tag{5.18}$$

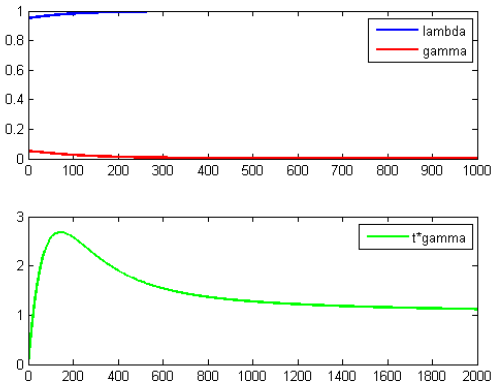
La scelta del **fattore d'oblio** $\beta(t,k)$ può essere interpretata come un modo per introdurre nella funzione costo solo un certo numero di misure recenti e quindi rilevanti per le proprietà del sistema.
La scelta del **guadagno** $\gamma(t)$ invece riflette il contenuto informativo dell'osservazione corrente, ad esempio una misura con elevato rumore dovrebbe avere un minor contenuto informativo e quindi guadagno più basso.

In questo progetto abbiamo deciso di operare scegliendo $\gamma(t)$ come suggerito da [11]. Nello specifico per stime *tempo-invarianti*, in cui si ipotizza la presenza di un parametro vero fisso, la convergenza

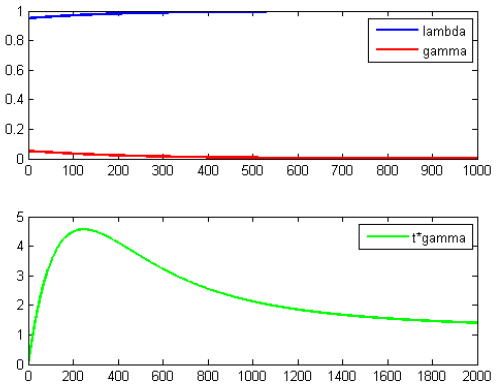
asintotica è garantita scegliendo un guadagno $\gamma(t) \to 1/t$, come visto nel paragrafo (5.3). Inoltre con la ricerca di Gauss-Newton si ottiene la stima a minor varianza possibile.
D'altro canto, nel transitorio, potrebbe essere il caso di scegliere $\gamma(t)$ in maniera diversa. Si è infatti scoperto empiricamente che la regolazione di questo parametro ha importanti effetti sulla velocità di convergenza dell'algoritmo (e diventerà ancora più importante nel caso distribuito, capitolo 7).
L'esigenza di dare poco peso ai vecchi dati, cioè di avere un buon comportamento nel transitorio dove le variazioni nella stima sono brusche, richiede un $\lambda(t) < 1$ e cioè $\gamma(t) > 1/t$. D'altra parte, è auspicabile avere $\lambda(t) \to 1$ ($t\gamma(t) \to 1$) per $t \to \infty$, per ottenere precisione asintotica. Dalla pratica emerge che una buona funzione che soddisfi i requisiti è della forma:

$$\lambda(t) = \lambda_0 \lambda(t-1) + (1 - \lambda_0) \tag{5.19}$$

Secondo la formula (5.19), $\lambda(t)$ converge a 1 con *rate* λ_0 partendo dalla condizione iniziale $\lambda(0)$; notare che questi 2 valori sono variabili di progetto.
In quanto a valori numerici, per soddisfare i requisiti esposti si prende $t\gamma(t)$ tale da avere un massimo compreso tra 2 e 5, che si raggiunge ad un tempo $t \in [50, 400]$. Per la traduzione di queste specifiche sulle variabili di progetto bisogna ricordare che $\lambda(0)$ influisce principalmente sull'intensità del massimo; mentre λ_0 regola il valore di t per cui si raggiunge il massimo e la durata temporale dello stesso.
Partendo dalla funzione consigliata da [11] (figura 5.1a), abbiamo modificato leggermente i parametri in modo da avere una dinamica che si spegnesse più lentamente (fig. 5.1b); la variabilità di $\hat{\theta}(t)$ risultava infatti sovrasmorzata precocemente al crescere di t.



(a) Funzione standard $\lambda(0) = 0,95$; $\lambda_0 = 0,99$



(b) Funzione adattata $\lambda(0) = 0,95$; $\lambda_0 = 0,995$

Figura 5.1: Grafico degli andamenti di $\lambda(t), \gamma(t), t\gamma(t)$

Nota

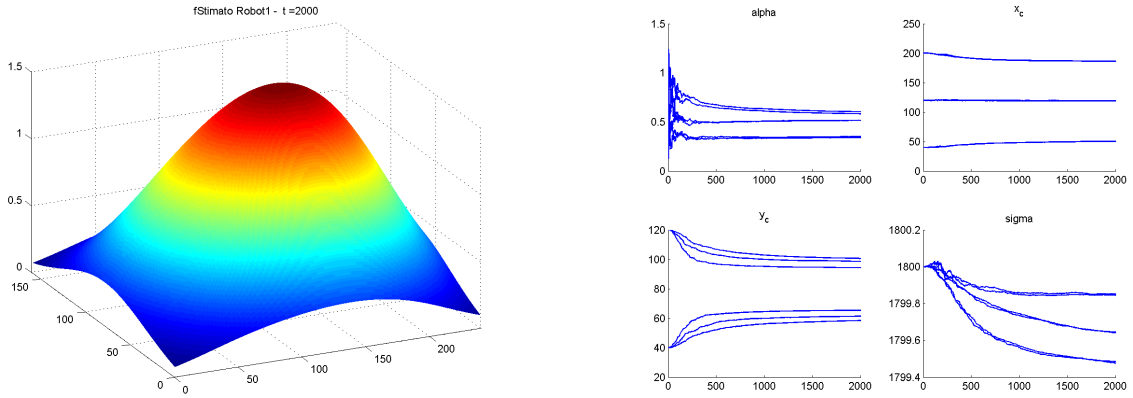
Le considerazioni appena effettuate in merito al fattore d'oblio, risultano un importante raffinamento nella versione centralizzata dell'algoritmo, infatti ne migliorano notevolmente i risultati, in quanto a velocità di convergenza e qualità della stima finale. Vedremo però che saranno condizione necessaria per la stabilità della versione distribuita (Capitolo 7).

5.5 Simulazioni

Allo scopo di testare l'algoritmo sviluppato, accuratamente e nelle più disparate condizioni applicative, abbiamo effettuato una serie di simulazioni su svariati campi realizzati in modo casuale.
Prima di esporre i risultati simulativi descriveremo le procedure effettuate per regolare in maniera ottima i parametri liberi del fattore d'oblio (paragrafo 5.5.1) e per decidere quale fosse la configurazione delle RBF di partenza più adatta all'algoritmo (paragrafo 5.5.2).
Anticipiamo che per le simulazioni qui riportate si è utilizzata sempre una distribuzione delle basi di partenza di tipo uniforme, spiegheremo più accuratamente il motivo di questa scelta nel paragrafo (5.5.2). Inoltre abbiamo scelto il numero di 2000 misure, che comportano 2000 cicli dell'algoritmo iterativo, perché ritenuto un numero sufficiente a mettere in luce l'andamento asintotico.
Dopo aver regolato i parametri riporteremo le simulazioni sugli ormai noti 3 campi significativi, utilizzati anche per analisi e stime nei capitoli precedenti. Oltre al risultato della stima, illustreremo anche l'andamento dell'errore di stima e dei parametri nei vari casi.

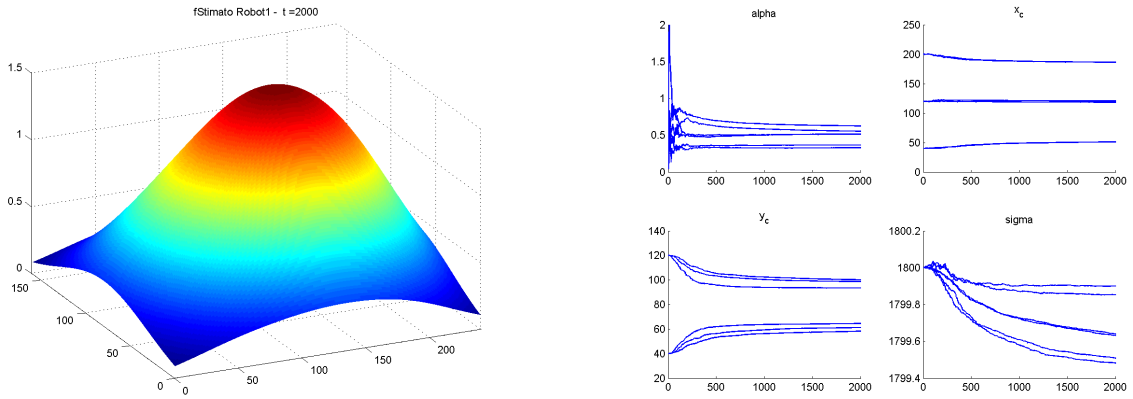
5.5.1 Tuning dei parametri

Una volta notato come la stima ottenuta utilizzando $\gamma(t) = 1/t$ risulti non eccelsa, si passa a introdurre la funzione come descritta nel paragrafo 5.4. I vari grafici fanno vedere la differenza nel risultato della stima per alcuni valori di λ_0.



(a) Campo stimato t=2000 (b) Andamento dei parametri

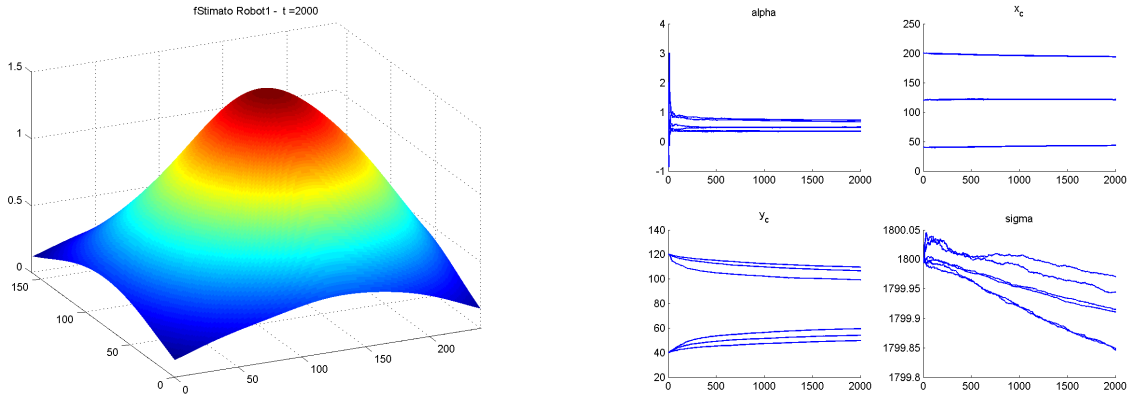
Figura 5.2: Base uniforme, Risoluzione = 2 (6 RBF), $\lambda_0 = 0.995$



(a) Campo stimato t=2000 (b) Andamento dei parametri

Figura 5.3: Base uniforme, Risoluzione = 2 (6 RBF), $\lambda_0 = 0.99$

Essendo la visualizzazione della stima finale poco significativa, si può passare ad analizzare l'andamento dell'errore di stima (fig. 5.5). Si nota coma l'introduzione del nuovo fattore d'oblio migliori



(a) Campo stimato t=2000

(b) Andamento dei parametri

Figura 5.4: Base uniforme, Risoluzione = 2 (6 RBF), $\gamma(t) = 1/t$

notevolmente la qualità della stima e la stabilità dell'algoritmo. In particolare si identifica nel valore $\lambda_0 = 0.995$ il miglior trade-off. Infatti per valori inferiori la dinamica risulta strozzata, mentre per valori superiori l'algoritmo diviene instabile e oscillante. Si può notare, per inciso, che come visto nel paragrafo 5.4, superato il valore di 0.995 si incorre in un picco della funzione $t\gamma(t)$ che supera in modulo il limite di 5 imposto dalla teoria.
Notiamo la presenza di un'ulteriore setting dei parametri che assicura stabilità, un andamento fluido della dinamica e ottima qualità della stima: si tratta della configurazione in cui si è spinto il rate λ_0 fino al valore $0,997$, aumentando contestualmente il valore iniziale $\lambda(0) = 0,97$ (color celeste in figura 5.5). Queste scelte hanno consentito di ottenere una andamento di $t\gamma(t)$ che presenta un picco ampio e piatto attorno a $t = 400$, i.e. attorno al limite massimo tollerabile, che raggiunge un intensità di $4,6$. Si capisce che questa configurazione sta appena dentro le specifiche, generando così una buona stima con dinamica più lenta e continua rispetto i parametri ottimi scelti.
Per il caso centralizzato che stiamo trattando in questa fase, conviene utilizzare $\lambda_0 = 0,995$, infatti esso ci consente di raggiungere più rapidamente un basso livello d'errore. Vedremo però, che per l'implementazione distribuita potrà essere utile rallentare la dinamica di Newton per favorire il consensus. Converrà quindi quest'ultima configurazione dei parametri.

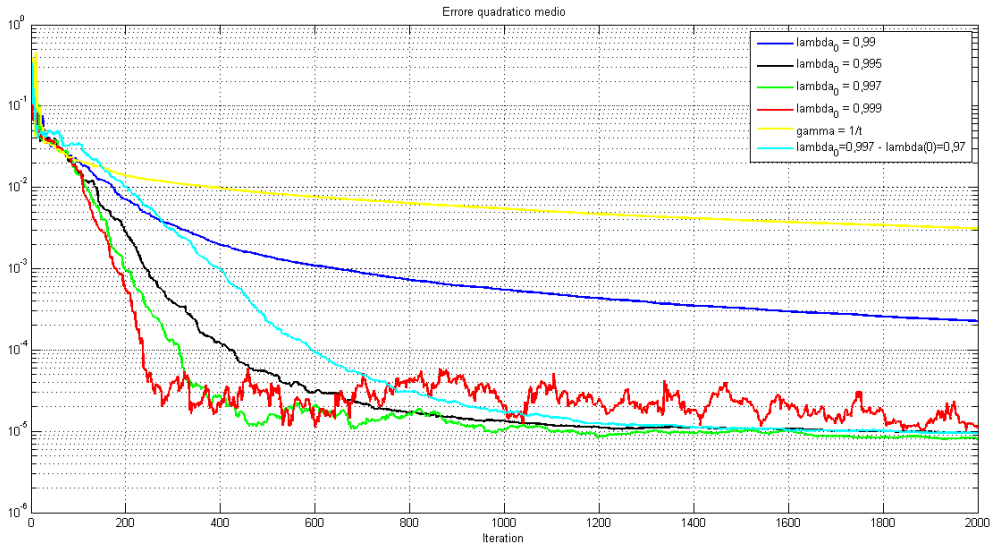
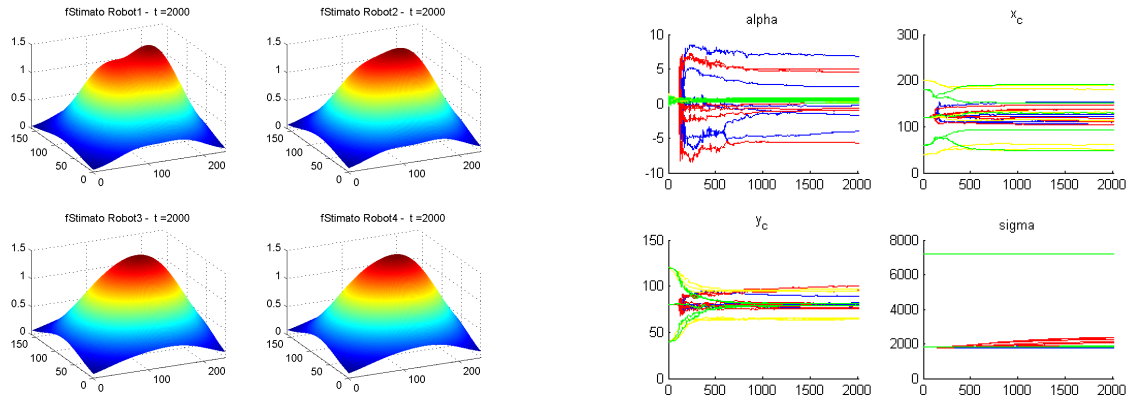


Figura 5.5: Andamento errore quadratico medio nelle diverse condizioni, in scala logaritmica

5.5.2 Scelta base di partenza

Come visto nel capitolo precedente (cfr.: 4.2.5), le stime PEM non lineari con direzione Newton-Raphson presentano per loro natura il rischio di incorrere in minimi locali. Questa problematica diventa ancora più sensibile e rilevante quando si passa all'algoritmo Ricorsivo: la presenza del fattore d'oblio comporta uno smorzamento della dinamica al crescere di t, rendendo più probabile la convergenza ad un minimo locale che non coincide con quello globale. Per questi motivi, variando la configurazione iniziale delle funzioni radiali, non è garantita né la convergenza allo stesso identico risultato, né il raggiungimento della stessa qualità di stima.
In particolare nelle figure seguenti abbiamo riportato: un esempio dei risultati dei processi di stima utilizzando le 4 diverse configurazioni iniziali sperimentate nel capitolo precedente (fig. 5.6); il grafico raffigurante il confronto tra i vari errori quadratici medi (fig. 5.7).



(a) Campo stimato t=2000 (b) Andamento dei parametri

Figura 5.6: Confronto tra i risultati delle stime con le diverse basi

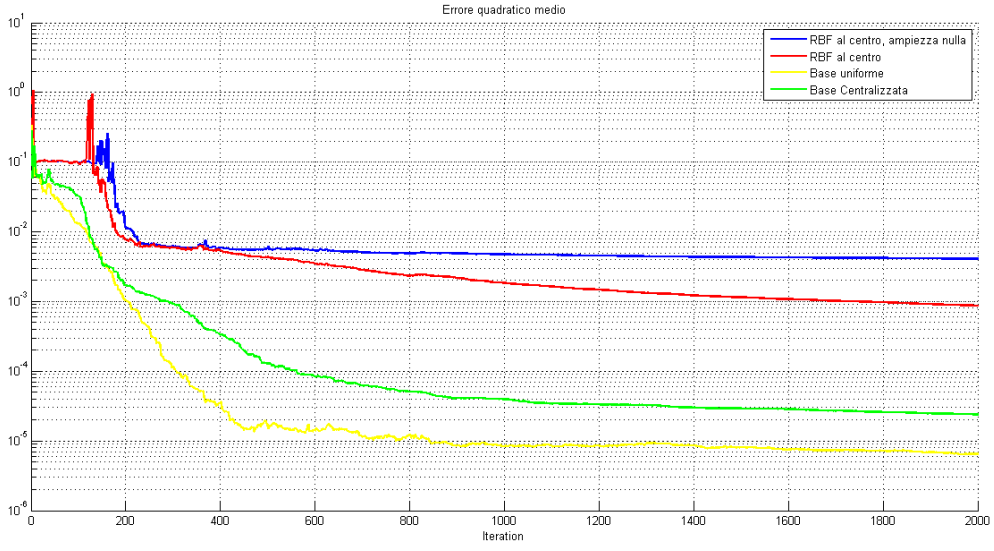


Figura 5.7: Confronti tra gli errori quadratici medi ottenuti con le diverse basi

Il tuning delle variabili di progetto del fattore d'oblio è quello selezionato dalle altre simulazioni di questo paragrafo, cioè ottimizzato per la **base uniforme**; come previsto infatti si vede che la stima migliore è proprio quella ottenuta con questa configurazione iniziale. Le altre configurazioni generano comunque una stima di qualità molto elevata, ad eccezione forse della prima tipologia di base. Questo comportamento scadente che si ottiene con la configurazione di tipo 1 è da imputare al fatto che essa prevede il settaggio delle ampiezze iniziali della gaussiane al **valore 0**: si è infatti notato, in fase di simulazione, che annullare alcuni dei parametri nelle condizioni iniziali genera spesso rallentamenti del

processo di stima o addirittura fenomeni di instabilità.
La scelta della base uniformemente distribuita è comunque dettata dalle considerazioni già esposte nel capitolo precedente sulla versione non ricorsiva dell'algoritmo e dal fatto che meglio si presta a stimare qualsiasi tipo di campo.

5.5.3 Campo poco variabile: confronto risoluzioni

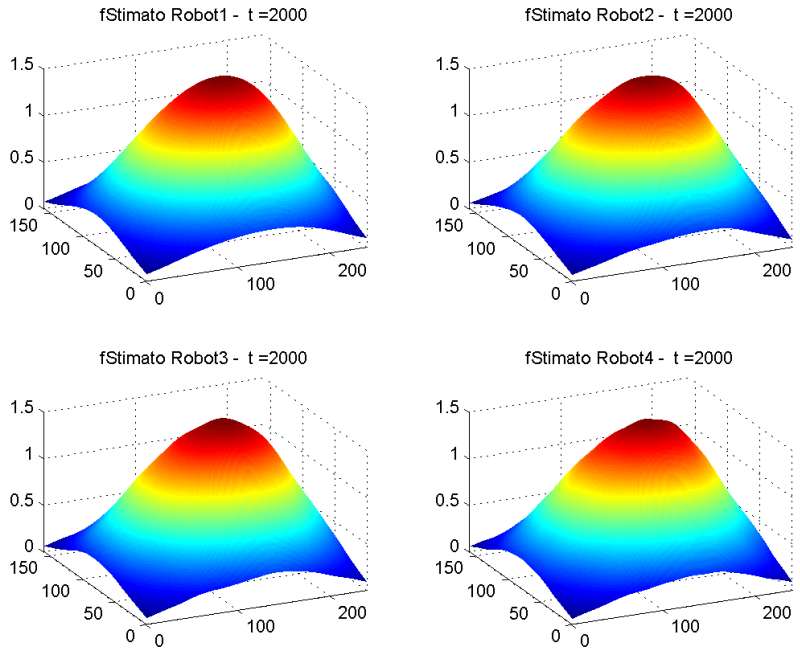


Figura 5.8: Stime campo poco variabile. Risoluzione 2, 3, 4, 5

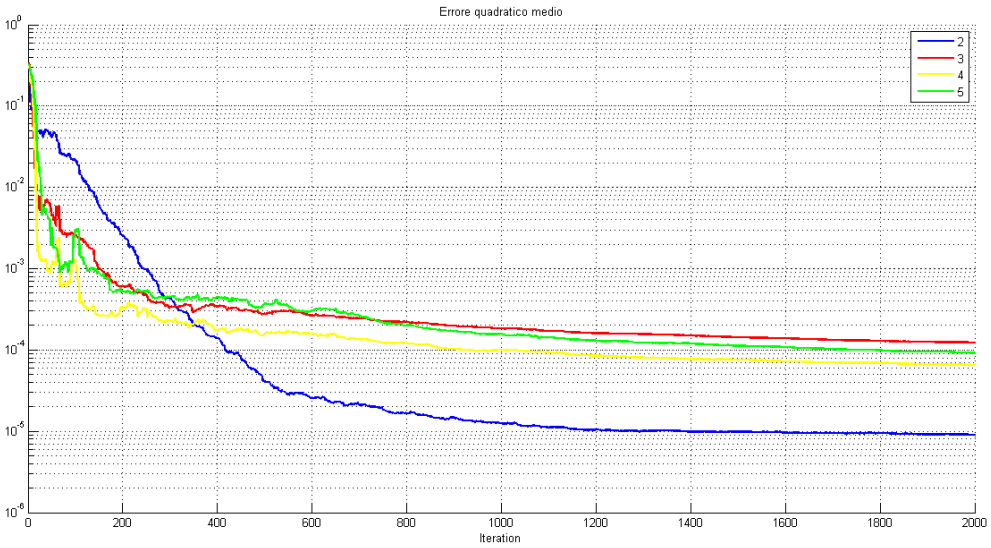


Figura 5.9: Confronti tra gli errori quadratici medi ottenuti con le diverse risoluzioni

Dalle figure emerge un'anomalia: notiamo infatti che la stima più precisa risulta essere quella a risoluzione inferiore. Questa simulazione mette in luce un limite dell'algoritmo, infatti quello che si vede è un fenomeno di *overfitting*. Essendo il campo molto semplice (si può infatti realizzare con sole 3 funzioni radiali), l'aumentare del numero di funzioni usate per la stima, e quindi del numero di parametri in gioco, genera un'incapacità dell'algoritmo di regolarli perfettamente. Sostanzialmente la campana in figura viene ricostruita con un numero molto elevato di RBF (da 12 per la risoluzione 3,

fino ad addirittura 36 per la risoluzione 5), questo comporta la tendenza, soprattutto in fase iniziale, a posizionare una gaussiana in ogni punto dove venga acquisita una misura, piuttosto che spostare poche funzioni base ed azzerare le altre. si necessita così di regolare finemente le varianze delle diverse RBF per seguire accuratamente la curva reale, ma il compito risulta un po' troppo preciso per l'algoritmo. In generale i risultati sono comunque molto buoni e si potrebbe pensare di eliminare gli errori dati dalla sovraparametrizzazione con uno smoothing finale dei riusultati.

5.5.4 Campo ai vertici: confronto risoluzioni

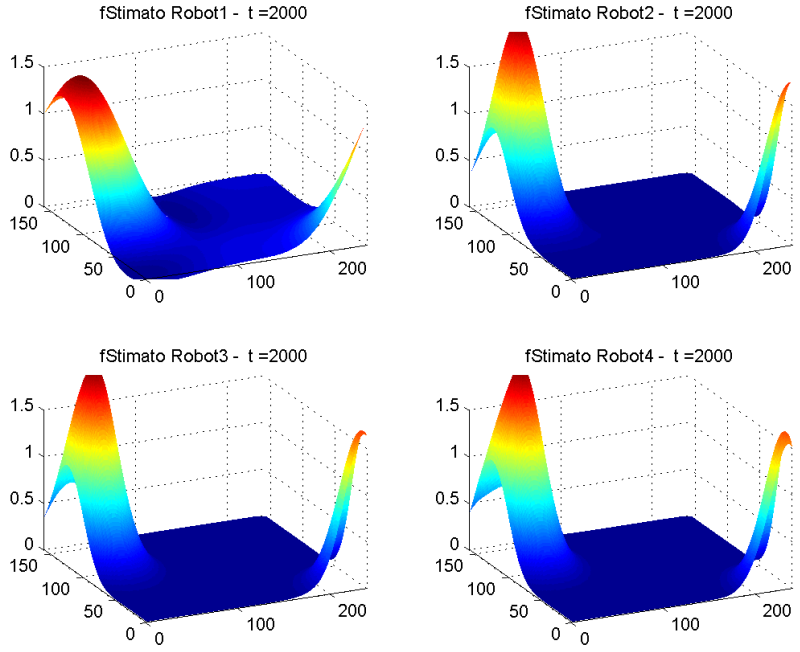


Figura 5.10: Stime campo ai vertici. Risoluzione 2, 3, 4, 5

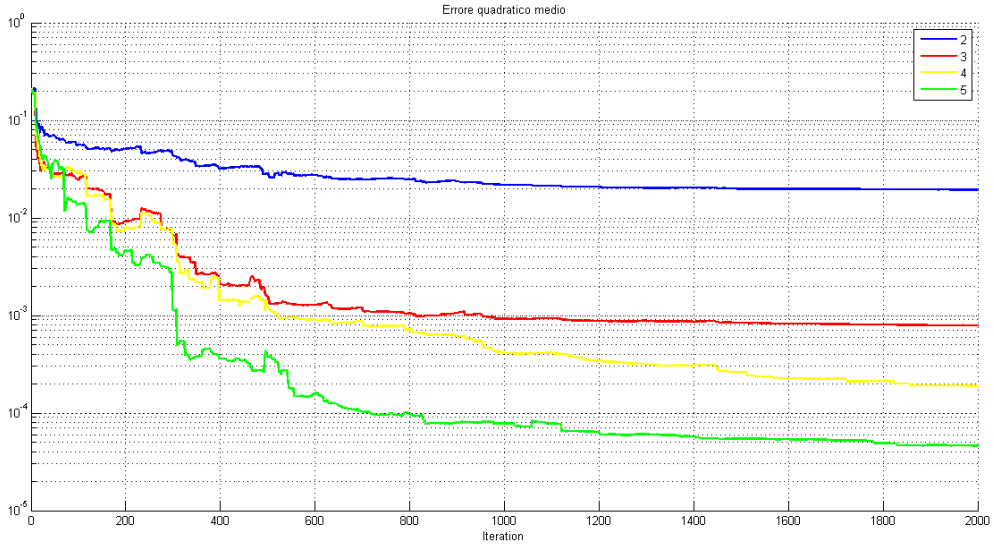


Figura 5.11: Confronti tra gli errori quadratici medi ottenuti con le diverse risoluzioni

Per questo campo molto particolare risulta necessaria un'elevata risoluzione per ottenerne una stima precisa. Si nota infatti come la versione a risoluzione bassa faccia fatica a seguire l'andamento preciso del campo vero. Imputiamo queste problematiche al fatto che la presenza di un ampia zona in cui si registrano misure nulle si traduca in un gradiente generalmente negativo, che smorza l'ampiezza

delle campane e ne aumenta la varianza. Per questi motivi l'aumentare del numero di basi, favorendo la precisione e la selettività della stima nelle diverse zone, contribuisce ad un sensibile miglioramento dei risultati.

5.5.5 Campo molto variabile: confronto risoluzioni

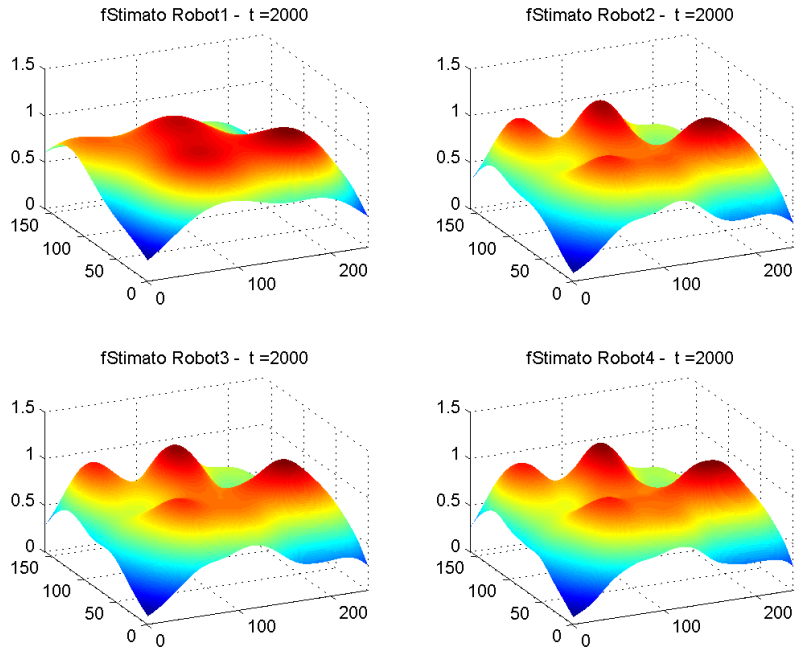


Figura 5.12: Stime campo molto variabile. Risoluzione 2, 3, 4, 5

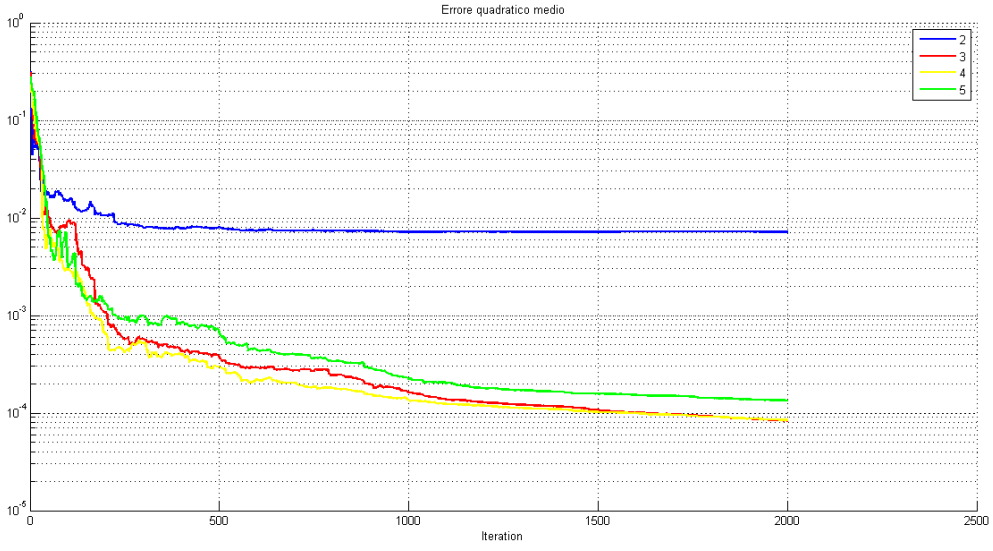


Figura 5.13: Confronti tra gli errori quadratici medi ottenuti con le diverse risoluzioni

La stima di questo campo molto ricco di variazioni, genera risultati ormai prevedibili. Notiamo che la versione con sole 6 gaussiane (risoluzione 2) non risulta essere sufficientemente parametrizzata per ricostruire correttamente l'andamento del campo; sappiamo infatti che questo esempio è stato generato con 12 funzioni. L'algoritmo genera comunque un buon abbozzo dell'andamento reale. All'aumentare della risoluzione poi aumenta anche la precisione della stima, fino a quando, alla risoluzione 5, si nota vagamente il fenomeno della sovraparametrizzazione che affiora, con qualche imprecisione ancora una volta sulle varianze delle funzioni base.

5.5.6 Risultati e limiti

Analizzate le simulazioni precedenti si può concludere che l'algoritmo funziona in linea di massima al meglio che gli è consentito dalla risoluzione utilizzata. Questa risulta infatti un parametro critico per la qualità delle stime ma, come emerge dalla prima parte della relazione, esiste un numero di basi (circa 20) che rappresenta in generale un ottimo trade-off tra pesantezza dell'algoritmo e qualità delle stime. Questo valore si può prendere come riferimento per ottene delle buone stime su qualsiasi campo, noto anche il fatto che oltre questo numero, l'aumento di complessità dato dall'introduzione di nuove basi si traduce solo in una piccolissima riduzione dell'errore quadratico per campi complessi, o addirittura in un innalzamento dello stesso per campi particolarmente semplici.
Notiamo inoltre che il primo fenomeno negativo messo in luce dalle simulazioni, sia in presenza di overfitting (campo poco variabile - risoluzione 5), sia in presenza di underfitting (campo ai vertici - risoluzione 2) è la stima imprecisa delle varianze delle RBF. L'algoritmo in queste situazioni di mal-condizionamento soffre quindi principalmente di questo problema.
In generale comunque la versione ricorsiva si dimostra un metodo di stima veloce e snello, pur consentendo di ottenere ottimi risultati. Riteniamo interessante, a questo punto, confrontarla con l'analoga versione non ricorsiva tratta nel capitolo precedente.

5.6 Confronto ricorsivo vs non ricorsivo

Presentiamo nel seguito il confronto dell'andamento dell'errore quadratico medio della stima di uno stesso campo, utilizzando le 2 versioni dell'algoritmo. Con la versione ricorsiva nelle misure implementata in questo capitolo, si ottengo risultati che sono qualitativamente inferiori rispetto alla versione standard dell'algoritmo. Caratteristica fondamentale però è l'onerosità computazionale, che viene decisamente ridotta con l'algoritmo ricorsivo. Ad esempio l'intera durate temporale della simulazione in causa è di soli 36,1123 secondi per la versione ricorsiva, contro i 275,3719 secondi della versione classica. Il tempo di calcolo è quindi ridotto quasi di un ordine di grandezza.
NB: Si ricorda che ogni iterazione della versione ricorsiva comprende anche la fase di acquisizione della misura, mentre la versione non ricorsiva opera con l'intero pacchetto di dati già noto.

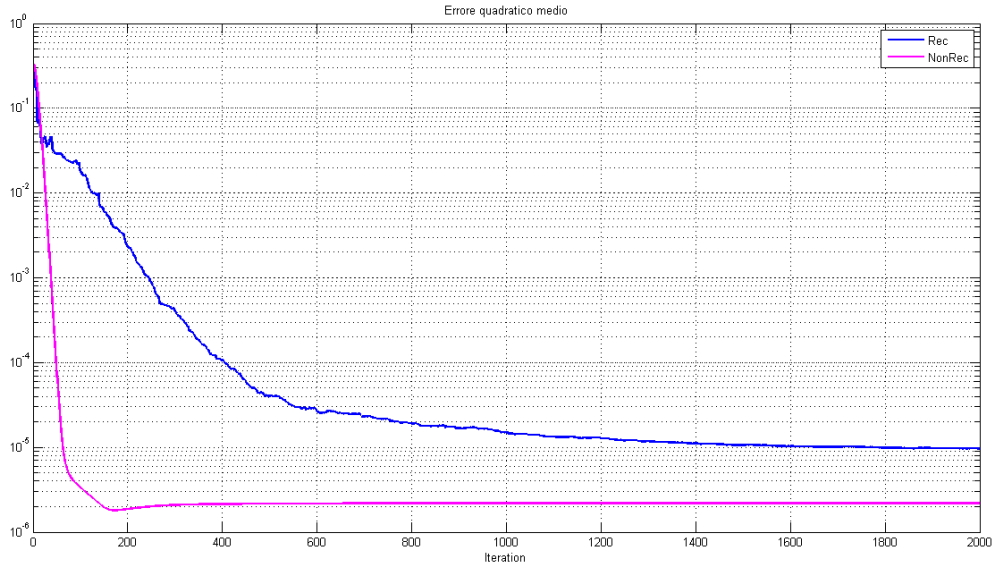


Figura 5.14: Confronto andamento dell'errore quadratico medio della versione ricorsiva e non ricorsiva nelle misure

Parte III

Stima distribuita

Capitolo 6

Stima PEM distribuita con direzione di Newton

Introduzione

Nella parte precedente abbiamo visto come utilizzare gli algoritmi di stima per modelli non lineari applicati al nostro caso, ovvero a reti neurali composte da funzioni RBF. Abbiamo inoltre analizzato le criticità ed i punti forza dell'algoritmo di Quasi-Newton.
La stima veniva eseguita da un unico agente che prendeva le misure su tutta la mappa e, sempre in maniera centralizzata, le elaborava ottenendo la stima del campo.
In applicazioni in cui la mappa è veramente molto grande diventa impossibile, in termini di tempo, stimare il campo tramite un unico agente. Si potrebbe pensare di utilizzare molti agenti per raccogliere i dati mandandoli poi ad un'unità centrale che li elabora e ne trae la stima del campo. Quest'implementazione sarebbe teoricamente fattibile, ma richiederebbe un mezzo di comunicazione molto efficiente, poiché la quantità di dati che i nodi slave dovrebbero inviare al master sarebbe davvero notevole. Se gli agenti si trovassero poi ad una distanza elevata dall'unità centrale, la potenza del segnale dovrebbe essere nettamente superiore rispetto quella che sarebbe invece richiesta per comunicare con un robot vicino. Inoltre l'unità centrale dovrebbe essere dotata di una potenza di calcolo decisamente rilevante per elaborare la massa di dati in tempi ragionevoli.
Utilizzare un *sistema distribuito di agenti* in cui ogni agente ha della capacità di calcolo propria per elaborare i dati risolverebbe questo genere di problemi. Ogni robot però può usufruire dei soli dati che misura ottenendo quindi, in linea di principio, una stima unicamente della propria parte di mappa. Quello a cui si tende è invece una stima globale del campo. La soluzione è quella di utilizzare gli algoritmi di consensus in sistemi distribuiti, ed in particolar modo quelli di *ottimizzazione convessa distribuita* [24]. Tali algoritmi fanno sì che, tramite la comunicazione tra i robot, questi riescano a convergere ad una stima ottima comune pur utilizzando misure su zone della mappa differenti.
Nel seguente capitolo vedremo come adattare tali algoritmi al nostro problema di stima. Infine verranno fatte varie simulazioni per capirne le criticità e le prestazioni in diverse situazioni.

6.1 Stima distribuita: Newton Raphson Consensus

Il problema che si propone adesso è quello di stimare il campo tramite n robot che effettuano misure sulla mappa equamente partizionata. Servendosi di tali misure e potendo "comunicare" tra di loro, si vuole che la stima del campo converga ad un risultato comune per tutti i robot e che ovviamente sia la migliore possibile.
Definiamo $\mathcal{Q}$ l'insieme di punti di tutta la mappa, ed applichiamo una partizione:

Definizione 1 (N-partizione)**.** *Una N-partizione di $\mathcal{Q}$, $\bar{\mathcal{Q}} = \cup_{i=1}^{n}\mathcal{Q}_i$ è una successione ordinata di sottoinsiemi di $\mathcal{Q}$ con le seguenti proprietà:*

- $\cup_{i=1}^{n}\mathcal{Q}_i = \mathcal{Q}$;
- $\mathcal{Q}_i \cap \mathcal{Q}_j = \emptyset \; \forall i \neq j$;
- *ogni sottoinsieme $\mathcal{Q}_i$ è chiuso e non vuoto.*

Ogni robot è inserito in una partizione $\mathcal{Q}_i$ nella quale esegue N_i misure ottenendo un inseme

$$\mathcal{N}_i = \{h(p_1), h(p_2), ..., h(p_{N_i}) \; ; \; p_t \in \mathcal{Q}_i \forall t = 1, ..., N_i\} \tag{6.1}$$

in cui p_t corrisponde al punto in cui viene eseguita la misura. Nel seguito, per una generica misura $h(p_t)$ nel punto p_t, adotteremo la notazione semplificata $h(t)$.
In queste condizioni, adottando la tecnica di stima PEM vista nella parte II, il generico agente *i-esimo*, dato l'insieme di misure $\mathcal{N}_i$ dovrà minimizzare una propria funzione costo pari all'errore quadratico medio:

$$V_N^i(\theta) := \frac{1}{N_i}\sum_{t=1}^{N_i}(\varepsilon_\theta^i(t))^2 = \frac{1}{N_i}\sum_{t=1}^{N_i}(h^i(t) - \hat{h}_\theta^i(t))^2 \tag{6.2}$$

Supponendo di avere n robot, ciò che si vuole ottenere è il θ che minimizza la funzione costo globale pari alla somma di quelle locali, che altro non è che la stima PEM (4.3) dato l'insieme di misure $\mathcal{N} = \cup_{i=1}^{n}\mathcal{N}_i$ quando ogni robot raccoglie un ugual numero di misure:

$$\bar{V}_N(\theta) := \sum_{i=1}^{n} V_N^i(\theta) \tag{6.3}$$

$$\hat{\theta} = \underset{\theta\in\Theta}{\text{Argmin}}\bar{V}_N(\theta) \tag{6.4}$$

Ciò che vorremmo è un algoritmo tale per cui la stima di ogni avente converga a quella ottima, ovvero che:

$$\theta_i(k) \longrightarrow \hat{\theta} \quad \forall i = 1..n \tag{6.5}$$

Questo è lo stesso contesto che si ritrova in [24], seguendo quindi la teoria proposta, applichiamo l'algoritmo di Newton alla funzione costo globale $\bar{V}_N(\theta)$ per ricavarne il θ che la minimizza. Analogamente a (4.12), si ottiene:

$$\theta_{k+1} = \theta_k - \lambda_k \left[\bar{V}_N''(\theta_k)\right]^{-1} \bar{V}_N'(\theta_k) \tag{6.6}$$

andando a sostituire la (6.3):

$$\theta_{k+1} = \theta_k - \lambda_k \frac{\sum_{i=1}^{n} V_N^{i'}(\theta_k)}{\sum_{i=1}^{n} V_N^{i''}(\theta_k)} \tag{6.7}$$

$$= \theta_k - \lambda_k \frac{\frac{1}{n}\sum_{i=1}^{n} V_N^{i'}(\theta_k)}{\frac{1}{n}\sum_{i=1}^{n} V_N^{i''}(\theta_k)} = \theta_k - \lambda_k \frac{y(k)}{Z(k)} \tag{6.8}$$

utilizziamo la seguente notazione semplificata

$$g_i(\theta_k^i) = V_N^{i'}(\theta_k^i) \tag{6.9}$$

$$H_i(\theta_k^i) = V_N^{i''}(\theta_k^i) \tag{6.10}$$

In (6.8) si può osservare che il passo di aggiornamento di Newton è calcolato come il rapporto di due medie, rispettivamente quella dei gradienti e quella delle Hessiane di ogni robot.
Prendiamo quindi la (6.8) come legge di aggiornamento di ogni robot:

$$\theta_{k+1}^i = \theta_k^i - \lambda_k \frac{y_i(k)}{Z_i(k)} \tag{6.11}$$

in cui $y_i(k)$ e $Z_i(k)$ sono una stima della media globale in (6.8). Supponendo che i parametri iniziali siano tutti uguali per ogni agente, i.e., $\theta_0^i = \theta_0^j \quad \forall i, j$, se ad ogni passo k di Newton è verificato

$$y_i(k) = \frac{1}{n} \sum_{i=1}^{n} g_i(\theta_k) \tag{6.12}$$

$$Z_i(k) = \frac{1}{n} \sum_{i=1}^{n} H_i(\theta_k) \tag{6.13}$$

si ha che $\theta_k^i = \theta_k^j \quad \forall i, j$ e converge al valore ottimo $\hat{\theta}$. Le condizioni (6.12) e (6.13) possono essere raggiunte utilizzando due algoritmi di consensus in media, lanciati in parallelo [21].

$$\mathbf{y}(k+1) = [y_1(k)...y_n(k)]^T = P^h \mathbf{y}(k) \tag{6.14}$$

$$\mathbf{Z}(k+1) = [Z_1(k)...Z_n(k)]^T = P^h \mathbf{Z}(k)$$

in cui P è una matrice bistocastica, ed h tende ad infinito.
Siccome le medie "vere" in (6.8) cambiano ad ogni nuovo θ_k, è chiaro che risulta proibitivo, per ogni passo di Newton, dover far ciclare l'algoritmo di consensus fino ad ottenere la stima esatta. Infatti comporterebbe un occupazione del mezzo di comunicazione e una tempo di stima troppo elevato.
La soluzione, presentata in [24], è quella di ottenere un *tracking* delle medie, i.e., le stime delle medie locali $y_i(k)$ e $Z_i(k)$ devono inseguire l'andamento di quelle vere.
Questo fenomeno di tracking può essere ottenuto utilizzando le seguenti formule

$$\begin{aligned} \tilde{\mathbf{y}}(k) &= \mathbf{y}(k-1) + \mathbf{g}(\theta_{k-1}) - \mathbf{g}(\theta_{k-2}) \\ \tilde{\mathbf{Z}}(k) &= \mathbf{Z}(k-1) + \mathbf{H}(\theta_{k-1}) - \mathbf{H}(\theta_{k-2}) \end{aligned} \tag{6.15}$$

Questa particolare implementazione fa si che la stima della media riesca ad inseguire la variazione di quella reale, ma questo solo se il rate di convergenza di $y_i(k)$ e $Z_i(k)$ è sufficientemente più veloce del rate di cambiamento di $\theta(k)$. Infatti se questo meccanismo di inseguimento non è "dominato" dal consensus tra gli $y_i(k)$ e $Z_i(k)$, i θ^i possono non convergere o addirittura l'algoritmo può divergere. A questo punto ci si può rendere conto dell'importanza del tuning di λ_k visto nella sezione 4.2.6 per limitare il passo di Newton. Come abbiamo visto, settando adeguatamente tale parametro, siamo riusciti ad ottenere una dinamica dell'errore di predizione, nonché dei parametri, molto più regolare e lenta. Tutto ciò aiuta le stime $y_i(k)$ e $Z_i(k)$ a stare "vicine" ai valori veri. Per limitare ulteriormente la dinamica dei parametri si può moltiplicare il passo di aggiornamento di Newton per un ulteriore costante $\varepsilon \in [0, 1]$. L'algoritmo che ne segue è il seguente [24].

Algoritmo 2. *Sia $\mathcal{N} = \cup_{i=1}^n \mathcal{N}_i$ un insieme di misure pari all'unione delle misure di ogni robot eseguite nella propria area di competenza $\mathcal{Q}_i$, P matrice di comunicazione doppiamente stocastica. Siano inoltre $\mathbf{g}(\theta_k) = [g_1(\theta_k)...g_n(\theta_k)]^T$ e $\mathbf{H}(\theta_k) = [H_1(\theta_k)...H_n(\theta_k)]^T$ in cui $g_i(\theta_k)$ e $H_i(\theta_k)$ sono calcolati come in (4.9) e (4.13) rispettivamente.*

Inizializzazione

$$\begin{aligned} &\theta_0 = \textit{(campo uniforme)} \\ &\mathbf{g}(\theta_0) = \mathbf{H}(\theta_0) = 0 \\ &\mathbf{y}(0) = \mathbf{H}(0) = 0 \end{aligned}$$

Al generico passo k

1. local update

$$\tilde{\mathbf{y}}(k) = \mathbf{y}(k-1) + \mathbf{g}(\theta_{k-1}) - \mathbf{g}(\theta_{k-2})$$
$$\tilde{\mathbf{Z}}(k) = \mathbf{Z}(k-1) + \mathbf{H}(\theta_{k-1}) - \mathbf{H}(\theta_{k-2})$$

2. consensus step

$$\mathbf{y}(k) = P\tilde{\mathbf{y}}(k)$$
$$\mathbf{Z}(k) = P\tilde{\mathbf{Z}}(k)$$

3. Newton step

$$\theta_k^i = \theta_{k-1}^i - \varepsilon\lambda_k \frac{y_i(k)}{Z_i(k)} \qquad \forall \quad i = 1, 2, ..., n$$

Come mostrato in [24] è possibile migliorare tale algoritmo con una sua forma "velocizzata" denominata *Fast Newton Rhapson Consensus* (FNRC) che riprenderemo nella sezione 6.2.

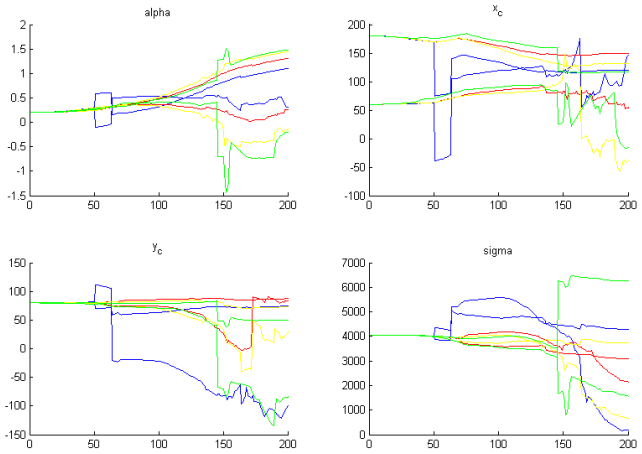
6.1.1 Simulazioni ed alcune considerazioni su ε

Come detto precedentemente, il consensus tra gli $y_i(k)$ e $Z_i(k)$ deve essere dominante rispetto la dinamica dei θ. E' quindi necessario far in modo che le stime delle medie $y_i(k)$ e $Z_i(k)$ stiano sempre "vicine" a quelle vere. A tal scopo abbiamo presentato la soluzione del fattore di smorzamento ε della dinamica dei θ, ma un altro fattore determinante è la legge di consensus che regola la comunicazione tra i robot. E' chiaro che più l'informazione si propaga tra essi ad ogni passo, più veloce sarà la convergenza alle medie esatte, e quindi migliore sarà l'inseguimento al variare dei θ. In questa sezione si vuol far vedere l'importanza di ε e del "protocollo di comunicazione" per ottenere il consensus finale tra i parametri stimati da tutti i robot.
Presenteremo quindi la dinamica dei θ^i di ogni robot per tre tipi di comunicazione, confrontandoli inoltre con tre valori decrescenti di $\varepsilon = \{1, 0.1, 0.01\}$.
I protocolli di comunicazione usati sono:

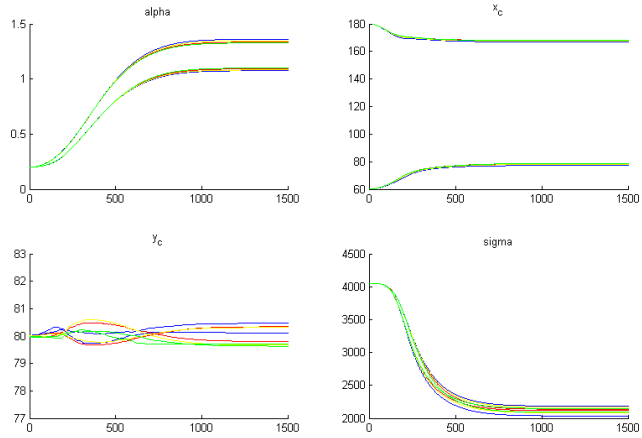
1. **Gossip simmetrico randomizzato:** Ad ogni iterazione vengono selezionati, in maniera casuale con probabilità uniforme, due robot i quali fanno la media tra i relativi $y_i(k)$ e $Z_i(k)$.
2. **Gossip simmetrico deterministico round robin:** A turno tutti i robot comunicano con un interlocutore casuale, facendo la media.
3. **Grafo di comunicazione circolante:** Ad ogni passo tutti i robot comunicano con i due vicini.

Sottolineiamo che tutti e tre i protocolli consentono il consensus in media. Infatti essi utilizzano matrici di comunicazione che sono tutte doppiamente stocastiche. Occorre però far attenzione che, nel caso del gossip simmetrico randomizzato, la convergenza alla media è da intendersi in senso probabilistico. Per ogni realizzazione non si convergerà mai alla media esatta ma ad un suo intorno. Questo come vedremo comporterà degli effetti negativi sull'algoritmo di Newton-Raphson Consensus.

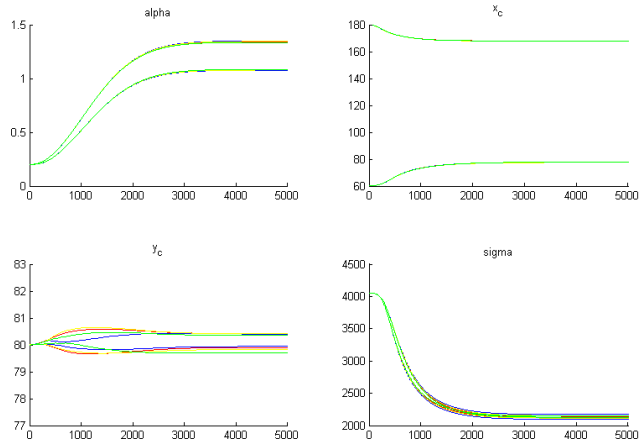
Innanzi tutto è necessario osservare in fig. 6.1a che, utilizzando il gossip simmetrico randomizzato, con $\varepsilon = 1$ l'algoritmo diverge. Questo è un chiaro esempio del fatto che quando il consensus non è dominante è possibile che l'algoritmo diverga. Diminuendo sempre di più ε si vede che il consensus tra i parametri migliora ma non è mai esatto. Analogamente, con il gossip simmetrico deterministico, per $\varepsilon = 1, 0.1$ non si ottiene un consensus esatto, mentre lo si ottiene con $\varepsilon = 0.01$. Il caso migliore è ovviamente con comunicazione circolante in cui, anche con $\varepsilon = 1$, si ottiene consensus esatto. La diversità di comportamento fra i tipi di comunicazione è dovuta al fatto che, come già spiegato, più alto è il rate di convergenza dell'algoritmo di comunicazione più il consensus di $y_i(k)$ e $Z_i(k)$ sarà dominante sul rate di cambiamento di θ. Ovviamente la velocità di convergenza dei tre tipi di comunicazione è in scala crescente rispetto l'ordine delle simulazioni.
Concludendo questa sezione, possiamo dire che affinché si abbia consensus della stima dei parametri è fondamentale che il rate di convergenza al consensus di $y_i(k)$ e $Z_i(k)$ sia significativamente più alto di quello di cambiamento di θ. Questo può essere ottenuto aumentando il primo, utilizzando protocolli di comunicazione migliori che però richiedono mezzi altrettanto efficienti, o diminuendo il secondo rallentando la dinamica di θ tramite l'ε, ottenendo però il risultato finale in tempi più lunghi.



(a) $\varepsilon = 1$.

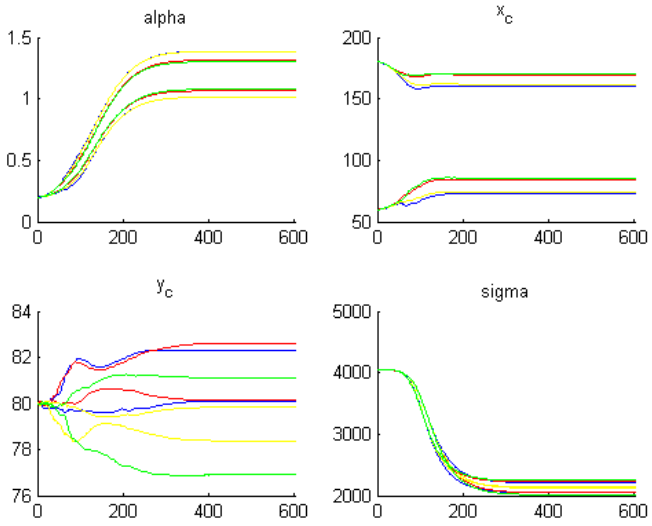


(b) $\varepsilon = 0.1$.

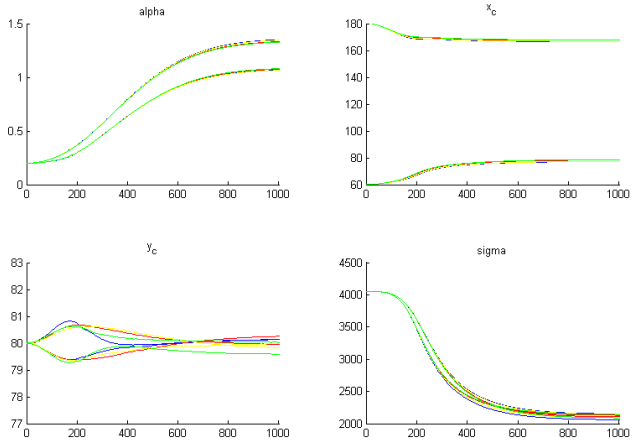


(c) $\varepsilon = 0.01$.

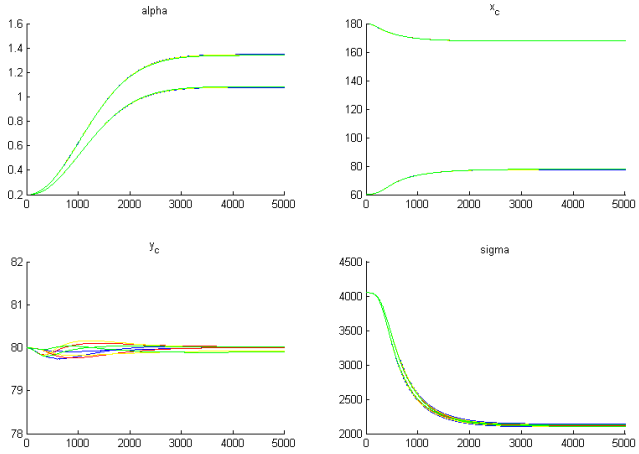
Figura 6.1: Gossip simmetrico randomizzato. Simulazione su campo poco variabile con base iniziale uniforme, risoluzione 1



(a) $\varepsilon = 1$.

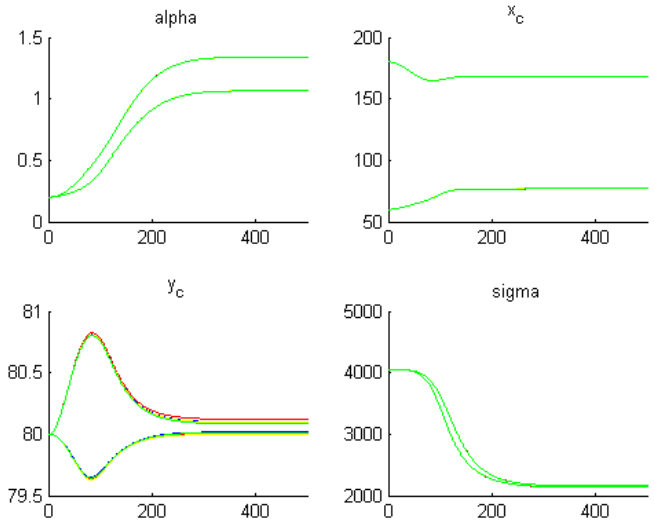


(b) $\varepsilon = 0.1$.

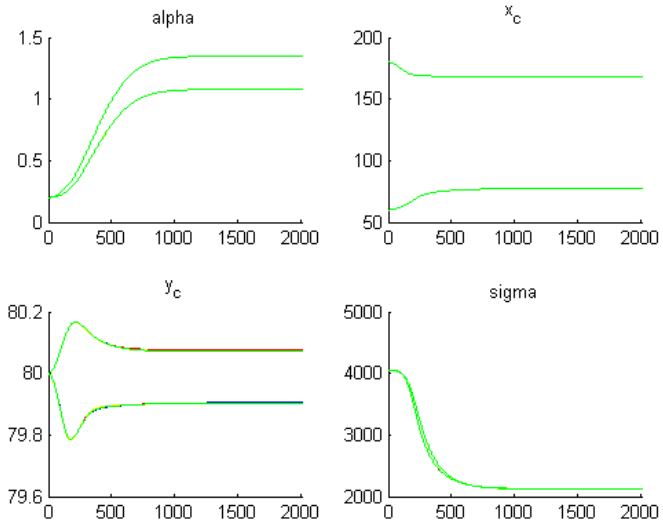


(c) $\varepsilon = 0.01$.

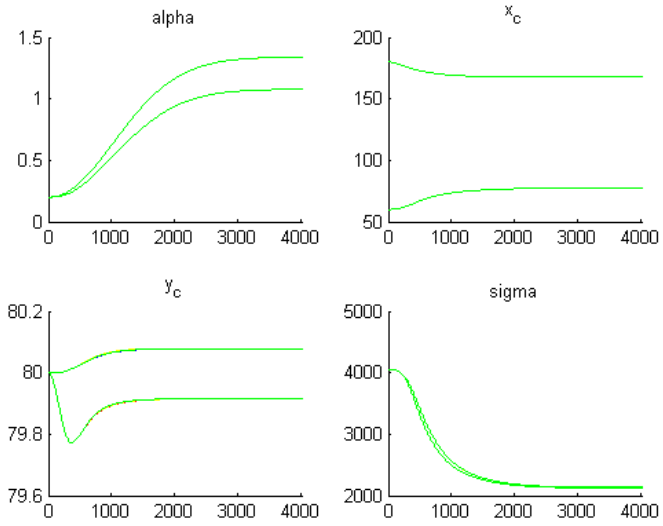
Figura 6.2: Gossip simmetrico deterministico. Simulazione su campo poco variabile con base iniziale uniforme, risoluzione 1



(a) $\varepsilon = 1$.



(b) $\varepsilon = 0.1$.



(c) $\varepsilon = 0.01$.

Figura 6.3: Grafo di comunicazione circolante. Simulazione su campo poco variabile con base iniziale uniforme, risoluzione 1

6.2 Fast Newton Raphson Consensus

In [15] ci si chiede se, nei problemi di consensus, o più in generale nei problemi di bilanciamento dei pesi distribuito, si possa aumentare il rate di convergenza degli usuali algoritmi:

$$\mathbf{y}(k+1) = P\mathbf{y}(k) \tag{6.16}$$

in cui P è una matrice doppiamente stocastica. Infatti, questa soluzione, denominata *first-order schedules*, pur essendo la più utilizzata in pratica, risulta di frequente essere lenta. La soluzione proposta in [15], denominata *static second-order schedules* (SSOS) è quella di utilizzare della "memoria", i.e. l'informazione al passo precedente $\mathbf{y}(k-1)$, ottenendo così una nuova regola di aggiornamento:

$$\mathbf{y}(k+1) = \varphi P\mathbf{y}(k) + (1-\varphi)\mathbf{y}(k-1) \tag{6.17}$$

Questa soluzione prevede quindi di aggiornare la stima delle medie con una combinazione tra l'informazione derivante dal passo di consensus e la stima al passo precedente. E' mostrato che se $\varphi \in (-\infty, 0] \cap [2, +\infty)$, (6.17) può non convergere. Invece se $\varphi \in (0, 2)$ si possono distinguere due diversi comportamenti:

- Se φ è fissato in maniera indipendente da P, tale che $0 < \varphi < 1$, l'uso di (6.17) non porta sostanziali miglioramenti;
- Se invece φ dipende da P, con $1 < \varphi < 2$, allora SSOS converge molto più rapidamente. Una facile ed efficacie scelta per correlare φ a P è utilizzare la relazione

$$\varphi = \frac{2}{(1 + \sqrt{1-\gamma^2})} \tag{6.18}$$

  dove $\gamma = \text{esr}(P)$, ovvero l'autovalore maggiore di P diverso da uno.

L'Algoritmo.2 può quindi essere modificato ottenendo la sua versione "velocizzata".

Algoritmo 3 (Fast Newton-Raphson Consensus)**.** *Sia $\mathcal{N} = \cup_{i=1}^{n}\mathcal{N}_i$ un insieme di misure pari all'unione delle misure di ogni robot eseguite nella propria area di competenza $\mathcal{Q}_i$, P matrice di comunicazione doppiamente stocastica. Siano inoltre $\mathbf{g}(\theta_k) = [g_1(\theta_k)...g_n(\theta_k)]^T$ e $\mathbf{H}(\theta_k) = [H_1(\theta_k)...H_n(\theta_k)]^T$ in cui $g_i(\theta_k)$ e $H_i(\theta_k)$ sono calcolati come in come in (4.9) e (4.13) rispettivamente.*

Inizializzazione

$$\begin{aligned}&\theta_0 = \textit{(campo uniforme)}\\ &\mathbf{g}(\theta_0) = \mathbf{H}(\theta_0) = 0\\ &\mathbf{y}(0) = \mathbf{H}(0) = 0\end{aligned}$$

Al generico passo k

1. *local update*

$$\begin{aligned}\tilde{\mathbf{y}}(k) &= \mathbf{y}(k-1) + \mathbf{g}(\theta_{k-1}) - \mathbf{g}(\theta_{k-1})\\ \tilde{\mathbf{Z}}(k) &= \mathbf{Z}(k-1) + \mathbf{H}(\theta_{k-1}) - \mathbf{H}(\theta_{k-1})\end{aligned}$$

2. *consensus step*

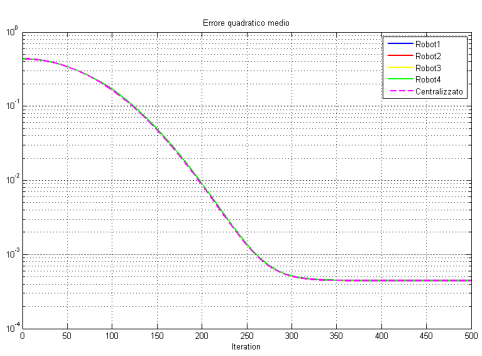
$$\begin{aligned}\mathbf{y}(k) &= \varphi P\tilde{\mathbf{y}}(k) + (1-\varphi)\mathbf{y}(k-1)\\ \mathbf{Z}(k) &= \varphi P\tilde{\mathbf{Z}}(k) + (1-\varphi)\mathbf{Z}(k-1)\end{aligned}$$

3. *Newton step*

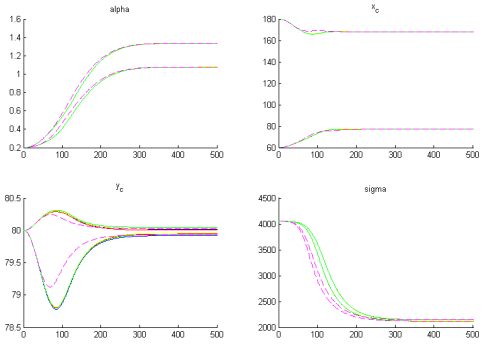
$$\theta_k^i = \theta_{k-1}^i - \varepsilon\frac{y_i(k)}{Z_i(k)} \qquad \forall \quad i = 1, 2, ..., n$$

6.3 Confronto tra stima distribuita e stima centralizzata

Dopo aver accertato che le stime di tutti i robot convergono ad una stima unica, ci si può chiedere se tale stima è uguale a quella del caso centralizzato. Tale domanda è lecita in quanto, se la stima distribuita portasse ad un risultato diverso o addirittura peggiore in termini di errore di predizione rispetto a quella centralizzata, nelle applicazioni che richiedono delle stime molto precise e affidabili, la soluzione distribuita sarebbe da scartare adottando invece quella centralizzata.

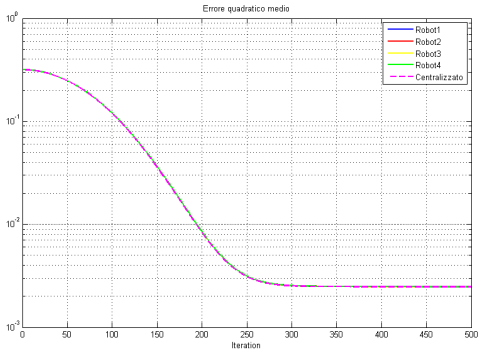


(a) Errore quadratico medio.

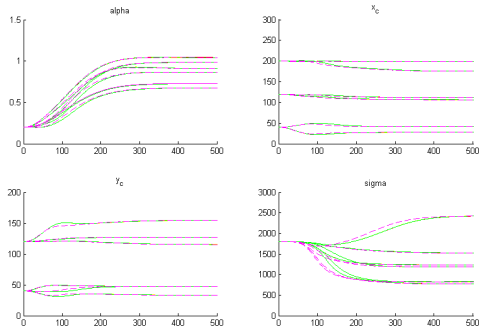


(b) Evoluzione dei parametri.

Figura 6.4: Confronto stima distribuita e stima centralizzata su campo poco variabile (fig. 3.1a).



(a) Errore quadratico medio.



(b) Evoluzione dei parametri.

Figura 6.5: Confronto stima distribuita e stima centralizzata su campo molto variabile (fig. 3.9).

Dalle figure 6.4 e 6.5, osservando l'andamento dei θ possiamo dire che la stima distribuita converge esattamente a quella centralizzata anche se con un numero di iterazioni leggermente superiore. Di contro però si ha che il carico computazionale di ogni agente è molto ridotto rispetto quello nel caso centralizzato. Infatti, supponendo che ogni agente raccolga N misure, ognuno di essi elaborerà N misure in parallelo. Nel caso centralizzato invece, ad ogni iterazione, dovranno esserne elaborate esattamente $N\cdot$(numero di robot). E' chiaro quindi che, in termini di tempo, un iterazione nel caso centralizzato durerà molto di più rispetto alla stessa nel caso distribuito. Questo ci porta a concludere che la stima distribuita potrà fornire la soluzione corretta in molto meno tempo. In applicazioni in cui il tempo di stima è una specifica chiave, la soluzione distribuita risulterebbe l'unica realmente fattibile.

Capitolo 7

Stima PEM Ricorsiva distribuita

Introduzione

Nei capitoli precedenti abbiamo illustrato lo sviluppo del nostro progetto. Partendo dall'analisi della versione lineare dell'algoritmo abbiamo identificato quale fosse il numero di basi ideale per la stima e quali fossero le strade corrette da seguire per ottenere un algoritmo snello, ottimizzato e applicabile al problema reale. Si è deciso così di perseguire la strategia di Metodi a Errore di Predizione (PEM) con funzione di costo quadratica. Una volta implementata e consolidata la versione centralizzata non ricorsiva, sono state sviluppate, da una parte la distribuzione dell'algoritmo di stima tramite *consensus* tra gli agenti e, dall'altra, la modalità *ricorsiva* centralizzata.
Dopo aver conseguito e collaudato gli obiettivi precedenti è stato possibile portare a conclusione il progetto sviluppando la stima in maniera **ricorsiva** e **distribuita**. La ricorsività dell'algoritmo si dimostra necessaria per soddisfare l'esigenza di tempo-reale insita nel tipo di applicazione a cui miriamo; questo perché l'idea di stima del campo distribuita con agenti robotici sottintende la necessità di elaborare le misure raccolte non appena esse si rendano disponibili e nel modo più veloce possibile.
Non si può infatti pensare di attendere la raccolta della totalità dei dati prima di iniziarne l'elaborazione poiché la modalità stessa di acquisizione non pretende un numero preciso di misure ma, bensì, richiede una stima che evolva di pari passo con l'acquisizione delle misure così da poter fermare l'algoritmo quando necessario, ottenendo la miglior stima possibile fino a quel momento.
Non si può nemmeno ricalcolare in maniera non ricorsiva le funzioni caratteristiche dell'algoritmo Newton-Raphson ad ogni nuova misura, questo perché ciò comporterebbe una quantità enorme di calcoli e dei tempi di elaborazione talmente elevati da occupare la maggior parte del tempo utile, mantenendo fermi gli agenti robotici.
Per quanto riguarda la strategia implementativa di quest'ultima parte del lavoro, si basa semplicemente sulla fusione delle parti precedenti, distribuendo la stima PEM ricorsiva con algoritmi di consensus. Come verrà illustrato nel corso del capitolo questa *semplice* strategia ha rivelato più di qualche inconveniente in fase di applicazione. Infatti convogliare due algoritmi, di cui uno non lineare e soggetto a rischi d'instabilità, ha reso evidente la necessità di:

- regolare finemente le funzioni costo e le variabili di progetto disponibili;
- elaborare algoritmi di comunicazione solidi per ottenere un consenso rapido e robusto tra i robot;
- introdurre informazione sulla dinamica locale di ogni agente nell'algoritmo di consenso;
- rallentare la dinamica della stima del parametro $\hat{\theta}(t)$ per sfuggire a eventuali problemi di stabilità.

Nel capitolo seguente andremo a sviscerare passo passo le procedure impiegate e le difficoltà incontrare durante l'implementazione di quest'ultima versione dell'algoritmo. Illustreremo, per concludere, i risultati ottenuti tramite simulazioni di funzionamento sugli ormai noti campi di test generici.

7.1 Algoritmo

L'implementazione per quest'ultima fase del progetto si basa, come già accennato nell'introduzione, nella fusione tra l'algoritmo *RPEM centralizzato* (5.16) e quello di *consensus* (Algoritmo 3). Sostanzialmente il procedimento è quello di ricavare le quantità d'interesse con un passo di Quasi-Newton ricorsivo:

$$\begin{aligned} g(t) &= \gamma(t)\psi(t)\varepsilon(t) \\ H(t) &= H(t-1) + \gamma(t)\left[\psi(t)\psi^T(t) - H(t-1)\right] \end{aligned} \tag{7.1}$$

Dopo aver fatto ciò, si utilizzano questi valori per passarli al consensus:

1. local update

$$\begin{aligned} \tilde{\mathbf{y}}(t) &= \mathbf{y}(t-1) + \mathbf{g}(\theta_{t-1}) - \mathbf{g}(\theta_{t-2}) \\ \tilde{\mathbf{Z}}(t) &= \mathbf{Z}(t-1) + \mathbf{H}(\theta_{t-1}) - \mathbf{H}(\theta_{t-2}) \end{aligned} \tag{7.2}$$

2. consensus step

$$\begin{aligned} \mathbf{y}(t) &= \varphi P\tilde{\mathbf{y}}(t) + (1-\varphi)\mathbf{y}(t-1) \\ \mathbf{Z}(t) &= \varphi P\tilde{\mathbf{Z}}(t) + (1-\varphi)\mathbf{Z}(t-1) \end{aligned} \tag{7.3}$$

3. Newton step

$$\theta_t^i = \theta_{t-1}^i - \varepsilon\frac{y_i(t)}{Z_i(t)} \qquad \forall \quad i = 1, 2, ..., n \tag{7.4}$$

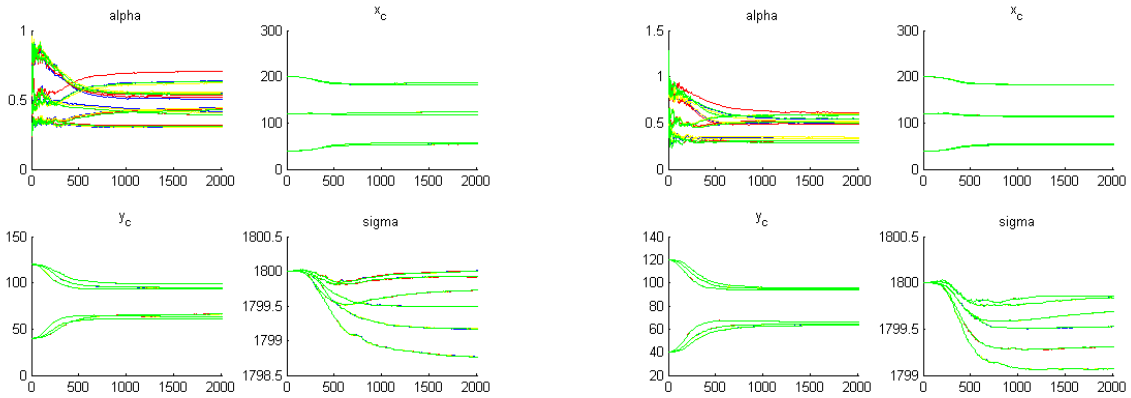
Si sfrutta così la caratteristica di tempo reale dell'algoritmo RPEM, che utilizza solo l'ultima misura per calcolare il gradiente, per implementare in maniera snella ed efficiente il consesus NRC tra i robot.

7.1.1 Robustezza della comunicazione

Il primo inconveniente che si è palesato durante i test del nuovo algoritmo è stato il fatto che quest'ultimo, sfruttando ad ogni iterazione solo l'ultima misura, restituisce passo per passo delle quantità che presentano una variabilità molto superiore rispetto alle stesse nella versione non ricorsiva. Infatti, come si può notare anche dai diagrammi sugli andamenti di *theta* riportati nel paragrafo 5.5.2, l'algoritmo genera un andamento del parametro molto meno fluido e continuo rispetto all'equivalente non ricorsivo, soprattutto nella fase iniziale. Questa caratteristica insita nella procedura si traduce in un elevato rischio di instabilità quando applicato in maniera distribuita. Nello specifico abbiamo notato che con un semplice algoritmo di consensus di tipo gossip randomizzato, dove le comunicazioni tra gli agenti avvengono a coppie e in maniera non molto frequente, l'algoritmo risultava essere instabile, divergendo. Ciò accade perché le variazioni del gradiente e le conseguenti variazioni del parametro risultano appunto essere troppo rilevanti, rischiando di allontanare le medie dei diversi robots tra un passo e l'altro.
Abbiamo perciò deciso di provare a rallentare la dinamica dell'algoritmo di stima favorendo il consensus, aumentando il rapporto tra il numero di comunicazioni e il numero di misure effettuate. Da questi tentativi è risultato che infittendo le comunicazioni tra i robot si ottiene una migliore condivisione della media sulle grandezze caratteristiche $\mathbf{y}(t)$ e $\mathbf{Z}(t)$ e una conseguente convergenza dell'algoritmo.
Per questo motivo nella versione ricorsiva si è dimostrata fondamentale l'applicazione di un protocollo di comunicazione più robusto del semplice gossip. La comunicazione ad anello, che era già stata presentata come un raffinamento nella versione non ricorsiva, risulta quindi una necessità per la versione ricorsiva. L'infittirsi delle comunicazioni e la scelta di renderle deterministiche comporta quindi un appesantirsi del protocollo, svantaggio che però rimane sempre ampiamente compensato dalla notevole riduzione del costo computazionale dovuto al calcolo ricorsivo nelle misure del gradiente della funzione costo.

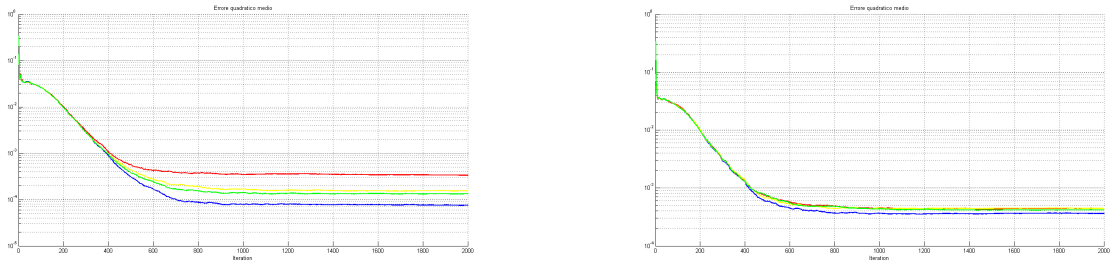
7.1.2 Correzione locale della dinamica

Le scelte illustrate nel paragrafo precedente (7.1.1), generano quindi una convergenza alla media molto robusta. Questa situazione può comportare anche uno svantaggio: c'è il rischio che man mano che la dinamica di θ rallenta convergendo alla media, si stabilizzino degli eventuali offset tra i parametri dei diversi agenti. Ciò accade per causa delle comunicazioni fitte che fanno si che tutti abbiano lo stesso gradiente e la stessa Hessiana, non prendendo più in considerazione le variazioni locali che farebbero convergere i parametri.
Per questi motivi, sempre considerando come l'andamento del parametro risulti meno fluido, l'introduzione del fattore φ che tiene conto della dinamica locale nel consensus (7.3), si dimostra condizione fondamentale per il funzionamento corretto dell'algoritmo.
In particolare riporto nelle figure (7.1) e (7.2) il confronto tra gli andamenti con o senza l'introduzione della dinamica locale. Si nota che, nel caso $\varphi = 1$, cioè senza considerare l'informazione locale, si possono ottenere stime anche migliori su uno degli agenti ma, le stesse, risultano non utili allo scopo finale, in quanto rimangono isolate dagli altri non convergendo a consensus. Si ipotizza quindi che un parametro abbia preso una direzione migliore degli altri, direzione che però viene seguita parallelamente dagli altri, senza convergenza, per causa della modalità di comunicazione. Introducendo invece il fattore φ, scelto secondo l'equazione (6.18), si ottiene un'ottima convergenza al consensus, seppur aspettandosi di poter ottenere una qualità della stima media che è leggermente inferiore alla migliore ottenuta nel caso precedente.



(a) $\varphi = 1$ (b) φ dato dalla (6.18)

Figura 7.1: Andamento dei parametri



(a) $\varphi = 1$ (b) φ dato dalla (6.18)

Figura 7.2: Andamento dell'errore quadratico medio

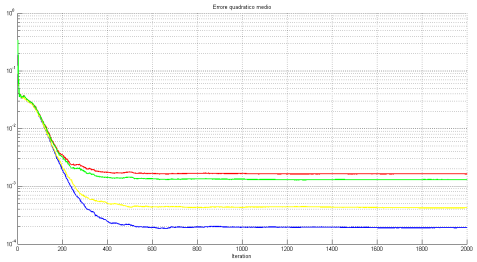
NB: purtroppo i grafici sull'andamento dei parametri sono come sempre poco rappresentativi perché richiederebbero uno zoom maggiore. D'altra parte è stata scelta questa rappresentazione per avere una visione globale del fatto che nella seconda immagine il colore dominante è sempre il verde (ciò denota perfetta sovrapposizione dei parametri), mentre nella prima si vede come i parametri non convergano. I grafici sull'errore quadratico sono invece molto significativi; si vede infatti come esso converga per i 4 robot nel secondo caso.

7.1.3 Tuning variabili di progetto

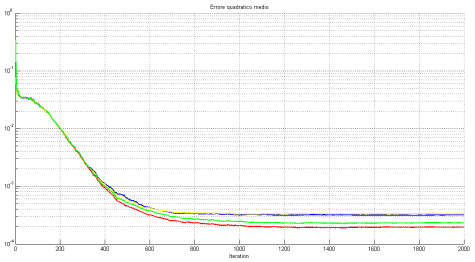
Come illustrato nei paragrafi precedenti, l'implementazione distribuita dell'algoritmo richiede di rallentare la dinamica di Newton per favorire quella del consensus. Questa esigenza si traduce nella necessità di scegliere un **fattore d'oblio** che smorzi la rilevanza dei dati in maniera meno rapida del corrispettivo per la versione centralizzata, ottenendo così un andamento dei parametri più lento ma duraturo. Si è così deciso di utilizzare i seguenti valori:

$$\lambda_0 = 0,997 \quad , \quad \lambda(0) = 0,97$$

Con questo settaggio, come già illustrato nel paragrafo (5.5.1),si ottiene la curva $t\gamma(t)$ che rispetta appena i limiti delle specifiche di progetto e, di conseguenza, la dinamica dei parametri più morbida possibile. Riportiamo (fig. 7.3) il confronto tra gli errori quadratici medi con le scelte ottime sopracitate e con quelle ottime per la versione centralizzata (campo utilizzato: poco variabile).



(a) $\lambda_0 = 0,995 \quad , \quad \lambda(0) = 0,95$



(b) $\lambda_0 = 0,997 \quad , \quad \lambda(0) = 0,97$

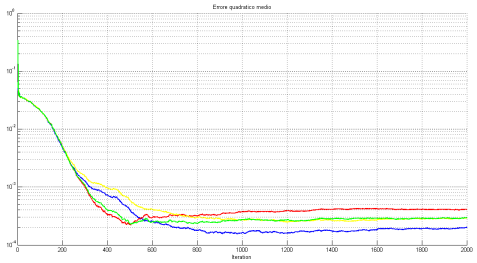
Figura 7.3: Andamento dell'errore quadratico medio nei 2 casi

7.1.4 Aggiustamento della dinamica

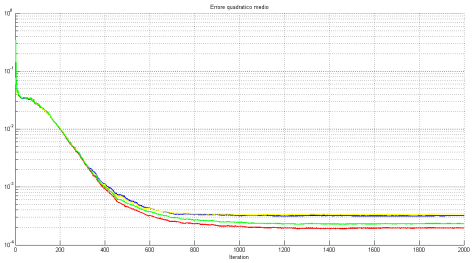
Per l'ottimizzazione dell'algoritmo ricorsivo si è dimostrato utile introdurre una matrice di pesi costante $\Lambda(t) = \bar{\Lambda}$ (quadrata) nell'aggiornamento della stima di $\hat{\theta}(t)$ (come suggerito dalla versione pesata esposta in [11, pp. 92-94]). Questo tipo di matrice è di forma diagonale e i valori sulla diagonale servono per accelerare la dinamica specifica dell'aggiornamento di alcuni parametri rispetto ad altri. Nel nostro caso i parametri α_i non hanno bisogno di essere modificati poiché compaiono linearmente nella funzione costo. Viceversa, i parametri σ_i, x_i, y_i compaiono non linearmente, si è deciso perciò di raddoppiarne la velocità di variazione, con un valore 2 nelle posizioni relative della matrice $\bar{\Lambda}$.

$$\begin{aligned} \hat{\theta}(t) &= \hat{\theta}(t-1) + \gamma(t) H^{-1}(t) \psi(t) \bar{\Lambda}^{-1} \varepsilon(t) \\ H(t) &= H(t-1) + \gamma(t) \left[\bar{\Lambda}^{-1} \psi(t) \psi^T(t) - H(t-1) \right] \end{aligned} \tag{7.5}$$

Il risultato finale non ha soddisfatto in realtà le nostre aspettative. Sebbene questo accorgimento sia in grado di migliorare (molto poco) la stima localmente, quando applicato al consensus, non la migliora, ma anzi aumenta la divergenza tra le stime dei diversi agenti e rende la dinamica meno stabile. Imputiamo questo risultato al fatto che l'amplificazione della dinamica locale possa portare anche ad amplificare eventuali differenze tra i parametri, che poi si propagano nel corso della stima.



(a) $\Lambda = \bar{\Lambda}$



(b) $\Lambda = \mathbb{I}$

Figura 7.4: Andamento dell'errore quadratico medio nei 2 casi

7.1.5 Problema dei minimi locali

La problematica si accentua notevolmente nella versione distribuita dell'algoritmo. Per ottenere risultati validi in questo caso siamo costretti ad utilizzare unicamente la versione con base iniziale uniformemente distribuita. In caso contrario l'algoritmo, nel tempo delle 2000 iterazioni canoniche riesce solamente ad abbozzare una stima del campo, mantenendo elevatissimi livelli d'errore. Ad esempio, nel caso riportato nelle figure (7.5) e (7.6), si ottiene una buona stima di uno dei picchi del campo ai vertici, mentre quello di quello più piccolo non vi è traccia.

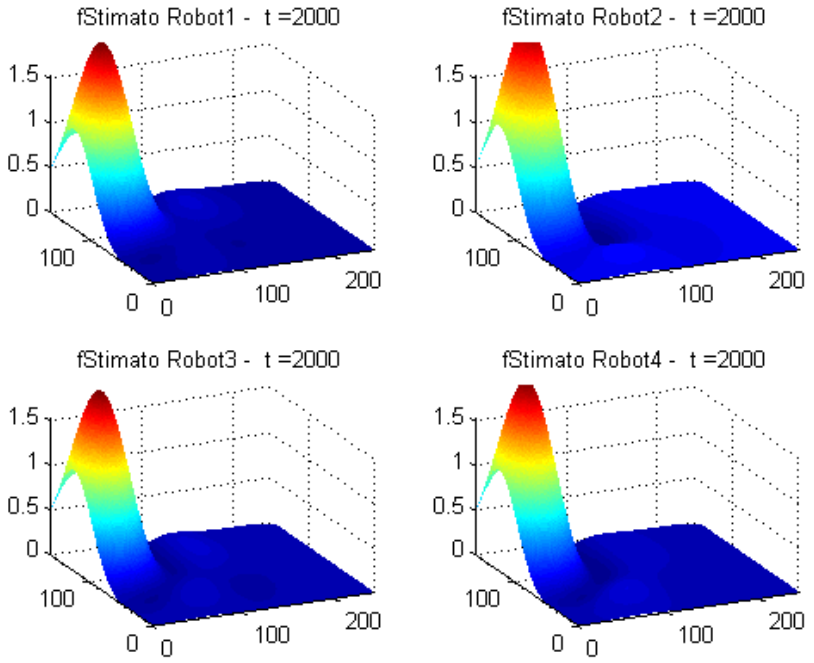


Figura 7.5: Stima campo molto variabile. Configurazione iniziale centrata

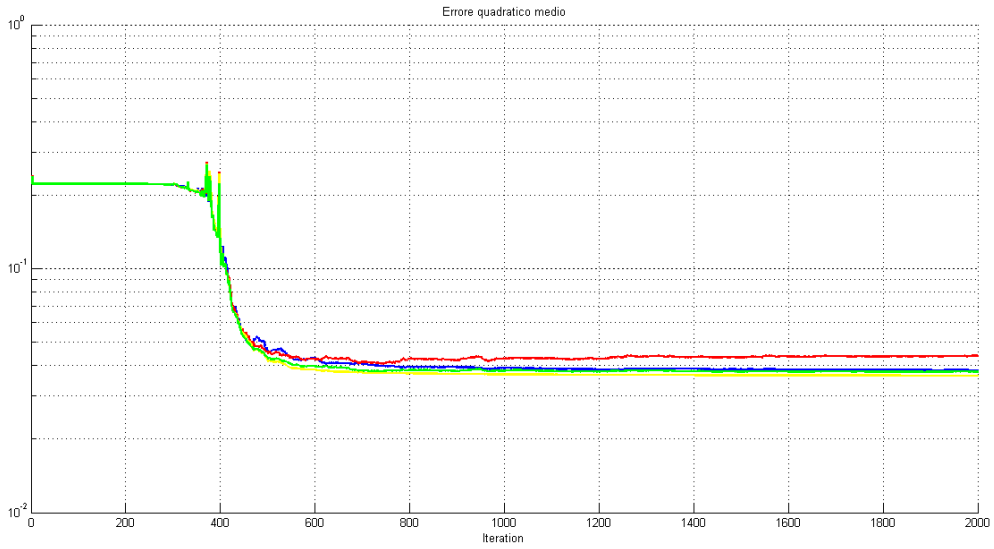


Figura 7.6: Errori quadratici medi dei diversi agenti. Configurazione iniziale centrata

7.2 Simulazioni

Riporto alcune simulazioni esemplificative dei risultati ottenuti con l'algoritmo.

7.2.1 Campo poco variabile

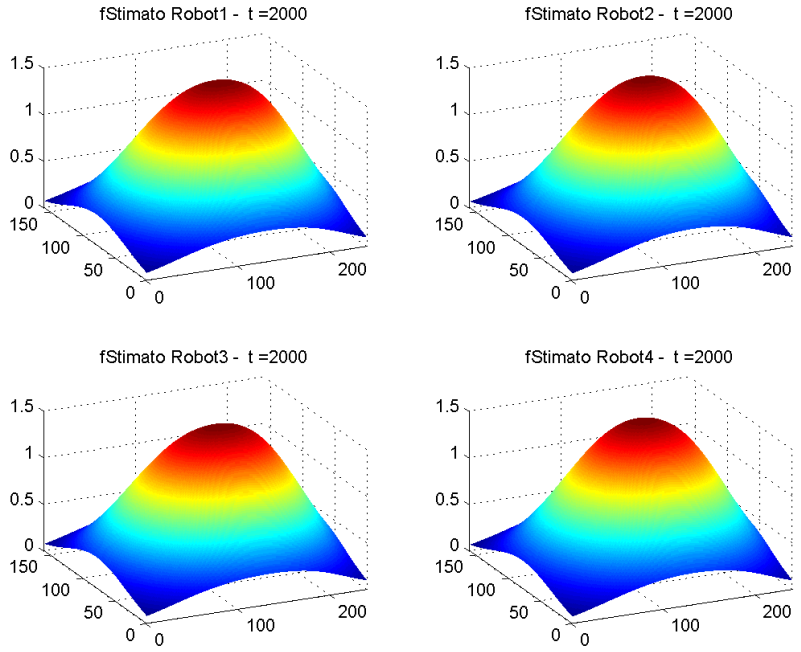


Figura 7.7: Stima campo poco variabile

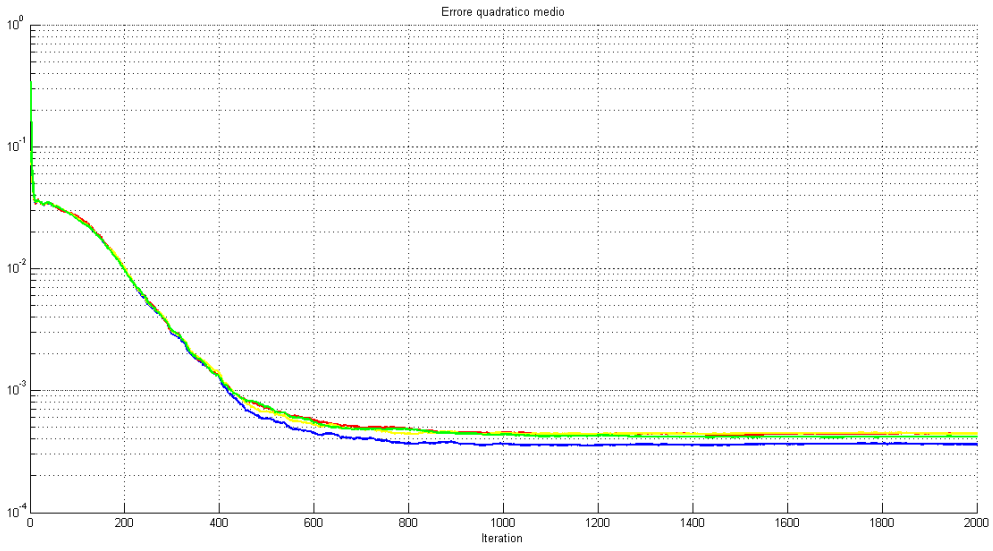


Figura 7.8: Errori quadratici medi dei diversi agenti

Simulazione effettuata con base uniformemente distribuita, risoluzione 2 (numero RBF = 6).

7.2.2 Campo ai vertici

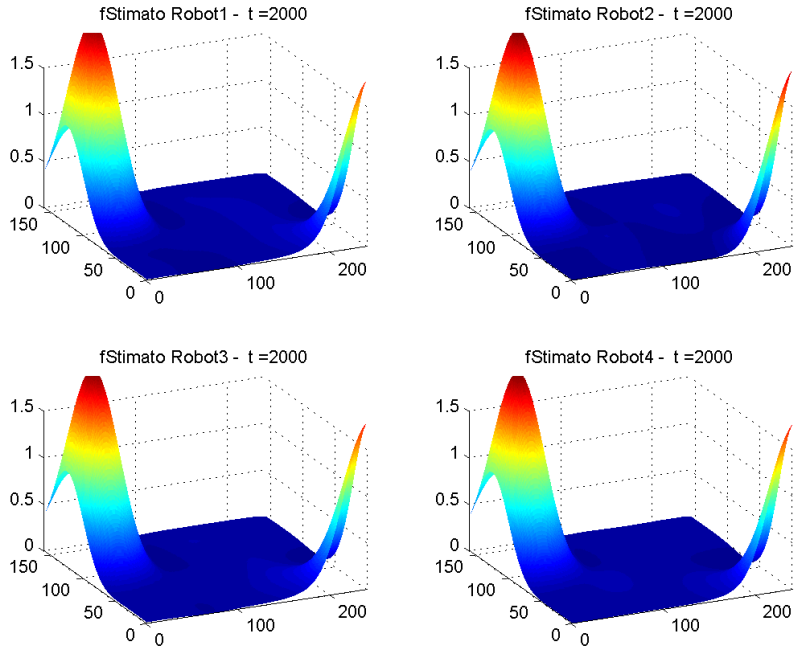


Figura 7.9: Stima campo ai vertici

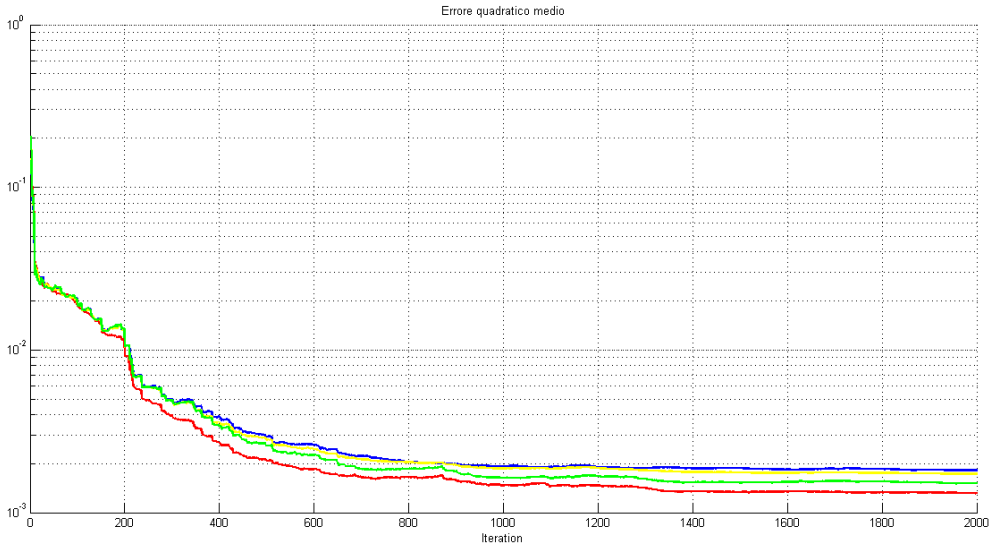


Figura 7.10: Errori quadratici medi dei diversi agenti

Simulazione effettuata con base uniformemente distribuita, risoluzione 3 (numero RBF = 12).

7.2.3 Campo molto variabile

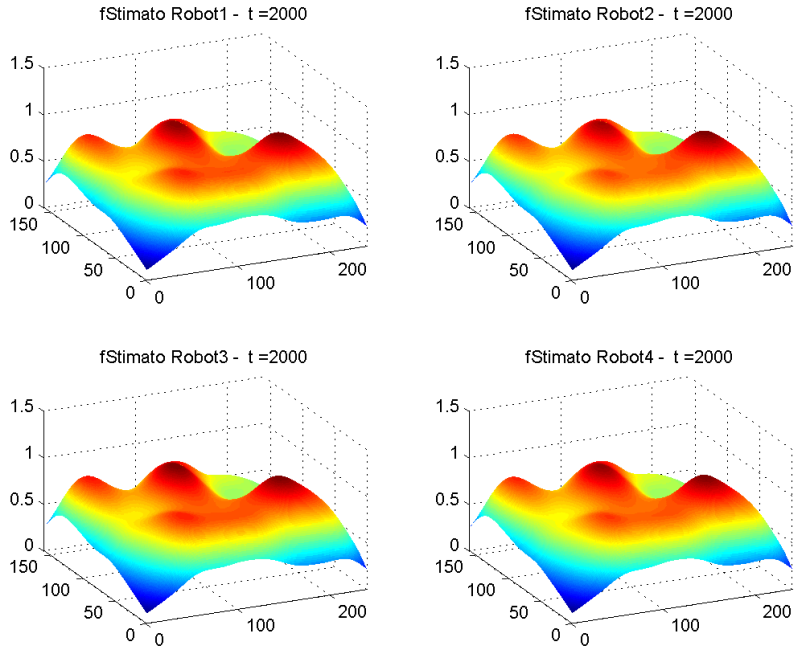


Figura 7.11: Stim1 campo molto variabile

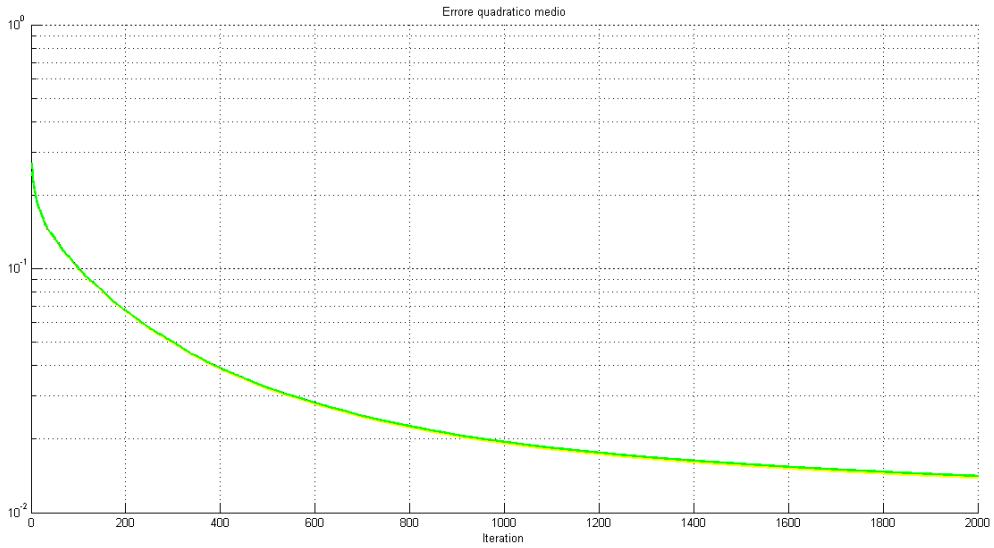


Figura 7.12: Errori quadratici medi dei diversi agenti

Simulazione effettuata con base uniformemente distribuita, risoluzione 4 (numero RBF = 24).

7.3 Confronto centralizzato vs distribuito

Dalle simulazioni si ottengo risultati simili a quelli della versione centralizzata e si nota come gli agenti tendano a raggiungere il consensus nelle loro stime. Purtroppo l'errore quadratico medio si mantiene superiore a quello ottenuto nella versione centralizzata, probabilmente per causa dei diversi accorgimenti utilizzati, peggiorativi per la stima, ma necessari per garantire il consensus.
Riporto nel seguito l'andamento dell'errore quadratico medio ottenuto da una simulazione su campo poco variabile.

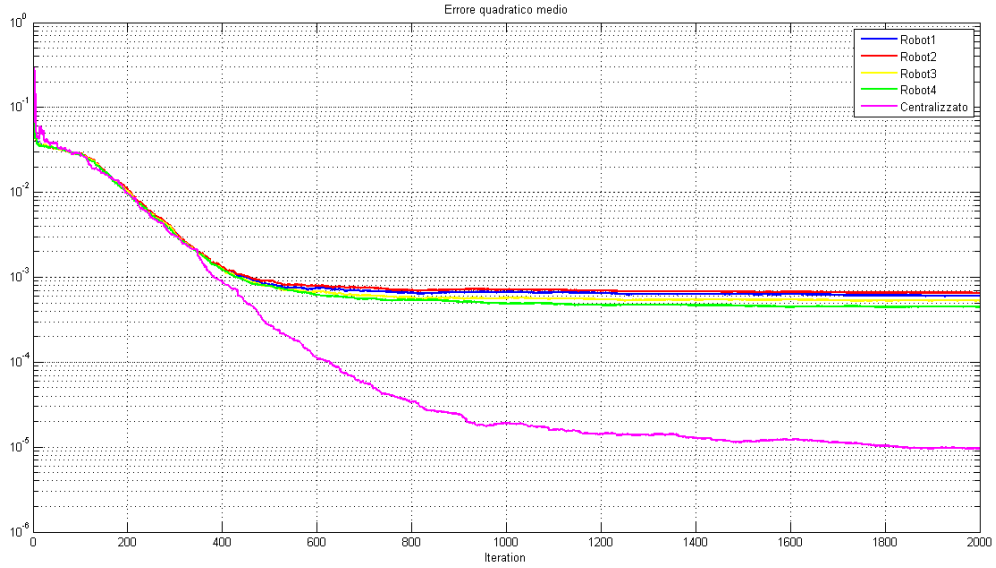


Figura 7.13: Confronto dell'errore quadratico medio della versione distribuita e centralizzata

Conclusioni

Nella prima parte del progetto è stato affrontato il problema di scelta del numero di basi per la stima del campo. Dai risultati ottenuti, con le basi statiche uniforme e centroidale, si è osservato che all'aumentare del numero di funzioni diminuisce l'errore di stima ma aumenta la varianza dei parametri. Si è notato, inoltre, che con numero troppo elevato di RBF la stima peggiora in quanto si verifica il fenomeno dell'overfitting. Questa analisi preliminare della stima ci ha portato a scegliere nelle parti successive un compromesso tra errore di stima e varianza dei parametri.

Nella seconda parte del progetto è stato implementato un algoritmo di stima su modello non lineare. Tale scelta ci ha consentito di posizionare le funzioni radiali in maniera ottima, ottenendo quindi a parità di errore di stima un numero di parametri inferiore, e di conseguenza una varianza, degli stessi, contenuta. Comunque sia la non linearità dell'algoritmo comporta il rischio di incorrere in minimi locali, di conseguenza si dimostra importante la scelta delle condizioni iniziali. Dalle simulazioni si è osservato che le stime migliori si ottengono scegliendo come base di partenza quella uniforme.
Inoltre, è stato fondamentale applicare un metodo di regolarizzazione per non incorrere nella divergenza dell'algoritmo. A tal proposito è stata anche utilizzata una particolare funzione per migliorare la direzione discendente data dal passo di Newton. Questa non è generalmente la soluzione migliore in quanto quest'ultima richiederebbe un costo computazionale molto maggiore, essendo esso già elevato.
Nella versione iterativa si è riscontrato il problema del tuning del fattore d'oblio, il quale risulta determinante ai fini della qualità della stima. Questo algoritmo risente, infatti, fortemente del settaggio delle variabili di progetto quali: valore iniziale e rate di convergenza della sequenza peso. In particolare, configurazioni errate di questi parametri possono generare stime dissimili dal campo vero o fenomeni d'instabilità. Prendendo spunto dalla pratica empirica tratta dalla letteratura, sono stati forniti dei range all'interno dei quali settare i parametri fino ad ottenere ottimi risultati. In ogni caso, purtroppo, la qualità finale della stima è risultata leggermente inferiore rispetto alla corrispettiva nel caso non iterativo; questo peggioramento è ampiamente compensato da una importante riduzione dei tempi di calcolo.

Nella terza parte sono state implementate le versioni distribuite dei precedenti due algoritmi. Con quella non ricorsiva tutti gli agenti convergono allo stesso risultato, identico a quello della versione centralizzata. Per raggiungere tale esito è necessario un numero di passi leggermente superiore rispetto al caso centralizzato ma con un costo computazionale di ogni iterazione molto inferiore.
Per quanto riguarda la versione ricorsiva, invece, l'implementazione è risultata meno efficace in quanto, probabilmente per la natura meno stabile dell'algoritmo, si sono presentati dei problemi di convergenza tra gli agenti. Per cercare di porvi rimedio si sono dovuti adottare degli accorgimenti peggiorativi per la qualità della stima. In particolare si è dovuta rafforzare la comunicazione tra gli agenti, utilizzando un protocollo deterministico ad anello. Inoltre, è stato necessario correggere il consensus applicando l'algoritmo SSOS che introduce un'informazione sulla dinamica locale del nodo.
Altro fenomeno rilevante è l'accentuata dipendenza dalle condizioni iniziali: se nella versione centralizzata, basi diverse potevano condurre a risultati leggermente diversi, in questo caso il settaggio preciso delle variabili di progetto e la scelta della base iniziale sono condizioni necessarie per un esito sensato della stima. Infatti, la modifica delle condizioni iniziali porta a stime che non riescono ad avvicinarsi a sufficienza al campo reale nei tempi presi in considerazione.
Concludiamo quindi, dicendo che, una volta settate accuratamente le condizioni iniziali, si ottengono risultati molto soddisfacenti ed in tempi relativamente brevi. In particolare, rispetto al caso non ricorsivo (in cui bisogna attendere di aver raccolto la totalità dei dati) pur ottenendo un risultato finale di qualità leggermente inferiore, già dopo poche misure si riesce ad avere una stima abbastanza accurata del campo.

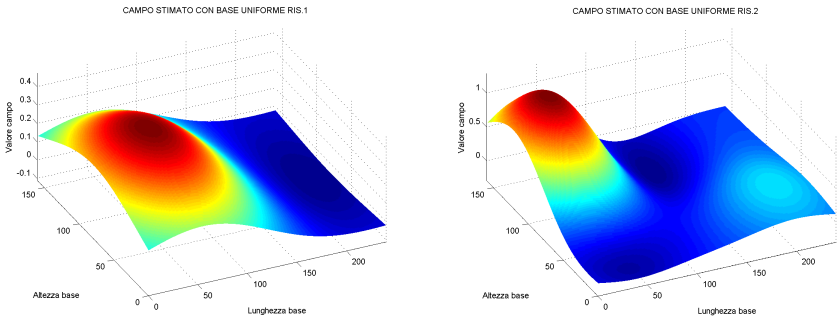
Partendo da questi presupposti, il progetto potrebbe prestarsi a diverse valide estensioni. Ad esempio si potrebbe partizionare in maniera ottima la mappa in aree di competenza in funzione delle zone di massima ampiezza o di massimo gradiente del campo. Così facendo si riuscirebbe a concentrare un più elevato numero di misure nei settori di maggior interesse, ottenendone una stima più precisa ed immediata, sprecando altresì meno risorse per le zone poco rilevanti. Un altro sviluppo interessante potrebbe essere l'implementazione di un algoritmo di stima adattiva del numero ottimo di RBF. In questo modo sarebbe possibile, teoricamente, stimare in maniera precisa qualsiasi campo, senza avere alcun tipo di informazione a priori e senza incorrere in sovra-parametrizzazione. Inoltre, sia la semplice stima lineare ai minimi quadrati, sia la stima PEM con direzione di Newton, viste in questo progetto, risultano sensibili alla presenza di valori anomali (*outliers*) che potrebbero comportare una forte riduzione nella qualità della stima, specialmente nella versione ricorsiva. Di conseguenza potrebbe essere molto utile utilizzare funzioni costo diverse da quelle finora trattate, che tengano meno in considerazione questo tipo di anomalie.

Appendice A

Figure stima centralizzata con basi statiche

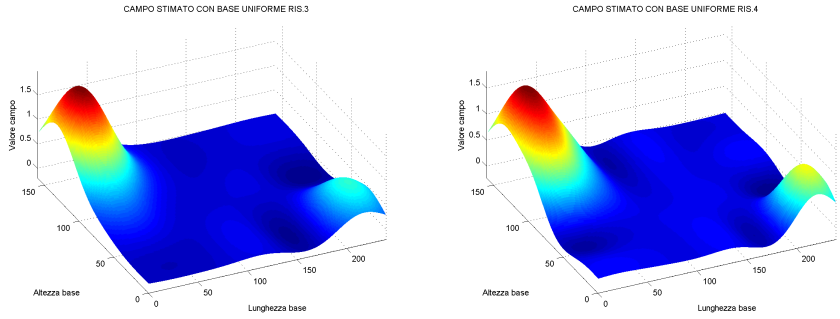
A.1 Campo ai vertici

A.1.1 Base Uniforme



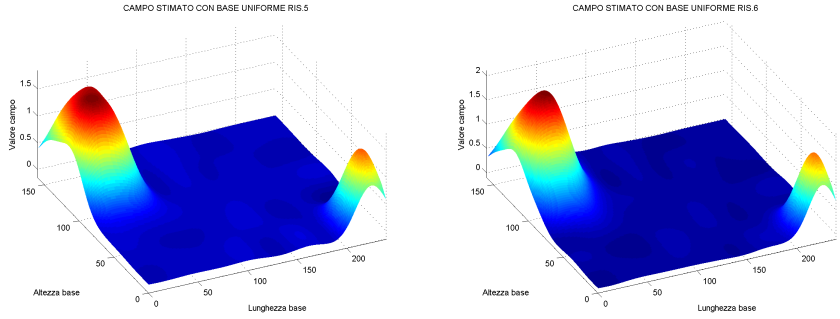
(a) Risoluzione 1

(b) Risoluzione 2



(c) Risoluzione 3

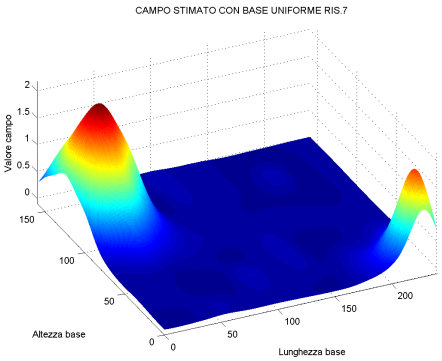
(d) Risoluzione 4



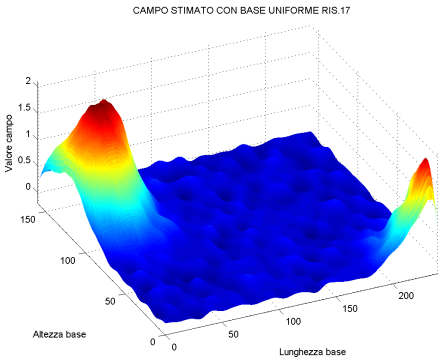
(e) Risoluzione 5

(f) Risoluzione 6

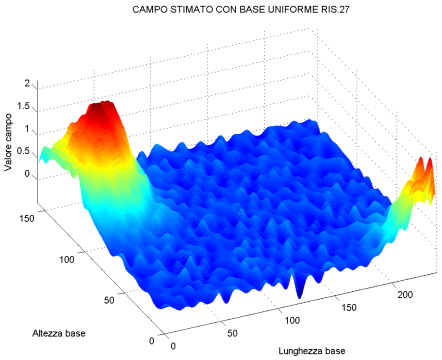
Figura A.1: Evoluzione stima Base Uniforme



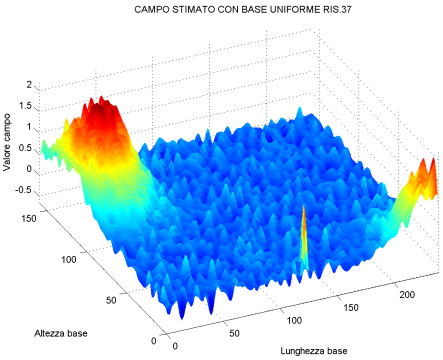
(a) Risoluzione 7



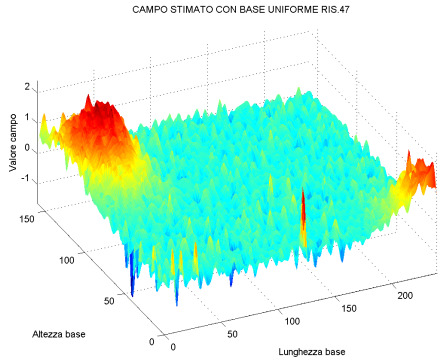
(b) Risoluzione 17



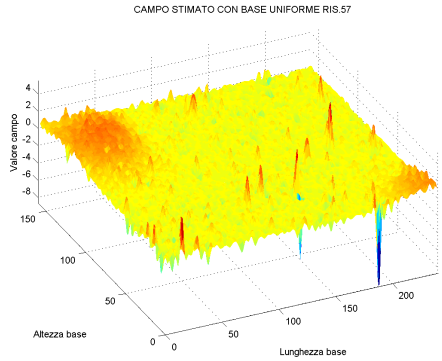
(c) Risoluzione 27



(d) Risoluzione 37



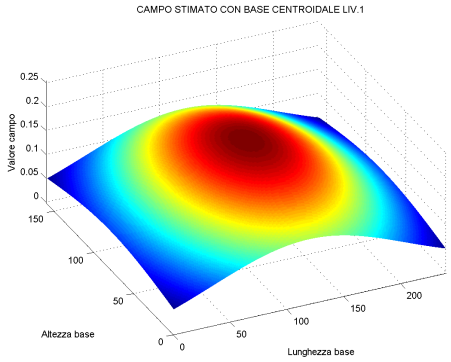
(e) Risoluzione 47



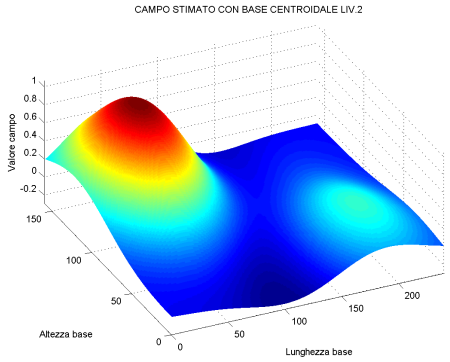
(f) Risoluzione 57

Figura A.2: Evoluzione stima Base Uniforme

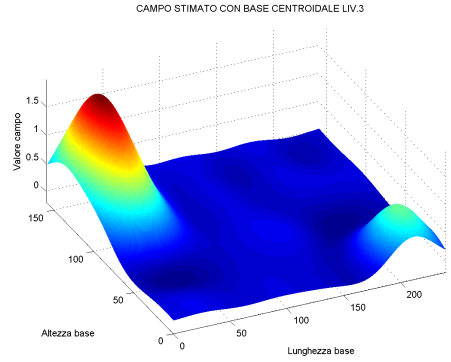
A.1.2 Base Centroidale



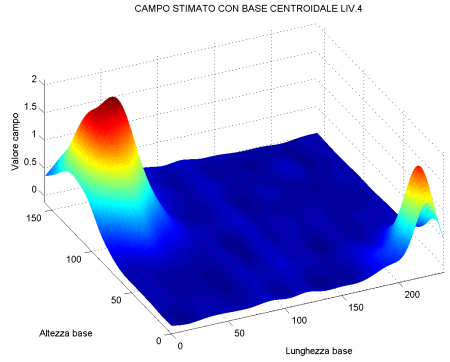
(a) Livello 1



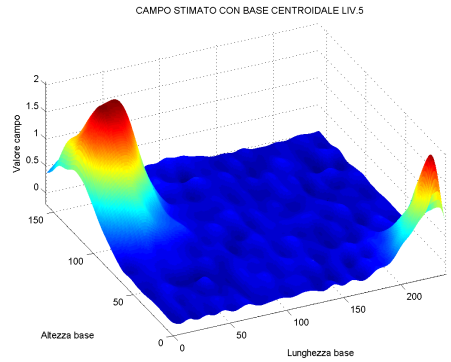
(b) Livello 2



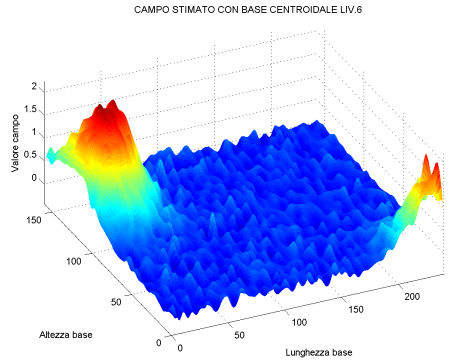
(c) Livello 3



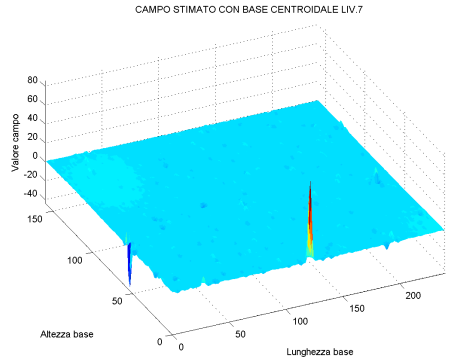
(d) Livello 4



(e) Livello 5



(f) Livello 6

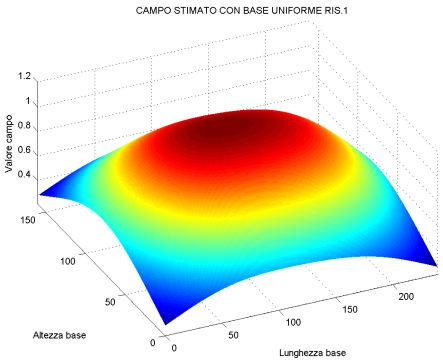


(g) Livello 7

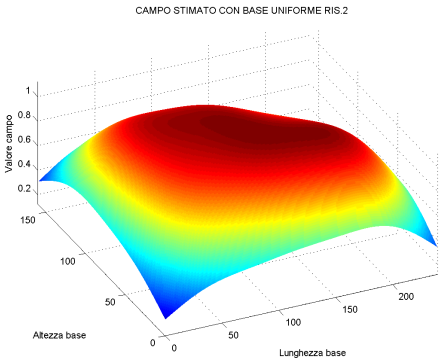
Figura A.3: Evoluzione stima Base Centroidale

A.2 Campo variabile

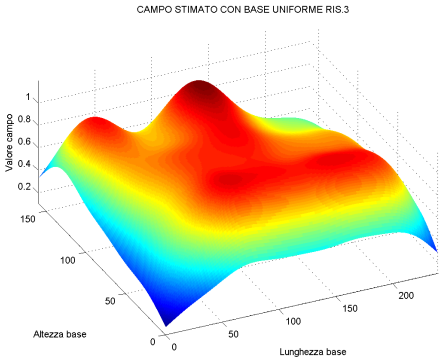
A.2.1 Base Uniforme



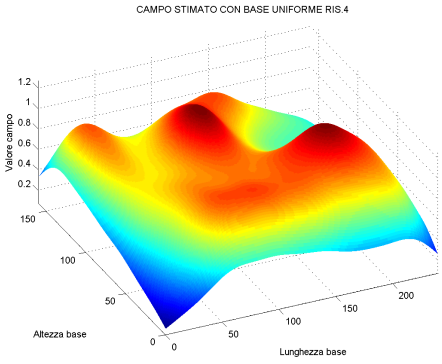
(a) Risoluzione 1



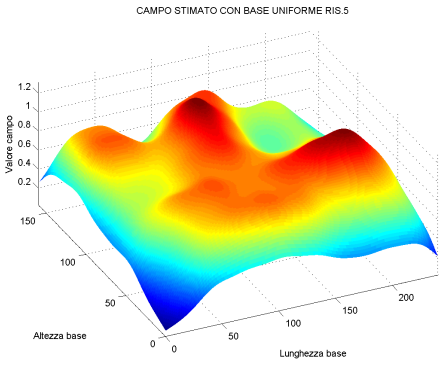
(b) Risoluzione 2



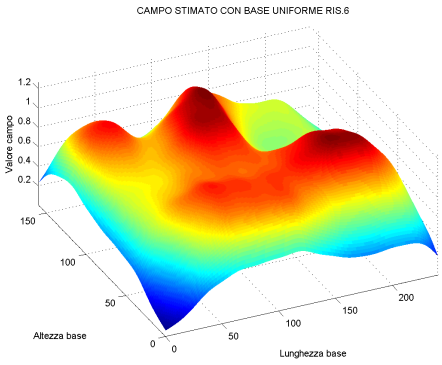
(c) Risoluzione 3



(d) Risoluzione 4

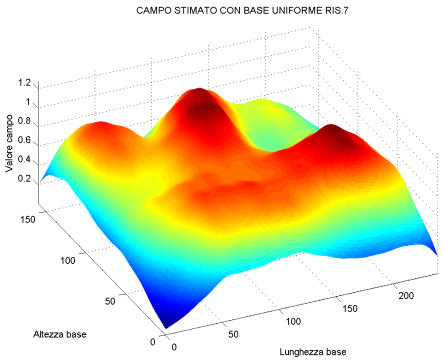


(e) Risoluzione 5

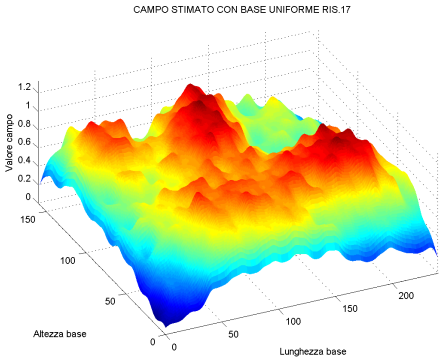


(f) Risoluzione 6

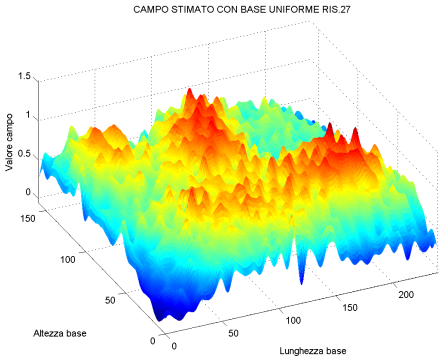
Figura A.4: Evoluzione stima Base Uniforme



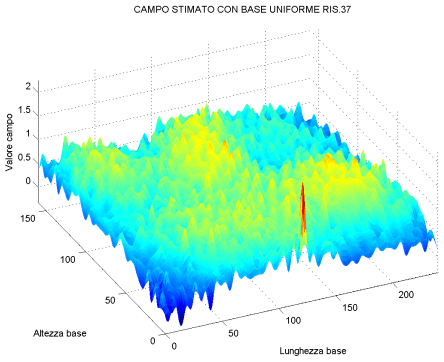
(a) Risoluzione 7



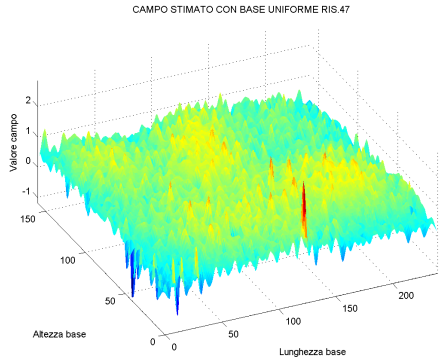
(b) Risoluzione 17



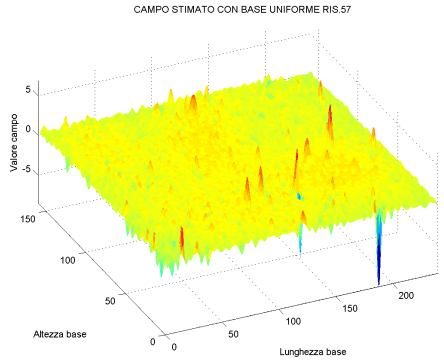
(c) Risoluzione 27



(d) Risoluzione 37



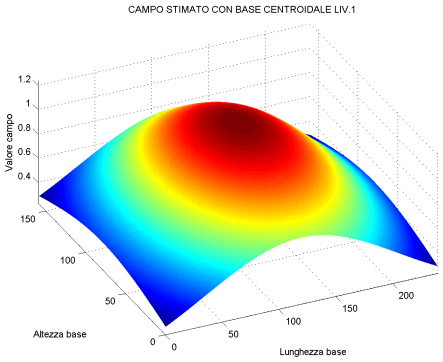
(e) Risoluzione 47



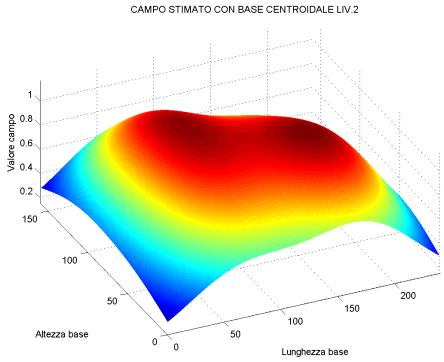
(f) Risoluzione 57

Figura A.5: Evoluzione stima Base Uniforme

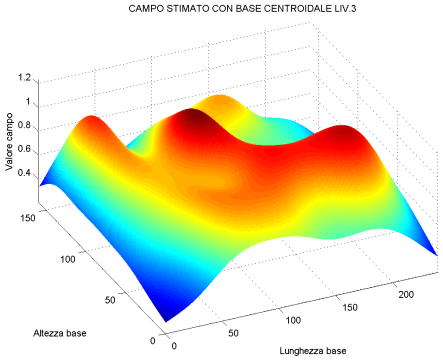
A.2.2 Base Centroidale



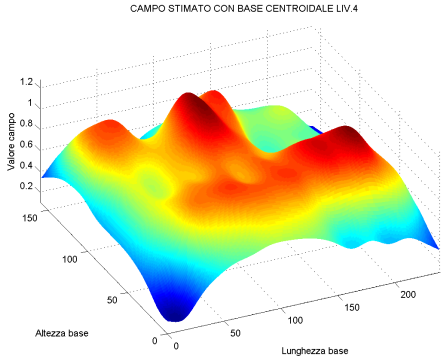
(a) Livello 1



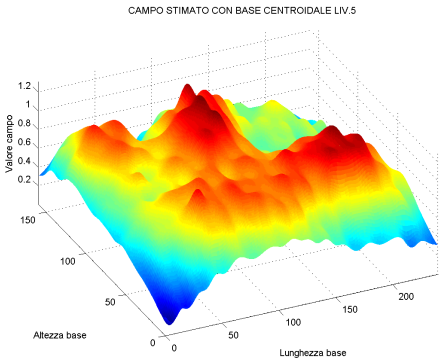
(b) Livello 2



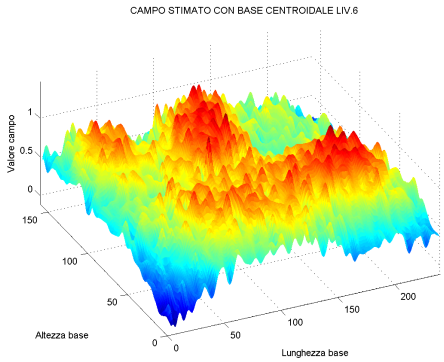
(c) Livello 3



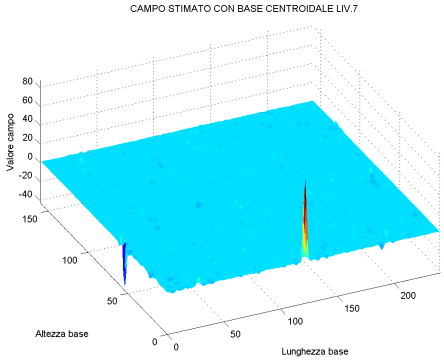
(d) Livello 4



(e) Livello 5



(f) Livello 6

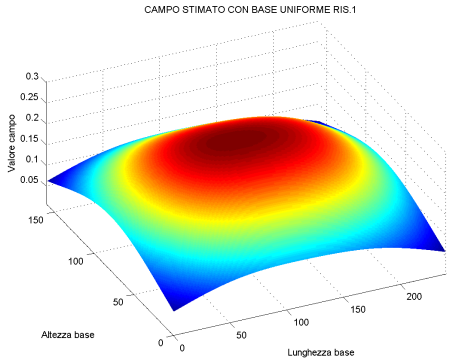


(g) Livello 7

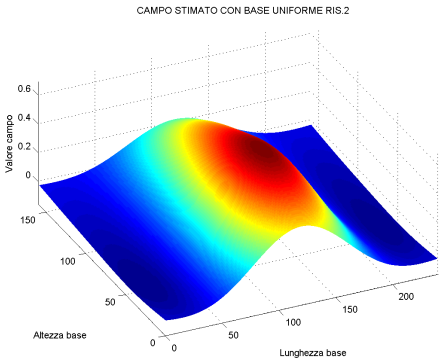
Figura A.6: Evoluzione stima Base Centroidale

A.3 Campo al centro

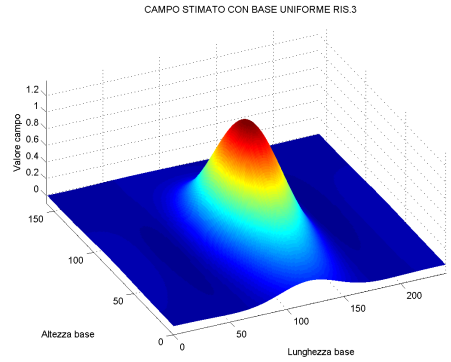
A.3.1 Base Uniforme



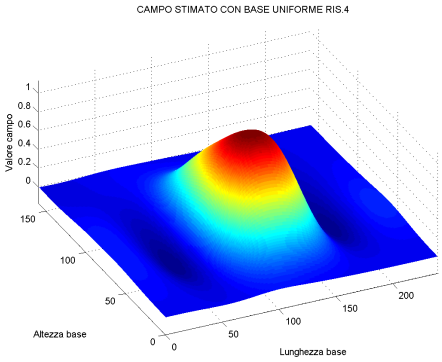
(a) Risoluzione 1



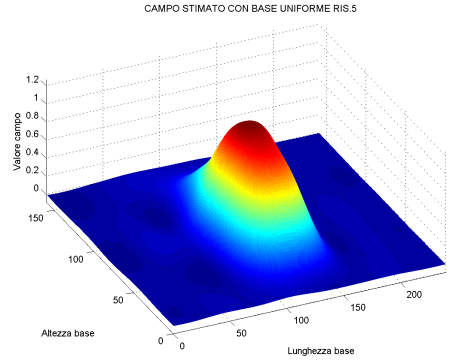
(b) Risoluzione 2



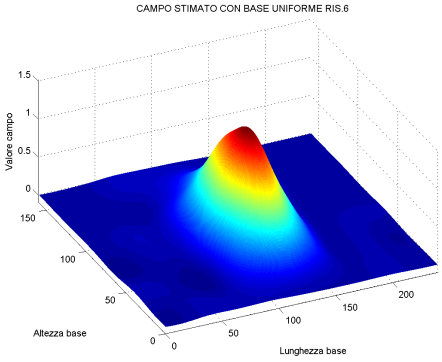
(c) Risoluzione 3



(d) Risoluzione 4

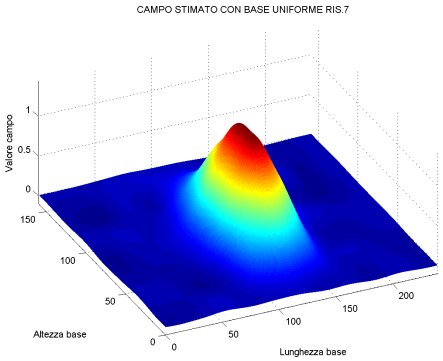


(e) Risoluzione 5

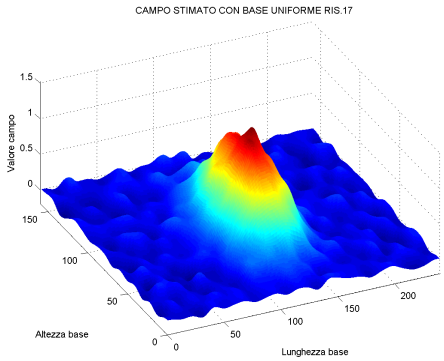


(f) Risoluzione 6

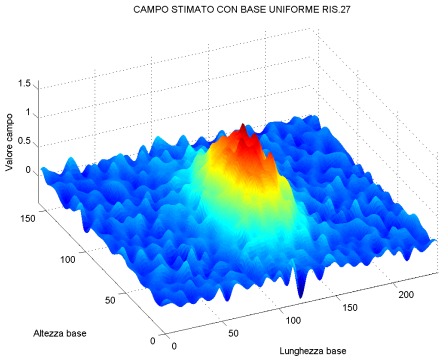
Figura A.7: Evoluzione stima Base Uniforme



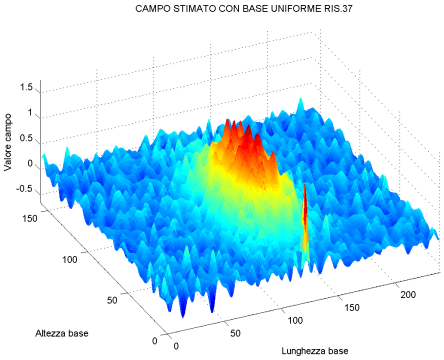
(a) Risoluzione 7



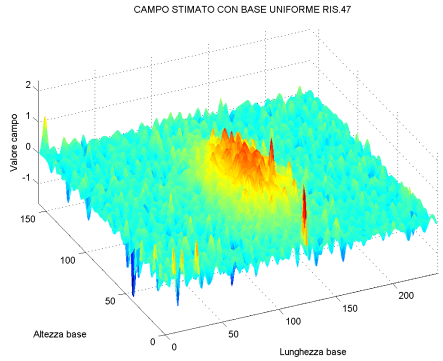
(b) Risoluzione 17



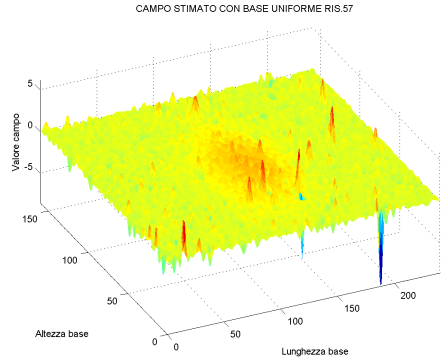
(c) Risoluzione 27



(d) Risoluzione 37



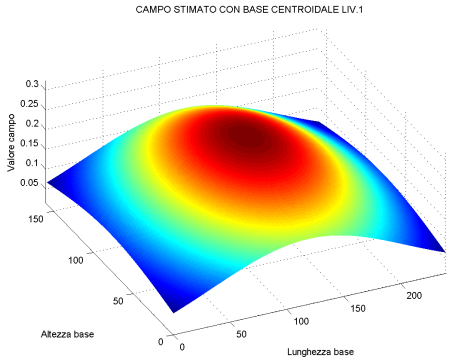
(e) Risoluzione 47



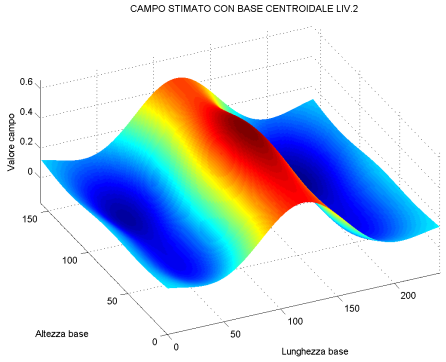
(f) Risoluzione 57

Figura A.8: Evoluzione stima Base Uniforme

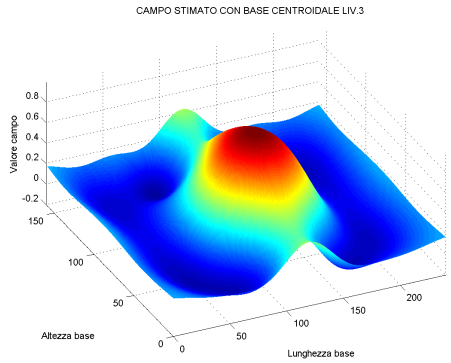
A.3.2 Base Centroidale



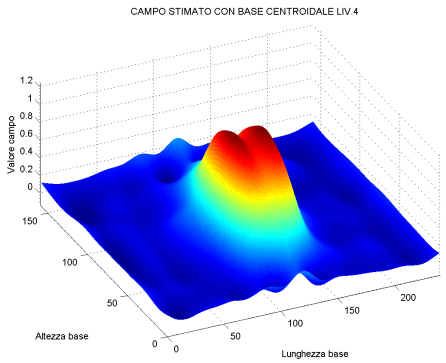
(a) Livello 1



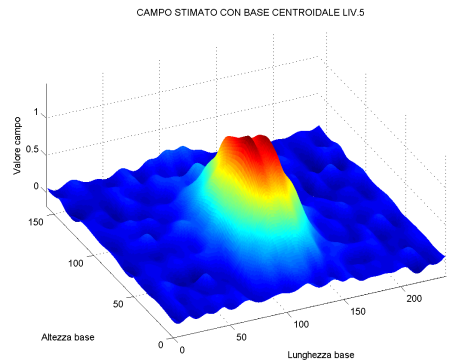
(b) Livello 2



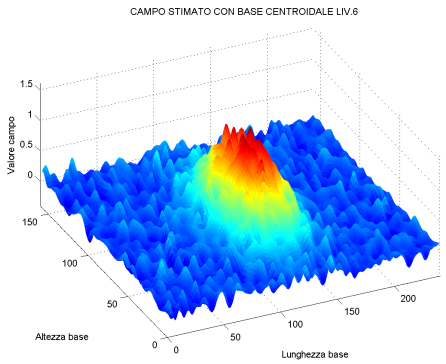
(c) Livello 3



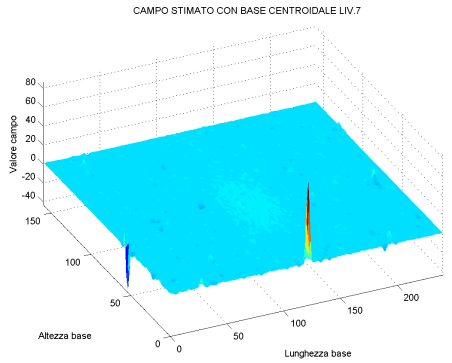
(d) Livello 4



(e) Livello 5



(f) Livello 6

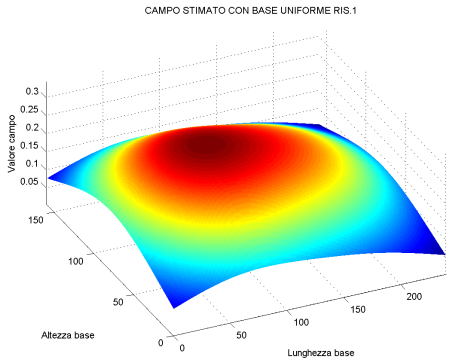


(g) Livello 7

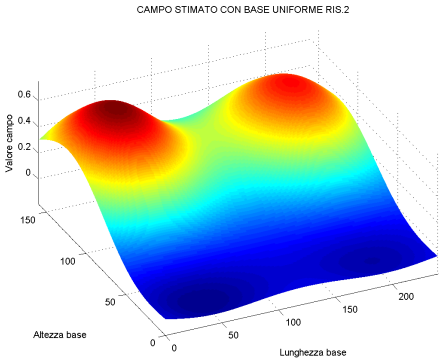
Figura A.9: Evoluzione stima Base Centroidale

A.4 Campo sul lato

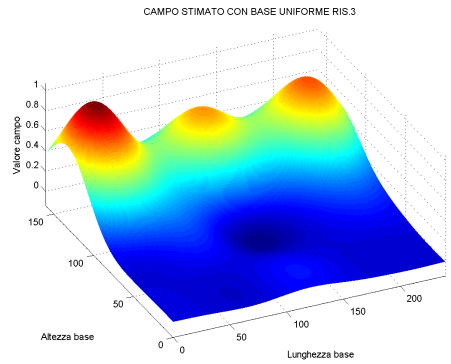
A.4.1 Base Uniforme



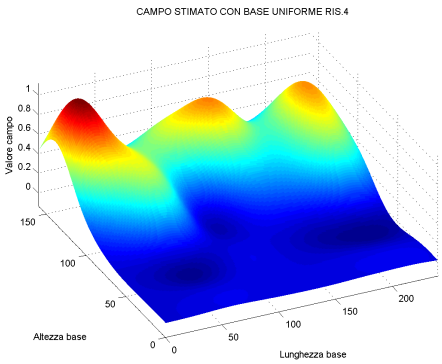
(a) Risoluzione 1



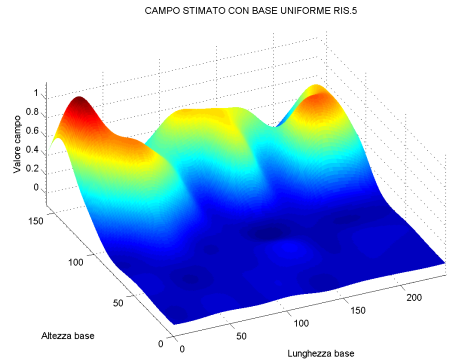
(b) Risoluzione 2



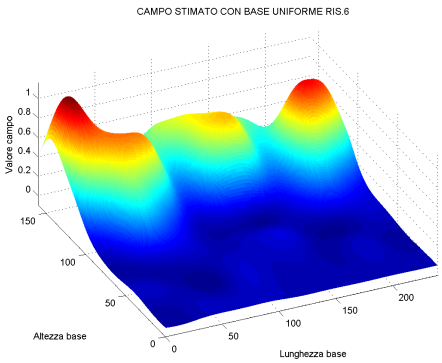
(c) Risoluzione 3



(d) Risoluzione 4

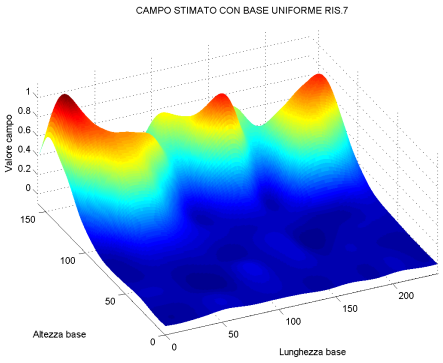


(e) Risoluzione 5

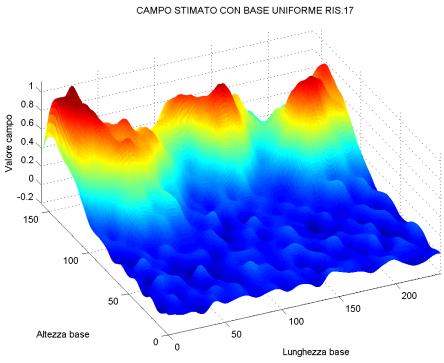


(f) Risoluzione 6

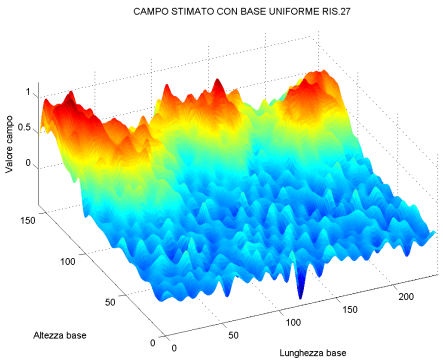
Figura A.10: Evoluzione stima Base Uniforme



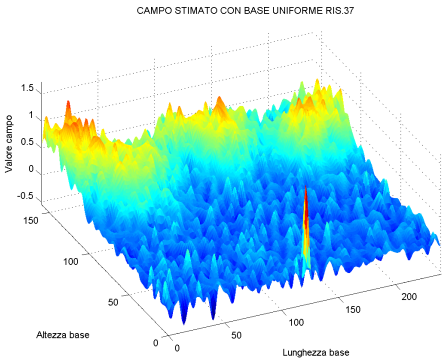
(a) Risoluzione 7



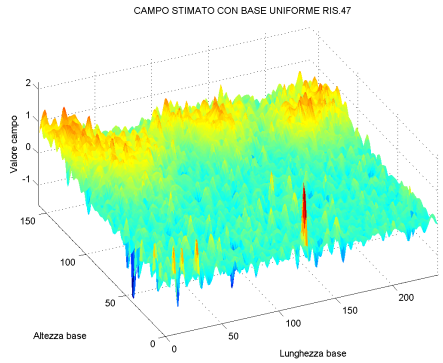
(b) Risoluzione 17



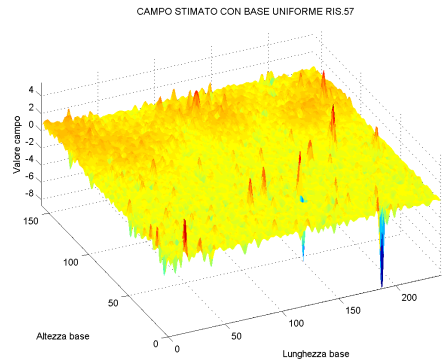
(c) Risoluzione 27



(d) Risoluzione 37



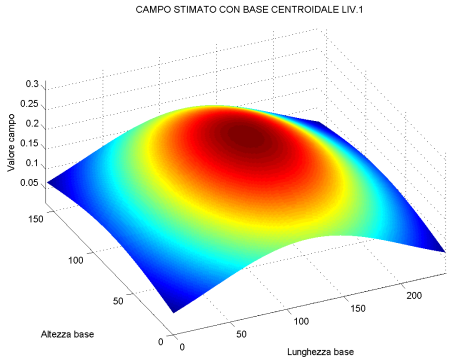
(e) Risoluzione 47



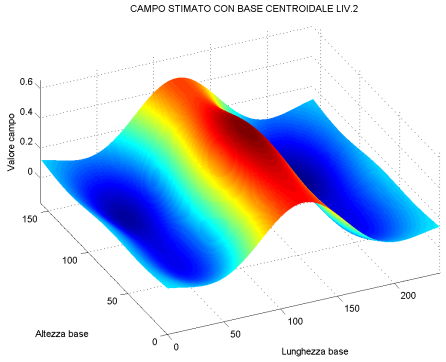
(f) Risoluzione 57

Figura A.11: Evoluzione stima Base Uniforme

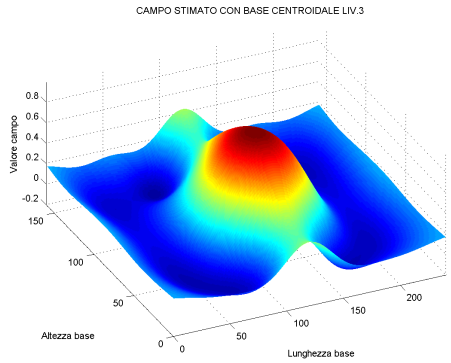
A.4.2 Base Centroidale



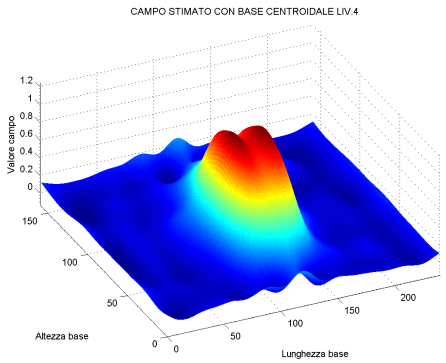
(a) Livello 1



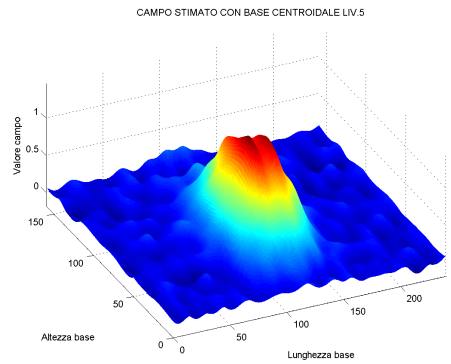
(b) Livello 2



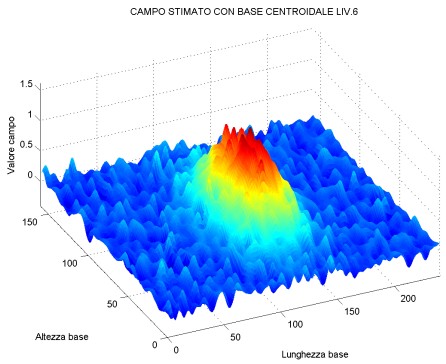
(c) Livello 3



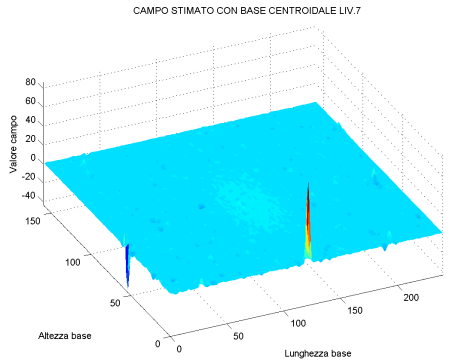
(d) Livello 4



(e) Livello 5



(f) Livello 6



(g) Livello 7

Figura A.12: Evoluzione stima Base Centroidale

Bibliografia

[1] G. Da Broi. "Modelli di reti neurali: multilayer perceptron e radial basis function". Tesi di Laurea: Ingegneria dell'Informazione. Università degli Studi di Padova, 2011-2012.

[2] C. G. Broyden. "Quasi-Newton Methods and their Application to Function Minimisation". English. In: *Mathematics of Computation* 21.99 (1967), pp. 368–381. ISSN: 00255718. URL: http://www.jstor.org/stable/2003239.

[3] E. Carta. "Le reti neurali RBF nell'analisi del colore marino da dati telerilevati". Tesi di Laurea: Ingegneria delle Telecomunicazioni. Università degli Studi di Pisa, 2000-2001.

[4] Sheng Chen, CFN Cowan e PM Grant. "Orthogonal least squares learning algorithm for radial basis function networks". In: *Neural Networks, IEEE Transactions on* 2.2 (1991), pp. 302–309.

[5] J. Cortés et al. "Coverage control for mobile sensing networks". In: *IEEE Transactions on Robotics and Automation* 20 (2004), pp. 243–255.

[6] Dennis e Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prenitce-Hall, 1983.

[7] M. Dunbabin e L. Marques. "Robots for Environmental Monitoring: Significant Advancements and Applications". In: *Robotics Automation Magazine, IEEE* 19.1 (2012), pp. 24 –39. ISSN: 1070-9932. DOI: 10.1109/MRA.2011.2181683.

[8] Matin Hanke Heinz W. Engl e Andreas Neubauer. *Regularization of inverse problems*. Kluer Accademic Publishers, 2000.

[9] A. Iglesias e A. Galvez. "Curve Fitting with RBS Functional Networks". In: *Convergence and Hybrid Information Technology, 2008. ICCIT '08. Third International Conference on*. Vol. 1. 2008, pp. 299 –306. DOI: 10.1109/ICCIT.2008.174.

[10] A Jadbabie, J Lin e A.S. Morse. "Coordination of groups of mobile autonomous agents using nearest neighbor rules". In: *IEEE Transactions on Automatic Control* 48 (2003), pp. 988–1001.

[11] Ljung e Soderstrom. *Theory and practice of recursive identification*. Prentice-Hall, 1983.

[12] L. Ljung. "Analisys of a General recursive prediction error identification algorithm". In: *Automatica*. Vol. 17, pp. 89–100.

[13] L. Ljung. "Convergence analysis of parametric identification methods". In: *Automatic Control, IEEE Transactions on* 23.5 (1978), pp. 770 –783. ISSN: 0018-9286. DOI: 10.1109/TAC.1978.1101840.

[14] L. Ljung. *System Identification, Theory for the User*. Prentice-Hall, 1987.

[15] S. Muthukrishnan, B. Ghosh e M. H. Schultz. "First- and Second-Order Diffusive Methods for Rapid, Coarse, Distributed Load Balancing". English. In: *Theory of Computing Systems* 31 (4 1998), pp. 331–354. ISSN: 1432-4350. DOI: 10.1007/s002240000092. URL: http://dx.doi.org/10.1007/s002240000092.

[16] P. Ogren, E. Fiorelli e N.E. Leonard. "Cooperative control of mobile sensor networks:Adaptive gradient climbing in a distributed environment". In: *Automatic Control, IEEE Transactions on* 49.8 (2004), pp. 1292 –1302. ISSN: 0018-9286. DOI: 10.1109/TAC.2004.832203.

[17] G. Picci. "Filtraggio Statistico (Wiener, Levinson, Kalman) e applicazioni". In: *Libreria Progetto* (2006).

[18] G. Picci. "Metodi statistici per l'identificazione di sistemi lineari". 2011.

[19] W. Ren e R. W. Beard. "Consensus seeking in multiagent systems under dynamically changing interaction topologies". In: *IEEE Transactions on Automatic Control* 50 (2005), pp. 655–661.

[20] A. Salata. "Controllo multi-agente per stima distribuita". Tesi di Laurea: Ingegneria del'Automazione. Università degli Studi di Padova, 2011-2012.

[21] L. Schenato. "Appunti di Progettazione di Sistemi di Controllo". 2012.

[22] D.F. Shanno. "Conditioning for Quasi-Newton methods for functions minimization". In: *Mathematics of computation* 24.111 (1970).

[23] S. Williams, L.T. Parker e A.M. Howard. "Terrain Reconstruction of Glacial Surfaces : Robotic Surveying Techniques". In: *Robotics Automation Magazine, IEEE* 19.4 (2012), pp. 59 –71. ISSN: 1070-9932. DOI: 10.1109/MRA.2011.2181769.

[24] F. Zanella et al. "Newton-Raphson Consensus for Distributed Convex Optimization". In: *IEEE Transactions on Automatic Control (submitted)* (20XX).