POLITECNICO DI MILANO

Facoltà di Ingegneria Industriale

Corso di Laurea in
Ingegneria Energetica

Algoritmi genetici per il model updating di turbogeneratori: analisi di sensibilità

Relatore: Prof. Nicolò BACHSCHMID
Co-relatore: Prof. Roberto RICCI

Tesi di Laurea Specialistica:

Manuel VIDONI
Matr. 734796

Anno Accademico 2009 - 2010

Indice

Ringraziamenti

Vorrei ringraziare i professori BACHSCHMID e RICCI, rispettivamente relatore e correlatore di questa tesi. Hanno dimostrato una grande disponibilità. Questo studio è stato veramente interessante e il loro aiuto mi ha permesso di completare questo lavoro in ottime condizioni.

Je voudrais également remercier mes parents, René et Odile qui m'ont, tout au long de ces deux années et demi, apporté un soutien sans faille. Malgré la distance j'ai toujours pu compter sur leur aide matérielle et morale et je leurs dois en grande partie la réussite de l'ensemble de mon parcours scolaire.

Un ringraziamento ai miei amici del gruppo di studio, Sergio e Paolo, con cui ho passato dei bei momenti nell'aula studio. Il loro aiuto è stato molto prezioso per superare le difficoltà di questo percorso formativo. Penso anche a tutti i miei altri compagni dell'università per la convivialità della loro accoglienza al Politecnico.

Un sentito ringraziamento a tutti i coinquilini che hanno condiviso un pezzo della mia vita, Giuseppe, Luca, Lorenzo, Claudio, Baldo, Flavia e Pasquale, e che hanno reso il mio soggiorno in Italia indimenticabile.

In fine, Sara, a te dedico questa tesi per il sostegno morale constante, per i tuoi incoraggiamenti, per la tua pazienza nella correzione dei miei errori d'italiano e per questi due anni passati insieme ... grazie.

I. Introduzione

Un modello matematico agli elementi finiti può approssimare il comportamento reale di una macchina. Esistono però diverse cause, come la discretizzazione del modello, le approssimazioni numeriche sulle quantità fisiche utilizzate o le imprecisioni costruttive, che impediscono una modellazione perfetta. La necessità di avere modelli più precisi nasce dal bisogno di potere controllare il funzionamento di certi componenti tecnologici ed è per questa ragione che è nato il processo del *Model Updating.*

Questa tesi è centrata sullo studio del model updating di turbo-generatori. Il model updating consiste in un'ottimizzazione del modello numerico di una macchina per farlo corrispondere a una serie di misure sperimentali. Si tratta di un processo abbastanza recente reso possibile dall'utilizzo di computer e permette di migliorare la precisione dei modelli agli elementi finiti.

Il model updating richiede una serie di operazioni preliminari prima di essere effettivamente eseguito. Si devono, per esempio, in una prima tappa misurare le risposte della macchina a diverse velocità. Richiede ovviamente una cautela specifica visto che da queste misure dipenderà la validità del modello finale. Una volta effettuate le misure, viene costruito un modello matematico della macchina da studiare che sarà la base sulla quale verrà eseguito l'updating. L'ultima tappa dell'intero procedimento prevede di usare una trasformazione matematica che permetta al modello di rispondere in modo più simile possibile alla risposta reale della macchina.

La prima problematica di questo processo è il significato fisico del modello ottimizzato. E' possibile riprodurre esattamente i valori delle misure in certi punti perdendo il significato fisico del modello. Altri metodi sono stati sviluppati per permettere di conservare questo significato fisico e ulteriori analisi sul modello aggiornato. Questi metodi si basano su degli algoritmi iterativi, più o meno efficienti. Il secondo problema sarà la scelta del metodo di aggiornamento iterativo. In questo lavoro ci interesseremo principalmente a metodi euristici sviluppati negli anni 70 che sono cresciuti in importanza in questi ultimi anni.

Abbiamo quindi scelto di utilizzare un algoritmo di tipo evolutivo che viene inizialmente applicato ad un modello agli elementi finiti abbastanza semplice. Quest'operazione ci permetterà di comprendere il funzionamento dell'algoritmo e di capire la parametrizzazione da adottare. Si passerà poi all'ottimizzazione di una macchina reale, una turbina a vapore, e studieremo come sono correlati i risultati della convergenza dell'algoritmo con la sensibilità dei parametri della modellazione.

Il lavoro di tesi è stato condotto secondo i seguenti passi:

- **Capitolo 2**: Studio del processo di model updating. Abbiamo visto le esigenze relative alle misure e alla parametrizzazione del modello studiato. E' stato fatto un confronto tra i metodi diretti e indiretti a traverso la descrizione di alcune tecniche di ciascun genere. Una volta studiati i metodi iterativi, sono stati presi in considerazione alcuni algoritmi euristici che permettono di realizzare l'ottimizzazione richiesta. L'attenzione è stata rivolta ad algoritmi di tipo genetico ed evolutivo che stanno alla base del funzionamento del programma impiegato nella parte sperimentale.

- **Capitolo 3**: Dopo avere capito gli interessi di questi metodi abbiamo potuto applicare le loro funzionalità su un modello *test_rig* per potere determinare quale parametrizzazione dell'algoritmo adottare per una massima efficienza. Un'analisi di sensibilità e uno studio della convergenza sono stati condotti in modo di capire la correlazione tra queste due entità.

- **Capitolo 4**: Come caso applicativo abbiamo realizzato e studiato l'updating su un modello di una macchina reale destinata alla produzione di elettricità. E' stata eseguita una verifica della correlazione evidenziata nel capitolo precedente.

2. Model Updating

Il bisogno di avere un modello teorico più vicino possibile alla realtà è diventato una necessità con l'elaborazione di macchine sempre più performante. Negli anni '50 sono nati i modelli a elementi finiti che hanno permesso di rappresentare le macchine reali in modo molto più dettagliato e conforme al loro comportamento dinamico. Siccome le esigenze tecnologiche sono cresciute in modo esponenziale nella seconda parte del ventesimo secolo, la modellazione a elementi finiti è diventata troppo importante per essere mantenuto nella sua forma primaria.

E stato quindi ideato il processo di model updating che permette di rivalutare i parametri del modello in modo tale che la risposta ottenuta con i nuovi parametri sia più conforme alla risposta reale del sistema. L'introduzione dei calcolatori ha fortemente aiutato lo sviluppo di questa teoria perché tratta un numero di dati enorme. Questo processo si basa in primo luogo su un certo numero di misure effettuate sulla macchina reale e, in secondo luogo, su una ottimizzazione tale da ridurre la distanza tra queste misure e la risposta analitica.

Il model updating pone quindi un problema già a livello della sua definizione perché avvicinarsi alla risposta misurata non significa per forza avvicinarsi alla risposta reale. Si consideri che, tra la risposta analitica e quella misurata, quella che corrisponde maggiormente alla realtà è quelle misurata, ove però influisce negativamente una serie di fattori. Un altro limite è imposto dal fatto che il numero di gradi di liberta e il numero di misure sono molto diversi per ragioni tecnologiche. È quindi necessario lavorare su misure stimate per completare il set di misure effettivamente raccolte o lavorare su un numero di gradi di liberta troncato in modo da avere abbastanza misure per trattare il problema.

Esistono due filosofie diverse quando si tratta di model updating. La prima è la rappresentazione puramente teorica che cerca di riprodurre esattamente il set di misure, modificando la struttura del modello analitico iniziale, senza preoccuparsi del significato fisico delle trasformazioni matematiche applicate. Si può applicare un metodo del genere nel caso in cui ci bastasse conoscere la risposta del modello per potere effettuare dei confronti o delle previsioni. La seconda filosofia invece cerca di conservare il significato fisico dopo l'updating. Si sceglie quindi un set di parametri sul quale verranno effettuate le modifiche in modo tale da poter capire l'influenza di un parametro sulla risposta. In questo modo il model updating lascia la possibilità all'utente di analizzare una risposta misurata e quindi di risalire ai parametri coinvolti nelle eventuali modifiche. Questo è molto utile nel caso dei turbo-generatori perché permette la scoperta dei difetti (cricche, componenti guasti, ecc...).

Da questa divergenza di idee di base nascono due famiglie di procedimenti, i metodi diretti e i metodi iterativi. Questi sono molto diversi nel loro funzionamento perché agiscono in modo differente sui parametri del modello analitico.

2.1 Misure

La parte di misurazione è la prima tappa nel processo di model updating. È uno step importante del processo perché le misure saranno considerate come le referenze da raggiungere dopo l'updating [1]. Le misure vengono effettuate durante dei test di vibrazione. L'utilizzatore applica una forzante al sistema di caratteristiche note e rileva la risposta data dalla macchina. Questa operazione è molto sensibile alle condizioni sperimentali in quanto il montaggio, il peso degli apparecchi di misura, il peso del dispositivo di eccitazione cambiano le condizioni di funzionamento della macchina rispetto a quelle nelle quali si troverà a lavorare una volta messa in esercizio. L'obiettivo è quindi di riprodurre il più fedelmente possibile le condizioni di utilizzazione. Alcuni elementi richiedono di non essere legati a nessuna fondazione per essere valutati, questo non è possibile sperimentalmente perciò si usano delle strutture con grande flessibilità per effettuare le misure.

La tecnologia attuale permette una buona valutazione delle frequenze proprie del sistema, fino all'1%. I modi di vibrare sono però molto più soggetti agli errori e potranno essere valutati solo al 10%. Il rumore è uno dei fenomeni principale che impedisce l'esattezza delle misure, può essere modellizzato con una distribuzione statistica con una certa media e una certa varianza. Eseguire più volte le misure permette di ridurre la varianza mediando i risultati ottenuti. Un'eccitazione di martellamento genera un rumore con una media nulla che è molto comodo perché facilmente rimuovibile.

Un altro problema è l'incompletezza dei dati. I problemi di campionamento per la conversione analogico-digitale impongono intervalli prestabiliti alle frequenze misurabili. Il teorema di Nyquist-Shannon impone la seconda condizione per il campionamento :

$$f_c \geq 2 f_s$$

La frequenza di campionamento deve essere più di due volte maggiore di quella dal segnale da misurare, quindi non si possono misurare senza perdite di informazioni i segnali che hanno una frequenza maggiore della metà della frequenza di campionamento. Le misure sono incomplete anche perché non tutte le componenti dei modi di vibrare possono essere misurate perché non si può accedere a certi nodi del modello. L'utilizzatore dovrà quindi usare tecniche di

riduzione del modello o di espansione dei dati per fare corrispondere i due vettori.

Per potere procedere con l'updating è necessario sapere a quale frequenza propria corrisponde ogni modo di vibrare, ma ciò non è sempre possibile visto che certe frequenze proprie sono veramente vicine tra loro e quindi praticamente impossibili da distinguere. Anche nell'intervallo di frequenze disponibile ci saranno delle incompletezze dovute a questo problema.

Una volta effettuate le misure ci sono ancora delle sorgenti di errore perché questi vengono registrati, ad esempio il trattamento dei dati può introdurre una media non nulla al rumore precedentemente descritto.

Possiamo quindi dire che le azioni preliminari alla misurazione devono essere fatte in modo molto curato, perché soggette a una serie di perturbazioni che le rendono imprecise e incomplete e quindi che gravano sul processo di updating teorico. La procedura richiederebbe dei dati precisi e adeguati con il modello analitico e i gradi di libertà scelti da l'utilizzatore.

2.2 Metodi diretti

Come indicato dal nome, questi metodi si eseguono in un piccolo numero di fasi senza richiedere iterazioni. La convergenza della soluzione è quindi assicurata e non necessitano tempi lunghi di calcolo.

Questi metodi hanno anche il vantaggio di riprodurre esattamente il set di misure effettuate. Ma questo vantaggio può manifestarsi anche come debolezza, visto che le misurazioni sono errate l'*Updating* effettuate riprodurrà tutti gli errori. Per esempio, il rumore, che perturba le misurazioni, viene integrato al modello analitico anche se non è una variabile di nostro interesse dal momento che le condizioni in cui si effettuano le misure sono diverse dalle condizioni operative. È richiesta quindi una cautela massima sulla procedura sperimentale che condurrà alla raccolta dei dati. Le frequenze proprie si misurano con buona precisione, al contrario i modi di vibrare rappresentano un problema perché la loro misura è delicata.

Per poter eseguire i calcoli c'è un pre-requisito che non è mai verificato: il numero di gradi di libertà misurati deve essere uguale al numero di gradi di libertà del modello analitico. Siccome non tutte le zone delle macchine studiate sono accessibili, un gran numero di gradi di liberta non può essere misurato. Inoltre le misure possono essere effettuate solo in un certo intervallo di frequenze, questo comporta che si possono conoscere solo certi elementi di certi modi di vibrare. Esistono due metodi per far coincidere queste due entità: la riduzione dei gradi di liberta del modello e l'espansione dei dati misurati.

2.2.1 Riduzione del modello

Consideriamo il modello classico senza smorzamento composto dalla matrice di rigidezza **K** e della matrice di massa **M**. Sia **f** la forzante applicata e **x** il vettore dei gradi di liberta.
Ricordiamo che queste matrici normalizzate verificano queste due relazioni:

$$\mathbf{U}^{\mathbf{T}}\mathbf{M}\mathbf{U} = \mathbf{I} \quad \text{e} \quad \mathbf{U}^{\mathbf{T}}\mathbf{K}\mathbf{U} = diag(\lambda_1,...,\lambda_n) \quad (1.1)$$

Con U la matrice degli autovettori e $(\lambda_i)_{i\in[\![1,n]\!]}$ gli autovalori corrispondenti

- Metodo statico [7]

Questo metodo si distingue dagli altri per la sua semplicità. Si chiama anche metodo di Guyan dal nome del suo inventore che l'ha introdotto in 1965.

Se separiamo le componenti di $\mathbf{x}$ che possiamo misurare ($\mathbf{x}_m$) dalle altre incognite ($\mathbf{x}_i$) avremo:

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_m \\ -- \\ \mathbf{x}_i \end{bmatrix}$$

Separiamo quindi i due spazi vettoriali secondo questo criterio, troviamo:

$$\begin{bmatrix} \mathbf{M}_{mm} & | & \mathbf{M}_{mi} \\ --- & | & --- \\ \mathbf{M}_{im} & | & \mathbf{M}_{ii} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{x}}_m \\ -- \\ \ddot{\mathbf{x}}_i \end{bmatrix} + \begin{bmatrix} \mathbf{K}_{mm} & | & \mathbf{K}_{mi} \\ --- & | & --- \\ \mathbf{K}_{im} & | & \mathbf{K}_{ii} \end{bmatrix} \begin{bmatrix} \mathbf{x}_m \\ -- \\ \mathbf{x}_i \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ -- \\ 0 \end{bmatrix}$$

Se trascuriamo la matrice d'inerzia nella seconda equazione vettoriale ottenuta con questa separazione avremo :

$$\mathbf{K}_{im}\mathbf{x}_m + \mathbf{K}_{ii}\mathbf{x}_i = 0$$

Quindi :

$$\mathbf{x}_s = -\mathbf{K}_{ii}^{-1}\mathbf{K}_{im}\mathbf{x}_m$$

Ed infine:

$$\begin{bmatrix} \mathbf{x}_m \\ -- \\ \mathbf{x}_i \end{bmatrix} = \begin{bmatrix} \mathbf{I} \\ ----- \\ -\mathbf{K}_{ii}^{-1}\mathbf{K}_{im} \end{bmatrix} \mathbf{x}_m = \mathbf{T}\mathbf{x}_m \quad \textbf{con} \quad \mathbf{T} = \begin{bmatrix} \mathbf{I} \\ ----- \\ -\mathbf{K}_{ii}^{-1}\mathbf{K}_{im} \end{bmatrix}$$

$\mathbf{T}$ è quindi la matrice di trasformazione che permette la riduzione del problema. Potremo quindi ricavare le nuove matrici di massa e di rigidezza moltiplicando l'equazione a destra per $\mathbf{T}$ e a sinistra per $\mathbf{T}^{-1}$. Le matrici ridotte saranno date da:

$$\mathbf{M}_R = \mathbf{T}^T\mathbf{M}\mathbf{T} \quad \textbf{e} \quad \mathbf{K}_R = \mathbf{T}^T\mathbf{K}\mathbf{T}$$

Questo metodo si chiama statico perché l'approssimazione fatta sulla matrice di inerzia da un risultato esatto solo per una frequenza nulla. In genere le misure si fanno su un intervallo che si allontana tanto dallo zero, per quello sono state inventate altre tecniche che possono essere applicate su un campo più

largo. Infatti la qualità di questo modello peggiora all'aumentare della frequenza.

- Metodo dinamico

Questo metodo riduce le matrice per una certa frequenza scelta dall'utente e quindi il modello ridotto sarà molto preciso per le frequenze intorno a quella scelta. Si può per esempio la frequenza media dell'intervallo che ci interessa però è spesso usata una media geometrica.

Se riprendiamo l'equazione matriciale a la frequenza ω_0 possiamo scrivere che:

$$(-\omega_0^2\mathbf{M_{im}} + \mathbf{K_{im}})\mathbf{x_m} + (-\omega_0^2\mathbf{M_{ii}} + \mathbf{K_{ii}})\mathbf{x_i} = 0$$

Quindi troveremo come nella parte precedente :

$$\begin{bmatrix} \mathbf{x_m} \\ \text{--} \\ \mathbf{x_i} \end{bmatrix} = \begin{bmatrix} \mathbf{I} \\ \text{----------} \\ -(-\omega_0^2\mathbf{M_{ii}} + \mathbf{K_{ii}})^{-1}(-\omega_0^2\mathbf{M_{im}} + \mathbf{K_{im}}) \end{bmatrix} \mathbf{x_m} = \mathbf{T}\mathbf{x_m}$$

Le matrice ridotte $\mathbf{M_R}$ e $\mathbf{K_R}$ si ricaverano come nella parte precedente.

- Metodo SEREP (*System Equivalent Reduction Expansion Process*)

Questo metodo permette di ottenere un modello ridotto che ha esattamente le stesse frequenze proprie del modello intero. Si basa su l'utilizzazione dei auto-vettori che vengono separati in due come nelle tecniche precedenti tra gradi di liberta in grado di essere misurati e i altri.

Se chiamiamo **U** la matrice dei modi di vibrare avremo:

$$\mathbf{U} = \begin{bmatrix} \mathbf{U_m} \\ -- \\ \mathbf{U_i} \end{bmatrix}$$

Costruiamo dopo la matrice $\mathbf{U_m^*}$ come segue:

$$\mathbf{U_m^*} = (\mathbf{U_m^T}\mathbf{U_m})^{-1}\mathbf{U_m^T}$$

$\mathbf{U}_m^*$ è chiamata matrice semi inversa di $\mathbf{U}_m$ (se $\mathbf{U}_m$ è di dimensione (p×q) $\mathbf{U}_m^*$ sarà di dimensione (q×p)).

La matrice **T** di trasformazione come descritta nei paragrafi precedenti sarà allora data da:

$$\mathbf{T} = \mathbf{U}\mathbf{U}_m^*$$

Si potrebbe dimostrare che questa matrice da un modello ridotto che ha esattamente le stesse frequenze proprie del modello intero iniziale.

2.2.2 Espansione del modello

L'altro metodo consiste a espandere il vettore delle misure in modo tale di farlo corrispondere con quello dei gradi di liberta. I valori aggiunti non vengono inventati, spesso si utilizzano i valori ricavati teoricamente dal modello analitico a elementi finiti.

- Metodo dinamico

Possiamo rifare la stessa suddivisione di prima tra coordinate misurate e coordinate incognite, avremo per una certa frequenza propria misurata $\hat{\omega}$ e **u** il modo di vibrare corrispondente la seguente relazione:

$$\left(\hat{\omega} \begin{bmatrix} \mathbf{M}_{mm} & | & \mathbf{M}_{mi} \\ --- & | & --- \\ \mathbf{M}_{im} & | & \mathbf{M}_{ii} \end{bmatrix} + \begin{bmatrix} \mathbf{K}_{mm} & | & \mathbf{K}_{mi} \\ --- & | & --- \\ \mathbf{K}_{im} & | & \mathbf{K}_{ii} \end{bmatrix} \right) \begin{bmatrix} \mathbf{u}_m \\ -- \\ \mathbf{u}_i \end{bmatrix} = \begin{bmatrix} 0 \\ -- \\ 0 \end{bmatrix}$$

Che ci darà:

$$\mathbf{u}_i = -(-\omega^2 \mathbf{M}_{ii} + \mathbf{K}_{ii})^{-1} (-\omega^2 \mathbf{M}_{im} + \mathbf{K}_{im}) \mathbf{u}_m$$

In questo modo avremo ottenuto la seconda parte del vettore **u** che ci mancava e quindi avremo realizzato un'espansione del set dei dati iniziali.

- Metodo modale

Questo metodo presuppone che i modi propri misurati sono combinazione lineare dei modi corrispondenti analitici. Chiamiamo $\hat{\mathbf{U}}$ la matrice dei modi misurati e **U** la matrice dei modi analitici. Riprendiamo anche

la notazione precedente coni indici ***m*** o ***i*** per indicare un dato realmente misurato o un dato incognito.

Avremo:

$$\hat{\mathbf{U}}_{\mathbf{m}} = \mathbf{U}_{\mathbf{m}}\mathbf{T}$$

Consideriamo ancora la matrice $\mathbf{U}_{\mathbf{m}}^{*}$ la matrice semi inversa della matrice $\mathbf{U}_{\mathbf{m}}$ e possiamo scrivere la matrice di trasformazione $\mathbf{T}$ come:

$$\mathbf{T} = \mathbf{U}_{\mathbf{m}}^{*}\mathbf{U}_{\mathbf{m}}$$

Possiamo quindi applicare dopo questa trasformazione ai modi di vibrare non misurati con i modi di vibrare analitici corrispondenti con la relazione:

$$\hat{\mathbf{U}}_{\mathbf{i}} = \mathbf{U}_{\mathbf{i}}\mathbf{T}$$

Avremo quindi con concatenazione di $\hat{\mathbf{U}}_{\mathbf{m}}$ e di $\hat{\mathbf{U}}_{\mathbf{i}}$ una matrice di misure espansa, dove la prima parte corrisponde alle misure reali e la seconda corrisponde a una stima dei modi di vibrare.

2.2.3 Metodo di minimizzazione

Il metodo dei moltiplicatori di Lagrange è un metodo semplice è facile da applicare per la minimizzazione di una funzione dove i parametri sono soggetti a una condizione. Descriveremo qua brevemente il funzionamento di questo metodo:

Siano una funzione da minimizzare $\varphi(x_1,\ldots,x_n) : \mathbb{R}^n \to \mathbb{R}$ e la famiglia di funzione $\phi_j(x_1,\ldots,x_n) : \mathbb{R}^n \to \mathbb{R},\ j \in [\![1,m]\!], m < n$ che serviranno a descrivere le costrizione del sistema. Cerchiamo il n-upplet $(a_1,\ldots,a_n)$ che verifica la condizione seguente:

$$\varphi(a_1,\ldots,a_n) = \min_{(x_i)\in G} \varphi(x_1,\ldots,x_n)$$

Con:

$$G = \left\{(x_i) \in \mathbb{R}^n,\quad \forall j \in [\![1,m]\!], \phi_j(x_1,\ldots x_n) = 0\right\}$$

Si definisce allora la funzione L come segue :

$$L(x_1,\ldots,x_n,\lambda_1,\ldots,\lambda_m) = \varphi(x_1,\ldots,x_n) + \sum_{j=1}^{m} \lambda_j \phi_j(x_1,\ldots,x_n)$$

Si risolve quindi il problema derivando la funzione L rispetto a tutte le sue coordinate dello spazio vettoriale :

$$\forall i \in [\![1, n]\!], \ \frac{\partial L(x_1, \ldots, x_n)}{\partial x_i} = 0$$

I coefficienti λ_j sono chiamati i moltiplicatori di Lagrange, questa tecnica permette quindi di passare da un problema di ottimizzazione con costrizione a un problema di ottimizzazione (quello della funzione L) senza costrizione.

Le tecniche di Model Updating che vedremo adesso usano sempre le stesse tre entità però in modo differente. Per i modelli non smorzati si considerano, la matrice di massa, la matrice di rigidezza e le coppie formate da frequenze proprie e modi di vibrare. Ogni tecnica di ottimizzazione si basa uno di questi tre personaggi e lo considera come preciso. La matrice di rigidezza è poco usata come referenza nella letteratura perché difficile da valutare, ci fermeremo quindi alle tecniche che sfruttano la matrice di massa o i dati modali come referenza.

2.2.4 Updating con la matrice di massa

- Updating della matrice degli auto-vettori

Siccome la matrice degli auto-vettori è incompleta rispetto al modello reale e con le varie imprecisioni delle misure, non avremo mai una matrice dei modi di vibrare che verifica la proprietà di ortogonalità con la matrice analitica di massa. Per rimediare a questo problema si può fare una prima ottimizzazione sulla matrice dei modi di vibrare misurati. Questo processo suppone anche di conoscere con precisione i coefficienti della matrice di massa analitica **M**. La matrice **U** di questa parte è una matrice $(n \times m)$ formata dai vettori propri concatenati in colone. Possiamo quindi ricercare la matrice ortogonale alla matrice di massa la più vicina alla nostra matrice di misura utilizzando il metodo di Lagrange con le funzioni seguente:

$$\varphi : M_{m,n}(\mathbb{R}) \to \mathbb{R}, \quad \mathbf{X} \to \left\| \mathbf{N}(\mathbf{X} - \hat{\mathbf{U}}) \right\|_2$$

E

$$\forall (i, j) \in [\![1, m]\!] \times [\![1, n]\!], \ \phi_{ij} : M_{m,n}(\mathbb{R}) \to \mathbb{R}, \quad \mathbf{X} \to \left(\mathbf{X}^{\mathbf{T}} \mathbf{M} \mathbf{X} - \mathbf{I} \right)_{ij}$$

Dove la matrice $\mathbf{N}$ è una matrice di peso che serve a omogeneizzare l'ordine di grandezza dei coefficienti della matrice $\hat{\mathbf{U}}$ scelta dall'utilizzatore e $\hat{\mathbf{U}}$ è la matrice dei modi di vibrare misurati. La famiglia delle funzioni ϕ_{ij} permette di stabilire la condizione di ortogonalità rispetto alla matrice di massa analitica perché dobbiamo annullarle tutte. Il parametro m definisce la lunghezza dei modi di vibrare quindi il numero di misure fatte a ogni modo di vibrare. Se applichiamo quindi il metodo di Lagrange con queste due funzioni, possiamo ricavare una matrice soluzione $\mathbf{U}$ dei modi di vibrare che sarà ortogonale alla matrice di massa analitica e che non sarà troppo lontana per la norma scelta alla matrice $\hat{\mathbf{U}}$. Si prende di solito $\mathbf{N} = \mathbf{M}^{1/2}$ come matrice di peso. La funzione *L* per quest'ottimizzazione sarà data da:

$$L(\mathbf{X}, \mathbf{\Gamma}) = \varphi(\mathbf{X}) - \sum_{(i,j)\in[\![1,m]\!]\times[\![1,n]\!]} \mathbf{\Gamma}_{ij}\phi_{ij}(\mathbf{X})$$

La matrice $\mathbf{\Gamma}$ è quindi composta di tutti i moltiplicatori di Lagrange e può essere resa unica imponendo la condizione di simmetria $\mathbf{\Gamma} = \mathbf{\Gamma}^{\mathbf{T}}$.
Tutti conti effettuati, viene:

$$\tilde{\mathbf{U}} = \hat{\mathbf{U}}(\mathbf{I} + \mathbf{\Gamma})^{-1}$$

Si può interpretare la matrice $\mathbf{\Gamma}$ come la differenza tra le misure e i vettori corretti. Facendo ancora qualche passaggio viene:

$$\tilde{\mathbf{U}} = \hat{\mathbf{U}}(\hat{\mathbf{U}}^{\mathbf{T}}\mathbf{M}\hat{\mathbf{U}})^{-1/2}$$

Disponiamo quindi adesso di un vettore degli auto-vettori che gode della proprietà di ortogonalità rispetto alla matrice di massa per eseguire le altre tappe del processo diretto di *Model Updating*.

- Updating della matrice di Rigidezza

A questo punto si può trattare la matrice di rigidezza supposta imprecisa e quindi da correggere. Chiameremo quindi $\mathbf{K}$ la matrice di rigidezza e $\tilde{\mathbf{K}}$ la matrice aggiornata di rigidezza.

Desideriamo che la matrice aggiornata sia il più vicino possibile alla matrice iniziale quindi desideriamo minimizzare questa funzione :

$$\varphi : M_{n,n}(\mathbb{R}) \to \mathbb{R}, \quad \mathbf{X} \to \frac{1}{2}\left\|\mathbf{N}^{-1}(\mathbf{X}-\mathbf{K})\mathbf{N}^{-1}\right\|_2$$

Con come nel caso precedente:

$$\mathbf{N} = \mathbf{M}^{1/2}$$

Vogliamo che la matrice aggiornata verifichi le due proprietà seguenti:

$$\begin{cases} \tilde{\mathbf{K}} = \tilde{\mathbf{K}}^{\mathbf{T}} \\ \tilde{\mathbf{U}}^{\mathbf{T}}\tilde{\mathbf{K}}\tilde{\mathbf{U}} = diag(\lambda_1, \ldots \lambda_n) \end{cases}$$

Possiamo quindi definire due famiglie di funzione di costrizione, una per ogni condizione da raggiungere. Per la condizione di simmetria possiamo definire la famiglia seguente:

$$\forall (i,j) \in [\![1,n]\!]^2 \ , \ \phi_{ij} : M_{n,n}(\mathbb{R}) \to \mathbb{R}, \quad \mathbf{X} \to \left(\mathbf{X}^{\mathbf{T}} - \mathbf{X}\right)_{ij}$$

Invece per l'altra condizione scriveremo:

$$\forall (i,j) \in [\![1,n]\!] \times [\![1,m]\!], \ \beta_{ij} : M_{n,n}(\mathbb{R}) \to \mathbb{R},$$

$$\mathbf{X} \to 2\left(\mathbf{X}\tilde{\mathbf{U}} - \mathbf{M}\tilde{\mathbf{U}} diag(\lambda_1, \cdots, \lambda_m)\right)_{ij}$$

Avremo quindi due matrici per ordinare i moltiplicatori di Lagrange la prima matrice $\mathbf{\Gamma_1}$ associata alla condizione di simmetria e di dimensione (n,n) e la matrice $\mathbf{\Gamma_2}$ di dimensione (n,m) associata alla seconda condizione. $\mathbf{\Gamma_1}$ può essere ritenuta unica se gli aggiungiamo la condizione di simmetria. Dopo avere fatto i conti, si può dimostrare che la matrice corretta è unica ed è ottenuta con la relazione seguente:

$$\tilde{\mathbf{K}} = \mathbf{K} - \mathbf{K}\tilde{\mathbf{U}}\tilde{\mathbf{U}}^{\mathbf{T}}\mathbf{M} - \mathbf{M}\tilde{\mathbf{U}}\tilde{\mathbf{U}}^{\mathbf{T}}\mathbf{K} + \mathbf{M}\tilde{\mathbf{U}}\tilde{\mathbf{U}}^{\mathbf{T}}\mathbf{K}\tilde{\mathbf{U}}\tilde{\mathbf{U}}^{\mathbf{T}}\mathbf{M} + \mathbf{M}\tilde{\mathbf{U}}\mathbf{\Lambda}\tilde{\mathbf{U}}^{\mathbf{T}}\mathbf{M}$$

Con: $\mathbf{\Lambda} = diag(\lambda_1, \ldots, \lambda_n)$

Questo metodo di *Updating* introdotto da Baruch alla fine degli anni 70 permette quindi di calcolare una matrice di rigidezza corretta con il presupposto che la matrice di massa è precisa e con una tappa intermedia di correzione dei valori misurati.

2.2.5 Updating con i dati modali

Questo metodo, introdotto da Berman e Nagy in 1983, contrariamente a quello precedente usa come riferimento i dati modali ottenuti con le misure. Abbiamo detto prima che si considerano le misure come l'entità la più vicina alla risposta reale della macchina quindi prendere il rischio di correggere questi dati non è considerato da tutti come un buon metodo. E stato quindi inventato un altro metodo basato sulle misure e quindi che cerca di migliorare la matrice di massa e la matrice di rigidezza sempre usando il metodo di Lagrange.

Conosciamo quindi la matrice degli auto-vettori misurati $\hat{\mathbf{U}}$ e cerchiamo in una prima tappa di ottimizzare la matrice di massa analitica $\mathbf{M}$. Chiameremo $\tilde{\mathbf{M}}$ la matrice di massa ottimizzata che deve essere vicina comunque alla matrice iniziale quindi cercheremo di minimizzare la funzione:

$$\varphi : M_{n,n}(\mathbb{R}) \to \mathbb{R}, \quad \mathbf{X} \to \frac{1}{2}\left\|\mathbf{N}^{-1}(\mathbf{X}-\mathbf{M})\mathbf{N}^{-1}\right\|_2$$

La letteratura consiglia di prendere $\mathbf{N} = \mathbf{M}^{1/2}$

La matrice corretta deve anche verificare le proprietà di ortogonalità con la matrice degli auto-vettori misurati:

$$\hat{\mathbf{U}}^{\mathbf{T}}\tilde{\mathbf{M}}\hat{\mathbf{U}} = \mathbf{I}$$

Si può quindi considerare la famiglia di funzione:

$$\forall (i,j) \in [\![1,n]\!],\ \phi_{ij} : M_{m,n}(\mathbb{R}) \to \mathbb{R}, \quad \mathbf{X} \to \left(\hat{\mathbf{U}}^{\mathbf{T}}\mathbf{X}\mathbf{U} - \mathbf{I}\right)_{ij}$$

Applicando il metodo dei moltiplicatori di Lagrange arriveremo al risultato seguente:

$$\tilde{\mathbf{M}} = \mathbf{M} + \mathbf{M}\hat{\mathbf{U}}\mathbf{M}_{\mathbf{U}}^{-1}(\mathbf{I} - \mathbf{M}_{\mathbf{U}})\mathbf{M}_{\mathbf{U}}^{-1}\hat{\mathbf{U}}\mathbf{M}$$

$$\text{Con: } \mathbf{M}_{\mathbf{U}} = \mathbf{U}^{\mathbf{T}}\mathbf{M}\mathbf{U}$$

Il risultato è simmetrico senza applicare nessuna condizione di simmetria su $\tilde{\mathbf{M}}$.

Si procede dopo con l'Updating della matrice di rigidezza con lo stesso procedimento. Si ottengono quindi in questo modo due matrici aggiornate $\tilde{\mathbf{M}}$ ed esistono altri metodi con altre matrice $\mathbf{N}$ di peso che portano a nuove matrici

aggiornate riproducendo sempre ugualmente le misure ottenute. Aggiornare prima la matrice di massa non è una regola si può anche procedere con un aggiornamento della matrice di rigidezza per prima e puoi aggiornare quella di massa.

2.2.6 Updating simultaneo con i dati modali

Un'alternativa all'aggiornamento in due passi e di aggiornare contemporaneamente le matrici di massa e di rigidezza con i dati modali misurati. In questo caso bisogna minimizzare la funzione:

$$\varphi : \left(M_{n,n}(\mathbb{R})\right)^2 \to \mathbb{R}, \quad (\mathbf{X},\mathbf{Y}) \to \frac{1}{2}\left\|\mathbf{N}^{-1}(\mathbf{X}-\mathbf{M})\mathbf{N}^{-1}\right\|_2 + \frac{1}{2}\left\|\mathbf{N}^{-1}(\mathbf{Y}-\mathbf{K})\mathbf{N}^{-1}\right\|_2$$

Con $\mathbf{N} = \mathbf{M}^{1/2}$ come nei casi precedenti pero questa formula pone un problema di omogeneità nel senso che il primo termine è senza unità e il secondo è omogeneo a $\left[s^{-4}\right]$. Avremo quindi dei coefficienti che saranno molto diversi e quindi le matrici non saranno modificate in modo equivalente. Si costata che la matrice di peso tenderà ad appesantire di più i coefficienti della matrice di rigidezza quindi saranno maggiori le trasformazioni applicate alla matrice di massa.

Dopo un metodo analogico a quelli precedenti si arriva a questi risultati.

$$\tilde{\mathbf{M}} = \mathbf{M} + \bar{\mathbf{M}} + \boldsymbol{\Delta}_{\mathbf{M}} + \boldsymbol{\Delta}_{\mathbf{M}}^{\mathbf{T}}$$

Con

$$\bar{\mathbf{M}} = \mathbf{M}\hat{\mathbf{U}}\mathbf{M}_{\mathbf{U}}^{-1}(\mathbf{I} - \mathbf{M}_{\mathbf{U}})\mathbf{M}_{\mathbf{U}}^{-1}\hat{\mathbf{U}}\mathbf{M}$$

La matrice di massa aggiornata nel caso di *Updating* con i dati modali:

$$\mathbf{M}_{\mathbf{U}} = \mathbf{U}^{\mathbf{T}}\mathbf{M}\mathbf{U}$$

$$\boldsymbol{\Delta}_{\mathbf{M}} = \left(\mathbf{I} - \mathbf{M}\hat{\mathbf{U}}\mathbf{M}_{\mathbf{U}}^{-1}\hat{\mathbf{U}}^{\mathbf{T}}\right)\mathbf{K}\hat{\mathbf{U}}\mathbf{P}^{-1}\boldsymbol{\Lambda}\hat{\mathbf{U}}^{\mathbf{T}}\mathbf{M}$$

$$\mathbf{P} = \mathbf{M}_{\mathbf{U}} + \boldsymbol{\Lambda}\mathbf{M}_{\mathbf{U}}\boldsymbol{\Lambda}$$

La matrice di rigidezza corretta lei è data da:

$$\tilde{\mathbf{K}} = \mathbf{K} + \bar{\mathbf{K}} + \mathbf{Q}\left(\hat{\mathbf{U}}^{\mathbf{T}}\mathbf{K}\hat{\mathbf{U}} + \boldsymbol{\Lambda}\right)\mathbf{Q}^{\mathbf{T}} - \mathbf{R}\hat{\mathbf{U}}\mathbf{Q}^{\mathbf{T}} - \mathbf{Q}\hat{\mathbf{U}}\mathbf{R}^{\mathbf{T}}$$

Con

$$\bar{\mathbf{K}} = -\mathbf{K}\hat{\mathbf{U}}\mathbf{Q}^{\mathrm{T}} - \mathbf{Q}\hat{\mathbf{U}}^{\mathrm{T}}\mathbf{K} + \mathbf{R} + \mathbf{R}^{\mathrm{T}}$$

$$\mathbf{Q} = \mathbf{M}\hat{\mathbf{U}}\mathbf{M}_{\mathrm{U}}^{-1}$$

$$\mathbf{R} = \mathbf{Q}\Lambda\mathbf{M}_{\mathrm{U}}\Lambda\mathbf{Q}^{-1}\hat{\mathbf{U}}^{\mathrm{T}}\mathbf{K}$$

2.2.7 Metodo ai valori propri multi-parametrizzato

Un altro metodo consiste nel trasformare il problema analitico lineare in un problema ai valori propri multi-parametrizzato che necessita un numero piccolo di valori misurate e che possiede il vantaggio di non avere a usare i metodi di riduzione o di espansione per fare corrispondere il numero di misure con il numero di gradi di libertà e quindi di non rendere non lineare il problema iniziale [2].

Parametrizzazione del sistema

L'insieme formato dal rotore e dei cuscinetti può essere modellizzato con un'equazione differenziale del tipo:

$$\mathbf{A}_2\ddot{\mathbf{u}}(t) + \mathbf{A}_1\dot{\mathbf{u}}(t) + \mathbf{A}_0\mathbf{u}(t) = \mathbf{S}\mathbf{p}(t)$$

Nella quale $\mathbf{A}_0$ rappresenta la matrice di rigidezza, $\mathbf{A}_1$ la matrice di smorzamento $\mathbf{A}_2$ la matrice d'inerzia del sistema. **p(t)** rappresenta la forzante che si applica al sistema.
Se il sistema è composto di n gradi di liberta i vettori $\ddot{\mathbf{u}}$, $\dot{\mathbf{u}}$, $\mathbf{u}$ e **p(t)** saranno composti di n coordinate e le matrici $\mathbf{A}_2$, $\mathbf{A}_1$ e $\mathbf{A}_0$ saranno di ordine n, sono anche simmetriche e definite positive. La matrice $\mathbf{S}$ d'*input* di dimensione $(n \times s)$ con $s \ll n$.

Il vettore di misura $\tilde{\mathbf{u}}$ della risposta si può scrivere come:

$$\tilde{\mathbf{u}}(t) = \mathbf{H}.\mathbf{u}(t) + \mathbf{R}(t)$$

H è la matrice di *output*, è di dimensione $(m \times n)$ con $m \ll n$. $R(t)$ è il vettore che rappresenta il rumore dovuta alla misura.

Per parametrizzare il modello si può scomporre le matrici $\mathbf{A_i}$, $i=0,1,2$ in una somma di matrice $\mathbf{A_{ik}}$ moltiplicate per un parametro adimensionale a_{ik} con $(i,k)\in[\![0,2]\!]\times[\![1,R_i]\!]$, dove R_i è il numero di parametri scelti per la matrice A_i. Avremo quindi una rappresentazione delle matrici in questo modo:

$$\mathbf{A_i}(a_i)=\sum_{k=1}^{R_i} a_{ik}\mathbf{A_{ik}}$$

I parametri a_{ik} sono inizialmente imposti al valore uno. Si cerca di costruire le matrici $\mathbf{A_{ik}}$ in modo tale che esse abbiano un significato fisico. In questo caso i parametri a_{ik} saranno dei numeri positivi reali perché dei valori nulli o negativi cambierebbero la natura fisica del sistema.

Risoluzione del problema matematico

Il problema che sarà risolto si può presentare come un sistema di k equazione di forma:

$$\begin{bmatrix} \beta_0\mathbf{A_{10}}\mathbf{g_1}+\beta_1\mathbf{A_{11}}\mathbf{g_1}+\beta_k\mathbf{A_{1k}}\mathbf{g_k} \\ \vdots \\ \beta_0\mathbf{A_{k0}}\mathbf{g_k}+\beta_1\mathbf{A_{k1}}\mathbf{g_k}+\beta_k\mathbf{A_{kk}}\mathbf{g_k} \end{bmatrix}=\begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}$$

$\mathbf{A_{rs}}, (r,s)\in[\![1,k]\!]\times[\![0,k]\!]$ sono degli endomorfismi che operano nello stesso spazio vettoriale.

$$\forall r, r\in[\![1,k]\!], \mathbf{A_{r1}},.....,\mathbf{A_{rk}}: G_r\to H_r$$

Le auto valori $(\beta_0,\beta_1,\cdots,\beta_k)\neq 0_{\mathbb{C}^k}$ devono verificare l'equazione:

$$\det(\sum_{s=0}^{k}\mathbf{A_{rs}}\beta_s)=0,\ \forall r\in[\![1,k]\!]$$

Se definiamo G come il prodotto tensoriale dei G_r possiamo anche definire l'operatore $A_r^{\dagger}$ come segue:

$$\mathbf{A_r^{\dagger}}: G\coloneqq G_1\otimes\cdots\otimes G_r\otimes\cdots\otimes G_k\to G_1\otimes\cdots\otimes H_r\otimes\cdots\otimes G_k$$

Per un elemento di G chiamato $\mathbf{g}=\mathbf{g_1}\otimes\cdots\otimes\mathbf{g_k}$ avremo l'uguaglianza seguente:

$$\mathbf{A_r^\dagger}(\mathbf{g_1}\otimes\cdots\otimes\mathbf{g_k}) = \mathbf{g_1}\otimes\cdots\otimes\mathbf{A_r g_r}\otimes\cdots\otimes\mathbf{g_k}$$

Questi operatori sono ovviamente commutativi, $\forall(i,j),\ i\neq j, \mathbf{A_i^\dagger A_j^\dagger} = \mathbf{A_j^\dagger A_i^\dagger}$.

Se ritorniamo agli operatori A_{rs} introdotti precedentemete, possiamo definire per ogni uno un operatore corrispondente $A_{rs}^\dagger$ che deve verificare:

$$\forall(r,s)\in[\![1,k]\!]\times[\![0,k]\!],\ Ker(\sum_{s=0}^{k}\mathbf{A_{rs}^\dagger}\beta_s)\neq 0$$

Si questa ineguaglianza è verificata allora esiste $f\in G$, $f\neq 0_G$ tale che:

$$\sum_{s=0}^{k}\mathbf{A_{rs}^\dagger}\beta_s\mathbf{f} = 0,\ \forall r\in[\![1,k]\!]$$

Per trovare un vettore f che verifica l'equazione precedente, definiamo Δ_s come un determinante della matrice dei A_{rs}:

$$\mathbf{\Delta_s} \coloneqq (-1)^s \det\begin{bmatrix} \mathbf{A_{10}^\dagger} & \cdots & \hat{\mathbf{A}}_{\mathbf{1s}}^\dagger & \cdots & \mathbf{A_{1k}^\dagger} \\ & \vdots & & \vdots & \\ \mathbf{A_{k0}^\dagger} & \cdots & \hat{\mathbf{A}}_{\mathbf{ks}}^\dagger & \cdots & \mathbf{A_{kk}^\dagger} \end{bmatrix}$$

Dove il cappello indica l'omissione della colonna numero $s+1$. Si può dimostrare che risolvere il problema delle auto valori è equivalente a risolvere il problema seguente che si limita alla ricerca di auto valori e auto vettori associati:

$$\forall s\in[\![1,k]\!],\quad \beta_s\mathbf{f} = \mathbf{\Delta^{-1}\Delta_s f}$$

$$con\ \ \mathbf{\Delta} = \sum_{s=0}^{k}\mu_s\mathbf{\Delta_s}$$

I μ_s sono scelti in modo di rendere la matrice $\mathbf{\Delta}$ invertibile.

Applicazione al processo di model updating

Si parte anche in questo caso dal presupposto che le matrici di massa e di smorzamento sono note. Con le equazioni di parametrizzazione possiamo scrivere il nostro modello nel caso non smorzato come:

$$\forall i \in [\![1,k]\!], \quad \left(-\hat{\omega}_i \mathbf{A_2} + \sum_{s=1}^{k} a_{0s} \mathbf{A_{0s}} \right) \mathbf{x}_i = 0$$

Dove

- $\omega_i = 2\pi f_i$ sono le pulsazioni proprie misurate dagli esperimenti
- $\mathbf{x_i}$ sono i corrispondenti auto-vettori non misurati
- i a_{0s} sono i parametri della matrice di rigidezza da determinare

Possiamo quindi per identificazione dei coefficienti mettere questo modello sotto la forma di quello matematico risolto più in alto con:

- $\mathbf{A_{r0}} = -\hat{\omega}_r^2 \mathbf{A_2}$ $\quad \forall r \in [\![1,k]\!]$
- $\mathbf{A_{rs}} = \mathbf{A_{0s}}$ $\quad \forall s \in [\![1,k]\!]$
- $\frac{\beta_s}{\beta_0} = a_{0s}$ $\quad \forall s \in [\![1,k]\!]$
- $\mathbf{g_r} = \mathbf{x_r}$ $\quad \forall r \in [\![1,k]\!]$

β_0 essendo direttamente legato alla matrice di massa conosciuta e siccome serve a normalizzare gli altri parametri, non sarà modificato e quindi la matrice di massa rimarrà uguale dopo l'*Updating*. Basta dopo risolvere il problema con il metodo descritto nella parte di risoluzione matematica.

Updating nel dominio delle frequenze per la matrice di rigidezza

In questo caso si suppongono note le matrici di massa e di smorzamento. Nel dominio delle frequenze possiamo trasformare la nostra equazione di partenza che diventerà:

$$\forall i \in [\![1,k]\!]$$

$$(-\omega_i^2 \mathbf{A_2} + j\omega_i \mathbf{A_1} + \sum_{s=1}^{k} a_{0s} \mathbf{A_{0s}}) \mathbf{U}(\omega_i) = \mathbf{SP}(\omega_i)$$

$\mathbf{P}(\omega_i)$ e $\mathbf{U}(\omega_i)$ sono le trasformate di Fourrier degli input e degli output del sistema alla frequenza ω_i. Supponiamo di avere misurato $\mathbf{P}$ grazie alla matrice d'*input* **S**.
Ricordiamo che $\tilde{\mathbf{u}}(t) = \mathbf{H}\mathbf{u}(t) + \mathbf{R}(t)$, se supponiamo l'assenza di rumore, avremo nel dominio delle frequenze $\tilde{\mathbf{U}}(\omega_i) = \mathbf{H}\mathbf{U}(\omega_i)$. Nel caso nel quale i punti di misura corrispondono esattamente ai punti di eccitazione, avremo che $\mathbf{H}$ e $\mathbf{S}$ saranno trasposte:

$$\mathbf{H} = \mathbf{S}^{\mathbf{T}}$$

Se separiamo la parte reale e la parte immaginaria dei vettori $\mathbf{U}(\omega_i)$ e $\mathbf{P}(\omega_i)$ otteniamo quattro vettori, $\mathbf{U_R}(\omega_i)$, $\mathbf{U_I}(\omega_i)$, $\mathbf{P_R}(\omega_i)$ e $\mathbf{P_I}(\omega_i)$ che verificano l'equazione seguente:

$$(-\omega_i^2\mathbf{A_2} + j\omega_i\mathbf{A_1} + \sum_{s=1}^{k} a_{0s}\mathbf{A_{0s}})(\mathbf{U_R}(\omega_i) + j\mathbf{U_I}(\omega_i)) = \mathbf{S}(\mathbf{P_R}(\omega_i) + j\mathbf{P_I}(\omega_i))$$

Se si separano le parti reali e le parti immaginarie si ottengono due equazioni:

$$\begin{cases} -\omega_i^2\mathbf{A_2}\mathbf{U_R}(\omega_i) - \omega_i\mathbf{A_1}\mathbf{U_I}(\omega_i) + (\sum_{s=1}^{k} a_{0s}\mathbf{A_{0s}})\mathbf{U_R}(\omega_i) - \mathbf{S}(\mathbf{P_R}(\omega_i) = 0 \\ -\omega_i^2 j\mathbf{A_2}\mathbf{U_I}(\omega_i) + \omega_i\mathbf{A_1}\mathbf{U_R}(\omega_i) + (\sum_{s=1}^{k} a_{0s}\mathbf{A_{0s}})\mathbf{U_I}(\omega_i) - \mathbf{S}(\mathbf{P_I}(\omega_i) = 0 \end{cases}$$

Se aggiungiamo un'equazione che descrive la conservazione dell'energia:

$$C_i = -((\mathbf{S}\mathbf{P}(\omega_i))^T\mathbf{U}(\omega_i))_R = (\mathbf{S}\mathbf{P_R}(\omega_i))^T\mathbf{U_R}(\omega_i) - (\mathbf{S}\mathbf{P_I}(\omega_i))^T\mathbf{U_I}(\omega_i)$$

Possiamo allora scrivere tutto il sistema sotto forma di equazione matriciale:

$$\left(\begin{bmatrix} \omega_i^2\mathbf{A_2} & \omega_i\mathbf{A_1} & -\mathbf{S}\mathbf{P_R}(\omega_i) \\ \omega_i\mathbf{A_1} & -\omega_i^2\mathbf{A_2} & -\mathbf{S}\mathbf{P_I}(\omega_i) \\ (-\mathbf{S}\mathbf{P_R}(\omega_i))^T & (\mathbf{S}\mathbf{P_I}(\omega_i))^T & \mathbf{C_i} \end{bmatrix} + \sum_{s=1}^{k} a_{0s}\begin{bmatrix} -\mathbf{A_{0s}} & 0 & 0 \\ 0 & \mathbf{A_{0s}} & 0 \\ 0 & 0 & 0 \end{bmatrix}\right)\begin{bmatrix} \mathbf{U_R}(\omega_i) \\ \mathbf{U_I}(\omega_i) \\ -1 \end{bmatrix} = 0$$

La simmetria delle matrici $\mathbf{A_0}$, $\mathbf{A_1}$ e $\mathbf{A_2}$ implica la simmetria della meta matrice dell'equazione. Il calcolo dei valori di $\mathbf{C_i}, i \in [\![1, k]\!]$ non necessita un gran numero di dati sperimentali. Questa forma è analoga alla forma del problema

visto più in alto se può quindi risolvere con lo stesso metodo per trovare i nuovi parametri.

Uno dei vantaggi di questo metodo è che la parametrizzazione scelta permette di mantenere un senso fisico dopo l'Updating poiché abbiamo separato il modello a elementi finiti in pezzi coerenti. Questa procedura si può applicare ai due casi per correggere la matrice di massa e nel secondo caso alla matrice di smorzamento con la parametrizzazione adatta. Tutti due i casi suppongono una matrice inizialmente conosciuta. La differenza tra questi due metodi è che il secondo caso necessita le misure della forzante e della risposta del sistema per assicurare l'unicità dei parametri ricavati. Permette anche di soddisfarsi anche di un piccolo numero di dati misurati (lo stesso che i parametri da ottimizzare nel primo caso). Nel secondo caso sono necessari solo nei punti di eccitazione del sistema. Un altro vantaggio è di non avere da completare il set dei dati ottenuti.

Lo svantaggio di questo metodo è il tempo di calcolo impiegato dal computer perché è in gioco un numero gigantesco di operazione, anche se le matrici considerate contengono un bel po' di zeri.

2.3 Metodi iterativi

I metodi iterativi sono una buona scelta perché permettono una scelta accurata dei parametri da ottimizzare e quindi permettono di mantenere il senso fisico del modello dopo l'*Updating.* Permettono quindi di ottenere dei modelli che offrono la possibilità di fare delle analisi su una macchina reale con il modello ottenuto.

Bisogna comunque usare matrice di peso per i dati modali e il modello analitico per permettere l'aggiornamento. Questa parte è molto difficile da riuscire perché richiede una buona conoscenza della natura fisica del problema.

Questi metodi sono basati sulla misura dei dati modali (frequenze proprie e modi di vibrare) e cercano di minimizzare la distanza tra i dati misurati e quelli ottenuti con il modello analitico aggiornato. Questa distanza è misurata da una funzione obiettivo, solitamente la somma della differenza dei quadrati. Se abbiamo i dati misurati scritti come $\hat{\mathbf{x}} = (\hat{x}_1, \ldots, \hat{x}_n)$ e i dati della k-esima operazione notati come $\mathbf{x_k} = (x_{1k}, \ldots, x_{nk})$ potremo scrivere il valore ritornato alla k-esima iterazione E_k come:

$$E_k = \left\| \mathbf{x_k} - \hat{\mathbf{x}} \right\|_2 = \sum_{i=1}^{n} (x_{ik} - \hat{x}_i)^2$$

Come l'abbiamo detto la parametrizzazione è il punto forte di questo metodo, adesso vedremo due metodi di parametrizzazione per il nostro modello analitico che danno un senso fisco ai parametri scelti.

2.3.1 Parametrizzazione del Modello

Una buona parametrizzazione del sistema permette una migliore efficienza dei metodi iterativi. E difficile scegliere bene i parametri che saranno ottimizzati perché non hanno tutti lo stesso ruolo nel sistema. Un modello può essere più o meno sensibile a certi parametri. I parametri sensibili sono quelli che creano una grande variazione della risposta per un piccolo spostamento relativo. Un altro fattore da considerare è anche la precisione della modellizzazione, è meglio dirigere l'aggiornamento sulle zone del modello meno conosciute perché, per mantenere il senso fisico, è sempre meglio non toccare alle zone note. Al riguardo dei tempi di calcolo e della semplicità degli algoritmi è sempre meglio usare un piccolo numero di parametri da utilizzare, il limite sul numero di misure spinge anche verso questa direzione.

Per le due parametrizzazione scelte indicheremo con p il numero di parametri da aggiornare e con $(a_i)_{i\in[\![1,p]\!]}$ l'elenco di questi parametri.

- Parametrizzazione segando il modello a elementi finiti.

La prima idea che svolgeremo è quella di Natke di sub-dividere le matrici analitiche in matrice della stessa dimensione però che isolano dei determinati pezzi del modello totale. Matematicamente si può descrivere in questo modo, per esempio per la matrice di massa:

$$\tilde{\mathbf{M}} = \mathbf{M}_0 + \sum_{i=1}^{p} a_i \mathbf{M}_\mathbf{i}$$

La conoscenza dei $(a_i)_{i\in[\![1,p]\!]}$ permette di capire quali sono le parti del modello modificate durante il processo. Ogni matrice $\mathbf{M_i}$ rappresenta la matrice di massa di uno degli elementi del modello finito da ottimizzare è sarà quindi riempita solo nel posto corrispondente all'elemento e vuota nel resto. La matrice $\mathbf{M_0}$ rappresenta la matrice iniziale di massa, $\tilde{\mathbf{M}}$ sarà quindi la matrice aggiornata. Per inizializzare l'*Updating*, una scelta è di uguagliare tutti i parametri a zero. In questo modo cercando i valori dei a_i intorno a zero si procede a delle modifiche sugli elementi finiti.

- Parametrizzazione fisica

Questa parametrizzazione considera le matrici analitiche come delle funzioni dei parametri fisici del sistema come il modulo di Young, le caratteristiche geometriche, la densità di massa ecc... Possiamo quindi scrivere lo sviluppo in serie di Taylor al primo ordine. Per esempio per la massa avremo:

$$\tilde{\mathbf{M}} = \mathbf{M_0} + \sum_{i=1}^{p} \delta a_i \frac{\partial \tilde{\mathbf{M}}}{\partial a_i}$$

Questo modello può quindi risultare equivalente al modello precedente con:

$$\mathbf{M_i} = \frac{\partial \mathbf{M}}{\partial a_i}, \quad i \in [\![1, p]\!]$$

Questo sotto intende di valutare le derivate solo all'istante iniziale e di tenerle durante tutto il processo. E possibile un altro approccio che consiste a rivalutare ad ogni passo le derivate per il calcolo della nuova matrice aggiornata. Se notiamo con l'esponente k le quantità del k-esima iterazione avremo la relazione di ricorrenza seguente:

$$\tilde{\mathbf{M}}_{k+1} = \tilde{\mathbf{M}}_k + \sum_{i=1}^{p} (\delta a_i)^{k+1} \frac{\partial \tilde{\mathbf{M}}_{k+1}}{\partial a_i}$$

2.3.2 Metodo con matrice di sensibilità

Scriviamo il vettore delle misure $\hat{\mathbf{z}}$ definito come segue:

$$\hat{\mathbf{z}} = \left(\hat{\lambda}_1, \hat{\mathbf{u}}_\mathbf{1}^\mathbf{T}, \ldots, \hat{\lambda}_p, \hat{\mathbf{u}}_\mathbf{m}^\mathbf{T} \right)^\mathbf{T}$$

Dove $\hat{\mathbf{u}}_i$ rappresenta il modo di vibrare misurato associato all'auto-valore $\hat{\lambda}_i$, non c'è per forza lo stesso numero di auto-valori che quello di auto-vettori in questa definizione. Sia **z** il vettore delle predizioni associato al vettore **z**:

$$\mathbf{z} = \left(\lambda_1, \mathbf{u}_\mathbf{1}^\mathbf{T}, \ldots, \lambda_p, \mathbf{u}_\mathbf{m}^\mathbf{T} \right)^\mathbf{T}$$

Sia $\mathbf{a}^\mathbf{k}$ il vettore dei parametri alla k-esima iterazione:

$$\mathbf{a}^\mathbf{k} = \left(a_1^k, \ldots, a_p^k \right)^\mathbf{T}$$

Sviluppiamo al primo ordine queste due quantità, avremo quindi:

$$\delta \mathbf{z}^\mathbf{k} = \mathbf{S}^\mathbf{k} \delta \mathbf{a}^\mathbf{k} \quad \text{con} \quad \begin{cases} \overline{\mathbf{a}} = \mathbf{a}^\mathbf{k} + \delta \mathbf{a}^\mathbf{k} \\ \hat{\mathbf{z}} = \mathbf{z}^\mathbf{k} + \delta \mathbf{z}^\mathbf{k} \end{cases}$$

E $\mathbf{S_k}$ la matrice di sensibilità descritta da:

$$\mathbf{S_k} = \left(\left. \frac{\partial z_i}{\partial a_j} \right|_{a_j = a_j^k} \right)_{(i,j) \in [\![1,M]\!] \times [\![1,p]\!]}$$

$\overline{\mathbf{a}}$ sarebbe il set di parametri che riproduce la risposta misurata $\hat{\mathbf{z}}$.

Caso di numero di misure maggiore di quello dei dati:

Useremo come funzione da minimizzare a ogni iterazione la funzione seguente che rappresenta l'errore, residua:

$$E_k(\mathbf{x}) = \left(\delta \mathbf{z}^{\mathbf{k}} - \mathbf{S}^{\mathbf{k}}\mathbf{x}\right)\left(\delta \mathbf{z}^{\mathbf{k}} - \mathbf{S}^{\mathbf{k}}\mathbf{x}\right)^{\mathbf{T}}$$

Derivando rispetto a $\mathbf{x}$ e uguagliando a zero viene $\mathbf{x} = \left(\left(\mathbf{S}^{\mathbf{k}}\right)^{\mathbf{T}}\mathbf{S}^{\mathbf{k}}\right)^{-1}\left(\mathbf{S}^{\mathbf{k}}\right)^{\mathbf{T}}\delta \mathbf{z}^{\mathbf{k}}$, possiamo quindi calcolare il set di parametri aggiornati all'iterazione k+1 con la formula seguente:

$$a^{k+1} = a^k + \left(\left(\mathbf{S}^{\mathbf{k}}\right)^{\mathbf{T}}\mathbf{S}^{\mathbf{k}}\right)^{-1}\left(\mathbf{S}^{\mathbf{k}}\right)^{\mathbf{T}}\delta \mathbf{z}^{\mathbf{k}}$$

Si itera quindi il procedimento fino a verificare la condizione di convergenza scelta dall'utilizzatore. Questo metodo comporta lo svantaggio di assegnare lo stesso peso a tutti i dati misurati, come il vettore delle misure comporta delle frequenze che sono abbastanza precise e dei modi di vibrare più imprecisi, l'aggiornamento verrà inquinato dai dati sui modi di vibrare.

Caso di numero di misure inferiore a quello dei dati:

Questo caso è frequente a causa delle problematiche nelle misure legato agli aspetti tecnologici. In questo caso la matrice $\left(\mathbf{S}^{\mathbf{k}}\right)^{\mathbf{T}}\mathbf{S}^{\mathbf{k}}$ non sarà invertibile e quindi non si può procedere come nel caso precedente. Ci sarà quindi un infinito numero di soluzione possibile e il problema diventa di definire un criterio per scegliere il set di parametri più opportuno. Spesso si sceglie il set di parametri il più vicino al set iniziale quindi il problema di minimizzazione diventa un problema con una condizione che si può risolvere con il metodo di Lagrange sviluppato più in alto.

Il problema è il seguente all'iterazione k, minimizzare:

$$\varphi : \mathbb{R}^p \to \mathbb{R}, \quad \mathbf{x} \to \mathbf{x}\mathbf{x}^{\mathbf{T}}$$

La famiglia di funzione imponendo la condizione all'iterazione k sarà:

$$\forall i \in [\![1, p]\!],\ \phi_i^k : \mathbb{R}^p \to \mathbb{R}, \quad \mathbf{x} \to \left(\delta \mathbf{z}^{\mathbf{k}} - \mathbf{S}^{\mathbf{k}}\mathbf{x}\right)_i$$

Che corrisponde alla condizione seguente: $\delta \mathbf{z}^{\mathbf{k}} = \mathbf{S}^{\mathbf{k}}\mathbf{x}$

Con quest'approccio e risolvendo con i moltiplicatori di Lagrange si ricava la soluzione seguente:

$$\mathbf{x} = \left(\mathbf{S}^{\mathbf{k}}\right)^{\mathbf{T}} \left(\mathbf{S}\left(\mathbf{S}^{\mathbf{k}}\right)^{\mathbf{T}}\right)^{-1} \boldsymbol{\delta}\mathbf{z}^{\mathbf{k}}$$

E quindi avremo all'iterazione k+1 il risultato:

$$\mathbf{a}^{\mathbf{k+1}} = \mathbf{a}^{\mathbf{k}} + \left(\mathbf{S}^{\mathbf{k}}\right)^{\mathbf{T}} \left(\mathbf{S}\left(\mathbf{S}^{\mathbf{k}}\right)^{\mathbf{T}}\right)^{-1} \boldsymbol{\delta}\mathbf{z}^{\mathbf{k}}$$

Uno dei grandi svantaggi di questo metodo è che linearizzando il problema con lo sviluppo al primo ordine il processo può rimanere intrappolato in una zona di minimo locale quindi perdere tanto in efficienza.

2.4 Metodi stocastici

In questa parte parleremo dei differenti algoritmi stocastici che permettono di realizzare il model updating iterativo. Gli algoritmi di questo genere sono spesso ispirati a fenomeni naturali o fisici e cercano di fare la trasposizione questi meccanismi [3][7].

Le tecniche classiche di ottimizzazione hanno lo svantaggio di rimanere bloccati negli estremi locali visto che sono basate sulla pendenza della funzione obiettiva, quindi se c'è un "buco" può non riuscire a risalire. Non è detto neanche che l'algoritmo riesca a convergere, quindi sono state sviluppate recentemente delle tecniche di ottimizzazione basate sulla probabilità che possono riuscire a superare le "trappole" dei minimi locali. Queste tecniche euristiche di ottimizzazione sono state sviluppate negli anni '60, sfruttando lo sviluppo recente dei nuovi mezzi di calcolo.

Qui parleremo essenzialmente di tecniche di due nature diverse, una basata su un'analogia fisica che si chiama Simulated Annealing (Ricottura Simulata) e di una famiglia di tecniche che si possono classificare come tecniche di calcolo evolutivo che si possono suddividere in tre tipi di ottimizzazioni diverse, avendo però un'ispirazione comune. L'idea ripresa da questi algoritmi è la teoria dell'evoluzione di Darwin.

In questo capitolo cercheremo di capire quali sono i parametri che influenzano l'efficienza, i vantaggi e gli svantaggi di ogni algoritmo.

2.4.1 Simulated Annealing (SA)

Questo processo di ottimizzazione è stato costruito facendo l'analogia con un fenomeno fisico [4]. Certi materiali hanno diversi livelli di energia in corrispondenza della loro configurazione molecolare. Il processo di Annealing è un processo che permette di raggiungere il livello più basso di energia possibile. Annealing significa ricottura, questo processo consiste nel ricuocere un materiale fino a quando non si scioglie. Poi il materiale si raffredda lentamente facendo diminuire discretamente la temperatura. Se la temperatura varia davvero lentamente, il processo riesce a raggiungere il minimo di energia: questo è il processo che viene simulato in modo tale da ottenere in metodo di ottimizzazione efficiente.

Nel 1877 Boltzman teorizzò il comportamento dei fluidi e descrisse come lo stato di un componente dipenda della posizione spaziale di ogni componente del fluido. Un numero limitato di atomi offre un gran numero di configurazioni possibili.

Chiamiamo $E(s)$ l'energia associata allo stato s del fluido, se il sistema è in equilibrio termico alla temperatura T la probabilità che il sistema sia nello stato s è data da:

$$\pi_T(s) = \frac{e^{-E(s)/kT}}{\sum_{w \in S} e^{-E(w)/kT}}$$

k è la costante di Boltzman e S l'insieme degli stati possibili.

L'algoritmo che ne deriva, detto *Metropolis algoritm* è costruito come segue:

- Si parte da un stato chiamato s_{old} con la sua energia $E(s_{old})$
- Un atomo scelto a caso viene perturbato, lo stato diventa s_{new} con energia $E(s_{new})$.
- Questo stato viene o accettato o rifiutato a seconda del criterio di Metropolis descritto da :

se $E(s_{new}) < E(s_{old})$ il nuovo stato è automaticamente accettato, altrimenti la probabilità che sia accettato è data da $P(accept - s_{new}) = e^{-\{E(s_{old}) - E(s_{new}))/kT\}}$
Se si itera questo processo la probabilità di essere in un certo stato tende a quella descritta da Boltzman.

Nel 1983 è stato scritto l'algoritmo di ottimizzazione analogo a quello di *Metropolis*. I componenti analoghi dell'ottimizzazione sono i seguenti:

Metropolis algoritm	**Optimization algoritm**
Stato del sistema	Parametri d'input scelti
Funzione d'energia	Funzione obiettiva
Temperatura	Variabile di controllo

Il nuovo stato nell'algoritmo è trovato come uno stato a caso generato in vicinanza dello stato. Si fa questa generazione usando uno spostamento aleatorio con una funzione di probabilità. Diverse funzioni sono state proposte, le principale usate sono le seguente:

- Aggiustamento lineare: si prende un parametro a casa e si cambia il suo valore usando una funzione uniformemente distribuita nel dominio di definizione del parametro.

- Aggiustamento con raggio fisso: si genera un nuovo punto sulla sfera di raggio definito e di centro lo stato precedente. Richiede la definizione di un nuovo parametro, il raggio.

- Aggiustamento normale: Ogni parametro viene ricalcolato usando la distribuzione normale centrata in zero Bisogna definire la varianza σ come nuovo parametro.

- Aggiustamento di Cauchy: Simile all'aggiustamento normale però si usa la distribuzione di Cauchy invece di quella normale.

L'algoritmo in dettaglio:

Si parte da una temperatura inizia T_0 e si itera l'algoritmo Mertropolis visto sopra finché il sistema all'equilibrio termodinamico. Una volta raggiunto, si moltiplica la temperatura per un coefficiente ρ tale che $0 < \rho < 1$. Di solito si usa un valore molto vicino a uno (0.95). Si procede così fino a quando ci sono posso pochi movimenti nel sistema e cioè il sistema è gelato.

Si può definire l'accettazione del sistema come un modo per controllare l'efficienza del l'algoritmo.

$$\phi_0 = \frac{Numero\ di\ mosse\ accettate}{Numero\ di\ mosse\ totale}$$

Si cerca di scegliere la temperatura T_0 tale che l'accettazione sia un valore compreso tra 0.5 e 0.9.

Se questo indicatore è molto inferiore a 0.5, c'è un gran rischio che l'ottimizzazione rimanga intrappolata in un minimo locale.

Sè molto superiore a 0.9, l'algoritmo passa la maggiore parte del tempo nello stato fuso e quindi effettua poco più che un percorso aleatorio.

Inconvenienti:

Il principale inconveniente deriva dal fatto che si deve scegliere un numero molto importante di parametri per fare funzionare questo algoritmo e spesso si può scegliere solo in modo empirico. Si può quindi pensare che questo algoritmo sia spesso usato al di sotto delle sue capacità reali visto che la parametrizzazione migliore è difficile da ottenere.

Semi procedura per il modello SA:

Si suppone di disporre di un insieme di stati (configurazioni) possibili G e di una funzione di evaluazione F .

Definire la "temperatura" iniziale $T(0)$ e $N(0)$
Impostare il contatore di generazione, $g := 0$
Definire lo stato iniziale S

While non convergenza ***do***

For $j = 1 : N(g)$
$\hat{S} = generate(S)$

Calcolo della convenieneza $F[\hat{S}(g)]$

Analisi e sostituzione se conviene
If $F[\hat{S}(g)] \le F[S(g)]$
do $S(g) = \hat{S}(g)$
else ***If*** $e^{-\left\{\frac{F(S)-F(\hat{S})}{T(g)}\right\}} \ge rand[0,1]$
do $S(g) = \hat{S}(g)$
End

Test di convergenza
If Test = true $\rightarrow$ non convergenza =false
Else $g := g + 1$
Calcolo di $T(g)$ e $N(g)$
End
End
End

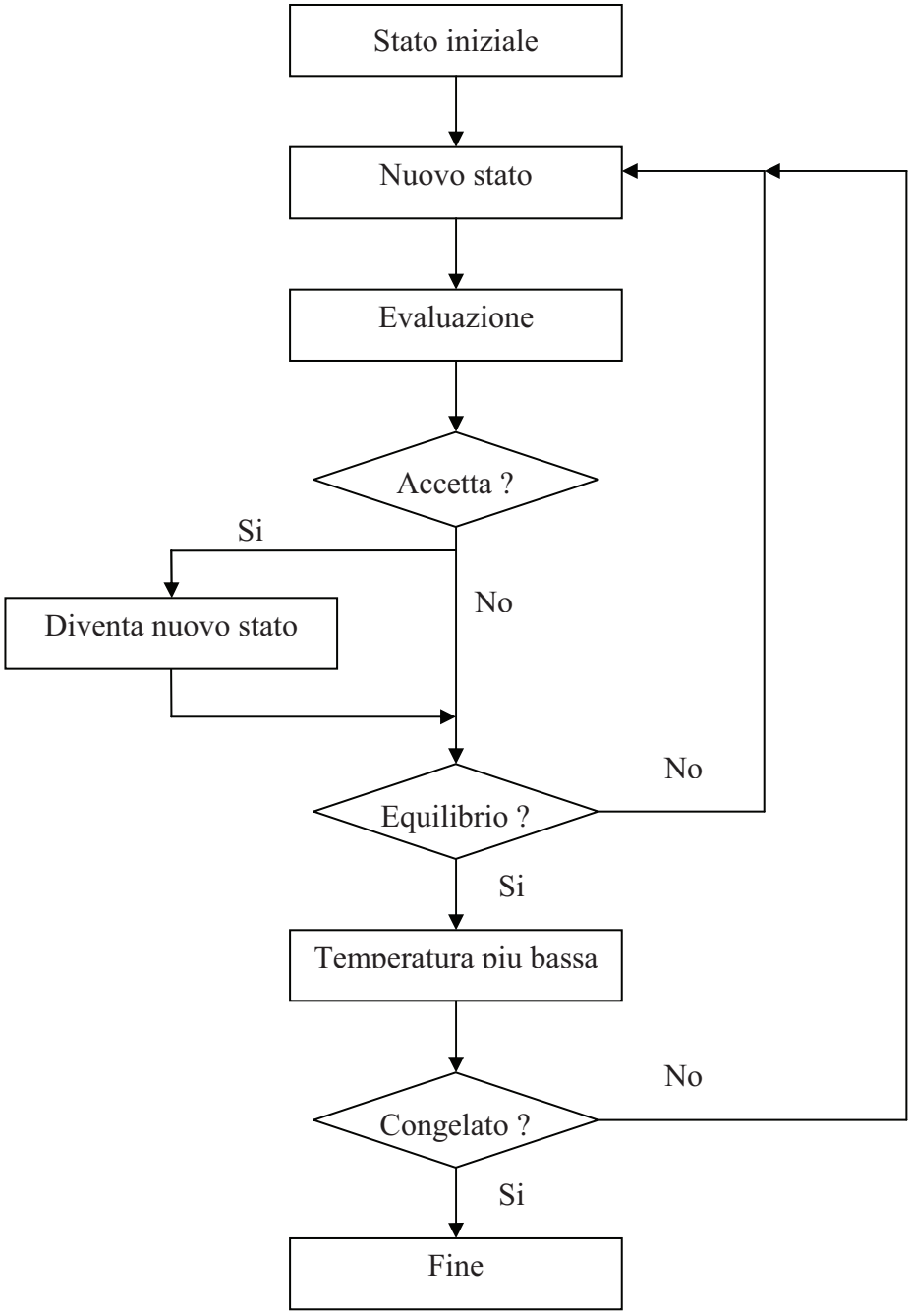
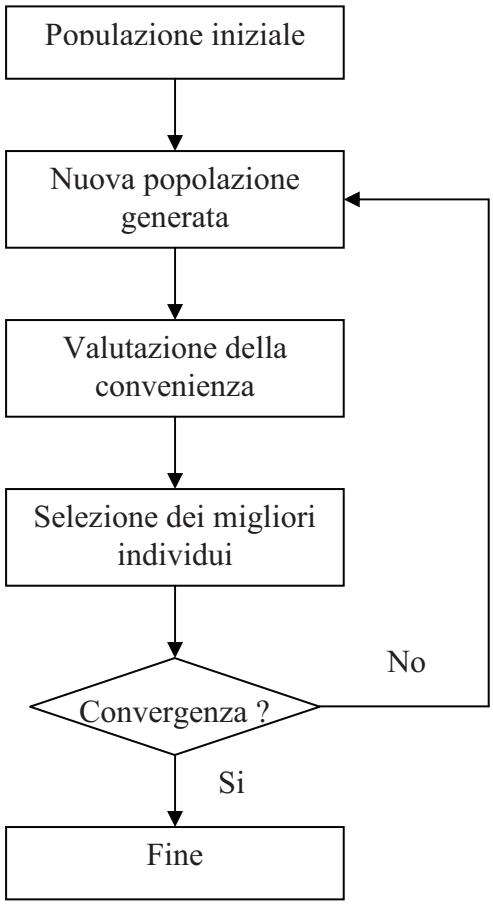


Figura 2.1 - Albero rappresentativo del Simulated Annealing

2.4.2 Algoritmi di calcolo evolutivo

Si possono considerare queste tecniche come delle metafore algoritmiche del principio di selezione naturale descritto da Charles Darwin. Uno dei punti forti di questo metodo è la semplicità concettuale dell'algoritmo. L'algoritmo si basa su una popolazione iniziale e crea una serie discendente con delle funzioni da definire [5]. Una volta create, verifica la loro convenienza con una funzione detta funzione obiettiva obiettivo che permette di misurare la distanza tra la soluzione generata dai nuovi parametri e la funzione da raggiungere.

Albero rappresentativo



Si pone quindi il problema di sapere quale funzione usare per generare i nuovi individui e come calcolare la convenienza di un set di parametri. Per quest'ultima azione si usa una funzione chiamata funzione obiettivo. Il vantaggio di questo tipo di algoritmo è che quello che l'usa non ha bisogna di conoscerla, non deve essere né esplicita né differenziabile. L'algoritmo non è

basato sulle pendenze della funzione ma solo sui suoi valori, questo è un vantaggio rispetto alle altre tecniche più classiche di ottimizzazione (tipo Newton).

2.4.3 L'algoritmo Genetico

Questo algoritmo è stato ideato da John Holland all'inizio degli anni 70 ed è stato riconosciuto come un'eccellente tecnica di ottimizzazione. Lo studio degli algoritmi genetici è stato sopratutto teorico fino agli anni '80 .

Consiste in un incrocio tra la selezione naturale e la genetica perché usa una codificazione binaria per i parametri scelti per l'ottimizzazione. Ogni catena di simboli binari è chiamata cromosoma. Nella selezione naturale, quando una caratteristica genetica da un vantaggio ad un individuo rispetto agli altri della stessa popolazione, sarà più probabile che questo sopravviva. Il punto dell'algoritmo genetico sarà quindi di selezionare gli individui che hanno le caratteristiche migliori al riguardo dell'ottimizzazione che si desidera fare e di trovare un modo per diffondere queste caratteristiche nella popolazione generata. In questo modo si può sempre migliorare la popolazione e renderla più conforme rispetto a un criterio prescelto.

Principio dell'algoritmo

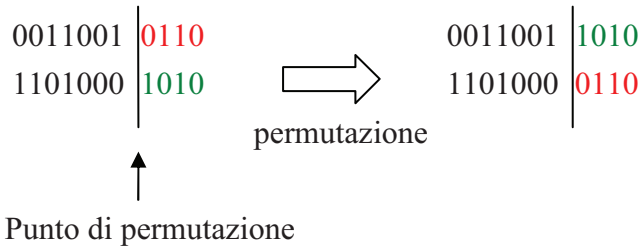
Anche per questo algoritmo c'è bisogno di una funzione obiettivo che permetta di misurare l'avanzamento dell'ottimizzazione. Si parte da una popolazione iniziale che può essere generata a caso o da dati conosciuti, alla quale si applica una serie di trasformazioni generiche, anche queste inspirate dalla natura, per cercare l'individuo ottimale.

Le trasformazioni applicabili ai cromosomi sono principalmente di tre tipi diversi:

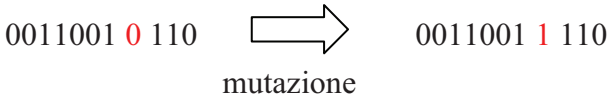
- Riproduzione: ogni cromosoma ha una probabilità di essere riprodotto in base alla sua convenienza nella risposta ottenuta. I più convenienti saranno riprodotti più volte, i meno convenienti saranno cancellati. La convenienza viene trovata in base alla funzione obiettivo obiettiva o a una tabella di classificazione predefinita dei cromosomi

- Permutazione: I cromosomi sono abbinati casualmente. Si definisce un punto di permutazione al di là del quale l'informazione viene tolta e scambiata con l'altro cromosoma. Questo operatore e applicato ad

ogni coppia di cromosomi con una probabilità tipicamente vicina a 0.6.



- Mutazione: questo operatore assegna ad ogni carattere binario una probabilità piccola di mutare (per il caso binario significa invertirsi) dell'ordine del 0.01-0.001.



Per non rischiare di perdere informazioni efficienti nella generazione della nuova popolazione si può pensare di selezionare i migliori individui e di salvarli in una banca chiamata *elite*. Questo permette di non abbassare il livello dopo ogni nuova generazione. Invece i peggiori elementi possono essere rimossi dalla popolazione e sostituiti con degli individui generati a caso che introdurranno delle proprietà che potranno essere migliore di quelli sostituti, si chiamano i *New blood.*

Parametrizzazione dell'algoritmo

Per definire totalmente questo algoritmo, bisogno imporre un certo numero di parametri :

- La taglia della popolazione N_P che definisce il numero iniziale di individui e il numero di individui selezionati ad ogni generazione.
- Le diverse probabilità per la mutazione, la permutazione e la riproduzione.
- Le dimensioni dell'*elite* e del *New blood.*

Il parametro più importante è N_P che dipende dalla taglia dei cromosomi perché una popolazione troppo grande aumenta il tempo di calcolo senza dare risultati

significativamente migliori e una popolazione troppo piccola potrebbe non trovare il minimo globale in un tempo accettabile.

Empiricamente il rapporto tra la taglia della popolazione e quella dei cromosomi, chiamato k deve essere compreso all'incirca tra 2 e 20. La codificazione binaria presenta un problema inerente alla sua forma, un numero finito di parametri può essere codificato per una dimensione di cromosoma fisso (2^N per una catena di lunghezza N).

Un problema più grave è la discontinuità, infatti, può essere che se un bit è diverso tra due catene, il risultato dato dalla funzione obiettiva sia molto diversa e quindi se pongono dei problemi di velocità di convergenza. Se un individuo è molto vicino all'ottimo e la sua convenienza è scarsa, non sarà trattenuto come soluzione e sarà difficile per l'algoritmo riuscire a trovare l'ottimo perché deve superare la barriera creata dalla discontinuità. Esiste anche il caso di convergenza troppo veloce quando un individuo molto più conveniente degli altri pero non ottimo è trovato i suoi discendenti, conquistano rapidamente il resto della popolazione intrappolando l'intera popolazione in un minimo locale e togliendo all'algoritmo genetico tutto il suo potere di ricerca spaziale.

Ci sono anche problemi legati all'operatore di permutazione che rompe le catene e quindi può separare informazioni togliendo le proprietà interessanti di certe sequenze di code.

Le prime applicazioni reali dei GA sono state effettuate all'inizio degli anni 90 per l'ottimizzazione per i sistemi energetici adesso è un metodo spesso usato in complemento di altri metodi di ottimizzazione più classiche.

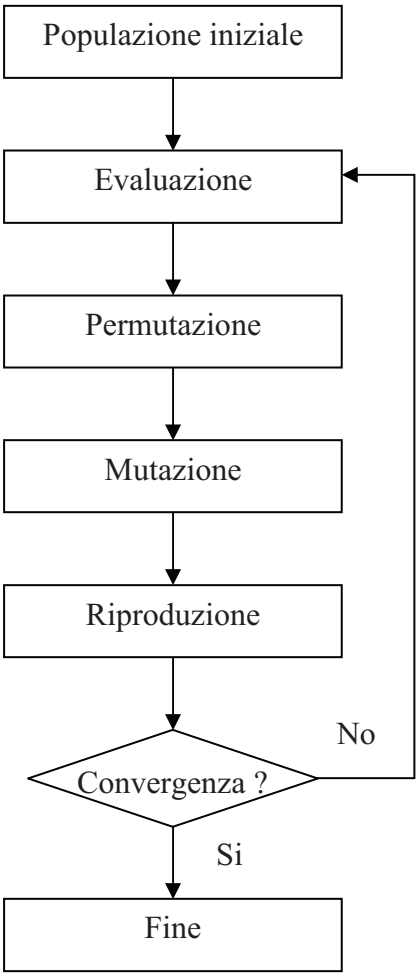


Figura 2.2 - Albero rappresentativo dell'algoritmo genetico.

2.4.4 Strategia Evolutiva (ES)

Questa famiglia di algoritmi è stata sviluppata da due scienziati tedeschi, Rechenberg e Schwefel negli anni '60 per la risoluzione di un problema di ottimizzazione. Cercavano di trovare una forma ottimale che minimizzasse la resistenza di un profilo aerodinamico. Tra il 1963 e il 1974 hanno sviluppato una teoria per la strategia evolutiva che è si è diffusa tra da un piccolo numero di persone, in particolare ingegnere civili e strutturali.

Questo metodo permette di esplorare uno spazio di soluzione usando i parametri sotto la loro forma primaria per risolvere il problema. Questo suppone che non ci sia bisogno di un sistema di codificazione prima di operare i cambiamenti e rappresenta quindi una semplificazione per la creazione degli algoritmi. Adesso ci sono tanti algoritmi derivati da questo principio, però in questo capitolo parleremo dei modelli derivati del modello generale $(\mu, \kappa, \lambda, \rho) - ES$ che spieghiamo adesso :

Modello $(\mu, \kappa, \lambda, \rho) - ES$

I simboli di questa notazione servono a fare capire subito al lettore la struttura dell'algoritmo con il seguente significato per i simboli :

- μ : Numero di genitori ad ogni generazione;
- κ : Numero di cicli massimo di sopravvivenza di un individuo;
- λ : Numero di figli creati ad ogni generazione;
- ρ : Numero di genitori coinvolti nella creazione di un figlio.

Il primo approccio fatto è stato quello del $(1+1)$-ES, cioè un genitore, un figlio e la scelta si fa ad ogni generazione tra il padre e il figlio in base alla sua convenienza. Questo è l'approccio più semplice che si possa immaginare, però si vedono subito gli svantaggi di tale metodo. Le variazioni compaiono lentamente, c'è poca diversità ad ogni generazione e la probabilità di un miglioramento dopo un salto di generazione è debole. Questo schema può quindi essere considerato come un'ossatura concettuale per ulteriori algoritmi più complessi.

(μ, λ) *-ES o* $(\mu + \lambda)$ *-ES:*

Due metodi diversi possono essere applicati per la selezione degli individui migliori. Una consiste a selezionare i μ migliori individui solo nella popolazione dei figli generati, ed è la soluzione (μ, λ)-ES. Con questo metodo,

si perde l'informazione contenuto dai genitori ad ogni generazione quindi è un problema perché ci possiamo allontanare dalla soluzione ottimale cercata e quindi c'è un rischio di divergenza non trascurabile. Si deve allora, in caso di uso di questo tipo di algoritmo, cercare un sistema che permetta di "salvare" gli individui migliori di una generazione per poter avere la possibilità di tornare indietro se le cose non vanno nel senso voluto, per esempio se il valore della funzione obiettivo crescesse di continuo. L'altro metodo consiste nel cercare gli individui nella popolazione formata dai genitori e dai loro figli in questo modo, anche se la generazione seguente non comporta miglioramenti, si possono sempre usare quelli più convenienti della generazione precedente per creare nuovi figli più convenienti. In questo caso la " memoria" è integrata nell'algoritmo di base e siamo allora sicuri che la funzione obiettivo non cresca. Possiamo concludere che per una semplicità maggiore è spesso considerata più valida la seconda soluzione.

I due altri parametri del modello generale introducono altre variazioni possibile. κ definisce la vita massima di un individuo e cioè il numero di generazioni massimo che può trascorrere. ρ indica il numero di genitori coinvolti nella generazione di un figlio quindi influirà direttamente l'operatore di ricombinazione dell'algoritmo.

Questi quattro parametri non permettono però di definire interamente il modello e la sua efficienza sarà determinata anche da altri parametri complementari elencati sotto :

- P : Popolazione iniziale;
- mut : L'operatore di mutazione;
- p_m : Probabilità di mutazione;
- rec : L'operatore di ricombinazione;
- p_r : Probabilità di ricombinazione;
- sel : L'operatore di selezione;
- Altri parametri associati ai diversi operatori.

Semi procedura per il modello $(\mu+\lambda)$*-ES:*

Definire $\mu, \lambda, \kappa, \rho, rec, mut, sel$ e i altri parametri legati.
Definire la popolazione iniziale $P(0)$ composta da μ individui.
Impostare il contatore di generazione, $g := 0$

While non convergenza ***do***

Generazione dei figli

$P_1(g) = rec[P(g)]$ / *ricombinazione della popolazione*
$P_2(g) = mut[P_1(g)]$/*mutazione della popolazione ricombinata*

Calcolo della convenienza

$Eval[P_2(g)]$

Selezione della prossima generazione

$P(g+1) = sel[P(g) \cup P_2(g)]$
Test di convergenza
If Test = true $\rightarrow$ non convergenza =false
Else $g := g+1$
End
End
End

Complementi:

Per capire a che ritmo avviene il processo di ottimizzazione, si introduce il concetto di tasso di progresso φ, definito come variazione tra una generazione e l'altra della distanza euclidiana tra l'ottimo e la posizione media della popolazione.
Per rappresentare lo spazio delle soluzioni si immagina un modello sferico dove il centro è l'ottimo cercato. Il raggio dove si trova un certo vettore y può essere definito come:

$$F(y) = c_0 + \sum_{k=1}^{N} c_k (y_k - y_k^*)$$

Con $\forall k \in [\![1, N]\!], c_i = 1$ e il simbolo $*$ indica che si tratta dell'ottimo ricercato. Se scriviamo $R_k = y_k - y_k^*$, costruiamo il vettore $\vec{R}$ avendo le coordinate R_k rappresenta il vettore delle distanze tra l'ottimo e il vettore y. Si nota $\hat{y}$ l'individuo mutato, possiamo quindi scrivere:

$$\hat{y} = y + M$$

Dove M è un vettore aleatorio.
Il tasso di progresso può essere definito come una speranza :

$$\varphi = E\left[R - r\right]$$

Dove r è la distanza dell'individuo mutato all'ottimo.

Regola del fattore 1/5:

Nel paragrafo precedente si è definito M come il vettore di mutazione creato come vettore aleatorio. E definito dalla legge di generazione (solitamente la legge normale $N(0,1)$) e dalla forza di mutazione σ in tale modo che :

$$M = \sigma(N(0,1), N(0,1), \ldots, N(0,1))^T$$

Se si mantiene la forza di mutazione costante durante tutta l'operazione potrà accadere che l'algoritmo perda in mutazione riuscita, questo significa che ad un certo punto saranno rare le mutazione che generano un figlio migliore del padre. Per questa ragione non si potrebbe mantenere questo parametro costante durante tutta l'ottimizzazione per avere una convergenza efficiente.

Una regola semplice è stata stabilita da Rechenberg e si chiama la *1/5 success rule.* Questa regola permette di controllare se l'algoritmo muta in modo efficiente e cioè se la forza di convergenza è ottimale o no. Empiricamente si è stabilito che σ è scelto bene se il rapporto tra le mutazioni che portano ad un miglioramento sull'insieme delle mutazioni è intorno a un valore di 1/5. Questo indice si può calcolare sulle 10 mutazioni precedente. Si può quindi pensare di adattare durante il processo il valore di σ per mantenere vicino a 0.2 il tasso di successo delle mutazioni pero questo comporta un rischio. Anche se questo processo accelera la scoperta dell'ottimo rende anche l'algoritmo più vulnerabile all'intrappolamento nei minimi locali.

L'auto-adattamento σSA :

Questa idea si oppone alla regola dell'1/5 di Retchenberg che è, per certi versi, contraria allo spirito della strategia evolutiva nel senso che tende a ridurre la forza di mutazione e quindi a rendere l'algoritmo deterministico. L'obiettivo di questo sistema è di impostare un sistema di auto-adattamento non imposto

dall'esterno, ma di integrarlo al processo di mutazione e di selezione. La forza di mutazione diventa quindi uno dei parametri da ottimizzare. Se un individuo è selezionato per la sua convenienza, si salvano con lui i suoi parametri. La trasformazione si può quindi rappresentare sotto questa forma :

$$\begin{pmatrix} \hat{p}_1(g) \\ \hat{p}_2(g) \\ \vdots \\ \hat{p}_i(g) \\ \vdots \\ \hat{p}_N(g) \\ \hat{\sigma}(g) \end{pmatrix} = \begin{pmatrix} p_1(g) \\ p_2(g) \\ \vdots \\ p_i(g) \\ \vdots \\ p_N(g) \\ 0 \end{pmatrix} + \sigma(g) \begin{pmatrix} N(0,1) \\ N(0,1) \\ \vdots \\ N(0,1) \\ \vdots \\ N(0,1) \\ 0 \end{pmatrix} + \xi \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ \vdots \\ 0 \\ \sigma(g) \end{pmatrix}$$

ξ è un parametro moltiplicativo di trasformazione di σ, il cui valore si attesta di solito intorno a uno e può essere generato in questo modo: $\xi = e^{\tau N(0,1)}$. τ è il cosiddetto fattore di apprendimento e definisce la velocità e la precisione dell'auto-adattamento.

Problematiche e analisi con più padri:

In questo caso, diverse tecniche sono possibili per generare i figli, se λ è un multiplo di μ si può pensare di attribuire un numero uguale di discendenti a testa. Si può anche pensare di scegliere un padre a caso, di applicargli le trasformazioni per dare un figlio e di applicare λ volte questa trasformazione. Si nota empiricamente che per essere efficienti dobbiamo avere $\mu \ll \lambda$, ovvero andare in tante direzione diverse per esplorare meglio lo spazio delle soluzioni. Lo svantaggio è quello di generare tanti individui inutili e di allungare molto il tempo di calcolo dell'algoritmo.

2.4.6 Programmazione Evolutiva (EP)

Come detto in precedenza questo genere di algoritmo sviluppato all'inizio negli Stati Uniti è in realtà molto vicino al suo cugino europeo. E stato introdotto in 1964 da Lawrence J. Fogel con un algoritmo del tipo $(3+3)$-ES per delle predizioni [7]. Si può in questo modo considerare che il modello del tipo $(\mu+\lambda)$-ES con $\mu=\lambda$ è una forma di Programmazione Evolutiva. Una differenza che spessa è accentuata per giustificare due nomenclature diverse è che nei processi ES la selezione si fa solo sul criterio data dalla funzione obiettiva e quindi dei migliori individui invece nell'EP la selezione è basata su dei tornei stocastici.

Un torneo stocastico $T(m,n)$ seleziona con una certa probabilità fissata esternamente i m migliori elementi relativamente alla funzione su un campione di n individui.

In effetto si applica una successione di tornei nei quali il migliore è scelto pero con una certa probabilità, non per forza, fino a ottenere il numero corretto d'individui per la prossima generazione cioè λ. Questo metodo non è sempre tenuto e quindi riduce ancora la piccola distanza tra ES e EP.

Gli scientisti che hanno studiato EP hanno sviluppato pure un metodo di auto-adattamento all'inizio degli anni 90. La generazione della nuova forza di mutazione è data dalla formula seguente:

$$\sigma(g+1)=\xi\sigma(g)$$

Con ξ definito da:

$$\xi=1+\tau N(0,1)$$

Dove τ è il parametro di apprendimento fissato dall'utilizzatore esternamente. Il suo valore ottimale è determinato empiricamente. Si tratta come nell'ES di una mutazione moltiplicativa.

La semi-procedura di quest'algoritmo è quasi identica a quella dell'ES, differisce del precedente nel numero di figli creati, solitamente lo stesso numero che i padri e nella selezione dei figli con il processo stocastico:

Semi procedura per il modello EP:

Definire μ, mut, sel_{stoc} e i altri parametri legati.
Definire la popolazione iniziale $P(0)$ composta da μ individui.
Impostare il contatore di generazione, $g := 0$

While non convergenza ***do***

Generazione dei figli

$P_1(g) = P(g)$ */duplicazione della popolazione iniziale*
$P_2(g) = mut[P_1(g)]$ */mutazione della popolazione ricombinata*

Calcolo della convenienza

$Eval[P_2(g)]$

Selezione della prossima generazione con i tornei stocastici

$P(g+1) = sel_{stoc}[P(g) \cup P_2(g)]$
Test di convergenza
If Test = true $\rightarrow$ non convergenza =false
Else $g := g+1$
End
End
End

L'operatore sel_{stoc} è un operatore che attribuisce une probabilità alta ai individui che hanno la migliore convenienza e una probabilità debole e che dopo seleziona gli individui in base a queste probabilità.

2.4.7 Caratteristiche comuni

Disaccoppiamento dei parametri:

In modo di esplorare meglio lo spazio delle soluzioni e permettere di focalizzare in certe direzioni per la ricerca si può usare un numero di forze di mutazione uguale al numero di parametri. In questo modo si ricerca le soluzioni non su una sfera probabilistica però su un'ellisse. Si sono ottenuti con questo metodo dei buoni risultati. Le forze di mutazione possono essere indipendenti tra di loro o correlati con una matrice di correlazione esternamente definita.

Ricombinazione:

Per il momento non abbiamo parlato della ricombinazione per generare i figli, questa tecnica lasciata da parte per un po' è adesso utilizzata come metodo efficiente per esplorare lo spazio in un modo alternativo. E il parametro ρ che definisce quanti genitori sono coinvolti nell'operazione. Per riprendere l'analogia con l'evoluzione naturale, nella natura abbiamo $\rho = 2$. Nei nostri casi non siamo costretti a limitarsi a solo due genitori, possiamo coinvolgere un numero grande di genitori in modo di modificare tanto i set di parametri. La ricombinazione si può fare in diversi modi elencati sotto:

- Ricombinazione uniforma: Si sceglie un padre a caso per ogni parametro con una probabilità uniforma.

- Ricombinazione intermedia: In questo sistema i valori sono calcolati come medie dei valori di un sotto gruppo di genitori o dell'intera popolazione dei genitori.

- Ricombinazione per tratti: Si scelgono casualmente diverse posizioni nelle catene di valori per separare tutti i figli in tratti comuni a tutti. Una volta questa decomposizione fatta si sostituiscono i tratti con quelli dei genitori.

L'esperienza ha mostrato che la ricombinazione, anche se meno intuitiva della mutazione permette di aumentare notevolmente la potenza degli algoritmi evolutivi e si è anche dedotto che più genitori partecipano alla formazione di un figlio più efficiente è l'algoritmo. L'ottimo è quindi di fare partecipare tutti i genitori alla costruzione dei discendenti.

Generazione della popolazione iniziale:

Per definire la popolazione iniziale che subirà le prime trasformazioni ci sono due principali metodi. Il primo è di generare casualmente tutte le coordinate degli individui assicurandosi che corrispondono all'intervallo stabilito. L'altro è di applicare gli operatori di mutazione a un padre conosciuto e di reiterare questo processo il numero di volte sufficiente a raggiungere il numero di padri voluto. Se lo scopo è di mantenere un certo senso fisico, il secondo metodo sembra più adeguato mentre se non si conosce un padre, il primo metodo permette di superare questo problema.

3. Model Updating di un *test_rig*

In questo capitolo studieremo un caso di updating di un modello a elementi finiti di un *test_rig* rotore. Useremo per il processo di updating un set di misure di vibrazioni sperimentali e un procedimento di tipo iterativo ed euristico che descriveremo successivamente. Gli obiettivi che si propone questo studio sono la valutazione dell'efficienza del processo e l'analisi di sensibilità dei diversi parametri sulla risposta del sistema. Abbiamo usato MatLab come programma per il calcolo della risposta e per l'applicazione dell'algoritmo di ottimizzazione.

3.1 Modello agli elementi finiti e parametrizzazione

Il rotore è stato modellizzato come una serie di elementi trava asezione circolare con determinate proprietà fisiche. E' costituito da quarantasette elementi, descritti ciascuno da undici parametri in quest'ordine:

- Numero dell'elemento;
- Diametro esterno di rigidezza [m];
- Diametro interno di rigidezza [m];
- Diametro esterno di massa [m];
- Diametro interno di massa [m];
- Lunghezza [m];
- Presenza di un cuscinetto o di un giunto (0 se non c'è il cuscinetto, 1 se c'è, 2 se è un giunto.);
- Modulo di Young del materiale costitutivo [Pa];
- Coefficiente di Poisson;
- Fattore di correzione per gli effetti di forma;
- Massa volumica [kg/ m^3].

Tutti questi dati sono contenuti in un file chiamato *test_rig.rot*. Questo file permette di generare un modello matematico del rotore che tiene conto in maniera opportuna delle carateristiche di massa e di rigidezza. Si vedono in Figura 1 queste due rappresentazioni. In questo disegno sono anche segnalate le posizioni dei cuscinetti, che sono gli elementi 4, 17, 25 e 44 del modello agli elementi finiti descritto.

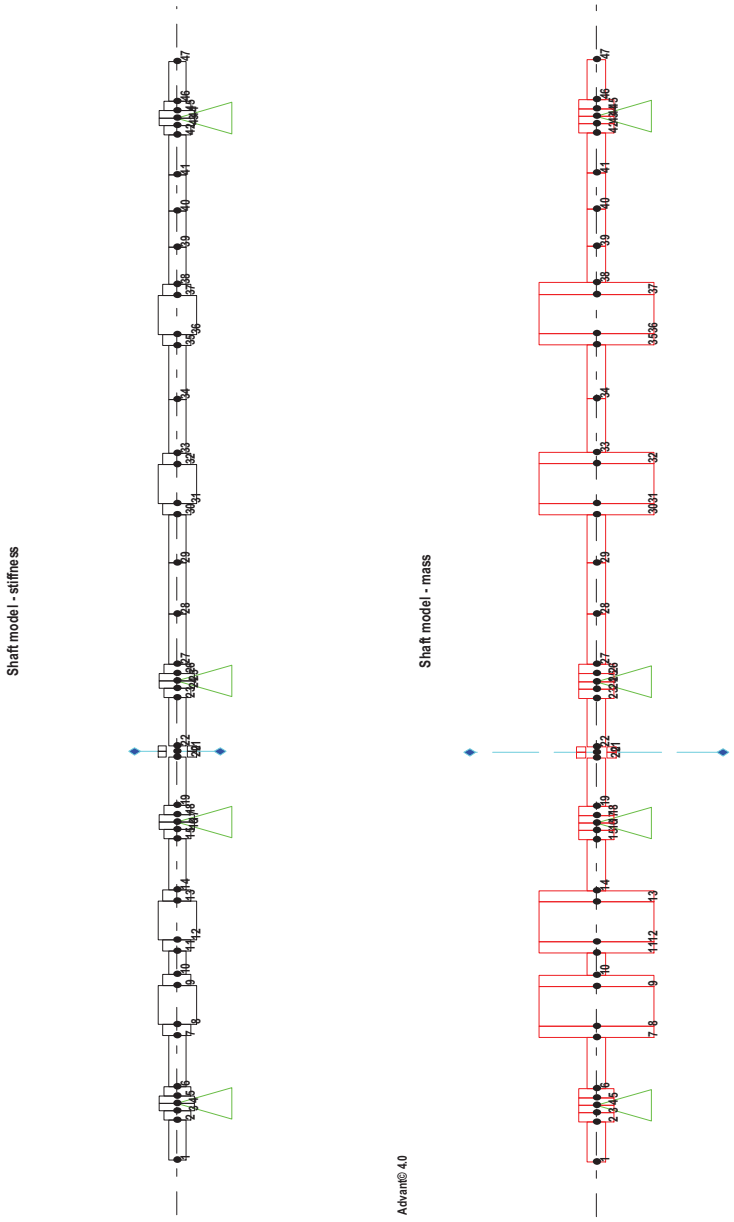


Figura 3.1 - Rappresentazione geometrica e di massa del rotore

Conosciamo le vibrazioni sperimentali in corrispondenza dei punti dove ci sono i cuscinetti e secondo i gradi di libertà **x** e **y** indicati in Figura 3.2:

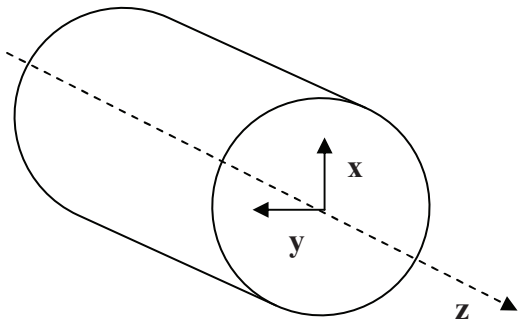


Figura 3.2 - Gradi di libertà

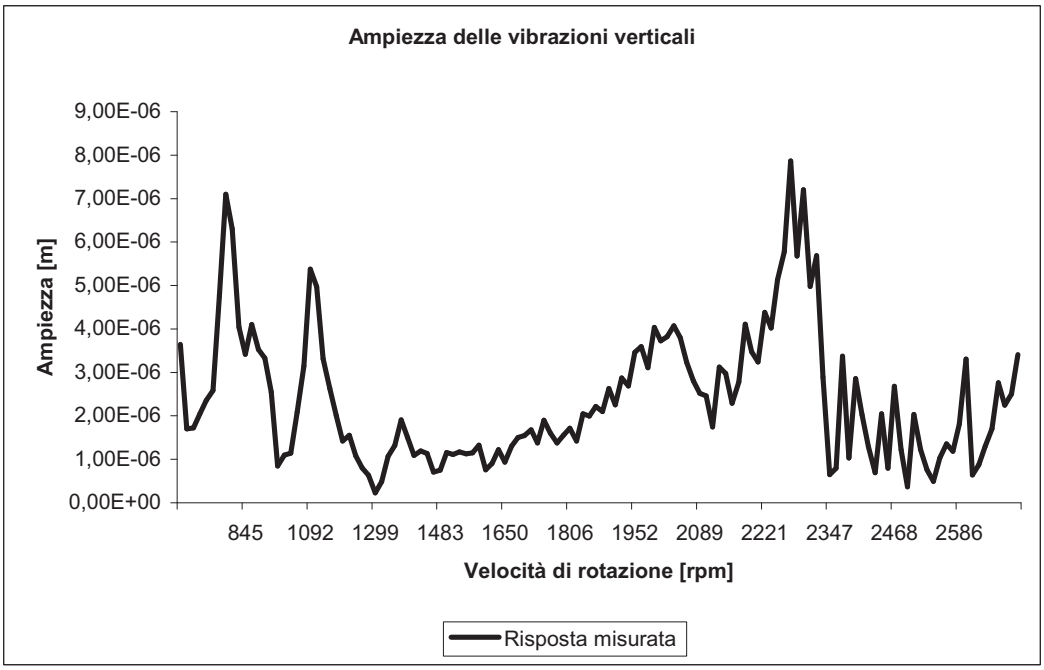
Avremo quindi un vettore, dove ogni coordinata corrisponde ad una velocità misurata (130 velocità diverse) e una matrice di dimensioni (8×130) dove per ogni velocità misurata è stata registrata la vibrazione in direzione **x** e **y** per ogni cuscinetto. Ogni elemento di questa matrice è riportato sotto la sua forma complessa come

$$A(\omega)e^{i\omega t+\varphi(\omega)}$$

dove $A(\omega)$ rappresenta l'ampiezza della vibrazione e $\varphi(\omega)$ la sua fase. Per esempio abbiamo rappresentato l'ampiezza e la fase delle vibrazione verticale misurate al livello del primo cuscinetto (Figura 3.3).

I cuscinetti saranno quindi modellati con le loro matrici di smorzamento e di rigidezza rispetto a questi due gradi di libertà. Solo due matrici sono sufficienti per la descrizione dei cuscinetti e sono $\mathbf{K}_i(\omega)$ e $\mathbf{C}_i(\omega)$ con $i\in\{1,2,3,4\}$ per il cuscinetto *i-esimo*. Tutti coefficienti dipendono dalla velocità di rotazione.

$$\mathbf{K}_i(\omega)=\begin{bmatrix}\mathbf{k}^i_{xx}(\omega) & \mathbf{k}^i_{xy}(\omega)\\ \mathbf{k}^i_{yx}(\omega) & \mathbf{k}^i_{yy}(\omega)\end{bmatrix} \quad \text{e} \qquad \mathbf{C}_i(\omega)=\begin{bmatrix}\mathbf{c}^i_{xx}(\omega) & \mathbf{c}^i_{xy}(\omega)\\ \mathbf{c}^i_{yx}(\omega) & \mathbf{c}^i_{yy}(\omega)\end{bmatrix}$$



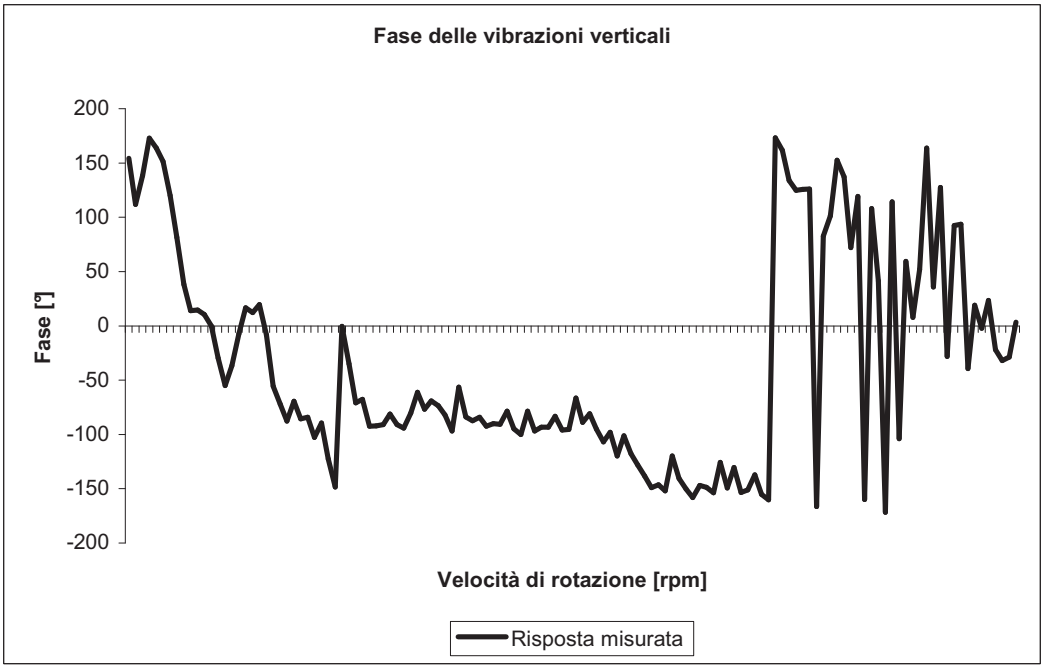


Figura 3.3 – Ampiezza e fase delle vibrazioni verticali del Nodo 4

Possiamo rappresentare graficamente questa matrice con delle curve in funzione della velocità. Ad esempio, per il primo cuscinetto, la rigidezza può essere rappresentata come in Figura 3.4 nell'intervallo di frequenze reso possibile dalle apparecchiature di misura.

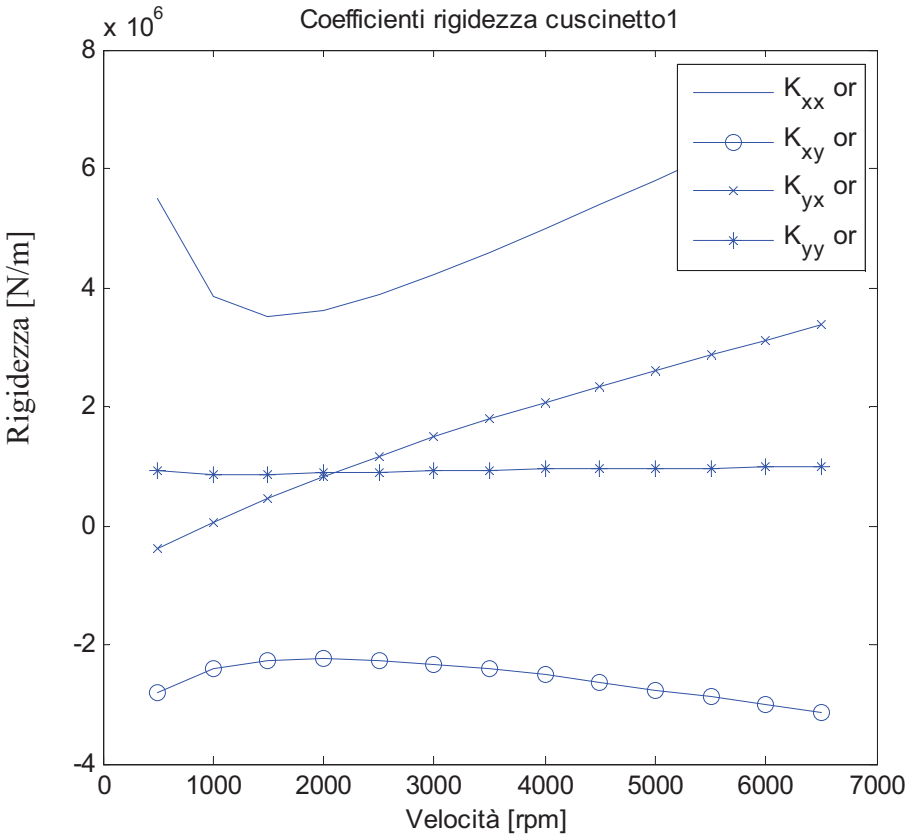


Figura 3.4 – Coefficiente di rigidezza del cuscinetto 1 in funzione della velocità.

Il vettore $\mathbf{a}$ dei parametri sarà quindi costruito con un metodo analogo a quello descritto nel Capitolo I sulla parametrizzazione secondo il modello agli elementi finiti. Questo vettore avrà quindi 32 coordinate, 8 per cuscinetto, messi una dietro l'altra dove ci sono, nell'ordine, i coefficienti legati a $\mathbf{k}_{xx}$, $\mathbf{k}_{xy}$, $\mathbf{k}_{xy}$, $\mathbf{k}_{yy}$, $\mathbf{c}_{xx}$, $\mathbf{c}_{xy}$, $\mathbf{c}_{yx}$, $\mathbf{c}_{yy}$. Ognuno di questi parametri moltiplicherà il coefficiente corrispondente nel modello aggiornato.

3.2 Metodo di ottimizzazione

Abbiamo visto nel capitolo 2 come un metodo iterativo euristico permetta di evitare il problema della convergenza nei minimi locali. Per procedere all'aggiornamento dei parametri del modello è stato scelto un algoritmo di strategia evolutiva del tipo $(\mu,\kappa,\lambda,\rho)-ES$ con i seguenti parametri:

- μ e λ scelti dall'utente;
- ρ=1, cioè ogni figlio avrà solo un padre;
- $\kappa=\infty$, questo significa che un individuo può vivere eternamente se è tra i μ migliori a ogni iterazione e che i padri competono con i figli.

Un altro parametro che sarà lasciato alla scelta dell'utente sarà la varianza sigma della distribuzione Normale centrata in zero (Figura 3.5) che permette di generare il vettore di perturbazione tra il parametro padre e il parametro figlio in questo modo:

$$a_{figlio} = a_{padre} + \sigma \begin{bmatrix} N(0,1) \\ \vdots \\ N(0,1) \end{bmatrix}$$

All'istante iniziale il vettore dei parametri sarà inizializzato a $\mathbf{a}^0 = [1,\ldots,1]^T$.

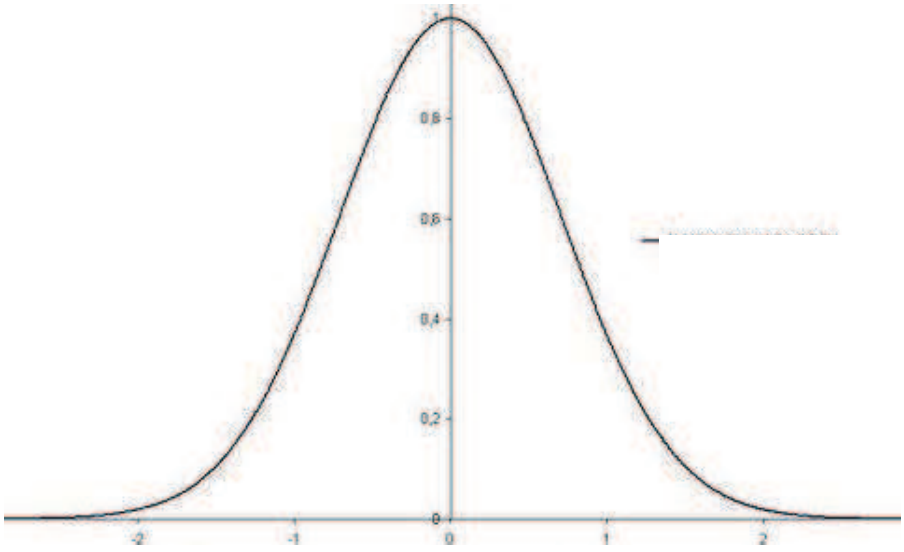


Figura 3.5 Legge Normale centrata ridotta $N(0,1)$

La funzione obiettivo usata in questo caso sarà la somma della differenza dei quadrati tra le vibrazioni generate dal modello aggiornato e le vibrazioni misurate. Se indichiamo con $\hat{\mathbf{x}}$ il vettore delle misure dei gradi di libertà e con $\mathbf{x(a_i)}$ il vettore dei gradi di libertà generato da un set di parametri $\mathbf{a}$ avremo l'errore relativo seguente:

$$E(\mathbf{a_i}) = \frac{\mathbf{(x(a) - \hat{x})^T (x(a) - \hat{x})}}{\mathbf{\hat{x}^T \hat{x}}}$$

Questa funzione ci permette di classificare gli individui generati e di selezionare i migliori, che saranno quelli che generano i valori più bassi di E.

Questo programma permette anche di scegliere un intervallo, che dovrà contenere il parametro aggiornato e di selezionare i parametri da aggiornare. E' stato quindi creato un file dove per ogni parametro a_i di $\mathbf{a}$ si scelgono b_i, c_i e d_i tali che:

$$\begin{cases} \forall k \in \mathbb{N}, \quad a_i^k \in [b_i, c_i] \\ \text{per } d_i = 0, \text{ il parametro non sarà modificato} \\ \text{per } d_i = 1, \text{ il parametro sarà modificato} \end{cases}$$

dove k indica il numero dell'iterazione.

E' stato implementato un algoritmo che segue le regole descritte nei capitoli precedenti con lo scopo di procedere all'aggiornamento dei parametri dei 4 cuscinetti con una parametrizzazione regolabile dall'utilizzatore. Nella parte successiva vedremo come influiscono questi parametri e quindi come massimizzare l'efficienza dell'algoritmo.

3.3 Analisi di sensibilità dell'algoritmo

In questa sezione cercheremo di capire empiricamente l'influenza dei parametri dell'algoritmo sulle sue caratteristiche in modo da stabilire quale sarà la parametrizzazione da scegliere per lo studio dell'updating.

3.3.1 Numero di figli

Il numero di figli λ corrisponde al numero d'individui generati ad ogni iterazione. Questi figli saranno in competizione con i loro padri. Intuitivamente possiamo già prevedere che più figli abbiamo, più la probabilità di trovare delle soluzioni migliori aumenta. Abbiamo quindi lanciato l'algoritmo con il set di cuscinetti iniziali facendo variare solo il numero di figli a tenendo fissati il numero di padri e la varianza della legge normale. I parametri scelti sono i seguenti:

- $\lambda = 1$, scegliamo un padre per capire meglio il fenomeno riguardante i figli;
- $\mu = 1, 2, 3, 4, 5$;
- $\sigma = 0,08$ valore consigliato nella letteratura;
- 100 iterazioni in modo da potere vedere abbastanza bene l'evoluzione.

Le curve in Figura 3.6 sono state ottenute facendo girare dieci volte l'algoritmo e mediando tutti i valori ottenuti. Questo ci permette di ridurre l'effetto dovuto alla varianza e di comparare in modo più efficiente i risultati. Possiamo quindi osservare che, come ci si poteva aspettare, aumenta la velocità di convergenza (in numero d'iterazioni) all'aumentare del numero di figli.

Vogliamo anche verificare che la convergenza non venga influenzata dal numero di figli perché, secondo lo stato dell'arte, la convergenza globale dovrebbe essere assicurata con un numero infinito d'iterazioni. Per verificare questo punto abbiamo lanciato il programma per effettuare mille iterazioni sempre a parità delle altre condizioni per un figlio o per 3 figli. Osserviamo in Figura 3.7 che, anche se nel secondo caso la convergenza è più rapida, il valore raggiunto non viene cambiato.

3.3.2 Numero di padri

Il numero di padri μ corrisponde al numero d'individui selezionati ad ogni iterazione, costituisce quindi un parametro che indica la forza di selezione dell'algoritmo. Una selezione difficile si ottiene con un numero di padri piccolo

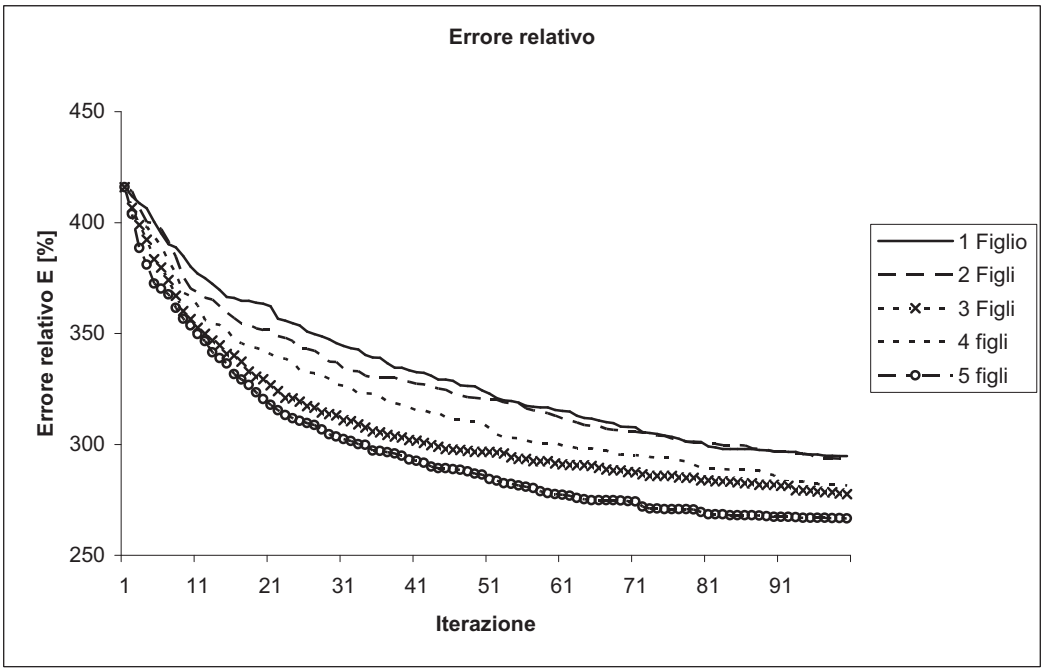


Figura 3.6 – Errore relativo per variazione del numero di Figli

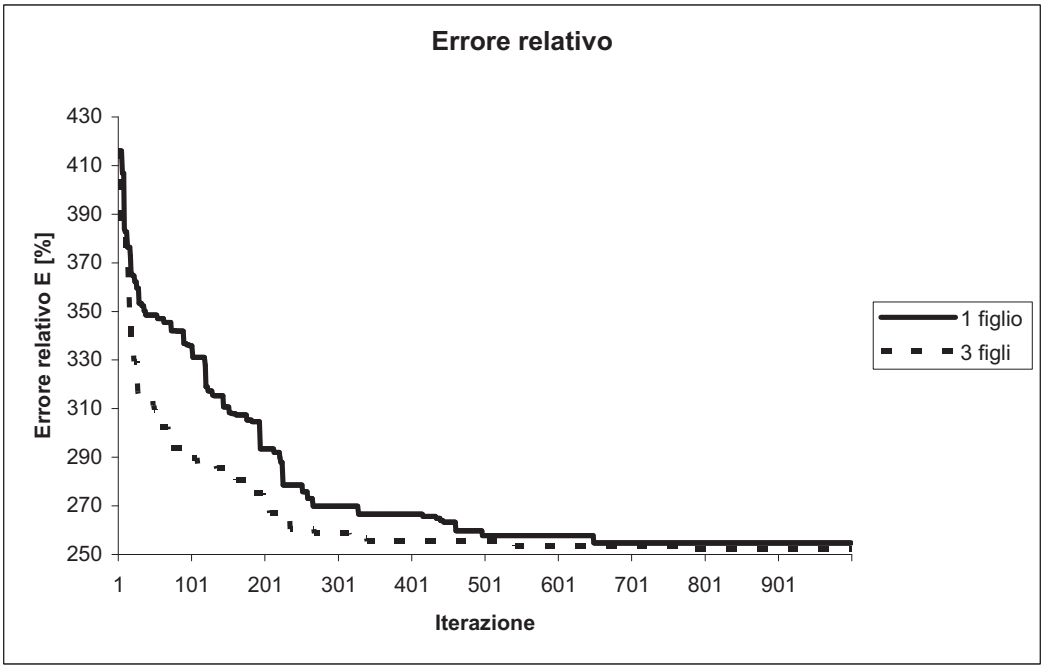


Figura 3.7 – Convergenza per diversi numeri di Figli

rispetto al numero di figli, invece se il numero di figli è piccolo rispetto a μ, un padre genererà pochi figli ma avrà una buona probabilità di avere un rappresentante nella generazione successiva. Abbiamo quindi imposto il numero di figli e fatto variare il numero di padri rispettando le condizioni seguenti:

- $\lambda = 8$;
- $\mu = 1,2,4,8$ in modo da generare lo stesso numero di figli da ogni padre;
- $\sigma = 0,08$ valore consigliato nella letteratura;
- 25 iterazioni dell'algoritmo permettono di osservare la manifestazione dei fenomeni senza tempi di generazione troppo lunghi.

Le curve rappresentate in Figura 3.8 sono, come nel caso precedente, ottenute mediando dieci ripetizioni in modo da togliere gli effetti dovuti alla varianza del fenomeno.

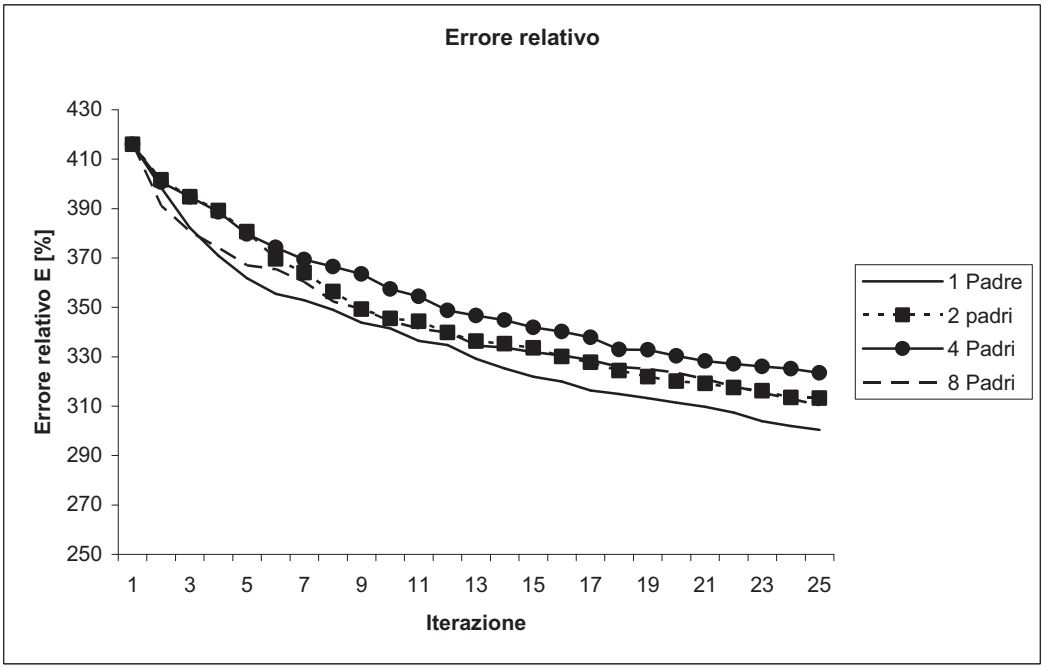


Figura 3.8 - Errore relativo per diversi rapporti padri/figli

Osserviamo quindi che l'algoritmo è più efficiente se la selezione è difficile, cioè se ci sono pochi eletti per passare da una generazione all'altra. Sceglieremo quindi nelle prossime operazioni un piccolo numero di padri rispetto al numero di figli in modo da sfruttare l'effetto positivo della selezione.

3.3.3 Varianza

La varianza σ definita in quest'algoritmo è la forza di mutazione e rappresenta la distanza media alla quale si sposta ciascuna coordinata ad ogni generazione. Aumentare la varianza permette di esplorare più ampiamente lo spazio delle soluzioni, però comporta spesso il rischio di non trovare una soluzione migliore quando siamo vicino all'ottimo (Figura 3.9). Al contrario, una forza di mutazione troppo bassa riduce le proprietà euristiche dell'algoritmo e lo rende più deterministico. L'evoluzione sarà quindi più lenta e con un tasso di successo nelle mutazioni maggiore (Figura 3.10). Ciò comporta tuttavia il rischio di rimanere rinchiusi nei minimi locali. Dovrebbe allora esistere un valore ottimo per il parametro sigma che permetta di generare figli in un raggio sufficientemente grande per esplorare lo spazio in modo efficiente senza perdere le caratteristiche interessanti degli antenati.

In quest'analisi abbiamo lanciato l'algoritmo con diversi valori della varianza a parità degli altri parametri. Abbiamo identificato un ottimo intorno a $\sigma = 0,085$ e rappresentato le curve di evoluzione dell'errore relativo separando i valori inferiori a 0,0085 e quelli superiori per identificare correttamente i diversi fenomeni. Abbiamo realizzato questo studio sul modello di base (1,1)-ES. Le curve (Figura 3.11 e Figura 3.12) sono mediate come nelle analisi precedenti.

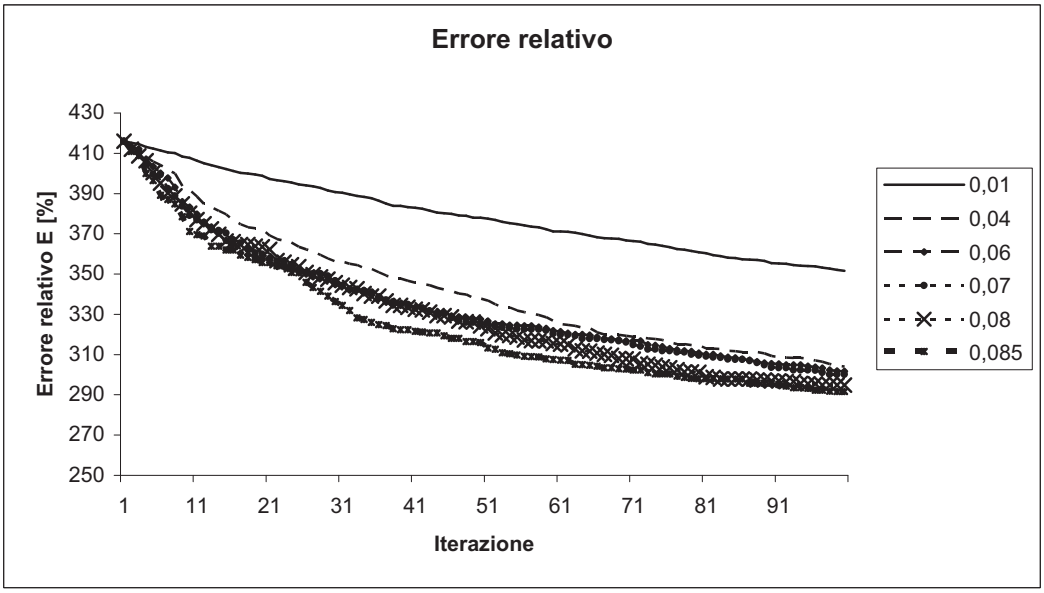


Figura 3.9 – Errore relativo per diversi valori di sigma inferiori a 0,085

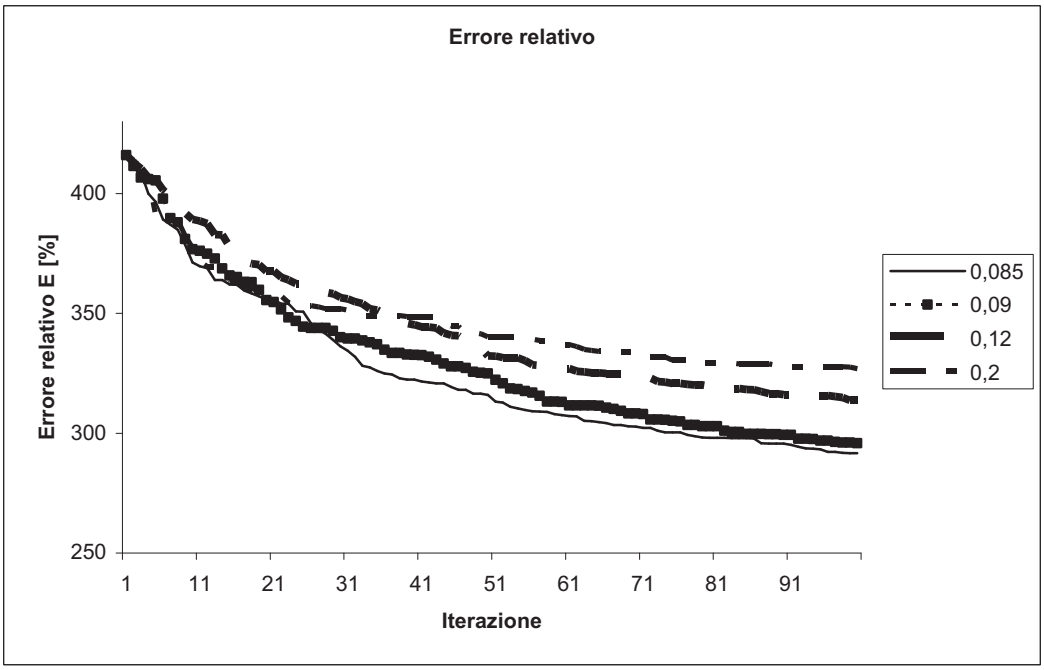


Figura 3.10 – Errore relativo per diversi valori di sigma superiori a 0,085

Sceglieremo quindi un valore di sigma uguale a 0.085 negli impieghi successivi del programma perché permette una buona ricerca nello spazio delle soluzioni pur mantenendo una buona correlazione tra figli e padri.

Possiamo vedere nelle seguenti curve i fenomeni descritti prima su delle curve non mediate. Nella prima, per una forza di mutazione inferiore all'ottimo, abbiamo una convergenza lenta con successo alto nelle mutazioni (Figura 3.11). La curva varia in modo quasi lineare verso il minimo.

Nel secondo caso (Figura 3.12) vediamo che i successi nelle mutazioni sono pochi ma permettono dei salti alti nella variazione della funzione obiettivo.

3.3.4 Conclusioni sull'analisi di sensibilità dell'algoritmo

In conclusione di questo studio preliminare possiamo dire che, per massimizzare l'efficienza dell'algoritmo, è importante generare tanti figli rispetto al numero di padri per aumentare la difficoltà della selezione. E importante anche usare il valore di sigma che realizza un buon compromesso tra ricerca spaziale e correlazione tra figli e padri.

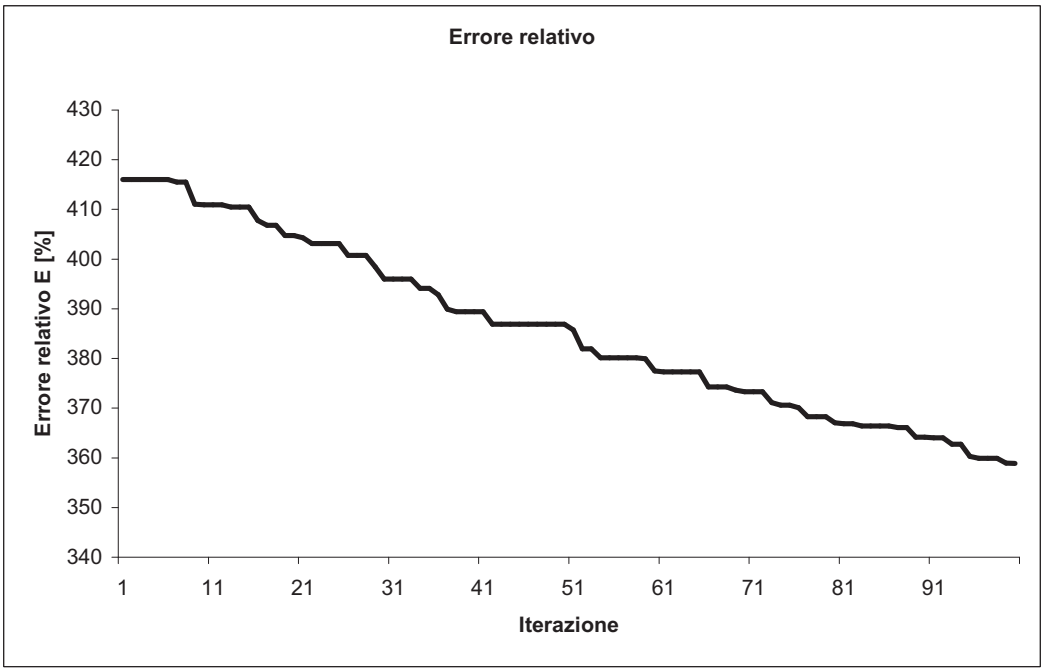


Figura 3.11 – Errore relativo per $\sigma = 0.01$

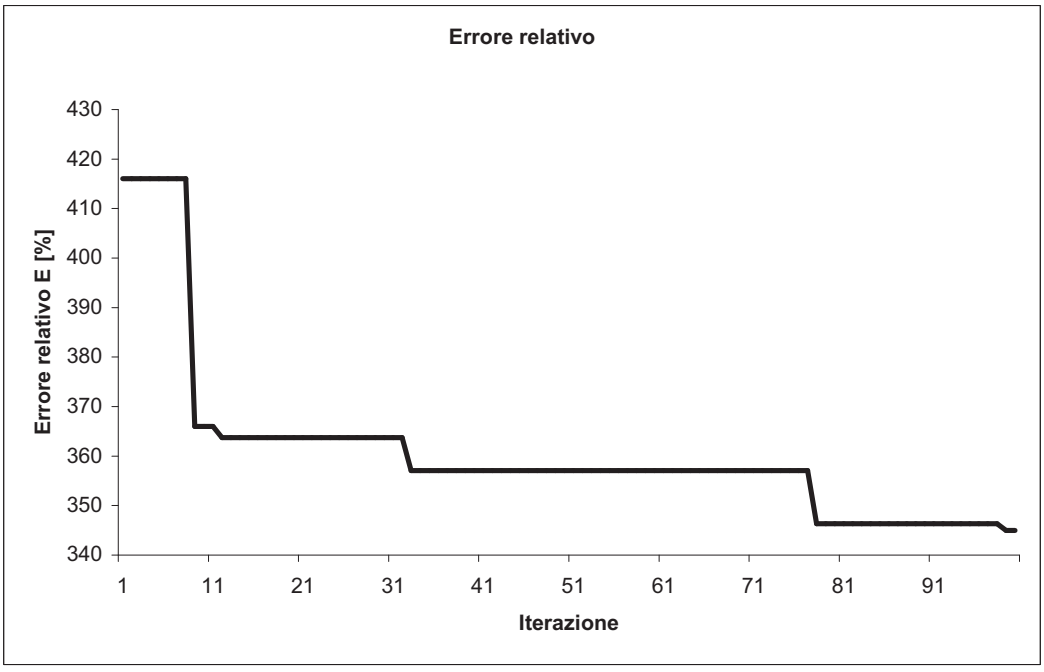


Figura 3.12 – Errore relativo per $\sigma = 0.2$

3.4 Analisi di sensibilità del modello

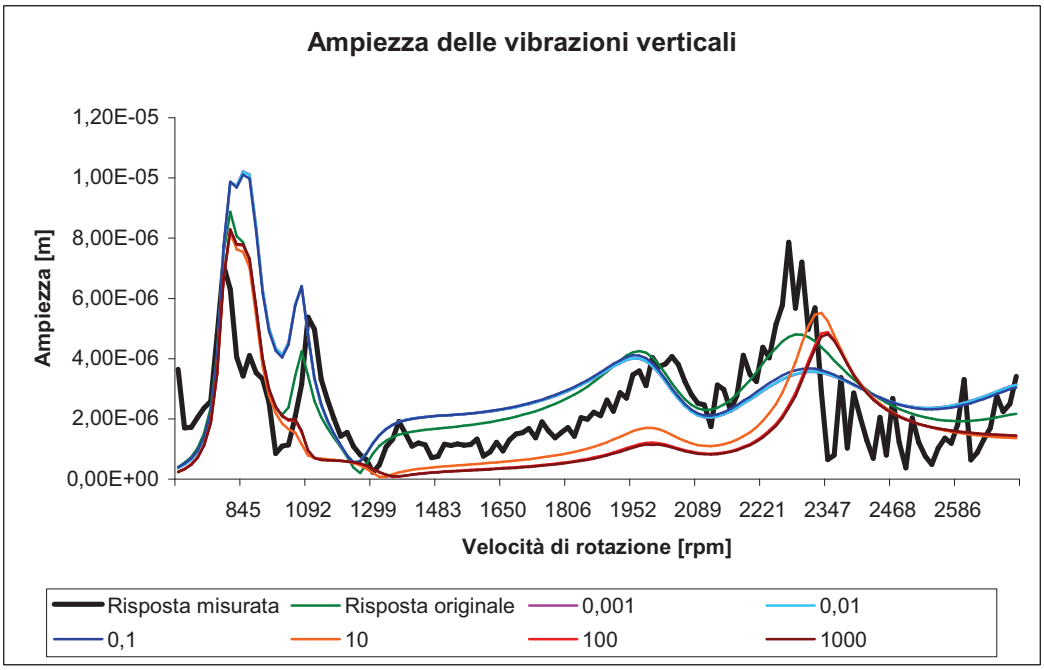
Abbiamo detto più in alto che ogni parametro aggiornato dovrà essere contenuto in un certo intervallo. Per definire questo intervallo occorre capire come variano le risposte del modello in funzione della variazione di ogni parametro. Analizzeremo quindi in questo paragrafo come variano le risposte date dal modello analitico aggiornato se facciamo variare solo un parametro del modello, moltiplicandolo per un coefficiente positivo, in modo da mantenere il senso fisico del modello.

Tratteremo il caso dell'ottimizzazione di tutti i parametri del primo cuscinetto dal primo all'ultimo e analizzeremo caso per caso l'influenza dei coefficienti sulla fase e sull'ampiezza delle risposte ottenute. I risultati potranno difficilmente essere interpretati tutti insieme perché la modifica di un parametro influisce su tutti i gradi di libertà in modo diverso e quindi non si può sapere a priori come verranno sovrapposte le diverse soluzioni ottenute. Si possono però interpretare i risultati di ogni variazione di parametro perché il significato fisico viene mantenuto.

Vista la simmetria di rivoluzione del modello, ci possiamo limitare allo studio della meta dei parametri. Per esempio, l'influenza di $k_{xx}, k_{xy}, c_{xx}, c_{xy}$ sulla vibrazione secondo **x** è analoga a quella di $k_{yy}, k_{yx}, c_{yy}, c_{yx}$ su **y** e l'influenza su **y** sarà analoga a quella su **x**. Per procedere a questo studio ogni coefficiente che definisce il modello dei cuscinetti è stato moltiplicato per una serie di valori (1/1000, 1/1000,1/10,1, 10, 100, 1000). Otterremo così una serie di sette curve sovrapposte per ogni parametro e per ogni grado di libertà.

3.4.1 Coefficiente di rigidezza k_{xx}

I coefficienti di rigidezza hanno un'influenza diretta sull'ampiezza di vibrazione, aumentare la rigidezza rispetto a un grado di libertà in un punto permette di diminuire l'ampiezza di vibrazione in questo punto rispetto a questo grado di libertà (Figura 3.13). L'influenza sull'altro grado di libertà è molto più debole come si può notare sul grafico corrispondente**.** Allontanandosi dal nodo dove è stato modificato il coefficiente, possiamo notare che l'influenza è quasi trascurabile, soprattutto per le basse velocità, e questo è dovuto all'inerzia del sistema e alla modellazione matematica (Figura 3.14). Per la fase possiamo notare che anche in questo caso l'influenza sull'altro grado di libertà è molto contenuta (Figura 3.15). Sul grado di libertà corrispondente invece vediamo uno spostamento laterale delle rispettive curve.



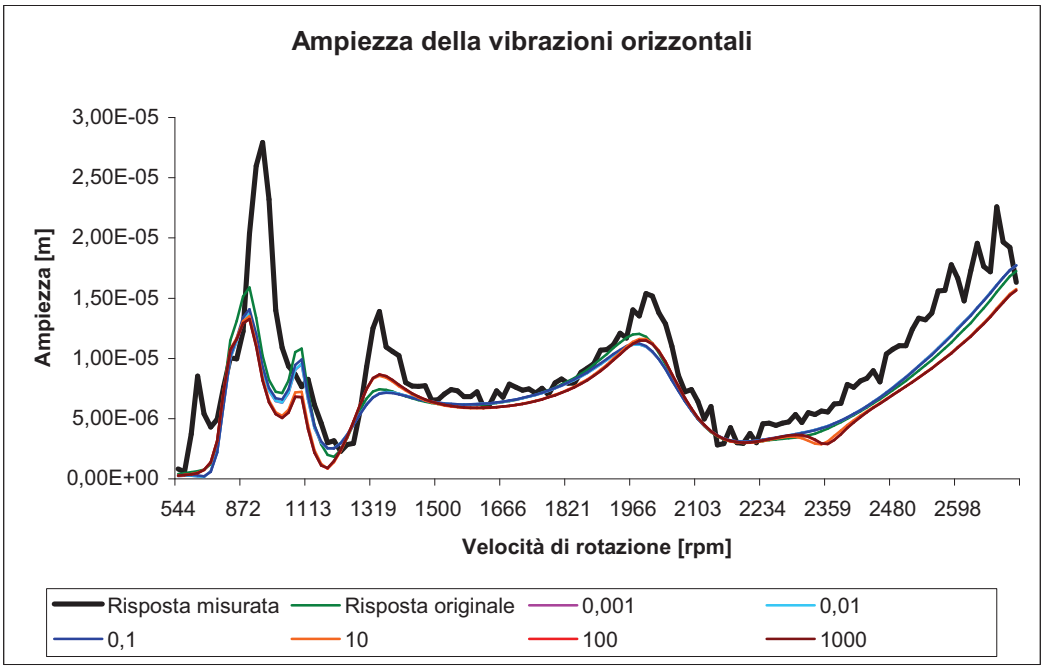
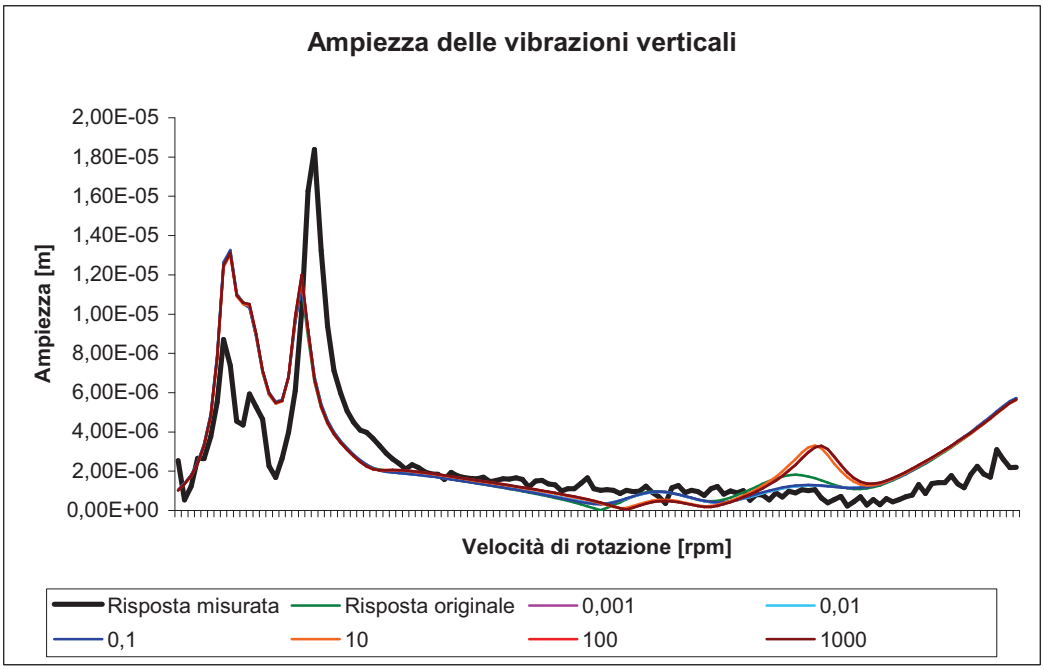


Figura 3.13 – Influenza sull'ampiezza delle vibrazioni verticali e orizzontali del Nodo 4 della variazione di k_{xx}



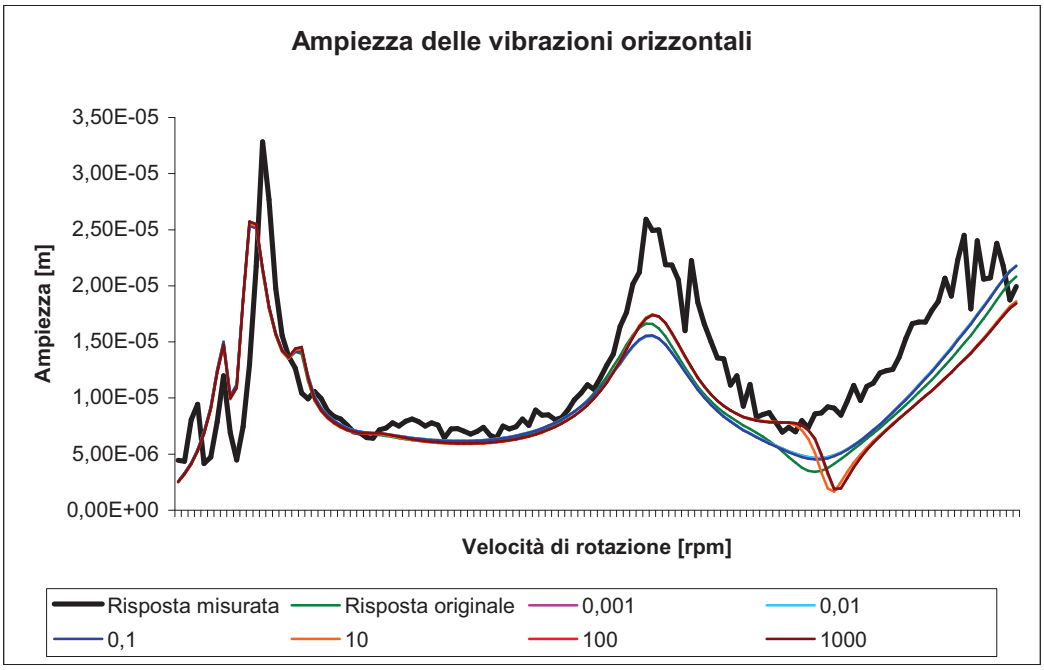
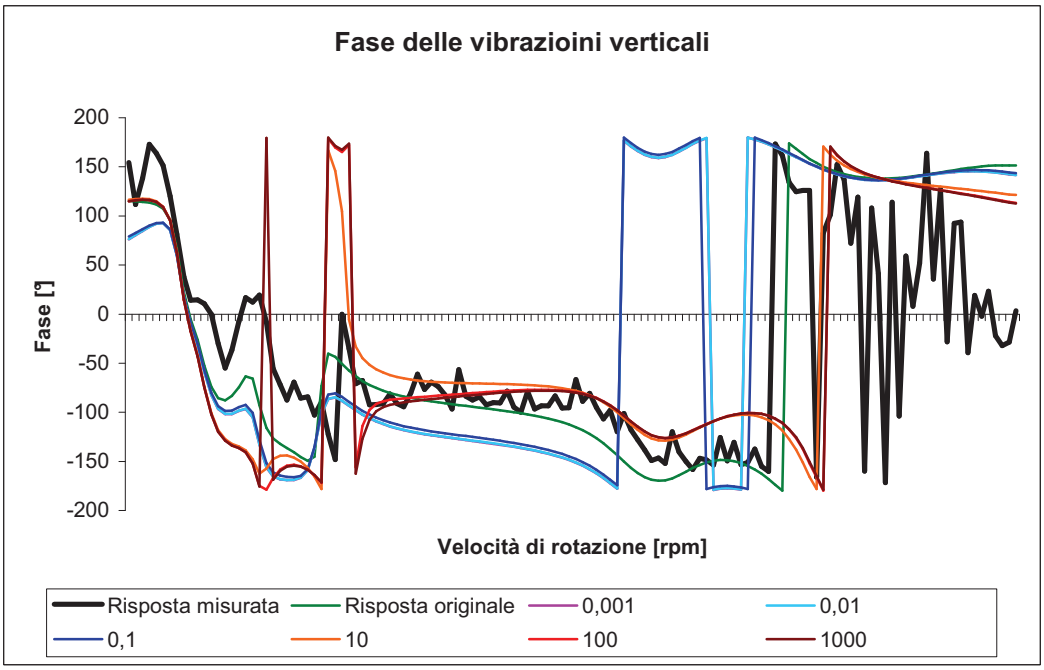


Figura 3.14 Influenza sull'ampiezza delle vibrazioni verticali e orizzontali del Nodo 44 della variazione di k_{xx}



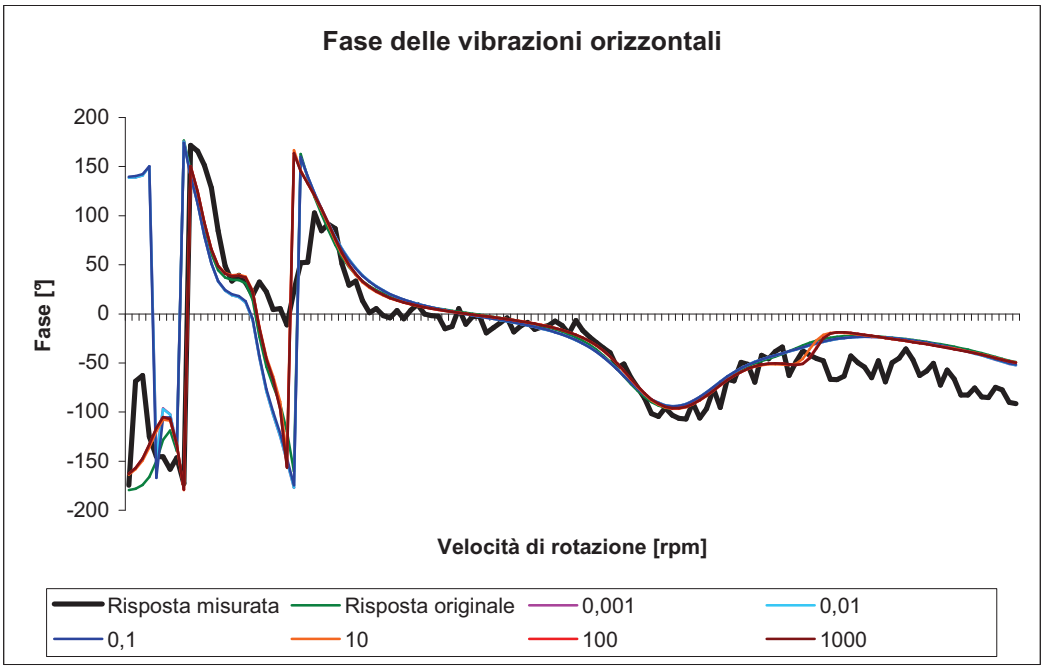
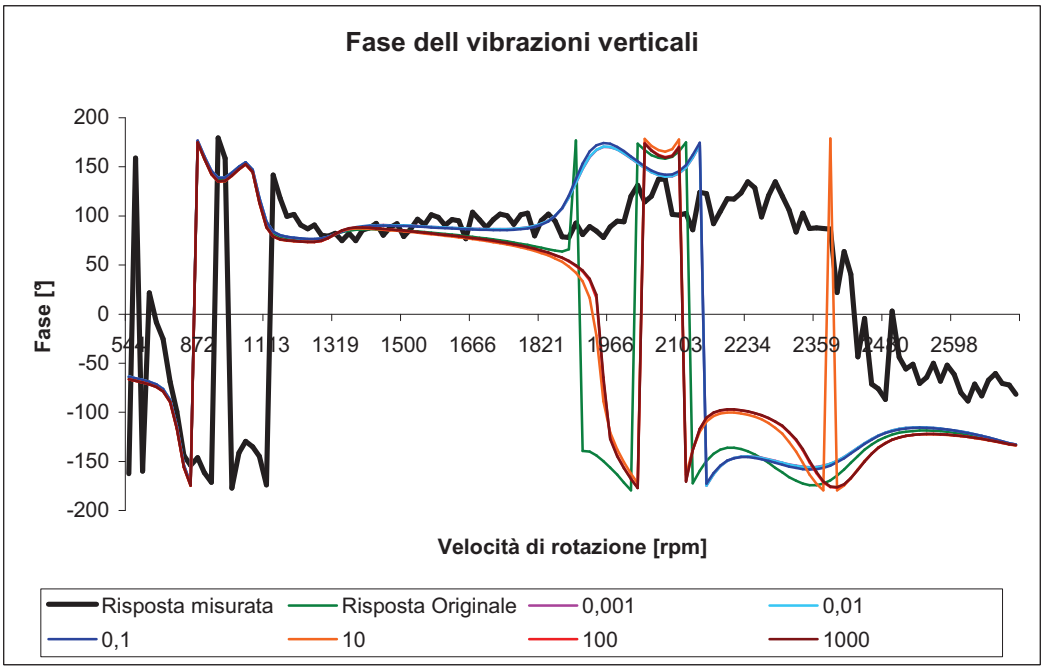


Figura 3.15 – Influenza sulla fase delle vibrazioni verticali e orizzontali del Nodo 4 della variazione di k_{xx}



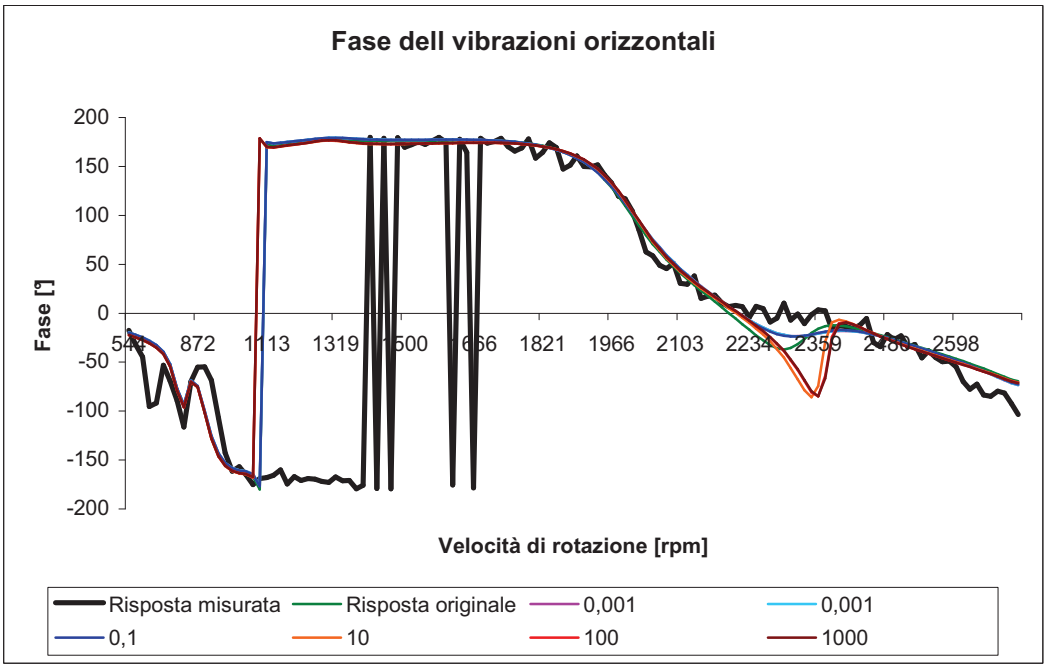
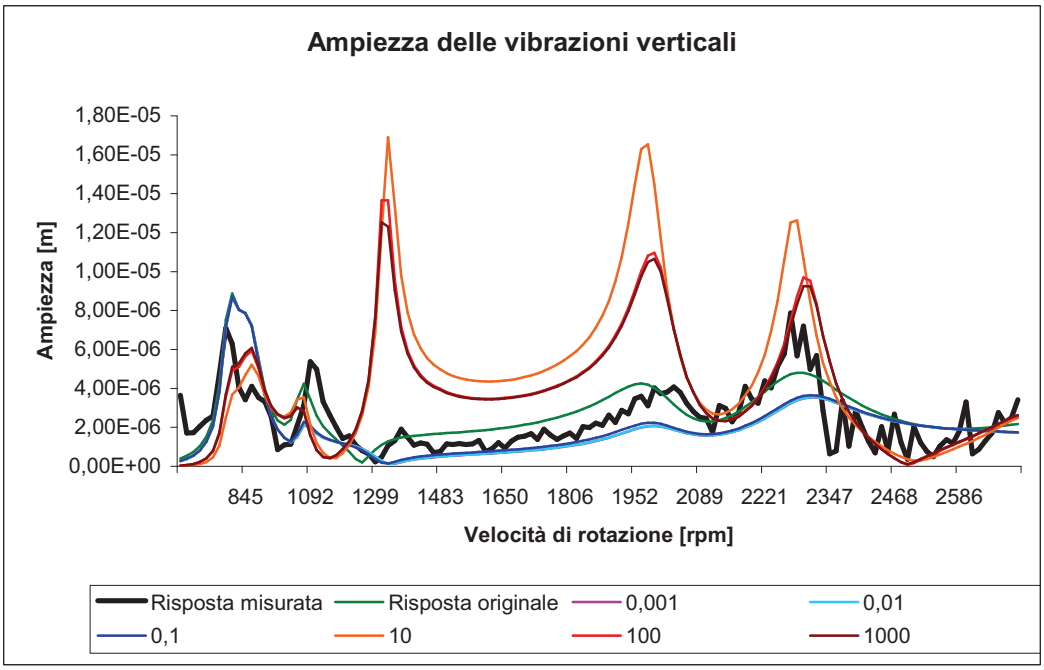


Figura 3.16 – Influenza sulla fase delle vibrazioni verticali e orizzontali del Nodo 44 della variazione di k_{xx}

Quelle ottenute moltiplicando il coefficiente per valori inferiori a uno si spostano a destra della curva iniziale invece le altre si spostano a sinistra. Se ci allontaniamo dal nodo dove è stata effettuata la modifica, vediamo un'influenza sostanziale (Figura 3.16) per la fase verticale però una variazione quasi nulla per la fase della componente orizzontale. La variazione della curva cambia a seconda che si aumenti o si diminuisca il valore di k_{xx}. Per un basso valore del coefficiente, la curva segue abbastanza bene la curva misurata, però per un valore superiore a quello originale del cuscinetto vediamo un'inversione del senso di variazione. Il coefficiente iniziale è quindi vicino a un punto critico per la fase su quel nodo. Si può notare anche un fenomeno di saturazione rispetto alla variazione del coefficiente. Per una moltiplicazione o divisione superiore a cento le curve non si spostano più. Questo fenomeno ci permette dunque di definire l'intervallo di variazione del parametro dell'updating legato a k_{xx} come uguale a $[0.1,100]$.

3.4.2 Coefficiente di rigidezza k_{xy}

Questo coefficiente definisce l'influenza di una variazione sul grado di libertà secondo **x** sul grado di libertà secondo **y**, Ci aspettiamo quindi delle variazioni importanti sui due assi, dovute a questo coefficiente, che crea un legame tra i due gradi di libertà. Se osserviamo in un primo tempo il modulo della vibrazione secondo i due assi (Figura 3.17) possiamo vedere delle grandi variazioni tra le risposte ottenute e in particolare sulle vibrazioni orizzontali. All'aumentare del coefficiente studiato vediamo un aumento dell'ampiezza di vibrazione verticale fino a un massimo raggiunto per un valore dieci volte superiore a quello originale. Se si oltrepassa questo valore, l'ampiezza torna a diminuire. Sembra che questo valore critico per la vibrazione verticale sia anche critico per l'ampiezza orizzontale se si tratta del fenomeno inverso. All'aumentare del coefficiente assistiamo ad una diminuzione dell'ampiezza di vibrazione fino ad un minimo che torna poi ad aumentare una volta superato il valore critico. Le curve ottenute per questo valore sono le peggiori per le risposte, i valori bassi del coefficiente permettono una buona approssimazione della risposta reale. Allontanandoci dal nodo 44, dove è stata eseguita la modifica, constatiamo, come nel caso precedente, una diminuzione dell'influenza del parametro, visto che le curve si ristringono per la vibrazione verticale invece sull'asse orizzontale, le variazioni rimangono più o meno uguali con sempre il valore critico intorno al valore definito prima (Figura 3.18). Per quanto riguarda la fase vediamo che le variazioni sono trascurabili sulla componente orizzontale e importanti sulla componente verticale (Figura 3.19).



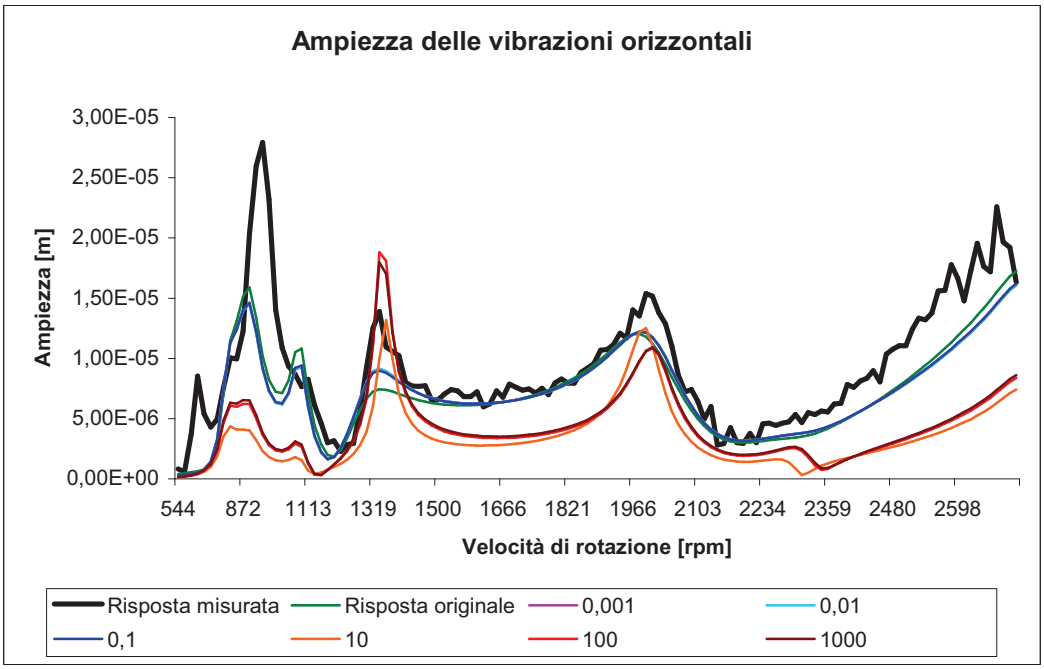
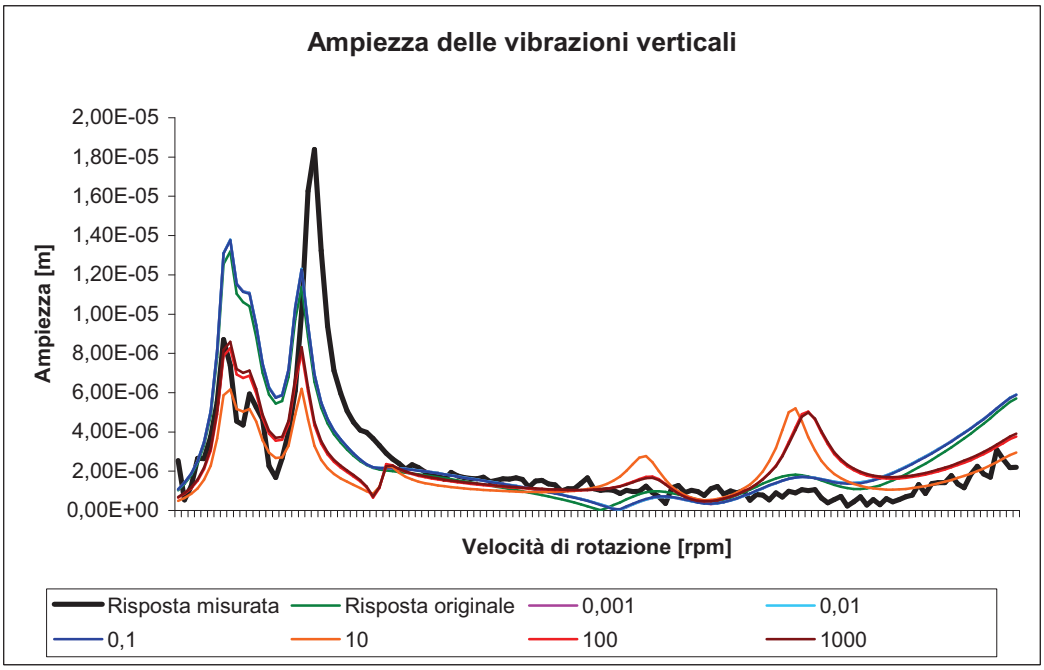


Figura 3.17 – Influenza sull'ampiezza delle vibrazioni verticali e orizzontali del Nodo 4 della variazione di k_{xy}



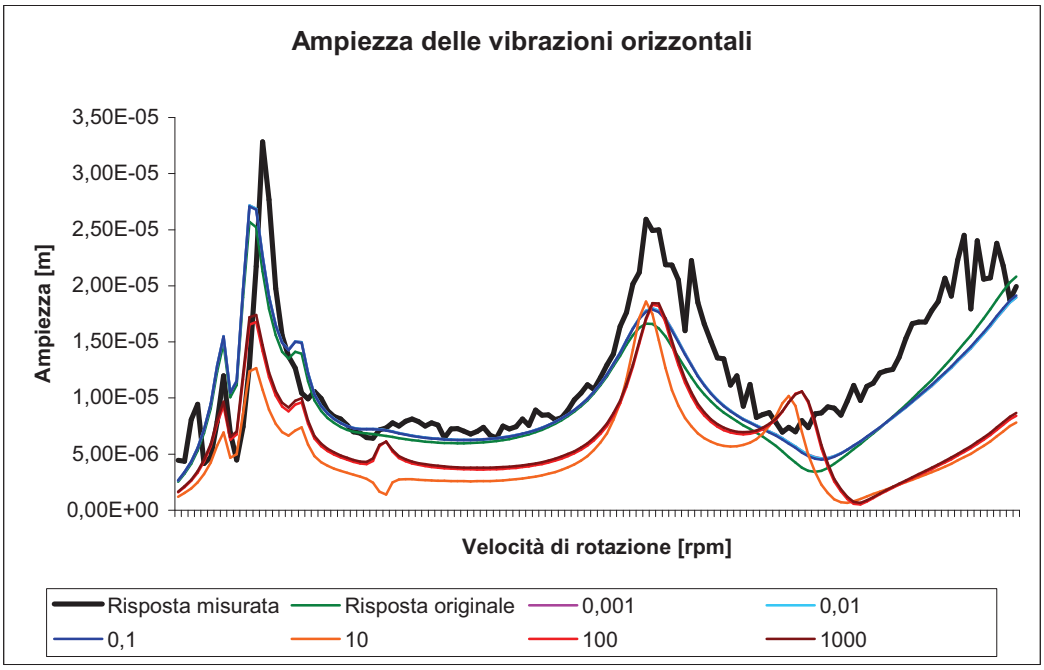
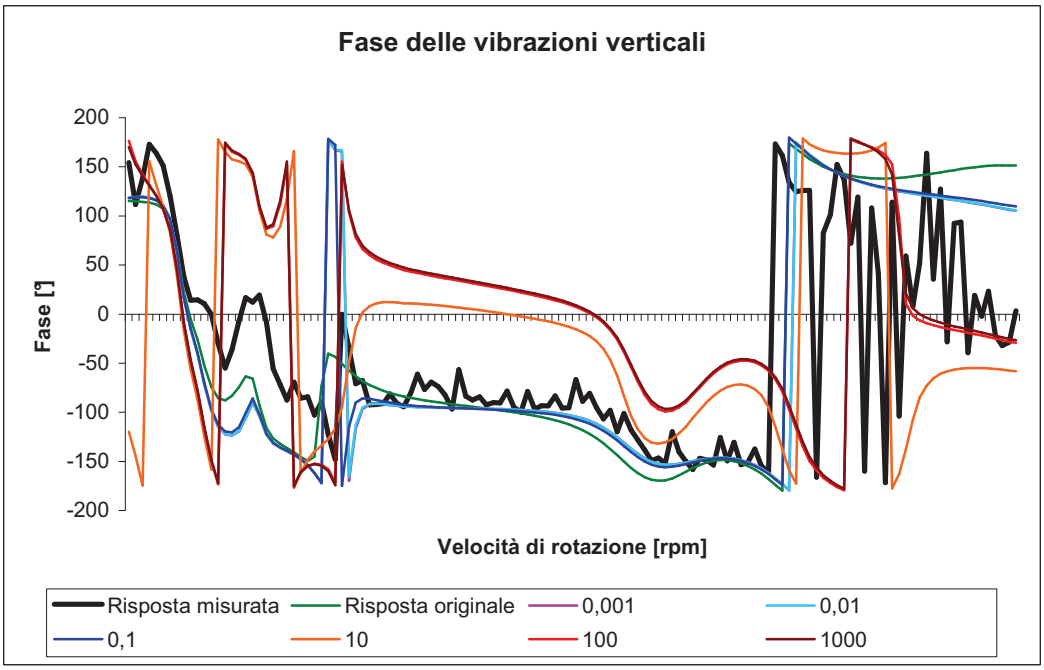


Figura 3.18 – Influenza sull'ampiezza delle vibrazioni verticali e orizzontali del Nodo 4 della variazione di k_{xy}



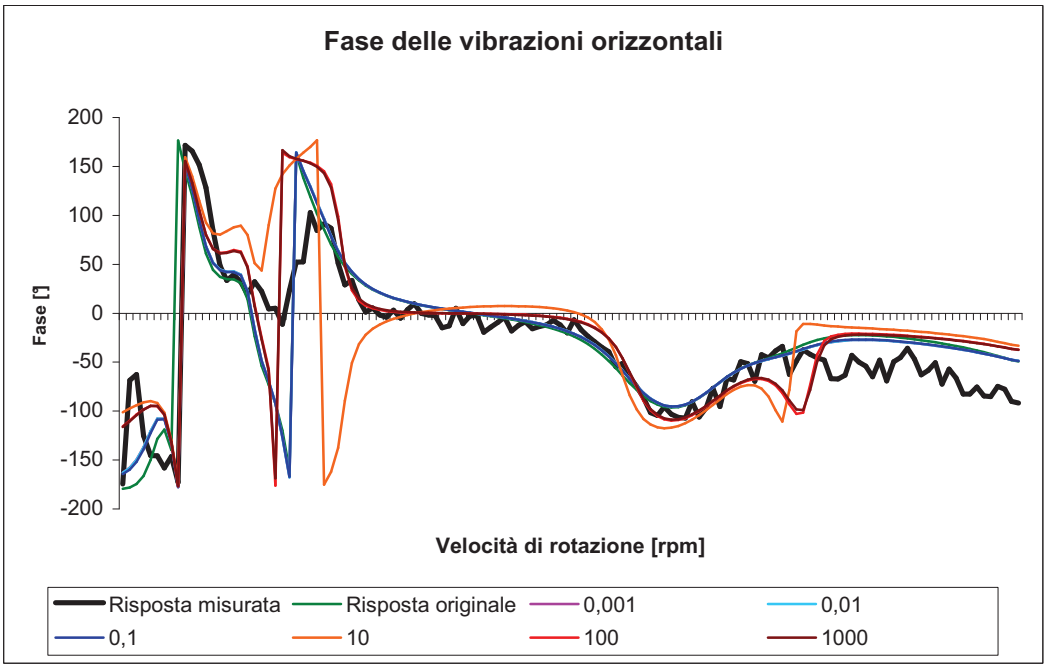
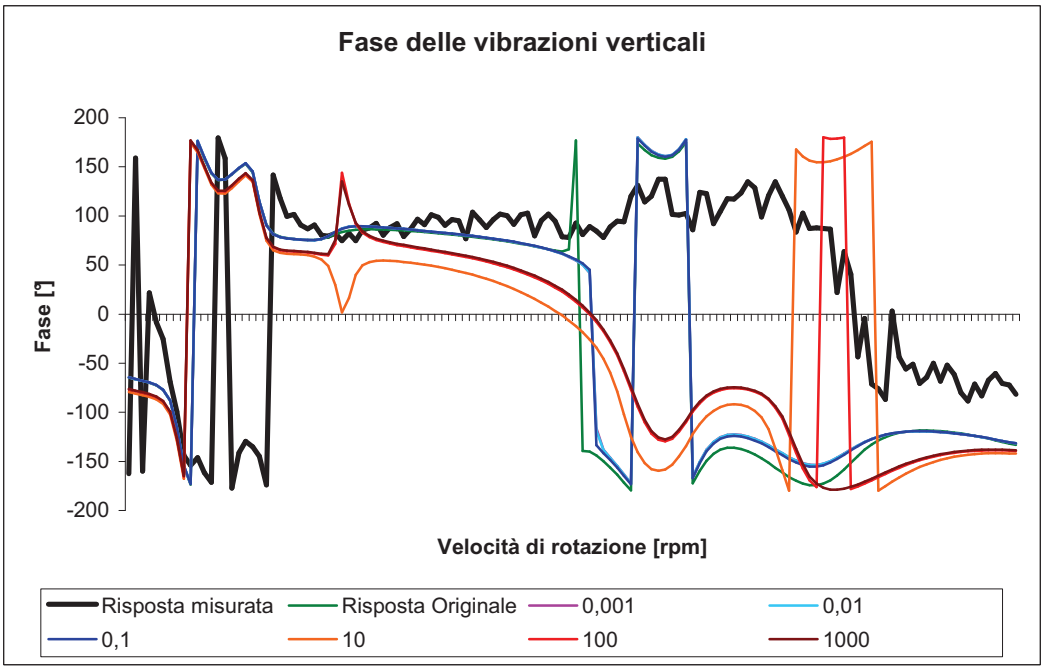


Figura 3.19 – Influenza sulla fase delle vibrazioni verticali e orizzontali del Nodo 4 della variazione di k_{xy}



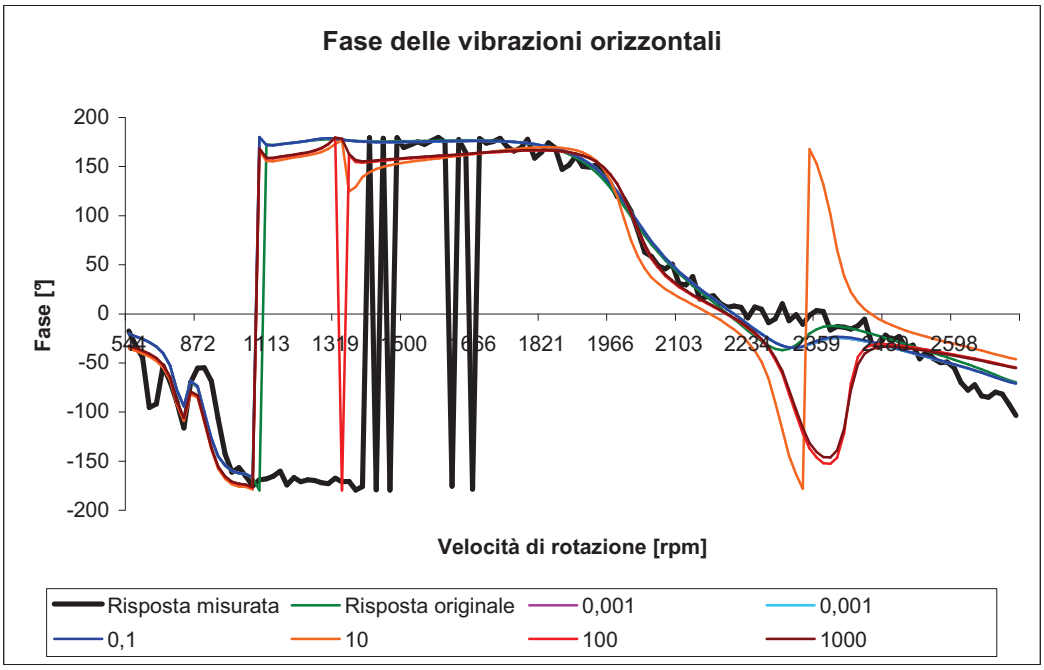
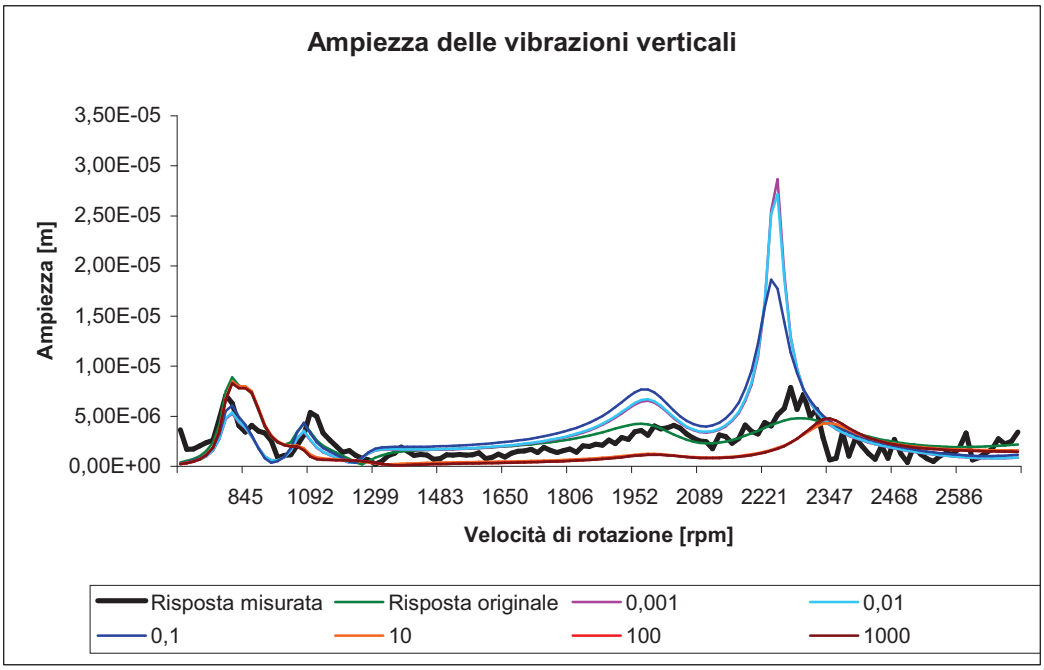


Figura 3.20 – Influenza sulla fase delle vibrazioni verticali e orizzontali del Nodo 44 della variazione di k_{xy}

All'aumentare del coefficiente k_{xy} vediamo uno spostamento verso l'alto della curva di fase e constatiamo anche che la curva è interpolata bene per piccoli valori del coefficiente. Man mano che ci si allontana nodo 4 possiamo notare come negli altri casi una diminuzione dell'influenza del parametro sulla risposta (Figura 3.20). Possiamo quindi concludere che questo parametro k_{xy} è un parametro molto sensibile del sistema e che la risposta ottenuta sembra più adeguata con un valore basso di k_{xy}.

3.4.3 Coefficiente di smorzamento c_{xx}

Questo coefficiente di smorzamento dovrebbe normalmente agire solo sulla componente verticale e, infatti, osserviamo che le curve che rappresentano l'ampiezza orizzontale sono molto strette (Figura 3.21). Si nota un'influenza molto debole del coefficiente per posizioni lungo l'asse della macchine lontane dalla posizione del cuscinetto modificato (Figura 3.22). L'influenza di questo coefficiente sembra abbastanza lineare, la diminuzione del coefficiente provoca un aumento dell'ampiezza di vibrazione visto che il sistema è meno smorzato. Si nota che compare una risonanza intorno a 2300 rpm che non è stata misurata quindi questo, è un grave problema per l'utente visto che il fine dell'updating è anche di potere localizzare le possibili risonanze. Per i valori bassi di questo parametro invece la modellazione è abbastanza corretta. Si vede in questa situazione il caso di un parametro che sembra poco sensibile visto che necessita di valori veramente alti del coefficiente per portare a una vera variazione della forma delle curve. Vedremo nella parte seguente come questa costatazione si traduce in termini di convergenza per questo parametro. La fase secondo la componente orizzontale subisce uno spostamento verticale invece l'atra componente non subisce nessuna variazione (Figura 3.23). Quando ci allontaniamo dal nodo 4 si riduce la dipendenza dal parametro ma si può constatare che incirca alla velocità di 2300 rpm l'introduzione di una risonanza si nota come l'abbiamo costata anche per il modulo secondo **x** nel nodo 4 (Figura 3.24). Per concludere, a proposito di questo parametro possiamo dire che esso influisce sopratutto sulle componenti secondo la direzione **x** e anche un valore troppo basso modifica in profondità la struttura del modello che si allontana così dalla realtà. Verso i 2300 rpm introduce una risonanza che non è stata rilevata dalle misure.



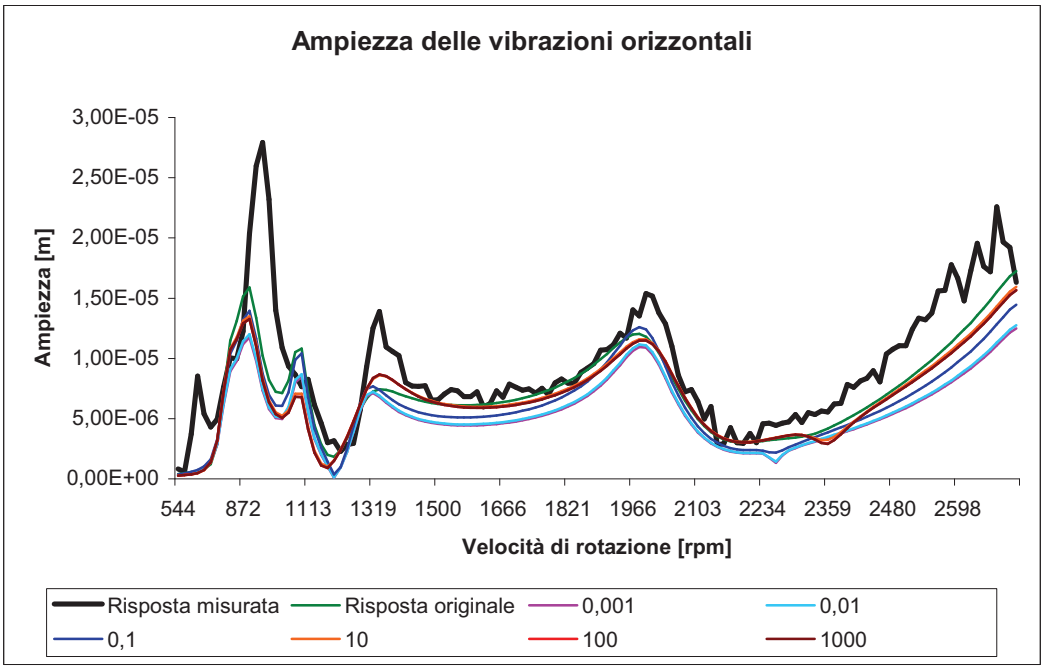
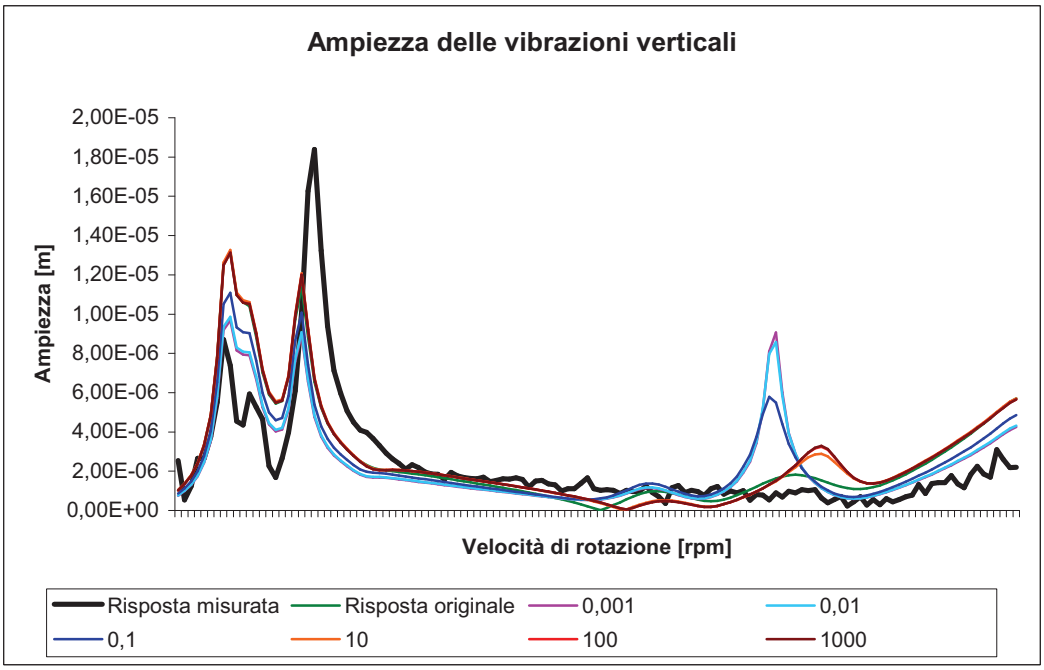


Figura 3.21 – Influenza sull'ampiezza delle vibrazioni verticali e orizzontali del Nodo 4 della variazione di C_{xx}



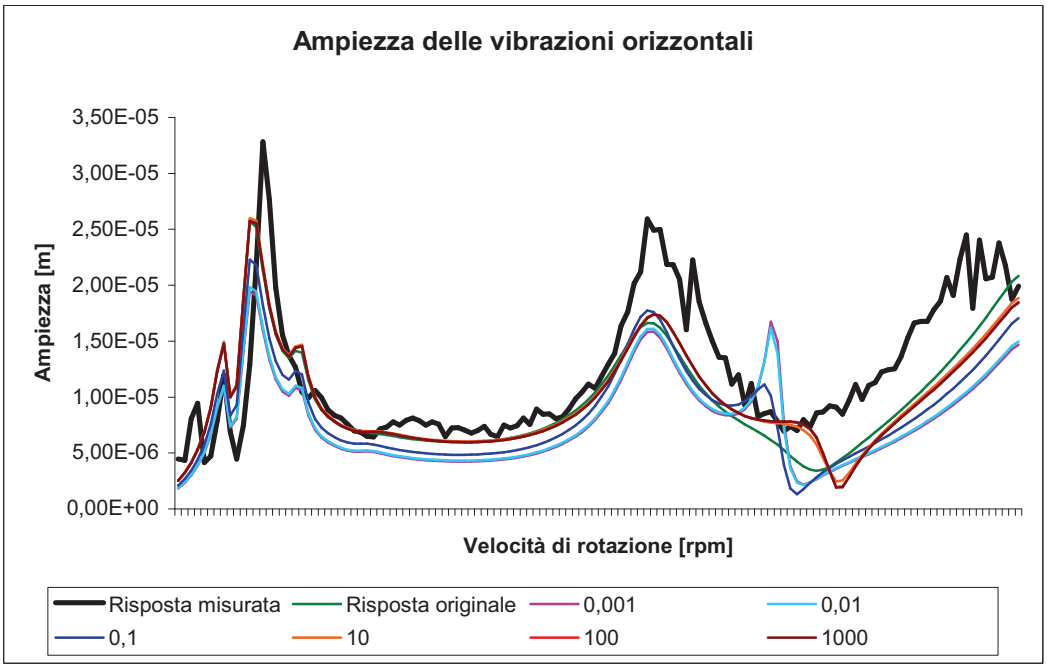
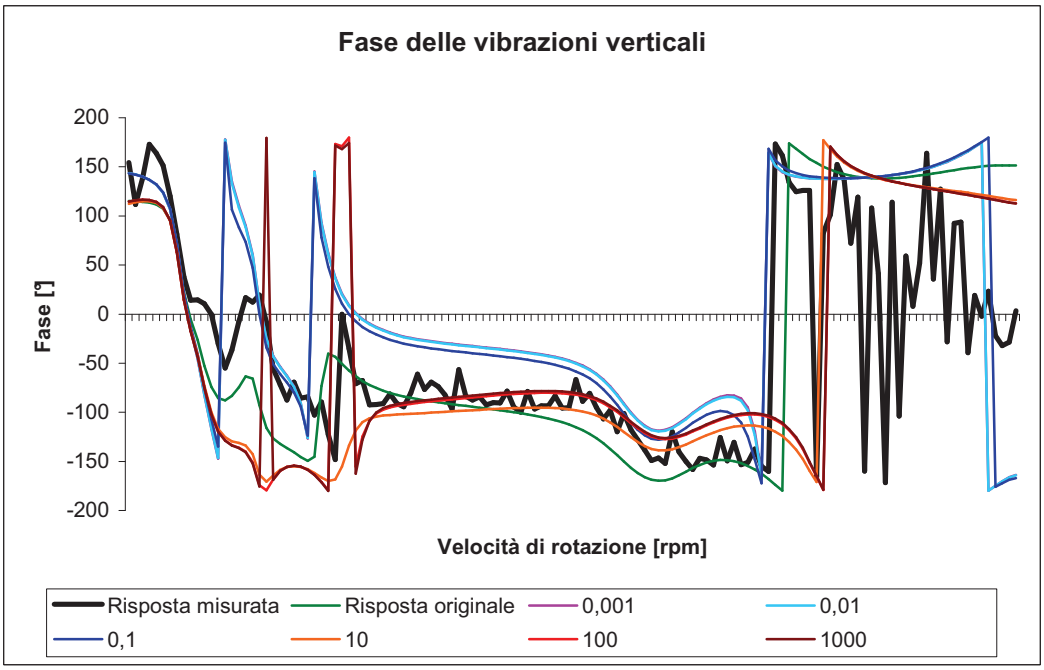


Figura 3.22 – Influenza sull'ampiezza delle vibrazioni verticali e orizzontali del Nodo 44 della variazione di C_{xx}



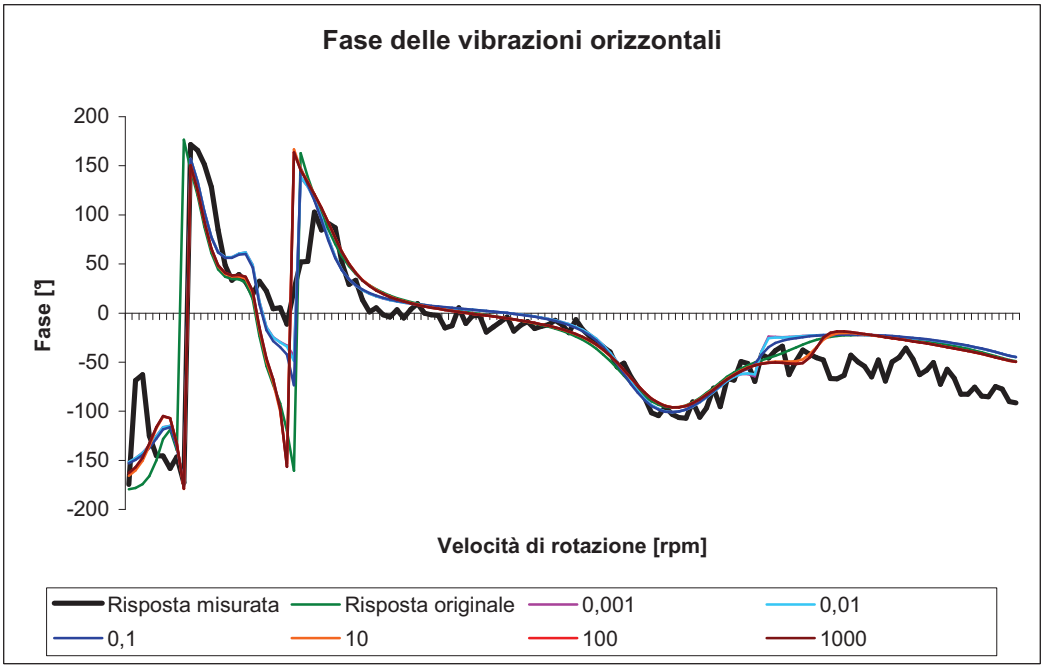
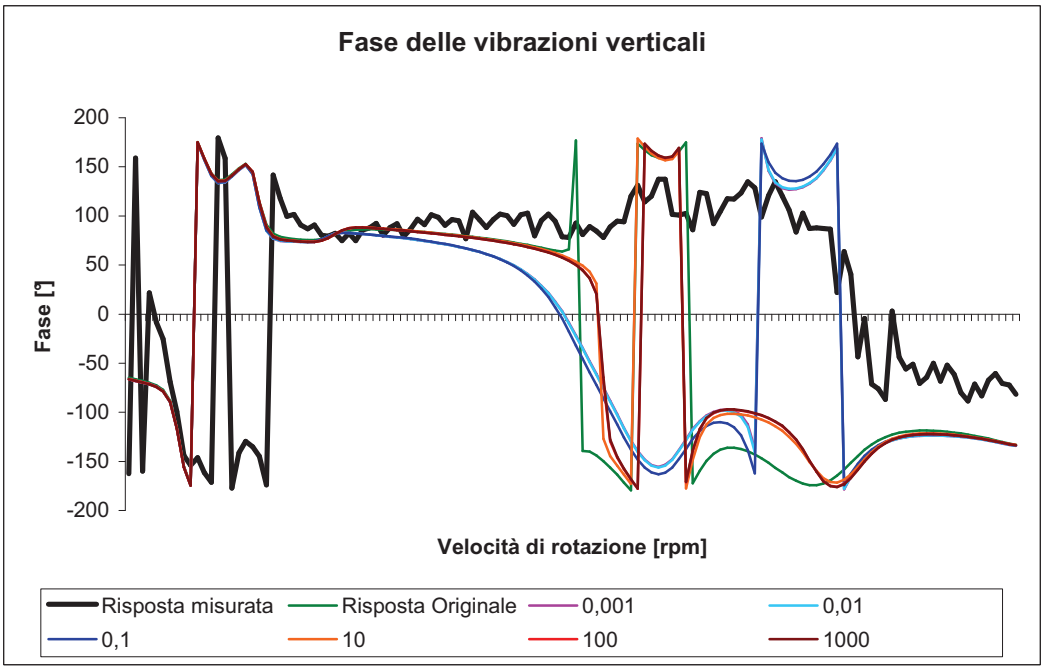


Figura 3.23 –Influenza sulla fase delle vibrazioni verticali e orizzontali ntale del Nodo 4 della variazione di C_{xx}



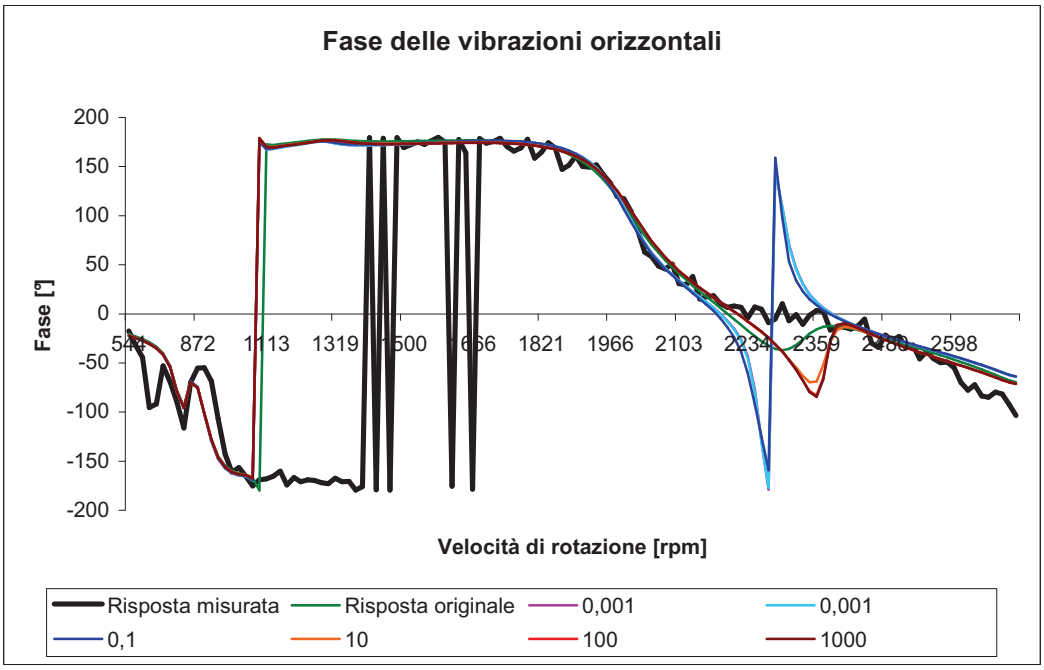


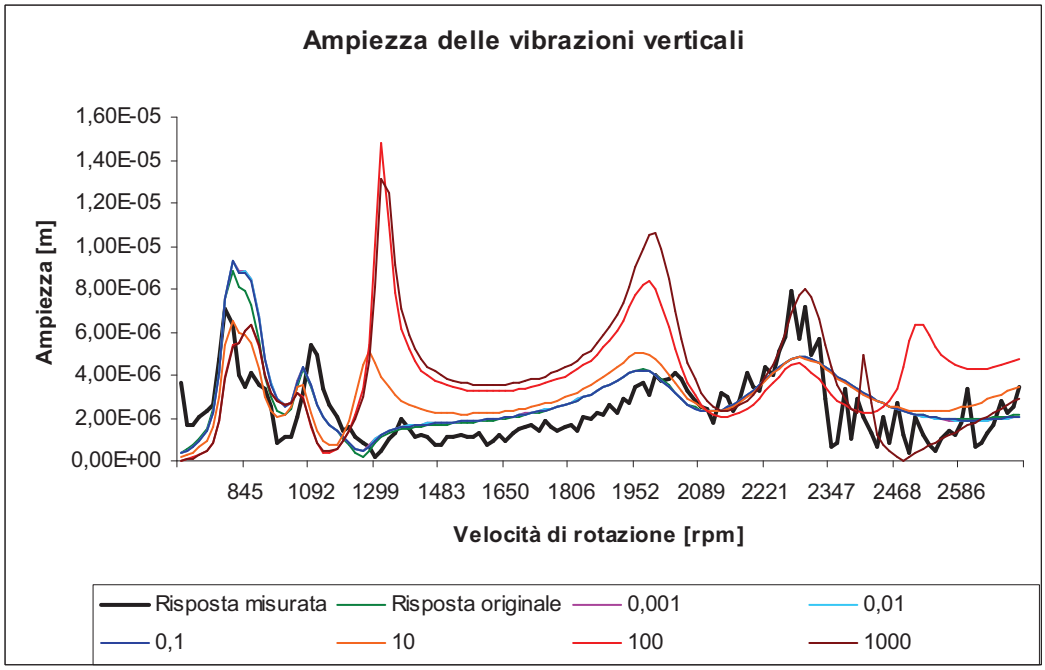
Figura 3.24 – Influenza sulla fase delle vibrazioni verticali e orizzontali del Nodo 44 della variazione di C_{xx}

3.4.4 Coefficiente di smorzamento c_{xy}

Il coefficiente di smorzamento trasversale influisce su entrambi i gradi di libertà come mostrato sui grafici in Figura 3.25. Le curve per un basso coefficiente moltiplicativo dello smorzamento sono affusolate. Una volta superato il valore iniziale, l'aumento del coefficiente conduce ad un aumento dell'ampiezza della vibrazione secondo **x** e a una diminuzione secondo **y.** Notiamo anche che, per alte velocità (maggiori di 2300 rpm), le risposte ottenute con alti valori del coefficiente sembrano divergere. Con la distanza i fenomeni sono simili, pero in un intervallo di variazione ristretto si mantiene il fenomeno di divergenza per alte velocità descritto precedemente (Figura 3.26). L'influenza sulla fase si nota sopratutto sull'asse **x**. Il modello è buono ancora una volta per valori bassi di c_{xy}, mentre per valori alti le curve si spostano verticalmente (Figura 3.27). Anche la fase subisce il distaccamento per alte velocità e alti valori del parametro. A questo punto possiamo dire che un valore troppo alto dello smorzamento trasforma la natura fisica del modello analitico che non corrisponde più a quello reale. L'algoritmo di ottimizzazione sarà quindi performante se spingerà la ricerca del coefficiente verso i valori bassi. L'ultima serie di grafici per la fase conferma il fenomeno di distaccamento per alte velocità (Figura 3.28).

3.4.5 Conclusione sulla sensibilità

Lo studio sulla sensibilità ci permette di definire l'intervallo nel quale vogliamo effettuare l'ottimizzazione dei diversi parametri. I risultati grafici ci hanno mostrato che le curve ottenute moltiplicando per 0.001 e per 0.01 del valore iniziale sono sovrapposte quindi possiamo definire 0.01 volte il valore iniziale come estremo inferiore. Allo stesso modo, per una moltiplicazione superiore a 100, le curve sono o sovrapposte o divergono totalmente rispetto alle misure. Imporremo quindi 100 volte il valore iniziale come estremo superiore all'intervallo. Definiamo il file *updatingcuscinetti.txt* con l'intervallo [0.01,100] per ogni parametro. Anche se l'influenza di un parametro diminuisce globalmente all'allontanarsi del punto dove è definito non possiamo trascorre le modifiche questo suppone quindi di effettuare l'aggiornamento di tutti i parametri contemporaneamente.



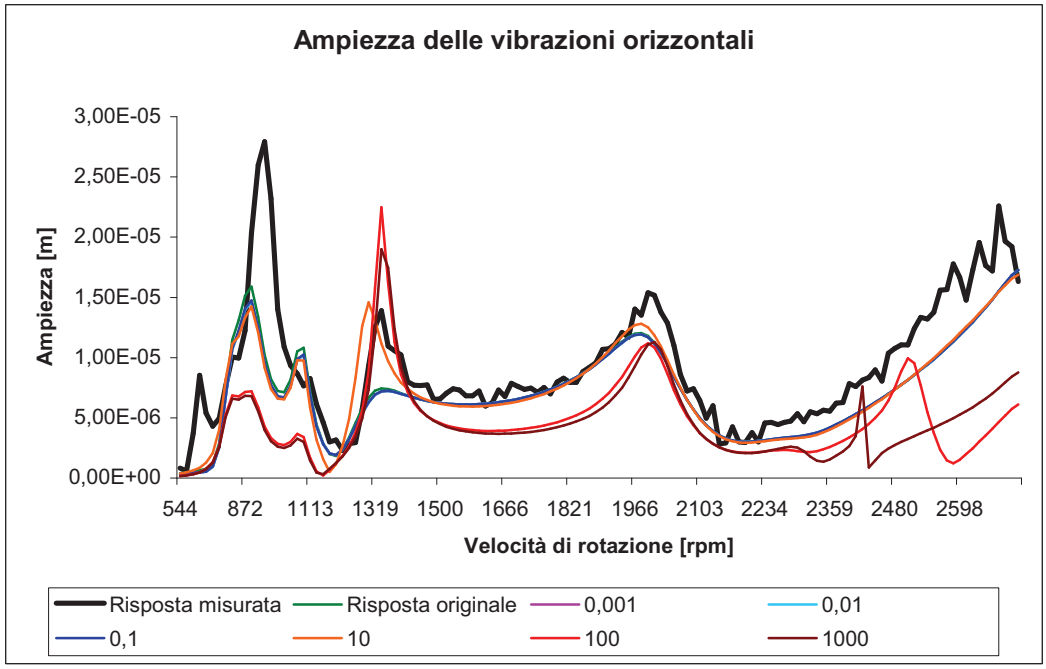
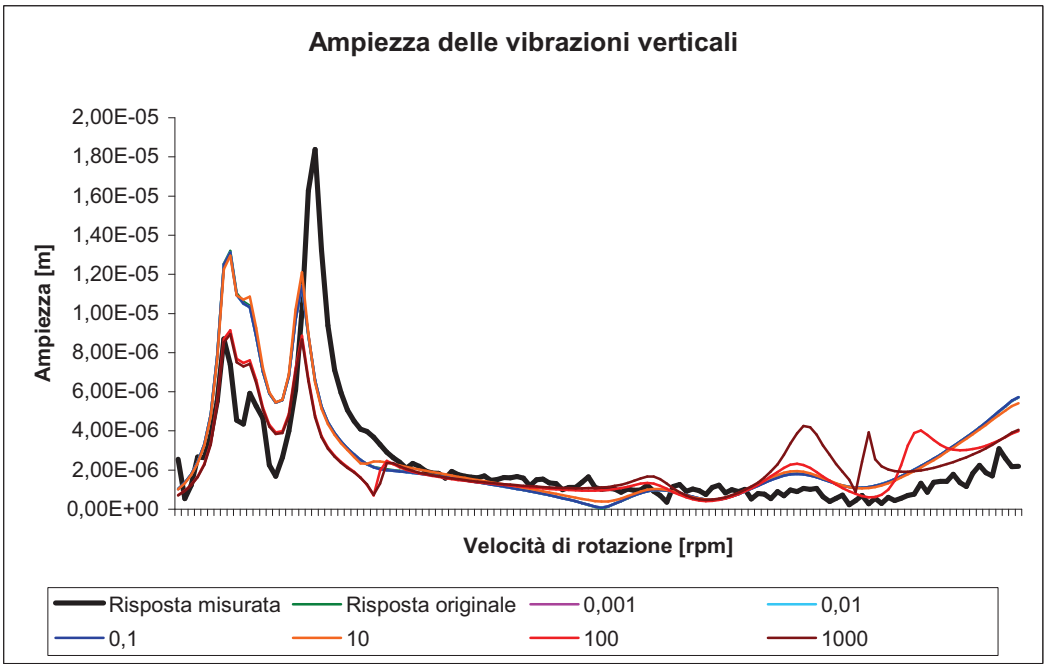


Figura 3.24 – Influenza sull'ampiezza delle vibrazioni verticali e orizzontali del Nodo 4 della variazione di C_{xy}



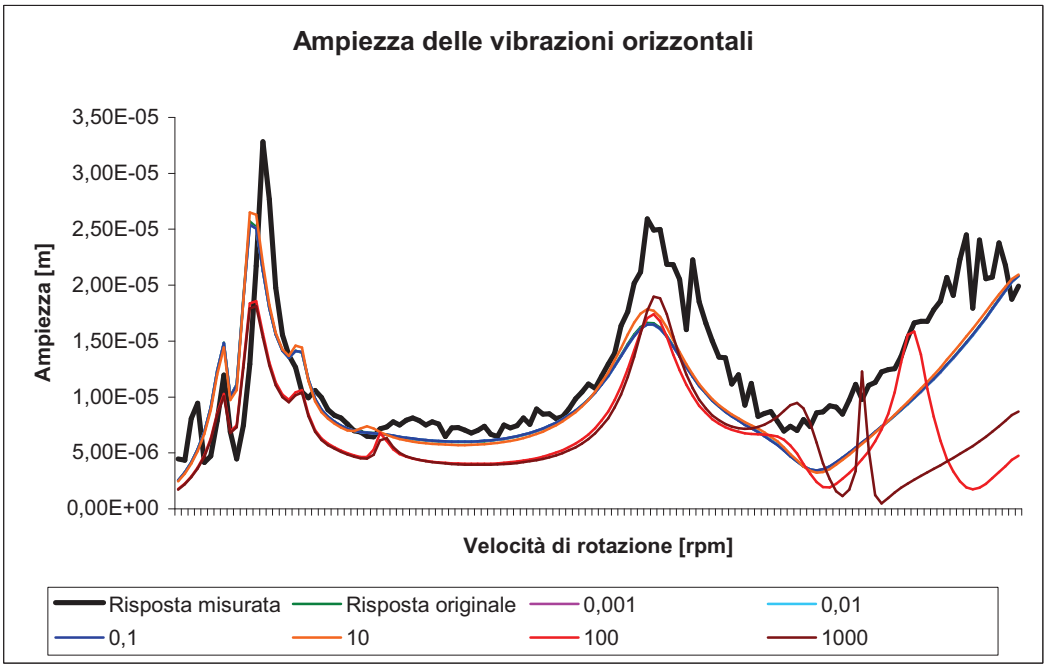
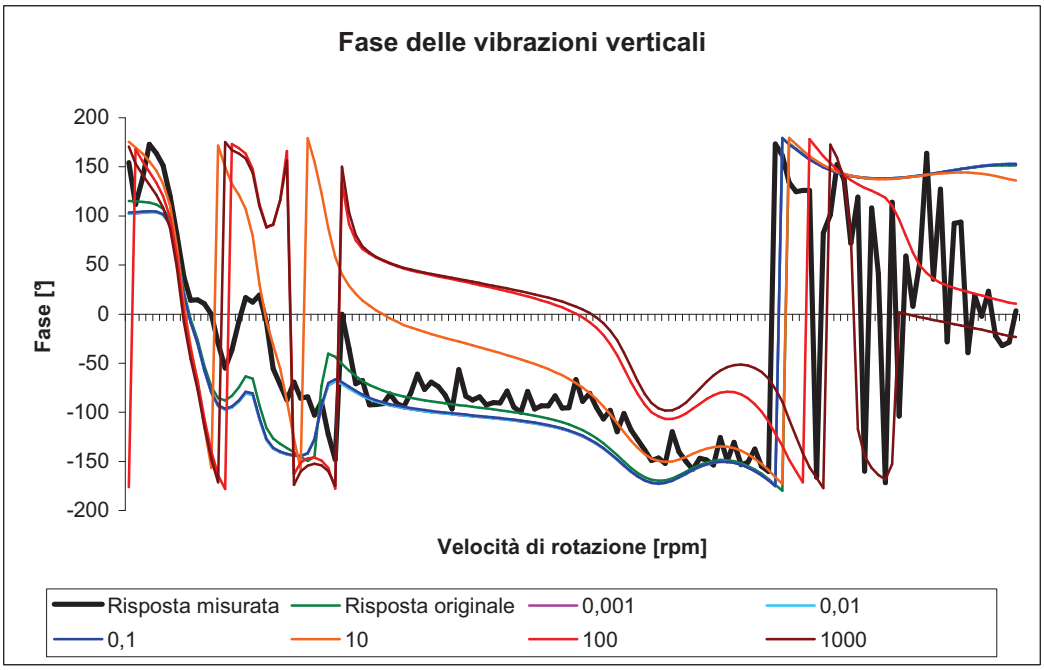


Figura 3.25 – Influenza sull'ampiezza delle vibrazioni verticali e orizzontali del Nodo 44 della variazione di C_{xy}



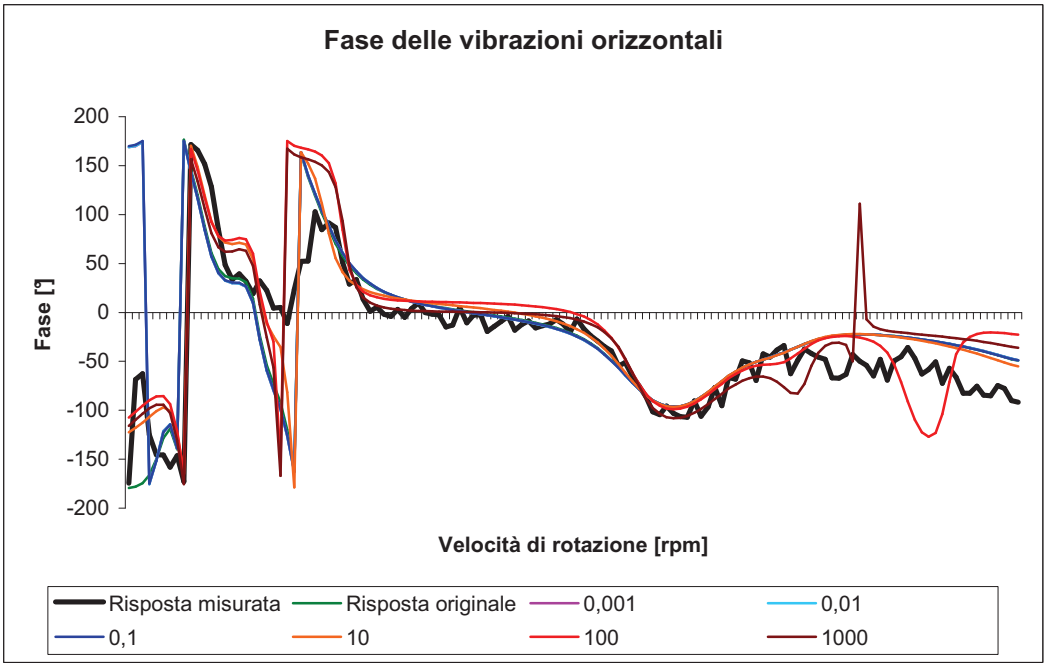
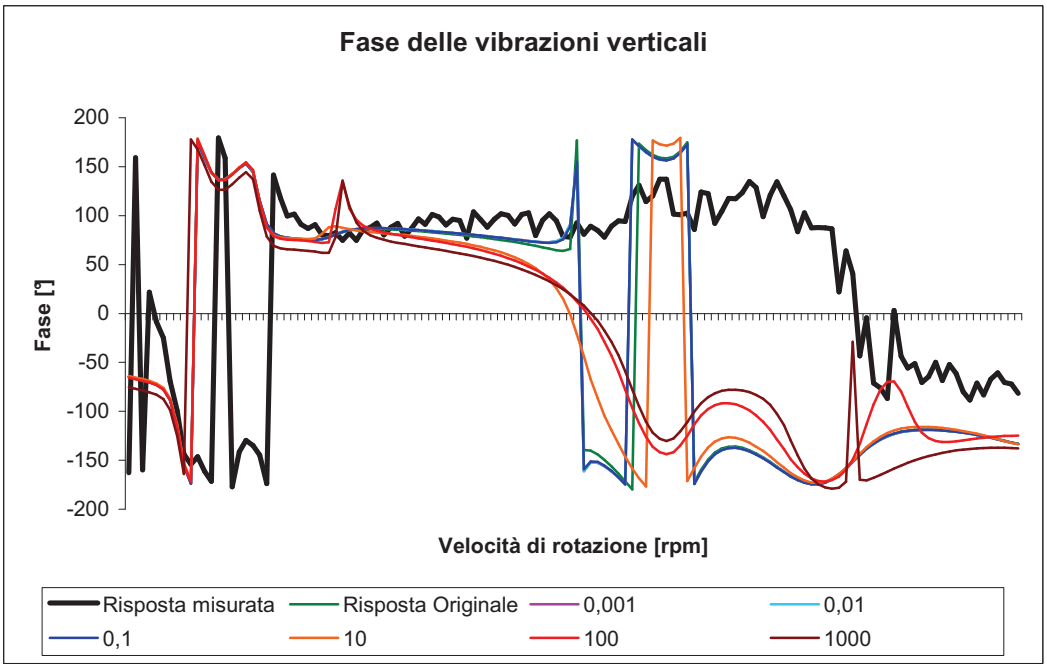


Figura 3.26 Influenza sulla fase delle vibrazioni verticali e orizzontali del Nodo 4 della variazione di C_{xy}



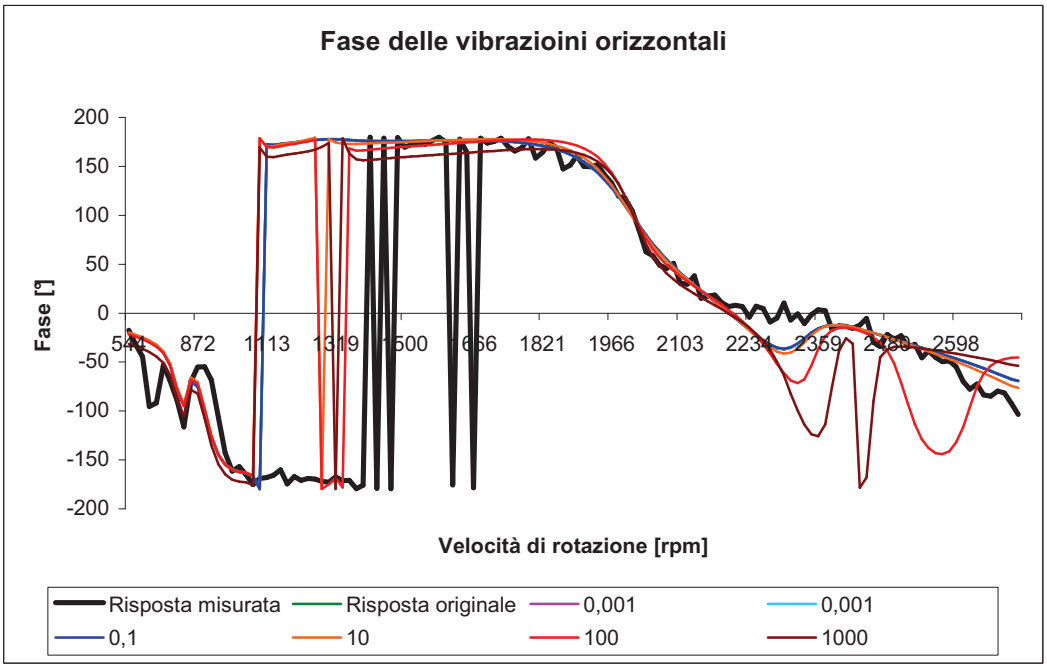


Figura 3.27 – Influenza sulla fase delle vibrazioni verticali e orizzontali del Nodo 44 della variazione di C_{xy}

3.5 Convergenza dell'algoritmo

L'obiettivo di questa sezione sarà di valutare l'efficienza dell'algoritmo rispetto ai parametri, cioè verificare se i parametri convergono verso valori specifici o se si distribuiscono uniformemente. Studieremo questo punto con due approcci diversi e confronteremo i due risultati per cercare di capire se la posizione iniziale del parametro nell'intervallo di variazione influisce o no sulla posizione finale del parametro aggiornato. Per poter effettuare questa comparazione, analizzeremo in un primo tempo la situazione di un padre originale che produce venti figli diversi. Riporteremo dopo su diversi grafici la dispersione dei risultati. In un secondo tempo lanceremo l'algoritmo su quaranta padri generati casualmente e ripeteremo lo stesso studio comparativo dei nuovi parametri ottenuti.

3.5.1 Primo caso

L'obiettivo di questo primo studio è di mostrare in che modo il punto di partenza iniziale di un parametro influisca sul valore finale ottenuto. Disponiamo di una serie di cuscinetti modellati in maniera abbastanza accurata e questo è un problema visto che l'updating modificherà poco i parametri dei cuscinetti e quindi sarà difficile capire il fenomeno di convergenza. Abbiamo pensato allora di generare quattro cuscinetti casuali dove ogni parametro è stato generato casualmente a partire del parametro corrispondente dei cuscinetti iniziali. Ogni parametro viene moltiplicato per un numero generato con la legge uniforme su [0.1,1.9] tramite una procedura Matlab (Appendice 1**:** Funzione *Generazione*) in modo da mantenere l'ordine di grandezza del valore iniziale. Una volta generati questi quattro cuscinetti, verrà lanciato venti volte l'algoritmo di ottimizzazione nella modalità (1,1)-ES con 150 iterazioni. Otterremo quindi 20 set di quattro cuscinetti figli. Per ogni parametro rappresenteremo la dispersione dei valori ottimizzati su un grafico rispetto alla posizione iniziale del padre. Per permettere una comparazione più facile dei risultati, tutti valori ottenuti saranno normalizzati rispetto ai parametri dei cuscinetti forniti (Appendice 2**:** Funzione *MatPF*). I valori ottenuti saranno in seguito raggruppati secondo la loro appartenenza a diversi intervalli per misura de la frequenza di ogni intervallo di valori (Appendice 3: Funzione *Ripartizione*).In questo studio sarà difficile trarre conclusioni se, per esempio, i risultati dovessero presentarsi sottoforma di una Gaussiana intorno al punto rappresentativo del parametro padre. Le spiegazioni possono in effetti essere due: o il padre è già vicino ad un ottimo, e quindi l'ottimizzazione agisce poco sul valore, o il parametro è in una zona dello spazio che porta la risposta in un minimo locale e quindi c'è un fenomeno in intrappolamento. Per esempio nel caso di K_{xy} del cuscinetto 1 si verifica questo fenomeno (Figura 3.28).

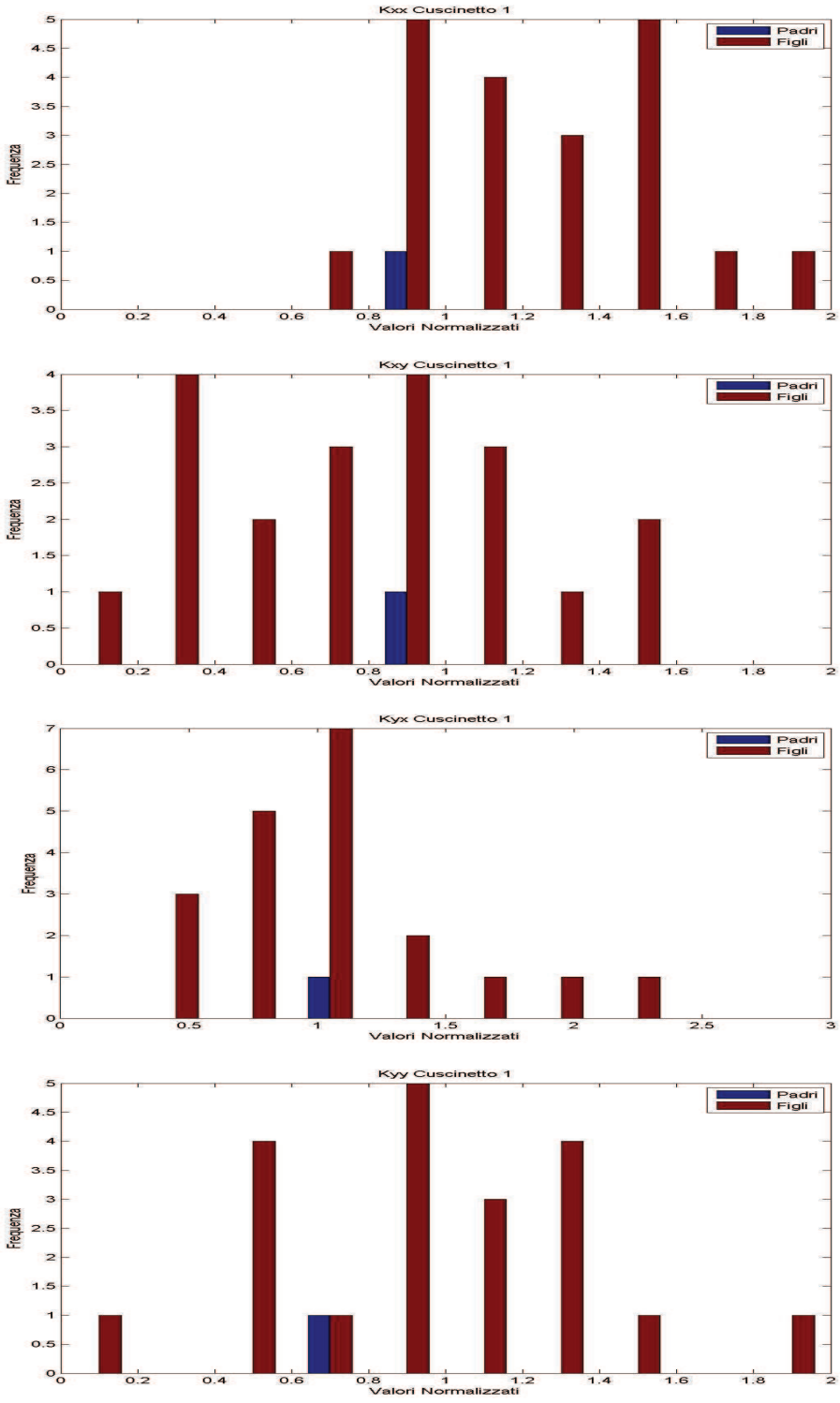


Figura 3.28 – Convergenza per K_{xx}, K_{xy}, K_{yx}, K_{yy} del cuscinetto 1

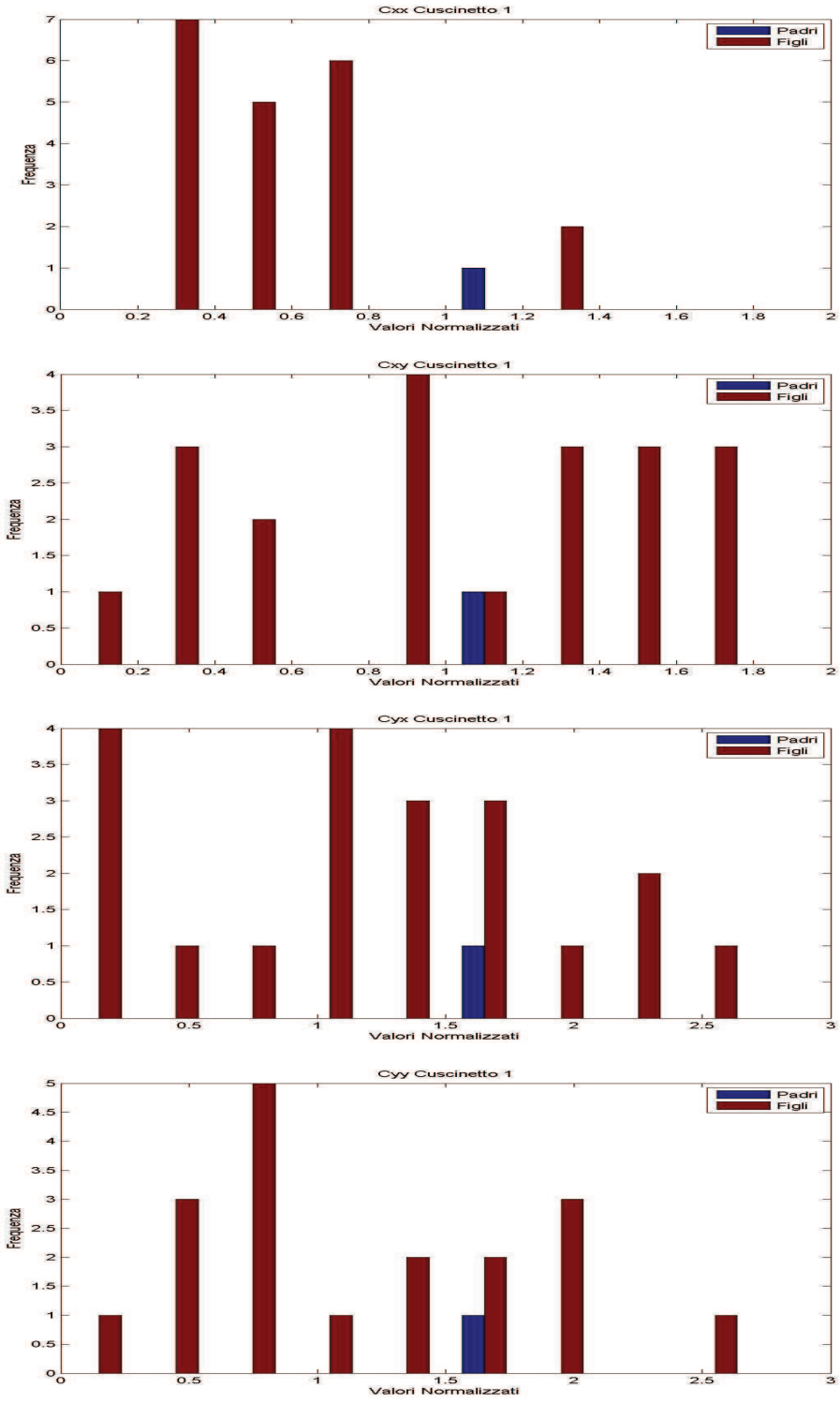


Figura 3.29 – Convergenza per C_{xx}, C_{xy}, C_{yx}, C_{yy} del cuscinetto 1

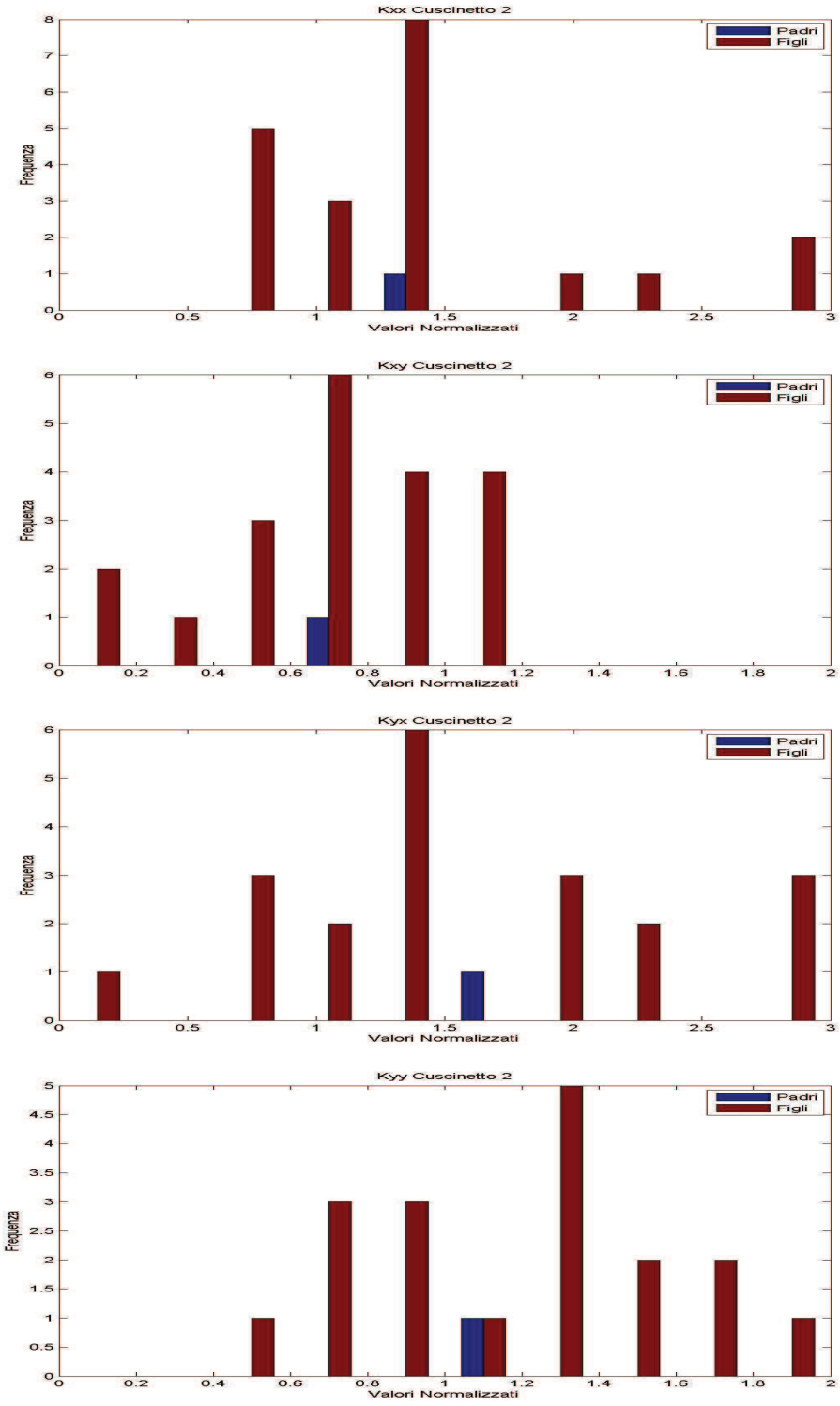


Figura 3.30 – Convergenza per K_{xx}, K_{xy}, K_{yx}, K_{yy} del cuscinetto 2

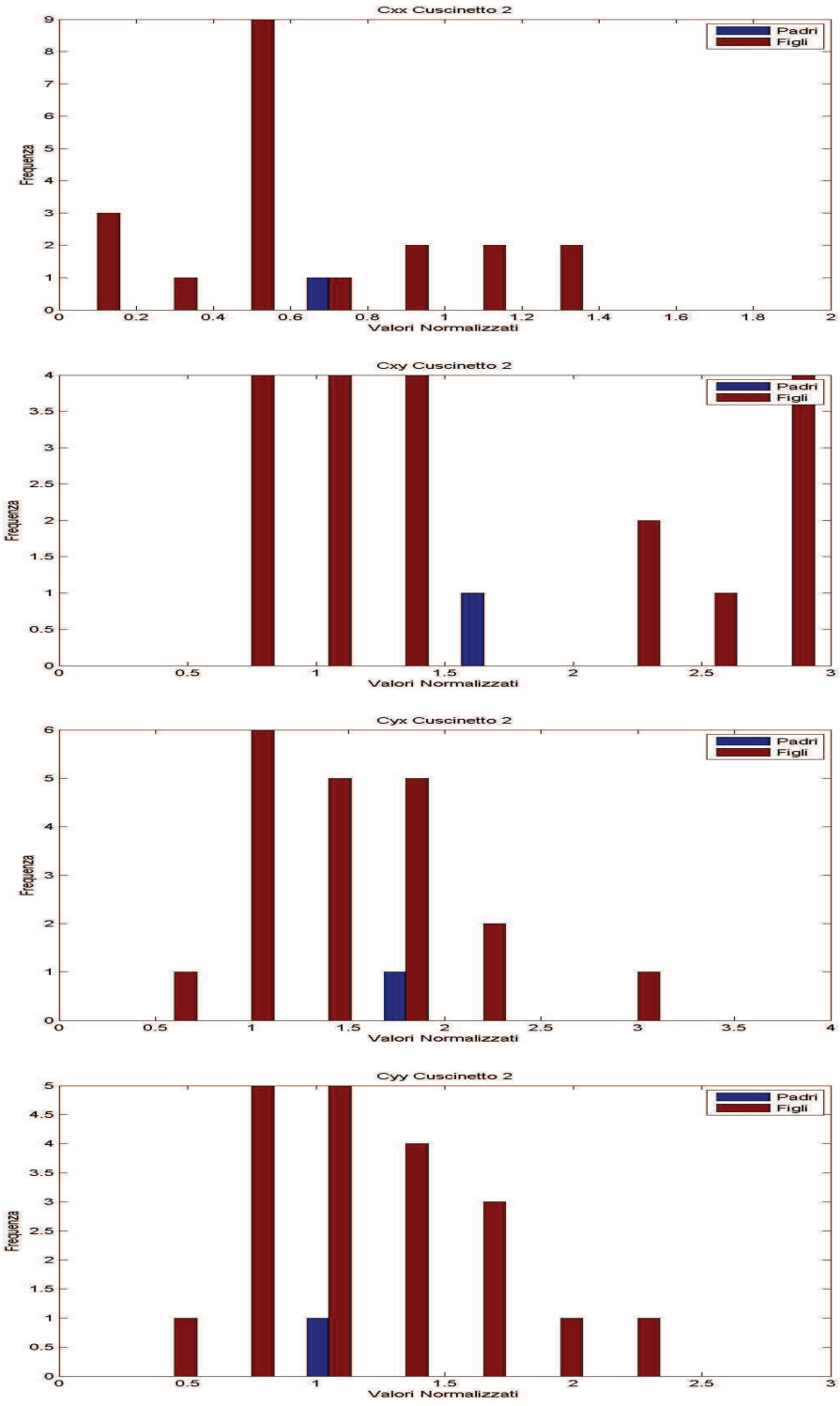


Figura 3.31 – Convergenza per C_{xx}, C_{xy}, C_{yx}, C_{yy} del cuscinetto 2

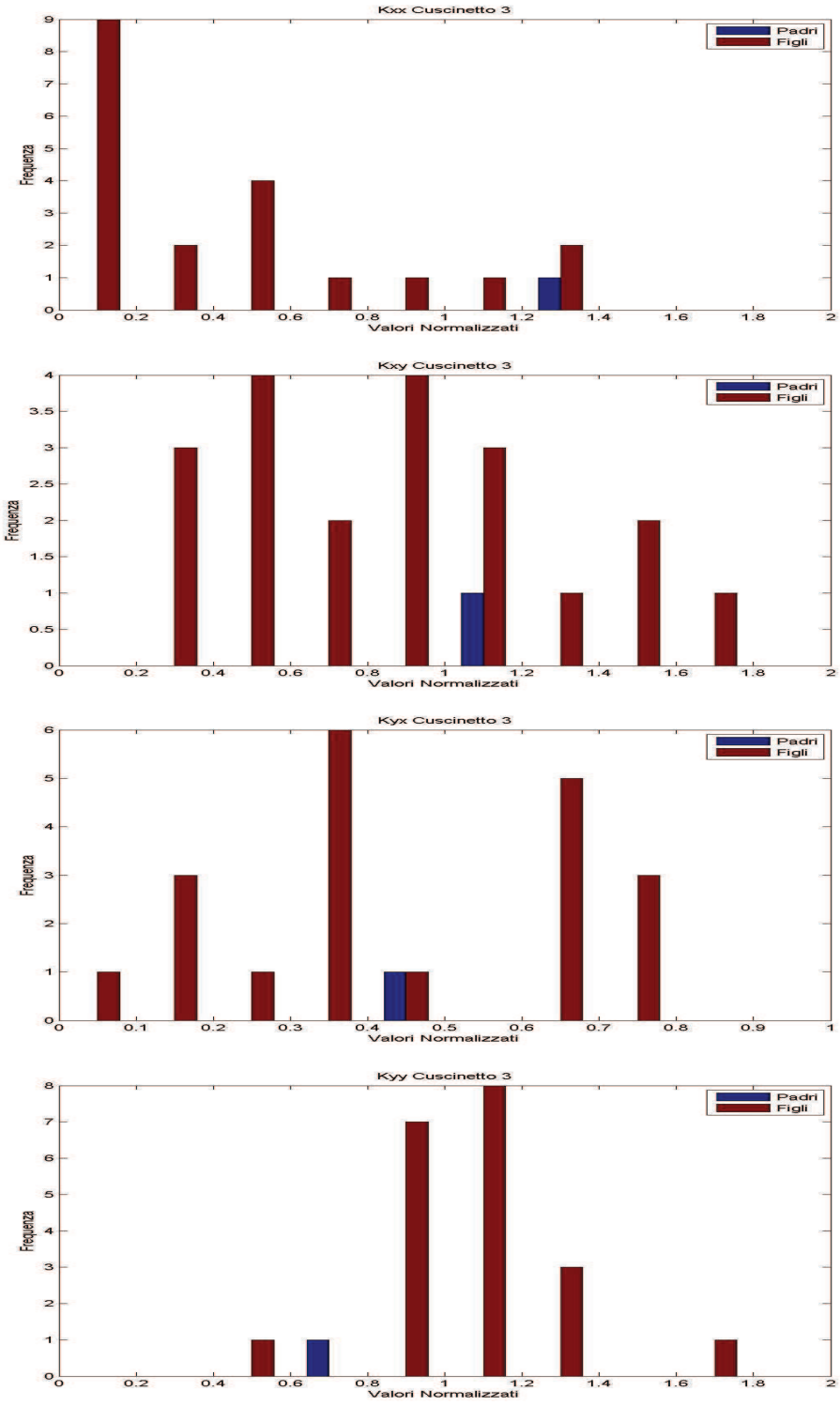


Figura 3.32 – Convergenza per K_{xx}, K_{xy}, K_{yx}, K_{yy} del cuscinetto 3

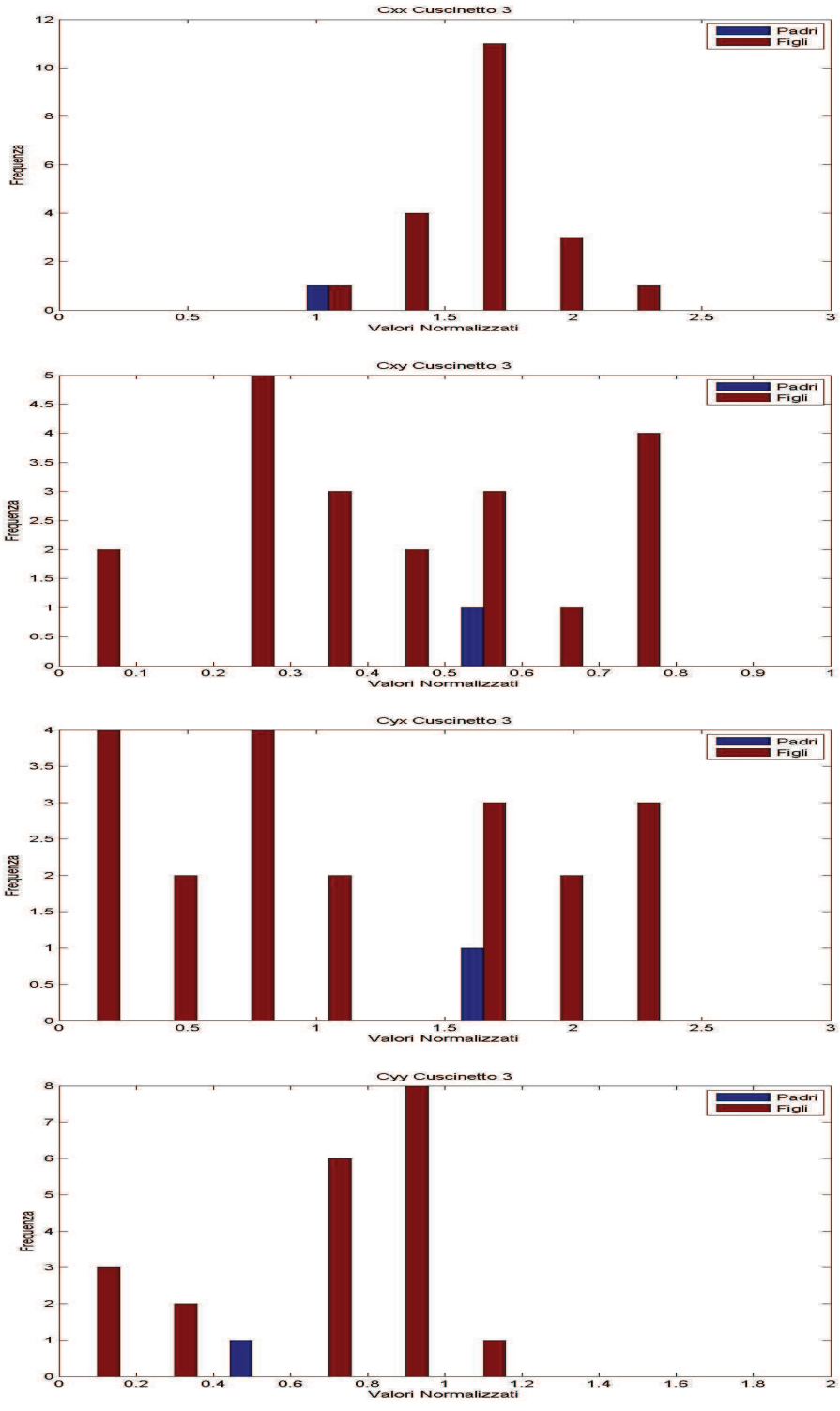


Figura 3.33 – Convergenza per C_{xx}, C_{xy}, C_{yx}, C_{yy} del cuscinetto 3

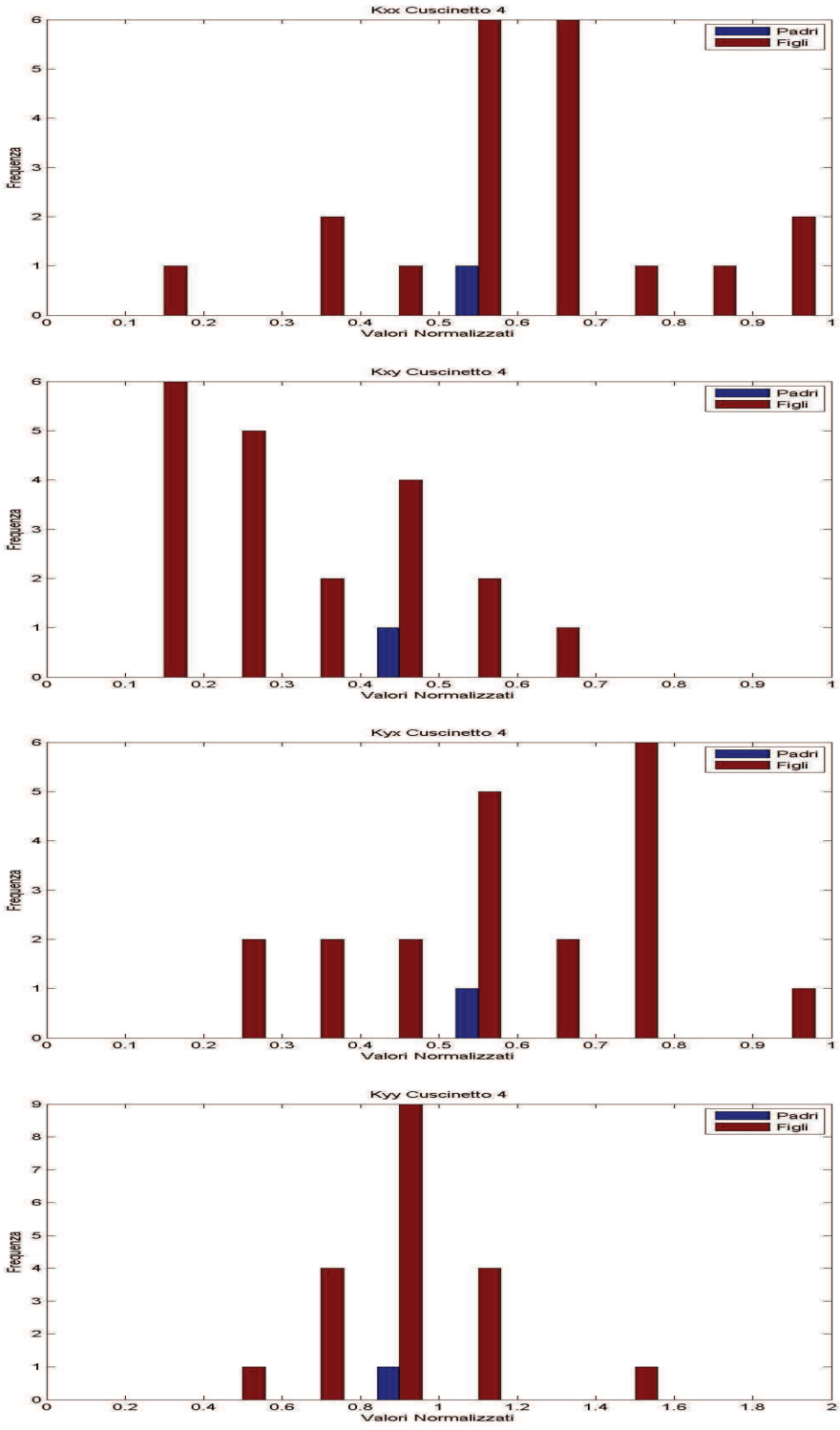


Figura 3.34 – Convergenza per K_{xx}, K_{xy}, K_{yx}, K_{yy} del cuscinetto 4

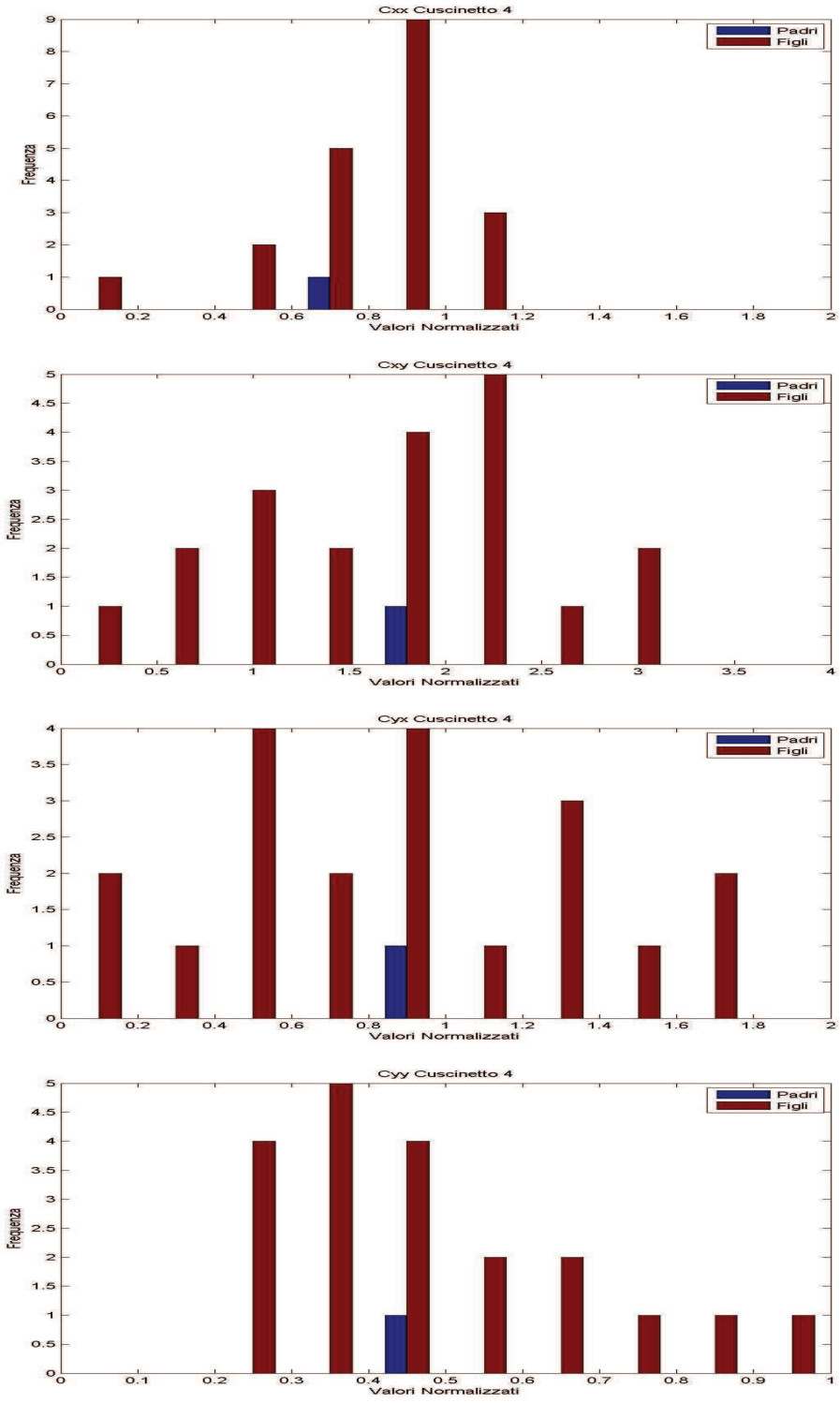


Figura 3.35 – Convergenza per C_{xx}, C_{xy}, C_{yx}, C_{yy} del cuscinetto 4

In altre situazioni, come nel caso di C_{xx} del cuscinetto 1, (**Figura 3.29)** vediamo che i valori si concentrano principalmente attorno al valore 0.5.

Un altro caso di convergenza osservata è quello di una convergenza uniforme su un intervallo dei valori dei parametri. Per esempio, per i coefficienti C_{yy} e C_{xy} del cuscinetto 1 (Figura 3.29), K_{yx} K_{yy}, C_{xy} del cuscinetto 2 (Figure 3.30 e 3.31), C_{xy} del cuscinetto 3 (Figura 3.33) e per C_{yx} del cuscinetto 4 (Figura 3.35) vediamo questo tipo di configurazione. Questo può significare che nell'intervallo in vicinanza del punto di partenza la variazione del parametro non modifica significativamente la risposta ottenuta, quindi possono essere parametri poco sensibili della modellazione. Viene dopo il terzo caso che è quello di ripartizione di tipo Gaussiana però in un posto diverso da quello di partenza. Questo indica la posizione di zone in cui il valore del parametro permette di ridurre bene la differenza tra il modello e le misure. Questo non significa però che la zona sia la migliore, perché può esserci un minimo locale determinato dal punto di partenza del valore. Esempi di questo tipo di convergenza sono soprattutto K_{xx}, K_{yy}, C_{xx}, C_{yy} del cuscinetto 3 (Figure 3.32 e 3.33). Il cuscinetto 3 sembra quindi avere un ruolo preponderante nell'updating. Per determinare la natura di questa zona conviene fare un secondo test di convergenza.

3.4.2 Secondo caso

In questo caso, per liberarsi delle condizioni iniziali, abbiamo usato l'updating su una serie di 40 set di cuscinetti diversi. In questo modo, se dovesse esserci convergenza, essa non dipenderà delle condizioni iniziali, visto che avremo dei valori iniziali dispersi su un intervallo. Per generare questi 40 set di cuscinetti padri useremo la stessa funzione (Appendice 1**:** Funzione *Generazione*). Lanceremo in seguito l'algoritmo (150 iterazioni) una volta sola per ogni padre e otterremo 40 set di cuscinetti figli. Questo permette di controllare se l'updating simultaneo di tutti i parametri tende verso una soluzione unica o se solo certi parametri tendono verso un ottimo.

Se guardiamo i risultati presentati vediamo che la convergenza di certi parametri rimane uniforme anche con questo tipo di test. Questo significa che il valore di questo coefficiente influisce pochissimo sulla risposta ottenuta e che quindi questi coefficienti sono poco sensibili del modello. Si può osservare questo fenomeno per i coefficienti K_{yx}, C_{yx}, C_{yy} del cuscinetto 1 (Figure 3.36 e 3.37), K_{yy} del cuscinetto 2 (Figura 3.38), K_{xy}, C_{xy} del cuscinetto 3 e C_{yx} del cuscinetto 4. Possiamo capire con questo studio i coefficienti che sono influenzato dalle condizioni iniziali.

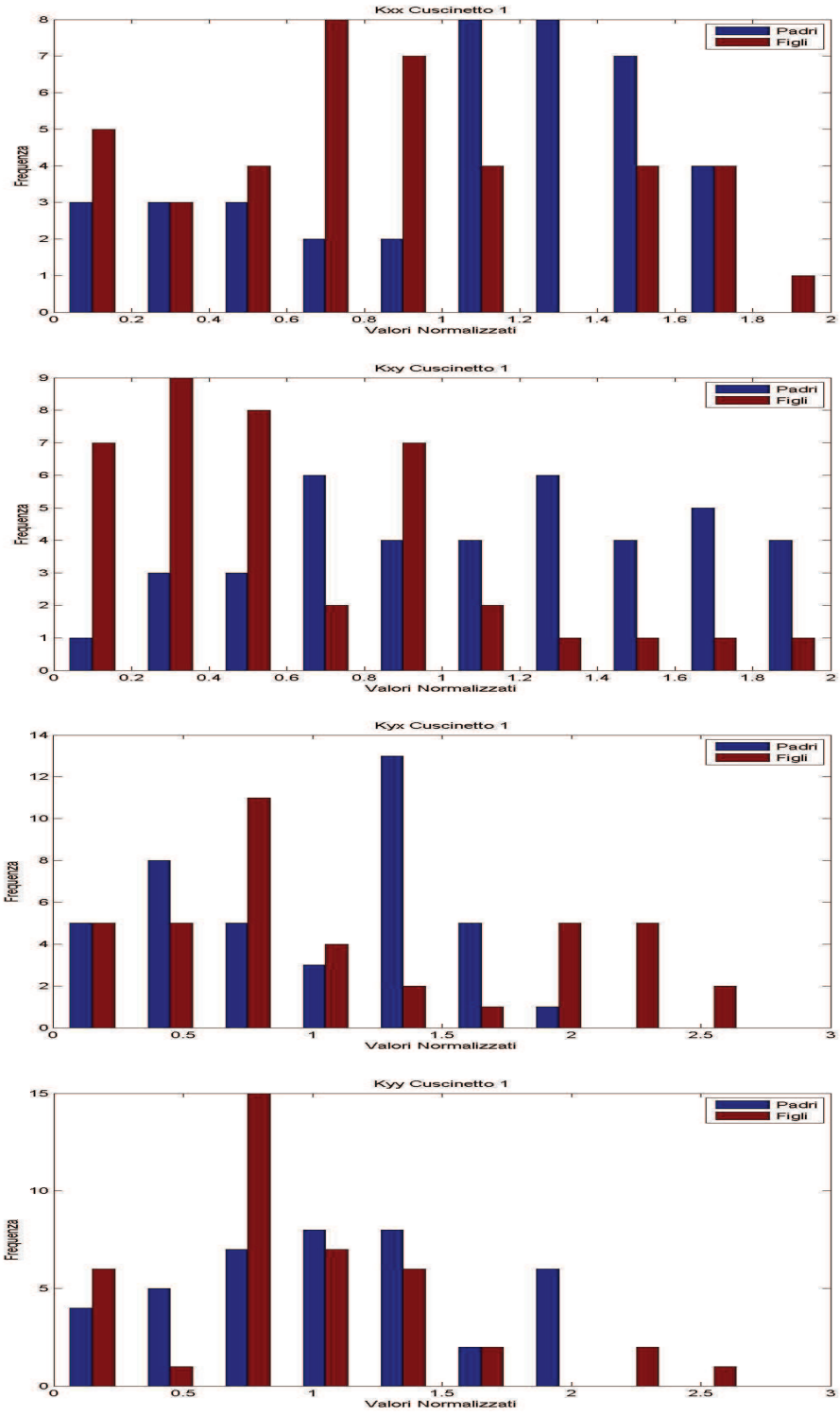


Figura 3.36 – Convergenza per K_{xx}, K_{xy}, K_{yx}, K_{yy} del cuscinetto 1

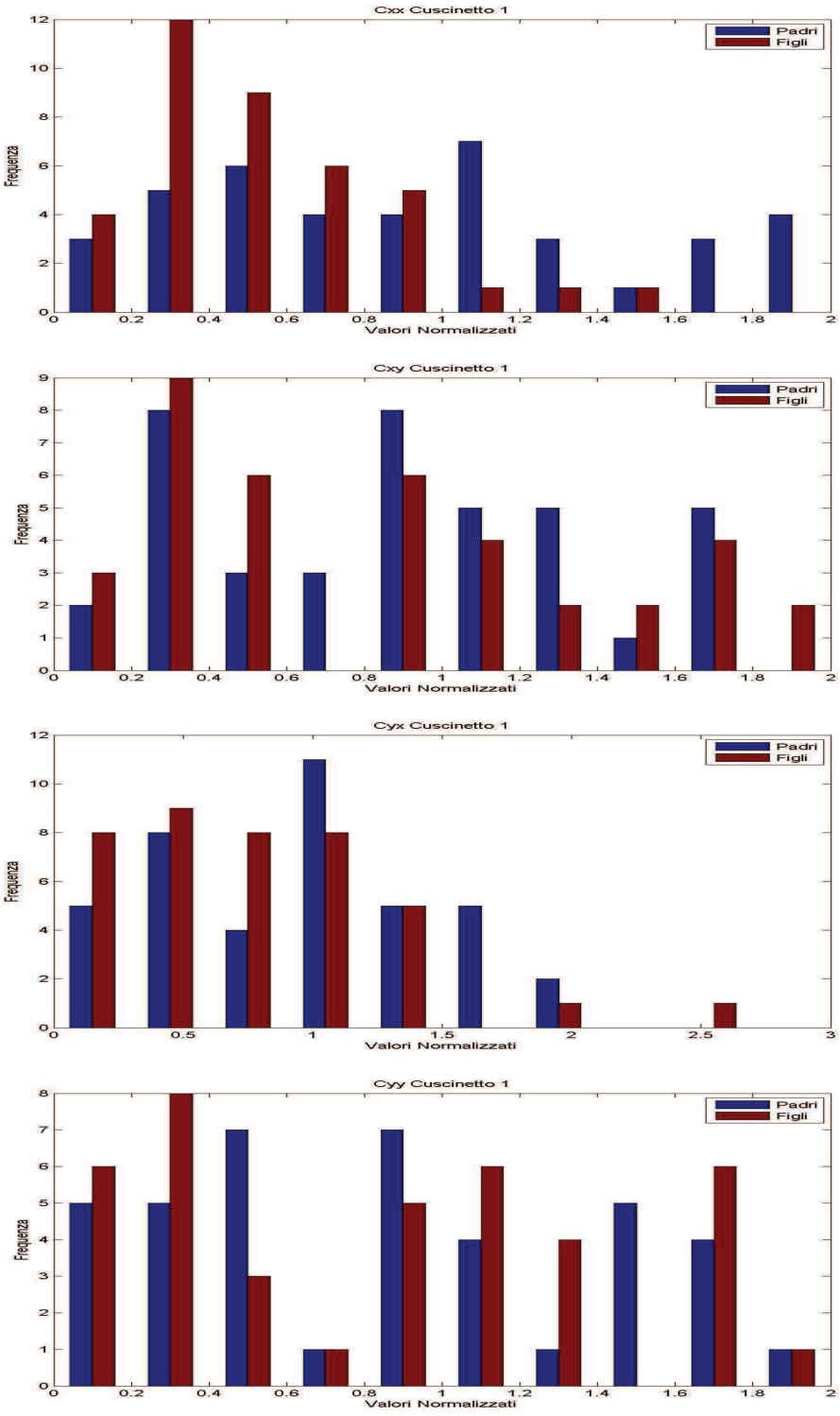


Figura 3.37 – Convergenza per C_{xx}, C_{xy}, C_{yx}, C_{yy} del cuscinetto 2

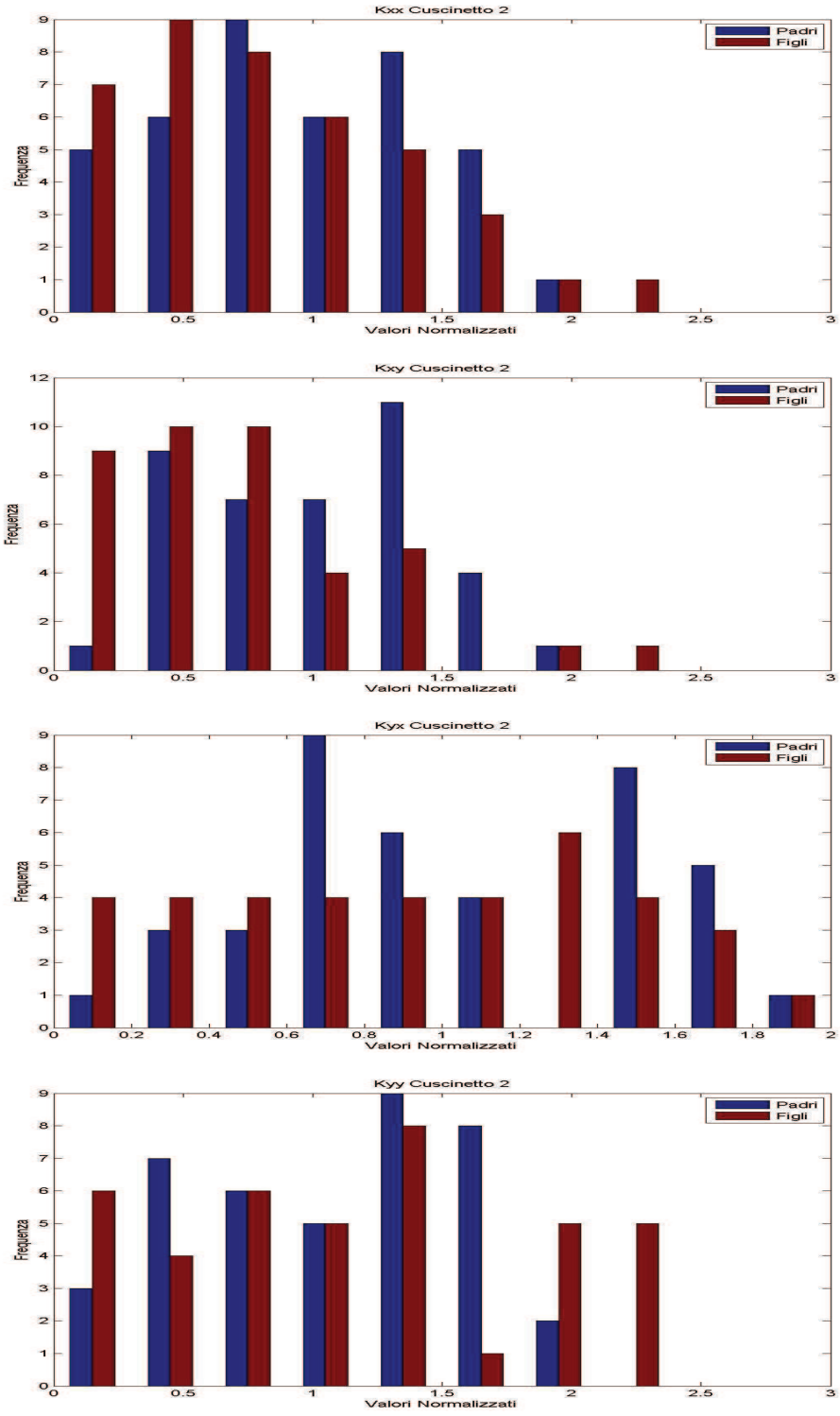


Figura 3.38 – Convergenza per K_{xx}, K_{xy}, K_{yx}, K_{yy} del cuscinetto 2

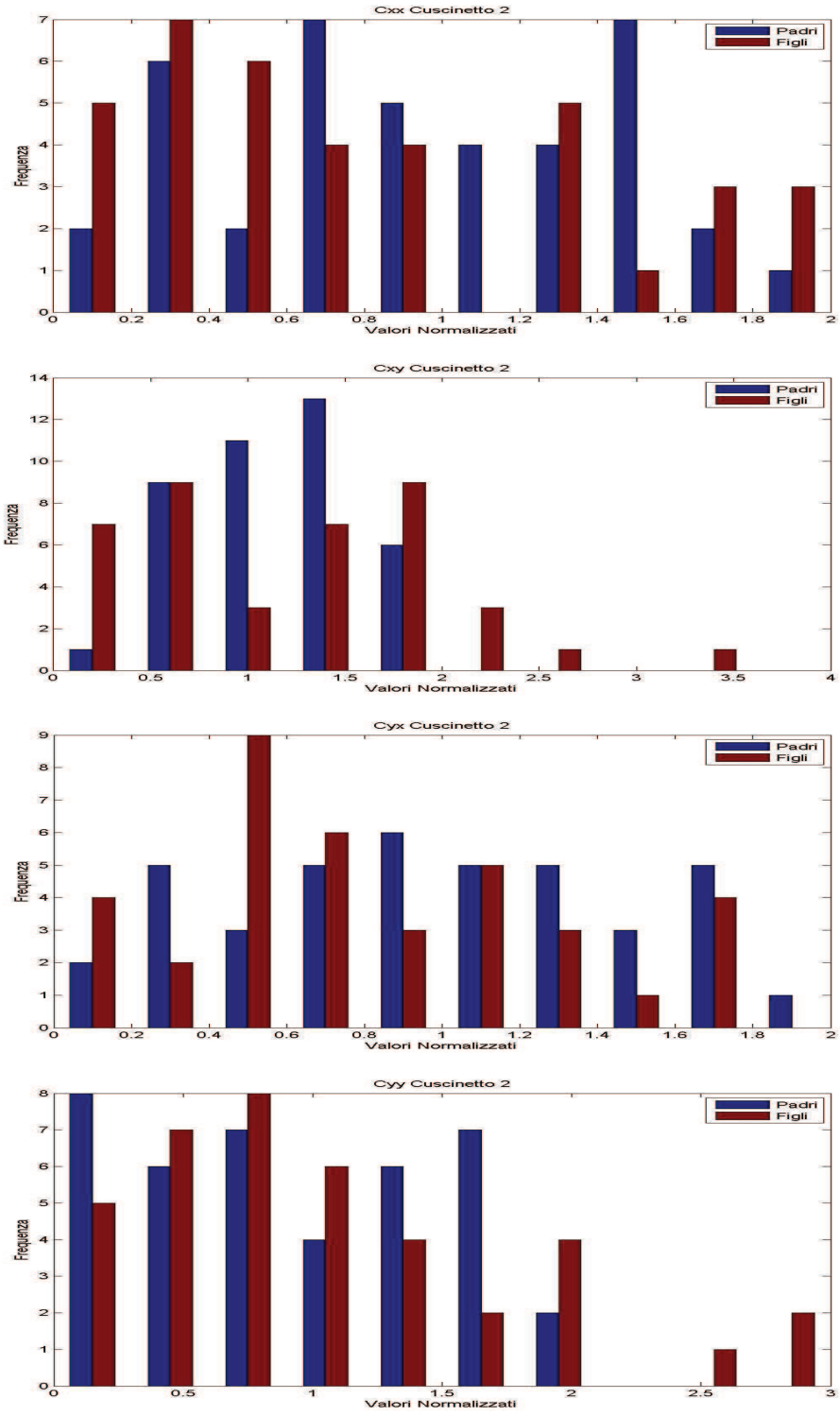


Figura 3.39 – Convergenza per C_{xx}, C_{xy}, C_{yx}, C_{yy} del cuscinetto 2

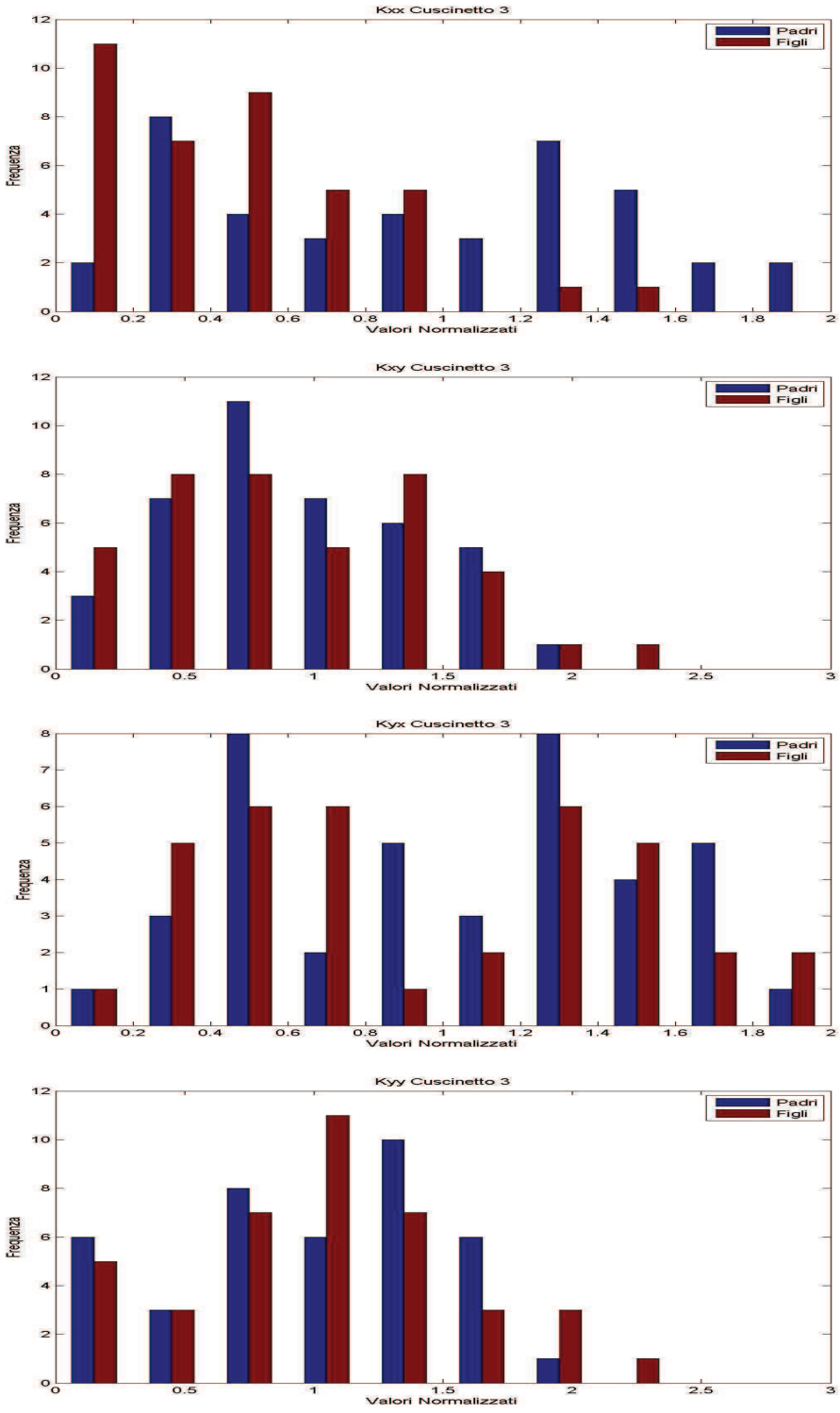


Figura 3.40 – Convergenza per K_{xx}, K_{xy}, K_{yx}, K_{yy} del cuscinetto 3

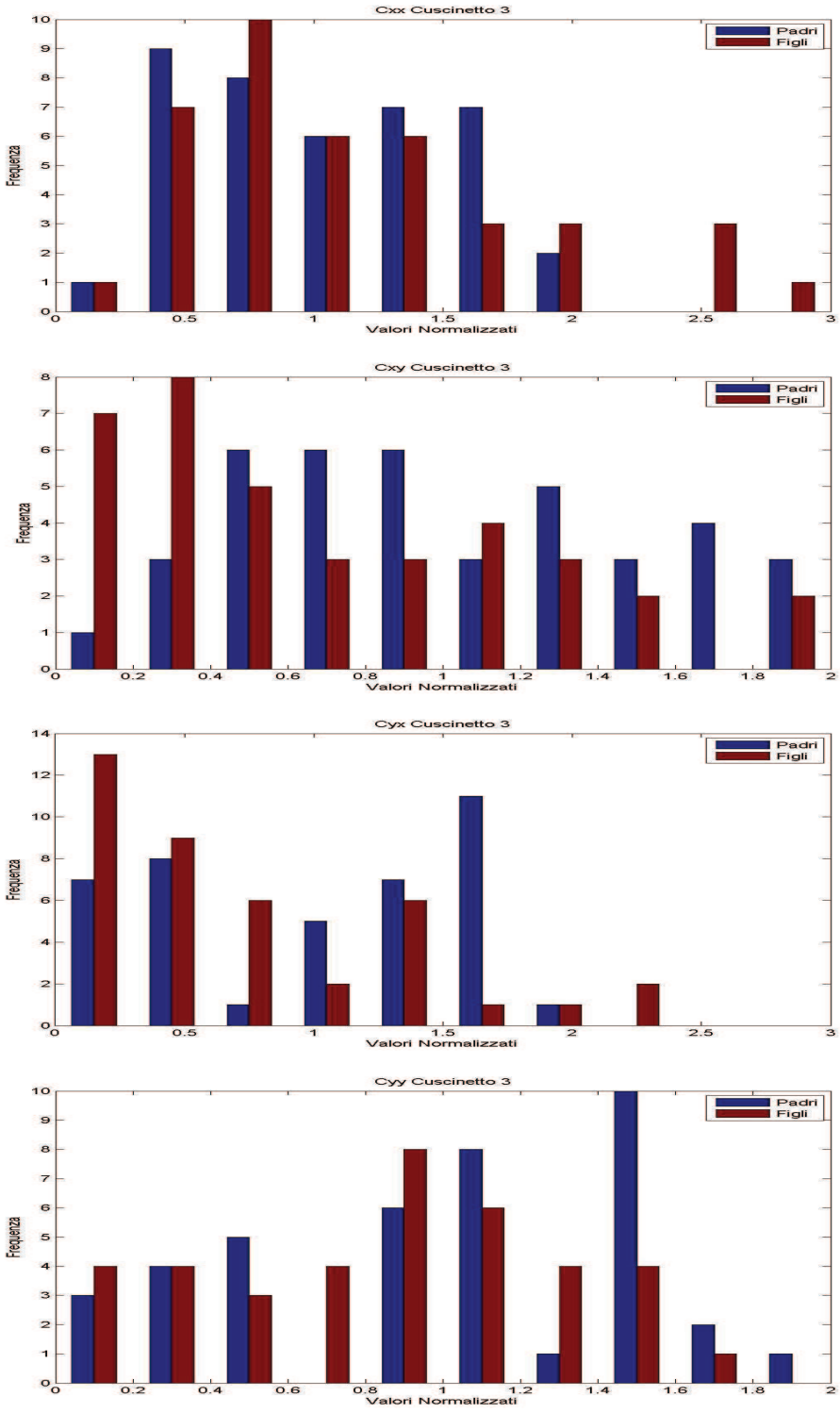


Figura 3.41 – Convergenza per C_{xx}, C_{xy}, C_{yx}, C_{yy} del cuscinetto 3

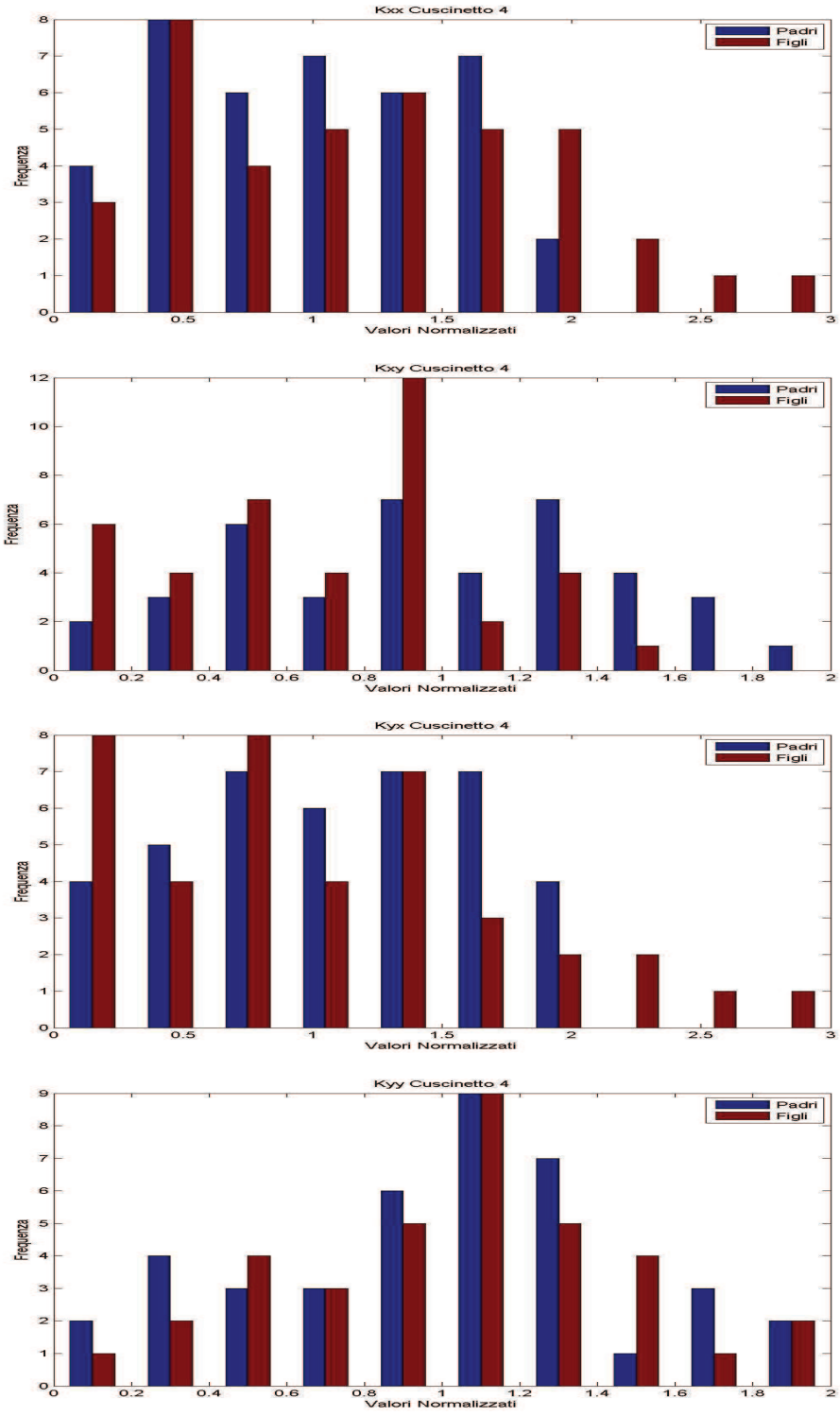


Figura 3.42 – Convergenza per K_{xx}, K_{xy}, K_{yx}, K_{yy} del cuscinetto 4

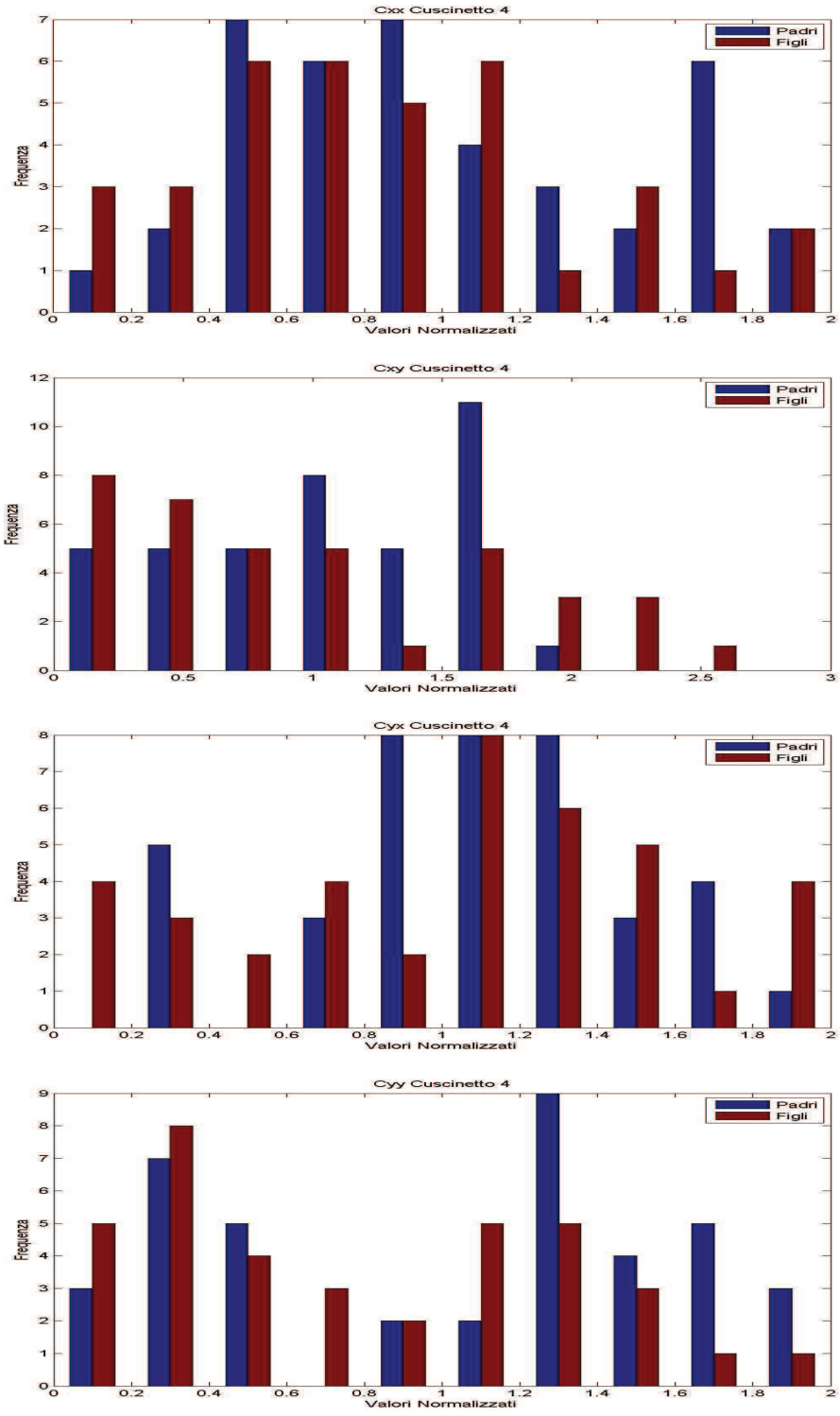


Figura 3.43 – Convergenza per C_{xx} , C_{xy} , C_{yx} , C_{yy} del cuscinetto 4

Per esempio, per i coefficienti K_{yx} del cuscinetto 1 (Figure 3.36), K_{xy}, C_{xx}, C_{yx}, C_{yy} del cuscinetto 2 (Figure 3.38 e 3.39), K_{xx} e C_{yy} del cuscinetto 4 (Figure 3.42 e 3.43) osserviamo una ripartizione uniforme dei valori ottenuti, anche se nel primo caso i valori convergono ad un valore vicino a quello iniziale. Questo significa che il valore iniziale è molto correlato al valore finale e che quindi, per una ripartizione iniziale dei padri, abbiamo una ripartizione uniforme dei figli sullo stesso intervallo. Fisicamente questo significa che questi parametri influiscono poco sulla risposta del modello. Infine altri parametri convergono allo stesso valore sia nel primo caso che nel secondo come C_{xx} del cuscinetto 1 (Figura 3.37) che converge intorno al valore normalizzato 0.5. Stessa cosa accade per i coefficienti K_{xx}, K_{yy}, C_{xx} e C_{yy} del cuscinetto 3 (Figure 3.40 e 3.41). Possiamo quindi considerare questi coefficienti ai quali possiamo aggiungere il coefficiente K_{xy} del cuscinetto 2 come quelli sensibili del sistema.

3.5.3 Conclusioni sulla convergenza

Alla luce di queste prime analisi possiamo considerare che l'ottimizzazione di ciascun parametro ha un'influenza diversa sull'avvicinamento della risposta del modello alla risposta misurata. Il modello è quasi indifferente alla variazione di certi parametri e invece molto influenzato alla variazione degli altri. Può essere interessante concentrarsi su quelli che permettono un vero miglioramento della risposta. Le condizioni iniziali determinano in parte il valore a convergenza e possono portare ad un minimo locale e non assoluto visto che lavoriamo a numero finito d'iterazioni. E' importante, per avere la miglior modellazione possibile, non limitarsi a una sola prova dell'updating. E' interessante constatare per esempio che, se si lancia l'updating solo sui 5 parametri descritti come sensibili, si procede ad una riduzione importante dell'errore mentre nel caso di parametri identificati come non sensibili osserviamo una riduzione molto più debole dell'errore (Figura 3.44).

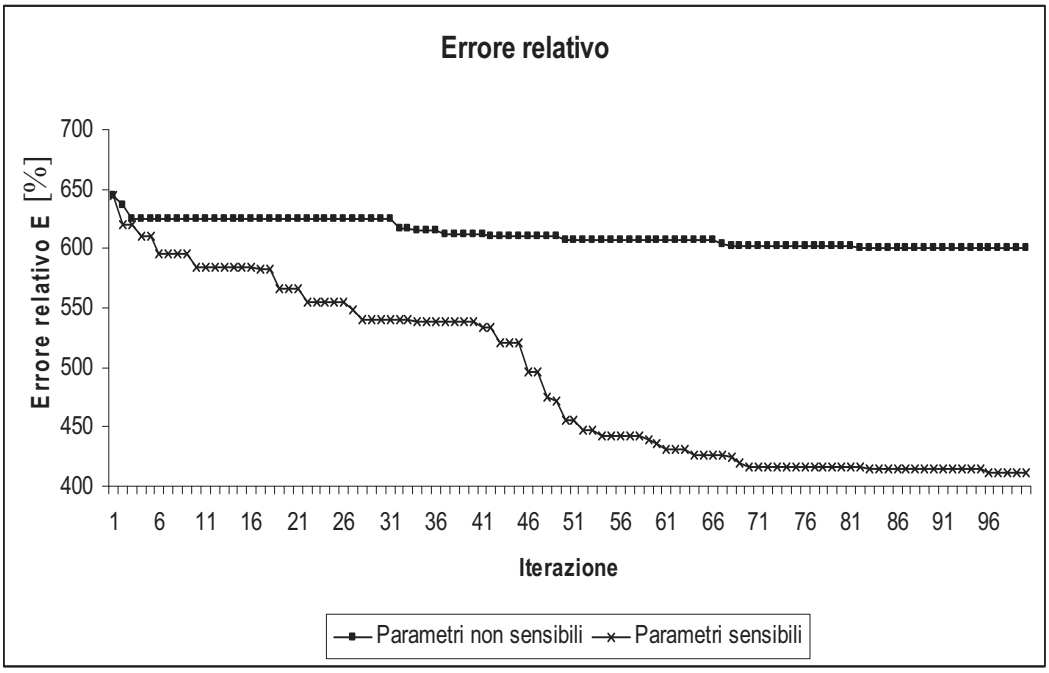


Figura 3.44 – Comparativo di ottimizzazione con parametri non sensibili e sensibili

IV. Model Updating di una macchina reale

4.1 Modello agli elementi finiti e parametrizzazione

Dopo avere studiato un caso teorico ci interesseremo al caso di una macchina reale, ovvero di una turbina a vapore per la produzione di energia. La modellazione sarà molto più complessa che nel caso precedente. Il file che descrive la struttura della macchina è composto questa volta da 167 elementi, sette cuscinetti nei nodi 4, 61, 78, 125, 136, 153, 160. I parametri da ottimizzare diventano 56. I calcoli dell'algoritmo saranno più lunghi visto che le matrici saranno di dimensioni molto maggiori. Un'altra differenza rispetto al modello teorico è la presenza di una fondazione pedestal invece di una fondazione modale. Ogni pedestal è modellato con otto coefficienti, quattro di rigidezza e quattro di smorzamento, alla maniera dei cuscinetti, che dipendono della velocità e che saranno quindi valutati a certe velocità prescelte. Il rotore disegnato è rappresentato in Figura 4.2. Lo studio dell'updating verrà effettuato come nel caso precedente come uno studio della convergenza dell'algoritmo. Abbiamo, come operazione preliminare, realizzato l'updating con i coefficienti forniti inizialmente per vedere di quanto si può migliorare il modello e per cercare un limite all'ottimizzazione (Figura 4.1).

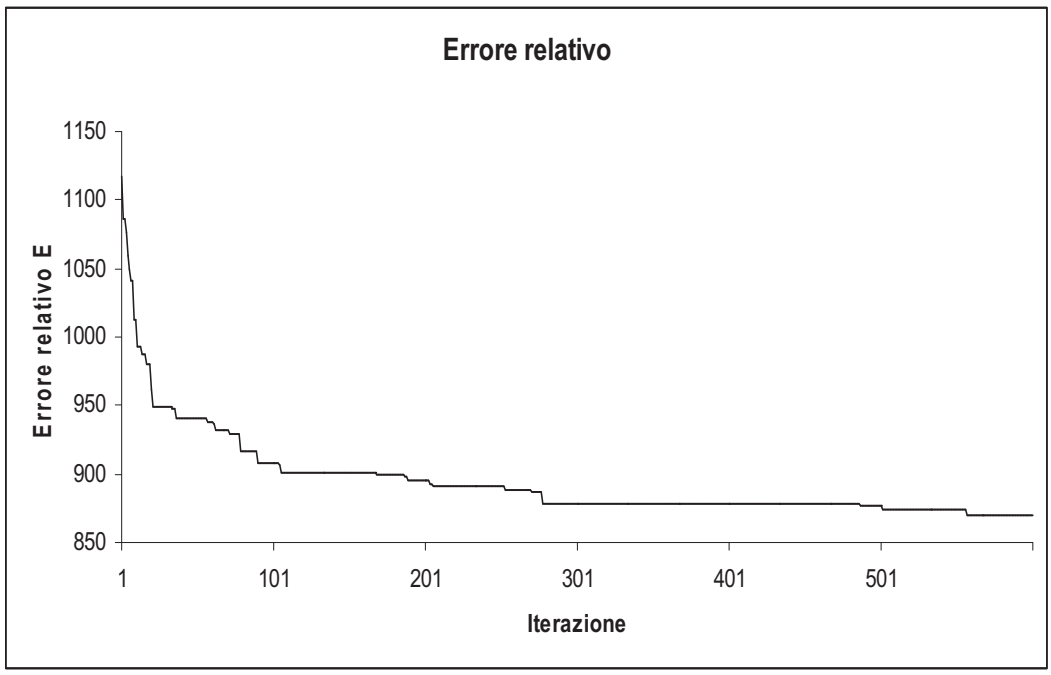


Figura 4.1 Errore relativo per una ottimizzazione della macchina reale

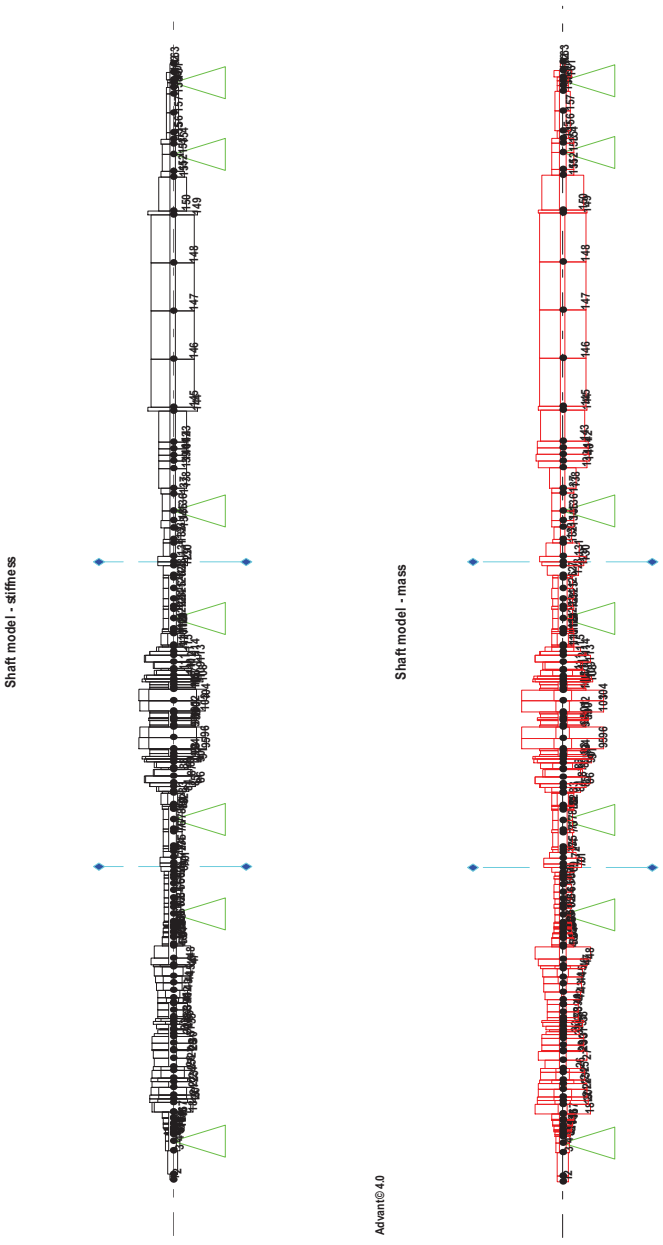


Figura 4.2 - Rappresentazione geometrica e di massa della macchina a vapore

4.2 Analisi di sensibilità del modello

In questo capitolo studieremo, come nel precedente, l'influenza dei parametri sul modello matematico del turbo-generatore. Dopo avere effettuato una serie di test di sensibilità, abbiamo scelto di presentare i risultati relativi al quarto cuscinetto perché sono i più significativi. Alla luce dello studio precedente, vediamo adesso che l'ottimizzazione avrà probabilmente un effetto maggiore su un cuscinetto rispetto agli altri. Analizzeremo ancora i quattro parametri: $k_{yy}, k_{yx}, c_{yy}, c_{yx}$.

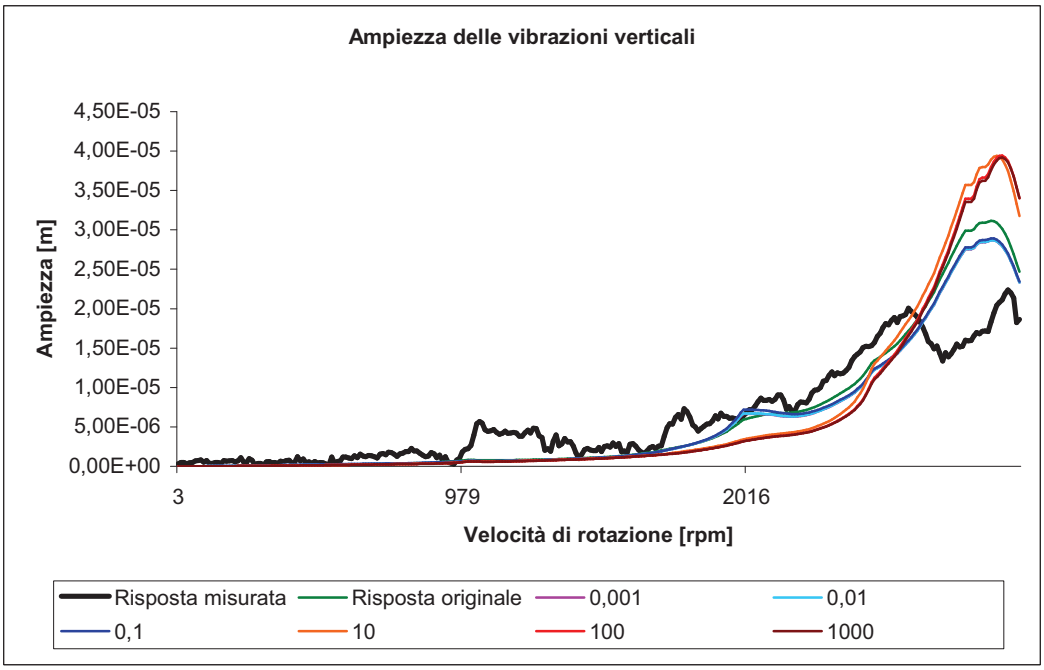
Il procedimento sperimentale sarà identico, ogni parametro sarà moltiplicato successivamente per 0.001, 0.01, 0.1, 10, 100, 1000 per generare dei cuscinetti diversi. Confronteremo in seguito le vibrazioni dovute al forzamento di questi modelli diversi al fine di capire l'influenza di tali parametri sulle vibrazioni. I grafici analizzati rappresentano ogni volta ampiezza e fase delle vibrazioni verticali e orizzontali.

4.2.1 Coefficiente di rigidezza k_{xx}

Come primo risultato possiamo vedere che i modelli reagiscono nello stesso modo per le basse velocità, infatti le curve sono veramente sovrapposte (Figura 4.3).

Per quanto riguarda le alte velocità, notiamo che l'ampiezza diminuisce all'aumentare della rigidezza e questo accade fino ad una velocità intorno ai 2600 rpm, valore per cui vediamo un'inversione delle curve e il modello più rigido manifesta le vibrazioni maggiori. Queste variazioni si notano meno per le vibrazioni orizzontali. C'è un leggero aumento delle vibrazioni, sempre per alte velocità, nel caso di un modello con alto coefficiente k_{xx}. In entrambi i casi si può dire che l'andamento è abbastanza rispettato.

Notiamo che le variazioni sulla fase sono debolmente visibili (Figura 4.4). Aumentare la rigidezza permette di alzare le curve di fase di circa 25° senza modificare sostanzialmente l'andamento. Le curve sono decrescenti come lo sono globalmente le curve misurate, anche se il range di variazioni è molto più stretto nel caso modellato. Abbiamo indicato per le vibrazioni orizzontali come cambiassero le risposte per variazioni del valore del parametro nel caso in cui ci si allontanasse dal cuscinetto 4 (Figura 4.5). Essendo le curve esattamente sovrapposte, possiamo asserire che l'influenza di un parametro si risente per lo più localmente. In questo caso, rispetto al caso del *test_rig* visto nel capitolo precedente, la distanza tra il cuscinetto modificato e il luogo delle misure è molto più evidente. Abbiamo scelto di rappresentare solo le vibrazioni orizzontali perché la situazione per le vibrazioni verticali è simile.



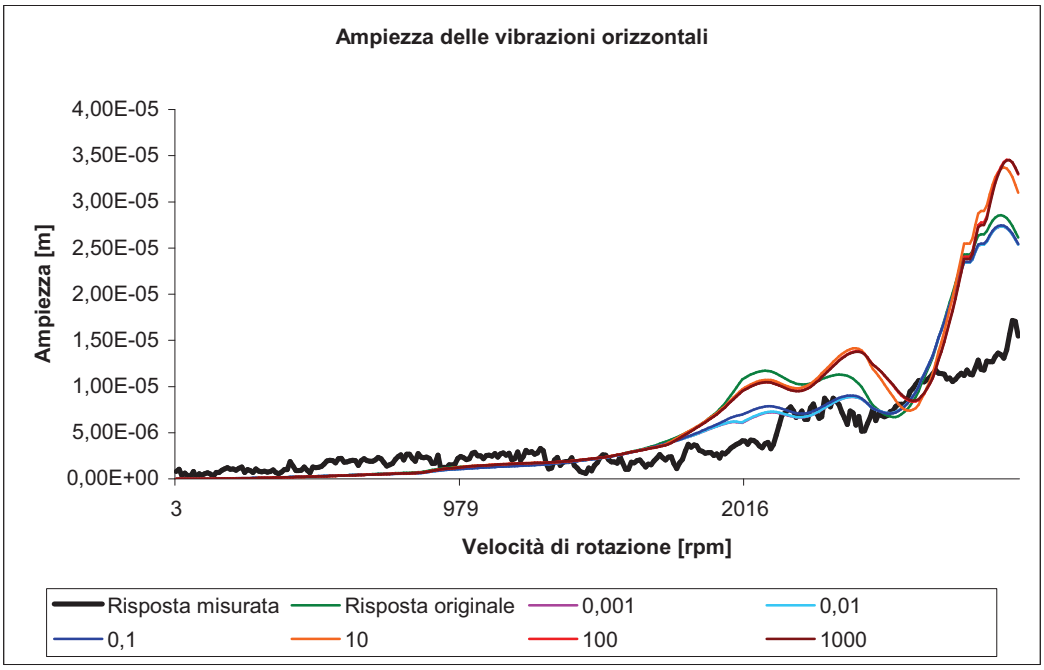
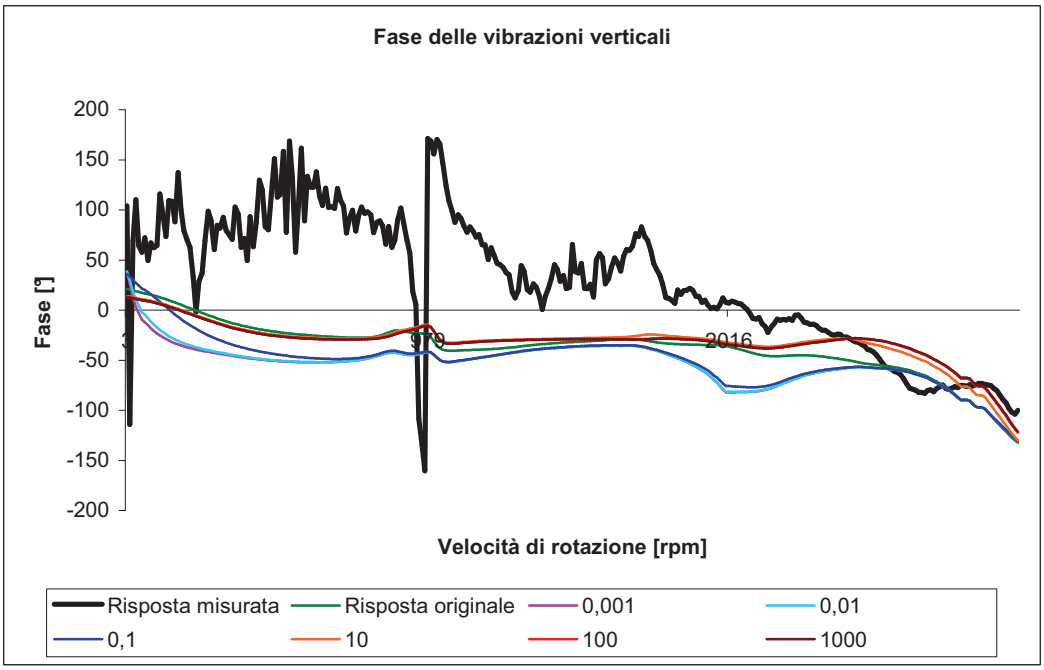


Figura 4.3 – Influenza sull'ampiezza delle vibrazioni verticali e orizzontali del Nodo 125 della variazione di k_{xx}



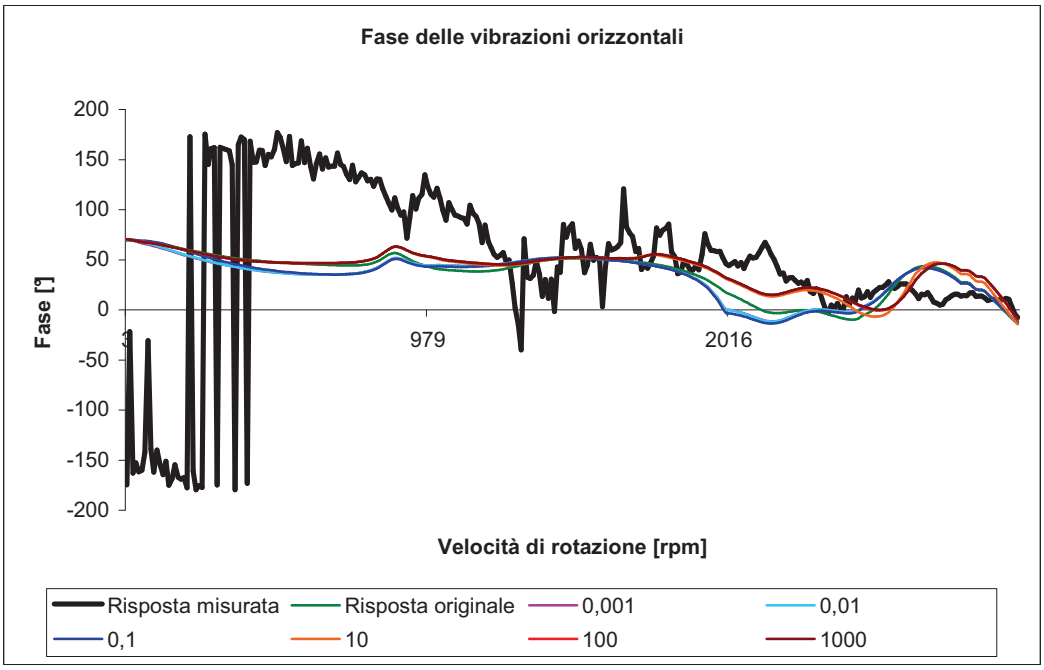
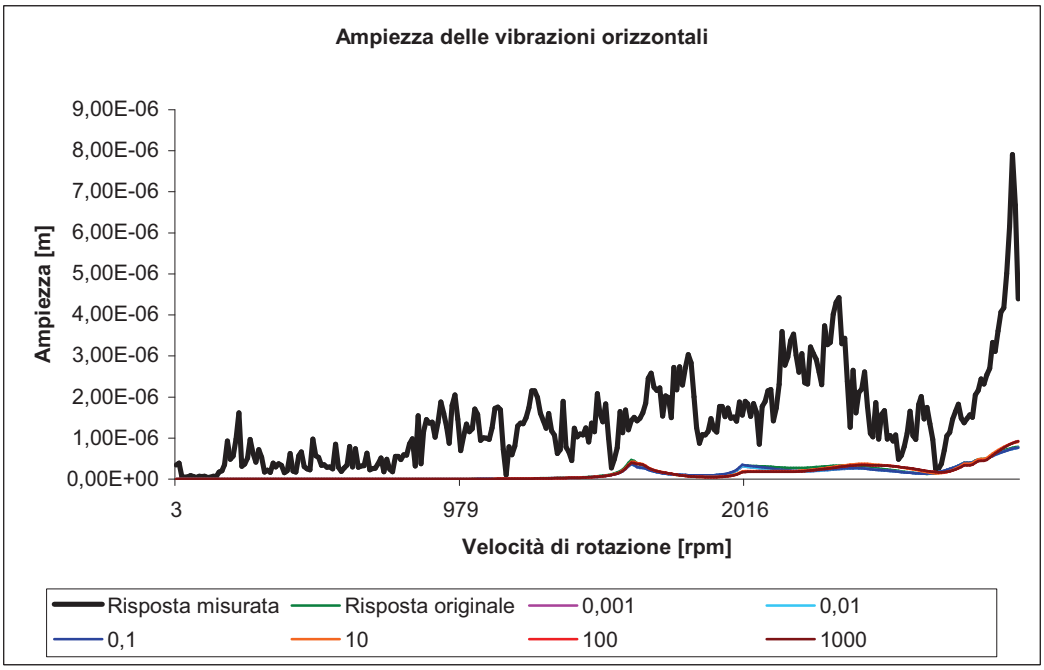


Figura 4.4 – Influenza sulla fase delle vibrazioni verticali e orizzontali del Nodo 125 della variazione di k_{xx}



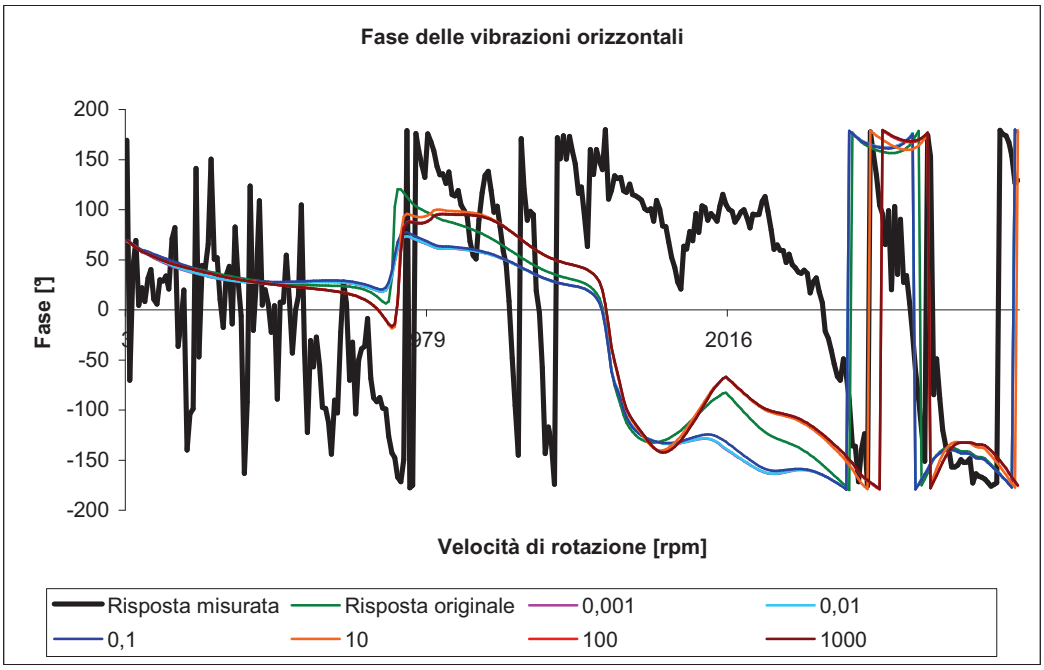


Figura 4.5 – Influenza sull'ampiezza e fase delle vibrazioni orizzontali del Nodo 4 della variazione di k_{xx}

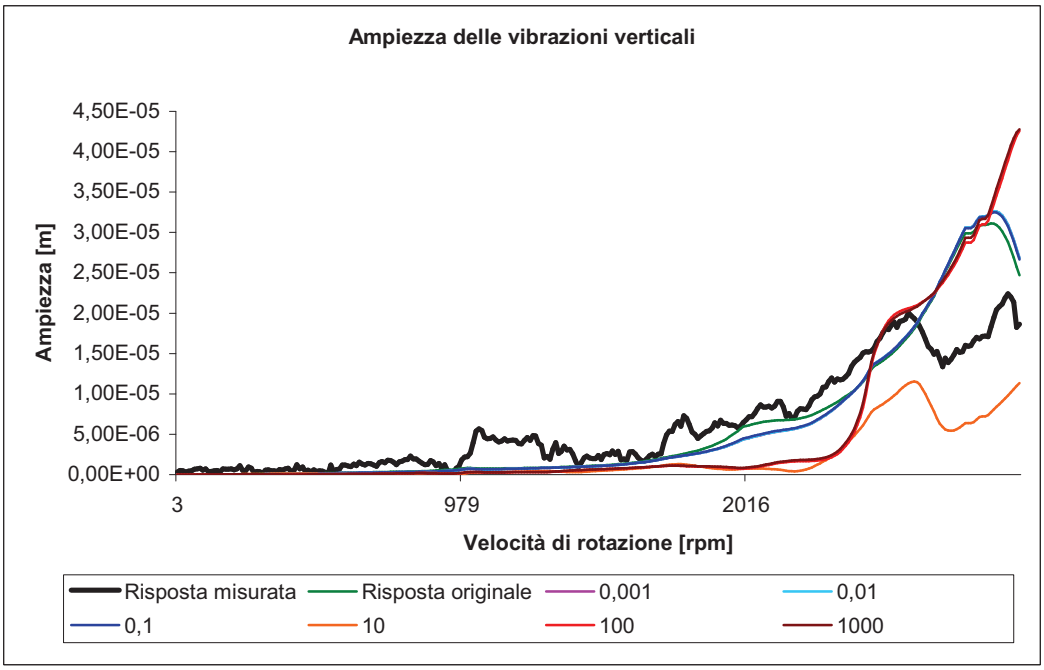
4.2.2 Coefficiente di rigidezza k_{xy}

Anche in questo caso vediamo che la separazione delle curve avviene per le alte velocità, circa 1700 rpm nel caso delle vibrazione verticali e circa 2300 rpm nel caso delle vibrazione orizzontali (Figura 4.6).

Osserviamo due famiglie di curve, quelle che sono moltiplicate per un coefficiente minore di 1 e quelle moltiplicate per uno maggiore di 1. Le prime hanno degli andamenti molto simili e abbastanza coerenti con le misure, le altre invece hanno un'ampiezza molto inferiore alla risposta misurata fino alla velocità di 2300 rpm e crescono fortemente fino a superare di tanto le misure dopo la velocità indicata. Questo fenomeno si ritrova nella variazione dell'ampiezza delle vibrazioni orizzontali. Per valori dei coefficienti molto piccoli, le curve sono raggruppate. Per valori superiori a uno, l'ampiezza della risposta cresce in maniera significativa. Il caso del coefficiente moltiplicato per 10 è diverso rispetto agli altri, e ciò potrebbe significare che questo sia un valore critico.

Questo punto di vista viene confermato dalle curve che rappresentano le fasi delle vibrazioni verticali (Figura 4.7). Il caso moltiplicato per 10 è molto diverso dagli altri e corrisponde ben poco alla misura effettuata. Ritroviamo anche in questo caso le due famiglie di curve, quelle per coefficienti minori di 1 e quelle per coefficienti maggiore di 1 che mostrano degli andamenti simili per un certo intervallo di velocità, ma si separano per velocità molto alte o molto basse.

Come per il coefficiente precedente la distanza annulla completamente l'effetto della variazione locale del modello come si può costatare sulle curve d'ampiezza e di fase delle vibrazioni orizzontali (Figura 4.8).



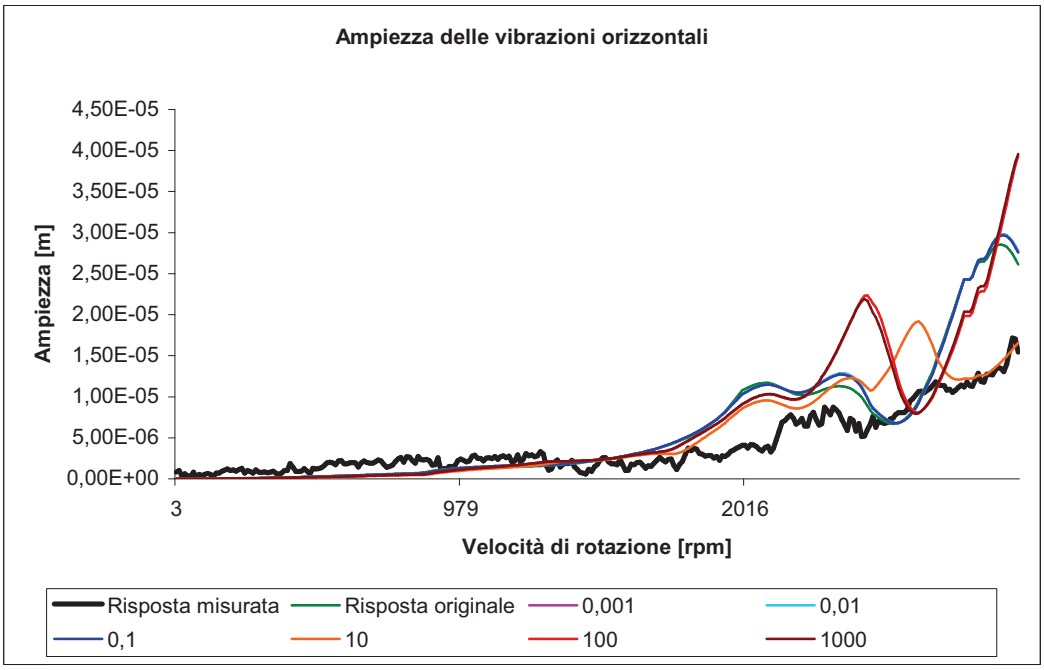
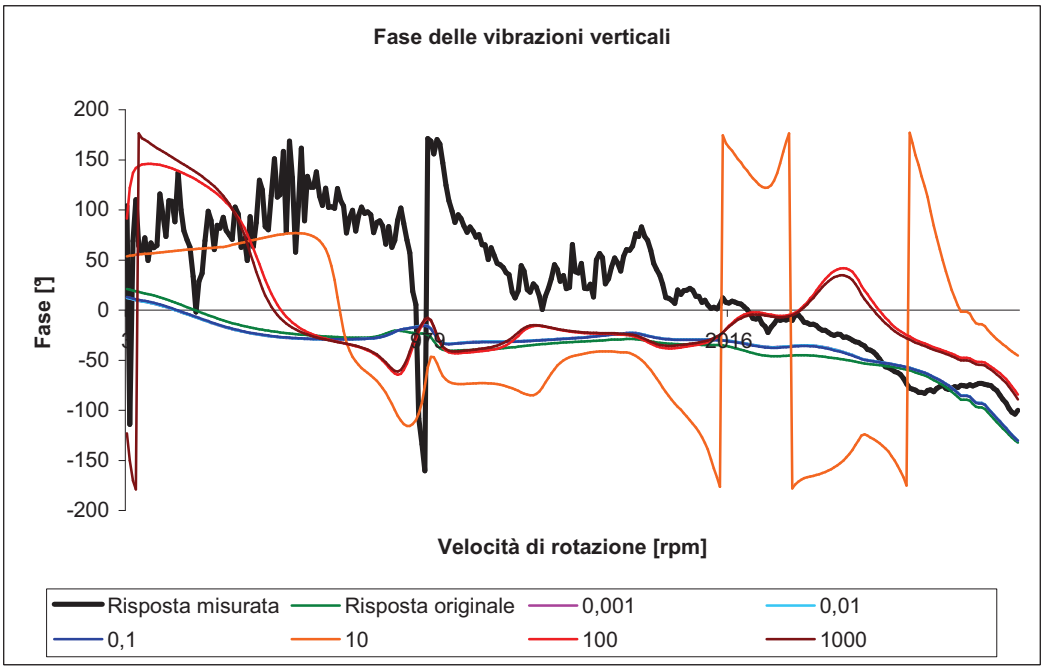


Figura 4.6 – Influenza sull'ampiezza delle vibrazioni verticali e orizzontali del Nodo 125 della variazione di k_{xy}



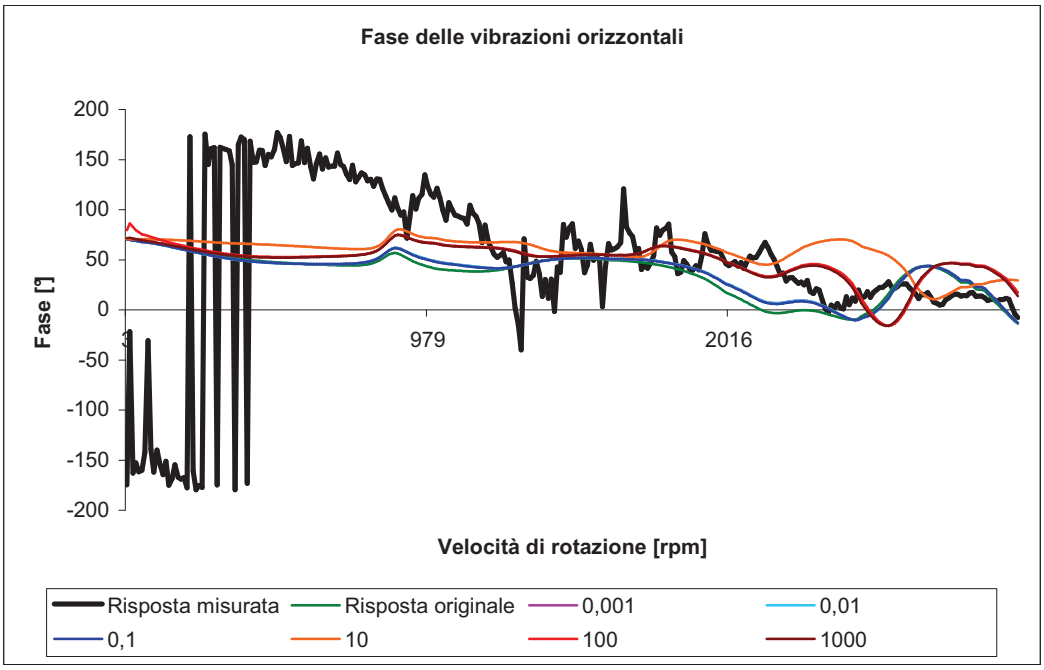
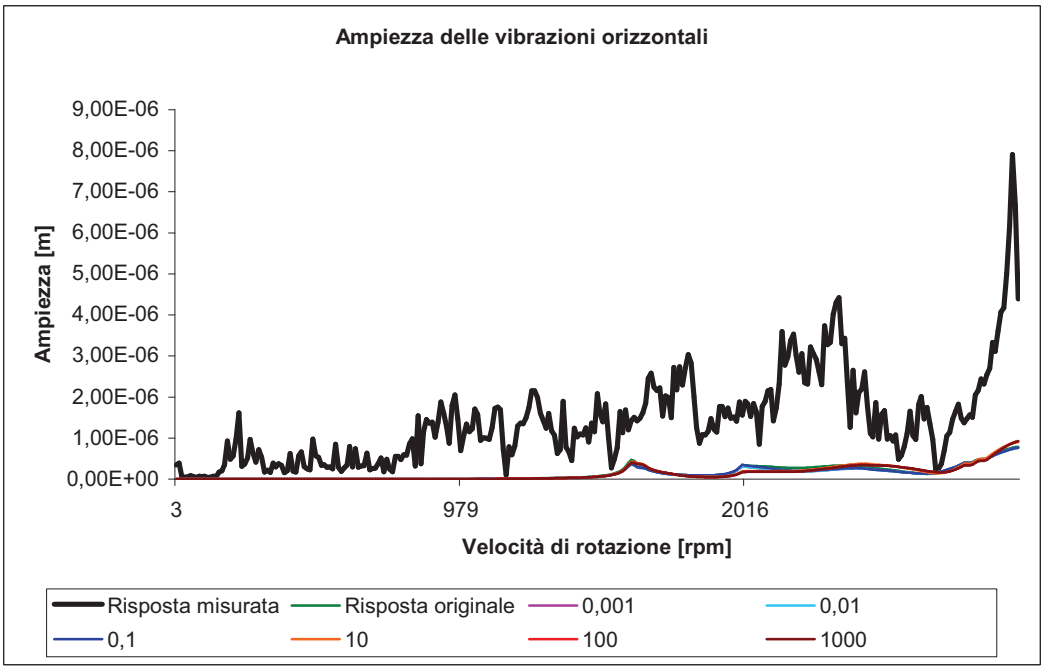


Figura 4.7 – Influenza sulla fase delle vibrazioni verticali e orizzontali del Nodo 125 della variazione di k_{xy}



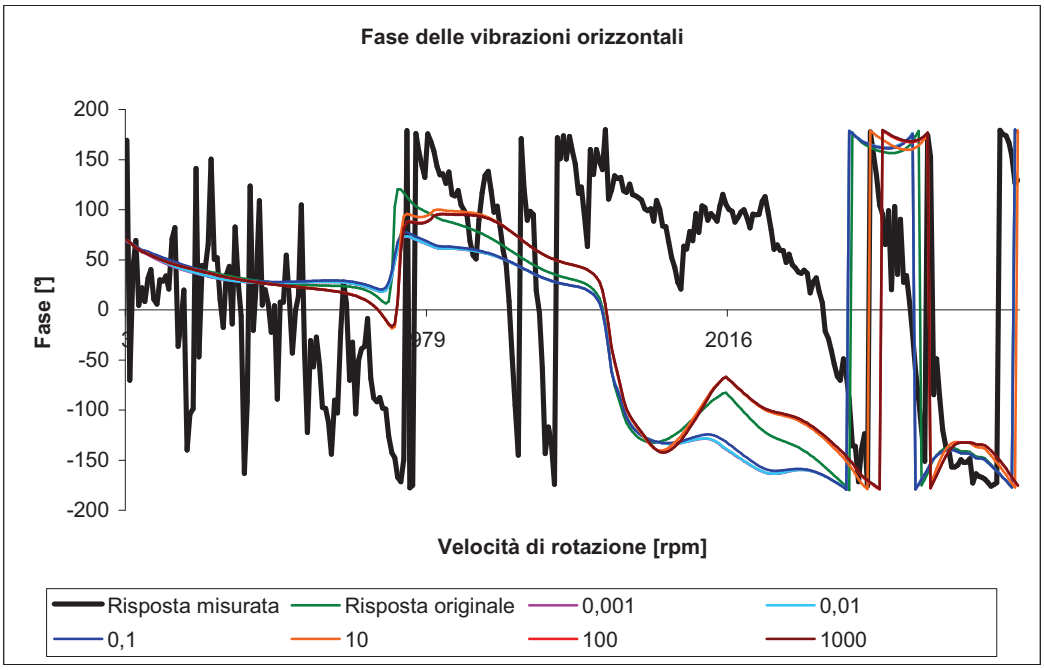


Figura 4.8 – Influenza sull'ampiezza el fase delle vibrazioni orizzontali del Nodo 4 della variazione di k_{xy}

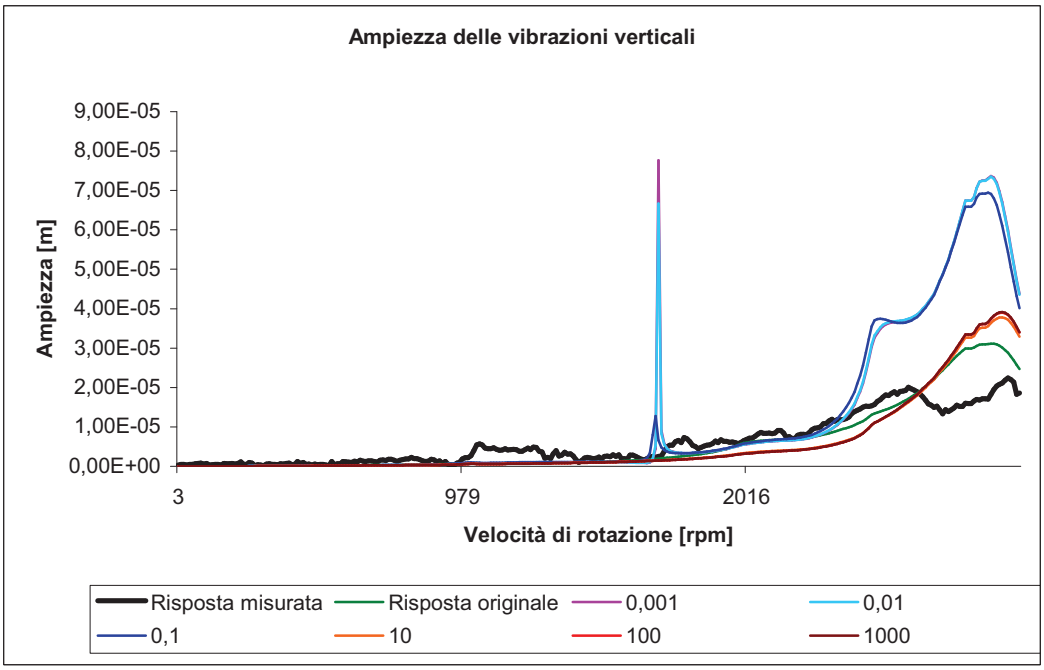
4.2.3 Coefficiente di smorzamento c_{xx}

Questo caso è di grande interesse per lo studio della macchina reale. Vediamo grandi variazioni tra le diverse curve, il che significa che il modello è molto sensibile rispetto a questo parametro (Figura 4.9). Per un valore alto del coefficiente le curve sono sovrapposte. La moltiplicazione del parametro che permette di avvicinare meglio le misure effettuate è quella unitaria. Questo significa che, probabilmente, il valore della convergenza sarà di quest'ordine di grandezza. Per piccoli valori del coefficiente di smorzamento le curve sono veramente lontane dalle curve misurate.

E' interessante notare l'introduzione di una risonanza intorno ai 1800 rpm sia per le vibrazioni verticali che per le vibrazioni orizzontali. Ma questo costituisce un difetto importante della modellazione perché spinge l'utente a usare precauzioni eccessive quando lavora intorno a questa velocità, visto che le misure reali non indicano che questa risonanza esiste.

Le curve di fase confermano che la modellazione, per piccoli valori dello smorzamento, è inadeguata (Figura 4.10). Le variazioni permettono soltanto un leggero spostamento verticale della fase.

L'aumento della distanza riduce l'effetto di queste variazioni, anche se ritroviamo la risonanza intorno alla velocità di 1800 rpm (Figura 4.11). Da tutte queste considerazioni possiamo concludere che questo parametro avrà un ruolo importante nell'updating e le considerazioni precedenti del *test_rig* ci spingono a pensare che la sua convergenza sarà notevole.



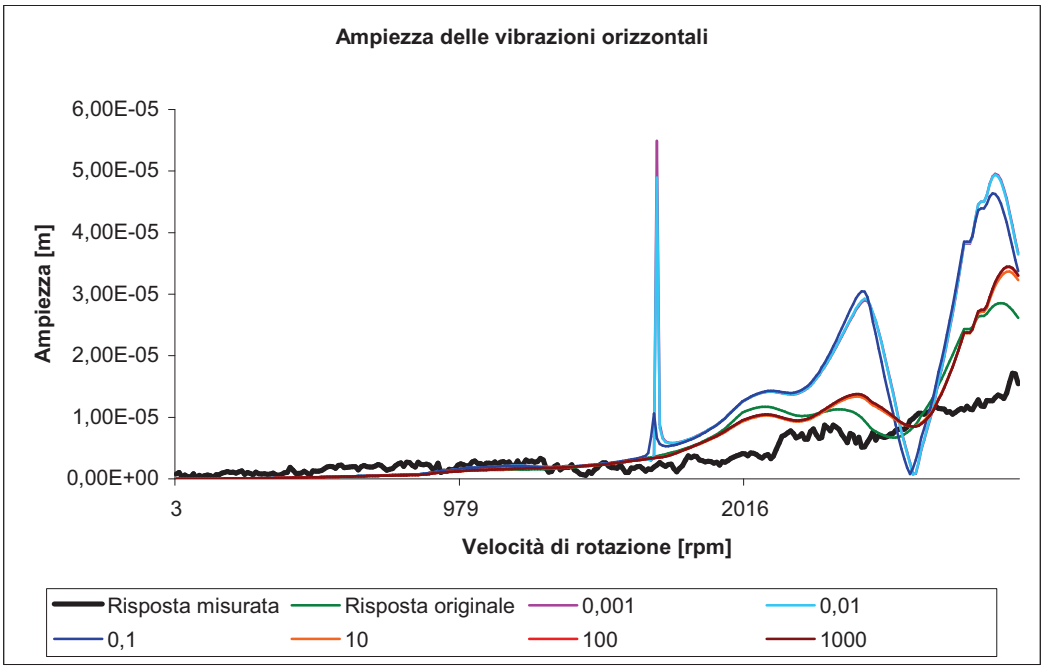
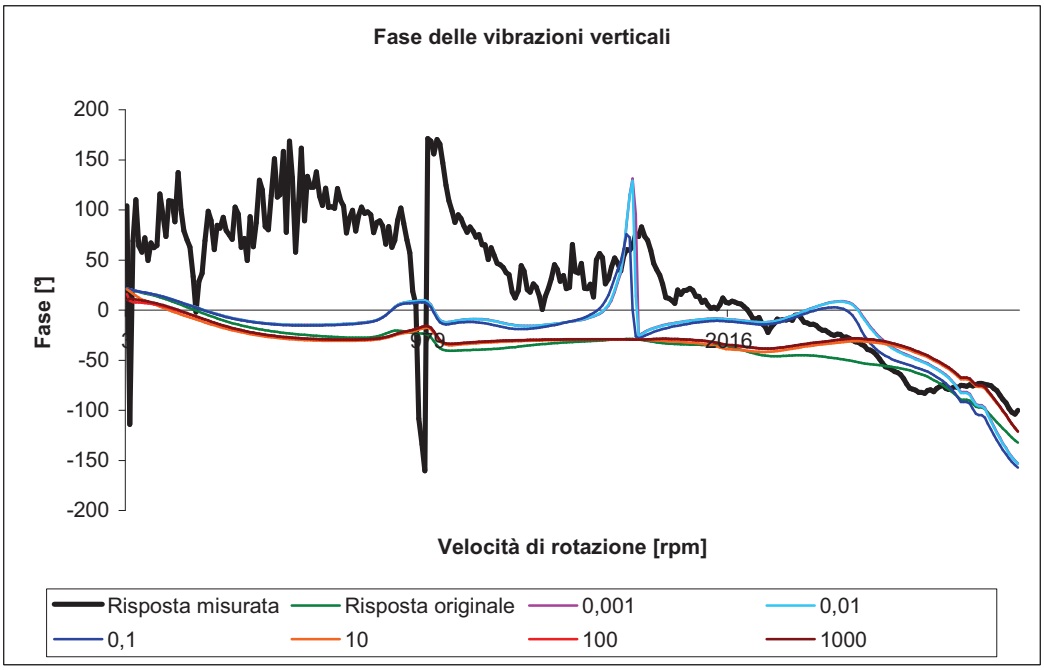


Figura 4.9 – Influenza sull'ampiezza delle vibrazioni verticali e orizzontali del Nodo 125 della variazione di c_{xx}



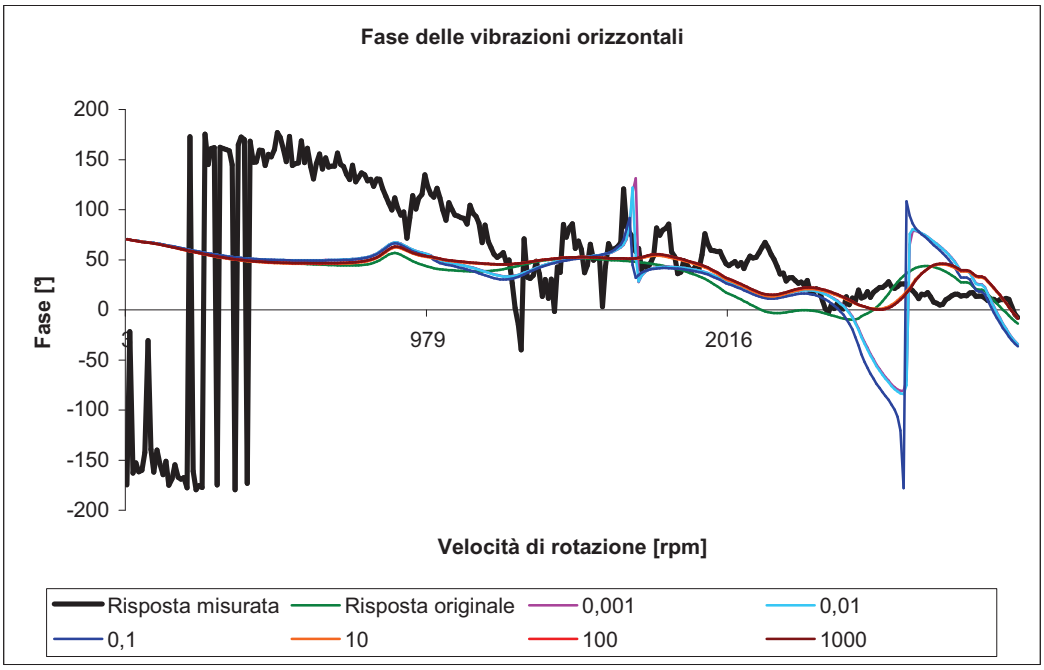
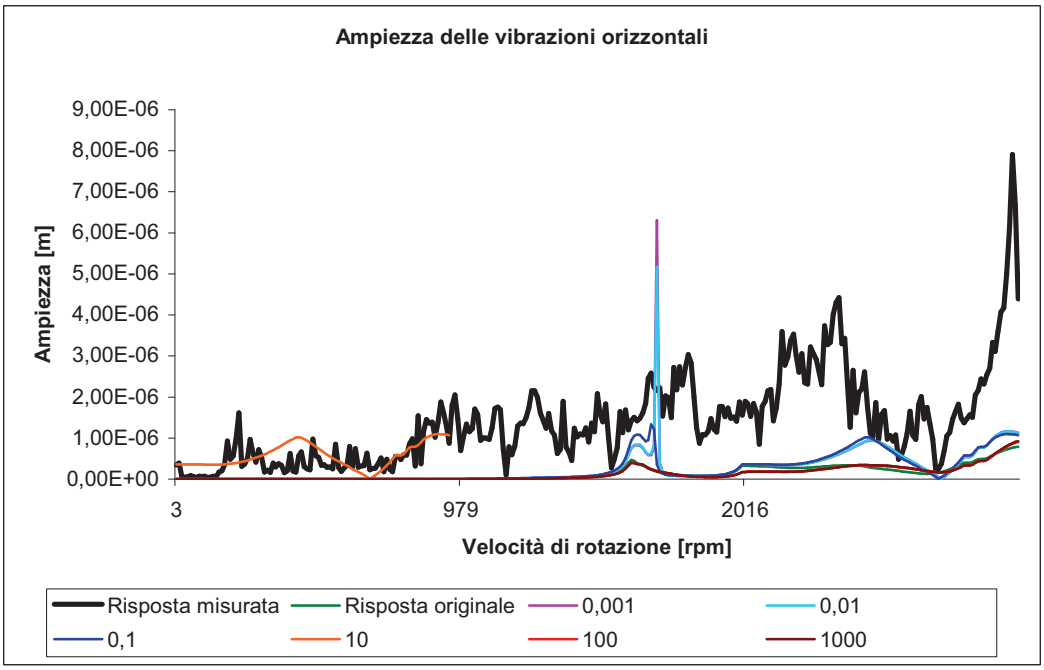


Figura 4.10 – Influenza sulla fase delle vibrazioni verticali e orizzontali del Nodo 125 della variazione di c_{xx}



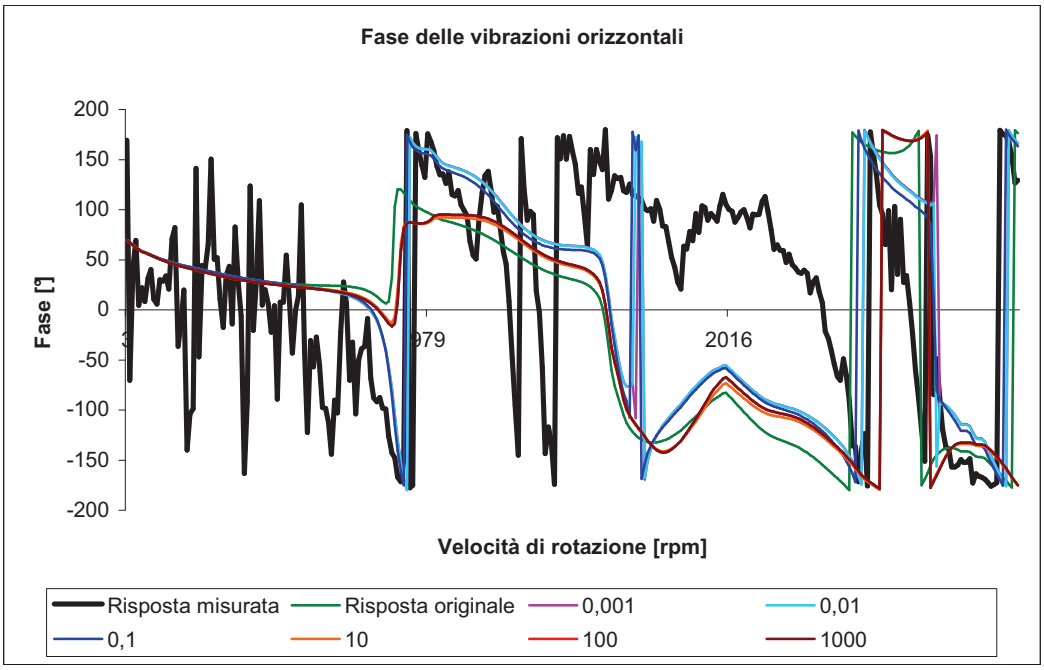


Figura 4.11 – Influenza sull'ampiezza e la fase delle vibrazioni orizzontali del Nodo 4 della variazione di c_{xx}

4.2.4 Coefficiente di smorzamento c_{xy}

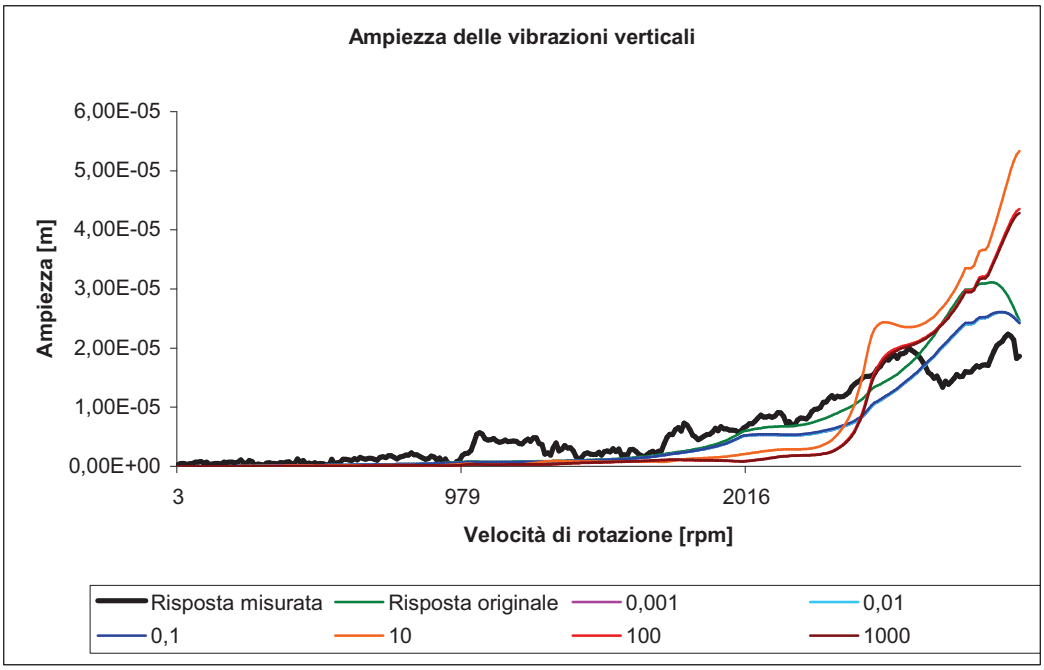
In questo caso piccoli valori dello smorzamento permettono di riprodurre più fedelmente l'ampiezza delle vibrazioni verticali e orizzontali misurate (Figura 4.12). Per uno smorzamento troppo alto, nel caso di basse velocità abbiamo un'ampiezza troppo limitata e viceversa per le alte velocità.

Le curve di fase si spostano verticalmente quando modifichiamo il valore di questo coefficiente (Figura 4.13). Anche in questo caso, la modellazione migliore è ottenuta per valori bassi del coefficiente che segue la pendenza decrescente abbastanza regolare della fase misurata.

La distanza tra il cuscinetto modificato e la vibrazione studiata, come negli altri casi, riduce tanto l'influenza del parametro (Figura 4.14) perché più lontano dal punto in cui è stata apportata la modifica effettuiamo la misura minore sarà l'effetto sentito. Possiamo notare comunque che la fase subisce delle grandi variazioni e che si generano due famiglie di curve a seconda che il coefficiente moltiplicativo sia maggiore o minore di 1. La risposta migliore si ottiene per la famiglia di curve con bassi valori del coefficiente c_{xy}.

4.2.5 Conclusioni sulla sensibilità della macchina reale

Confrontando questa analisi con i risultati del *test_rig* vediamo che qui la sensibilità dei parametri è minore, ovvero un solo parametro non può portare a delle variazioni significative tranne che in alcuni casi particolari come, ad esempio, nel caso del parametro c_{xx} del cuscinetto 4. Nel caso della macchina reale, la distanza gioca un ruolo maggiore per la riduzione dell'influenza di tale variazione. In effetti, in quasi tutti i casi, l'influenza sul nodo 4 di un parametro modificato nel nodo 160 è stata quasi nulla. Un punto comune con l'analisi precedente è che l'aumentare o il diminuire di più di due ordini di grandezza di ciascun parametro del modello rispetto ai valori iniziali dei cuscinetti forniti non porta più ad altre modifiche delle risposte e che quindi il range da scegliere per il model updating della macchina reale deve essere compreso nell'intervallo [0.01,100].



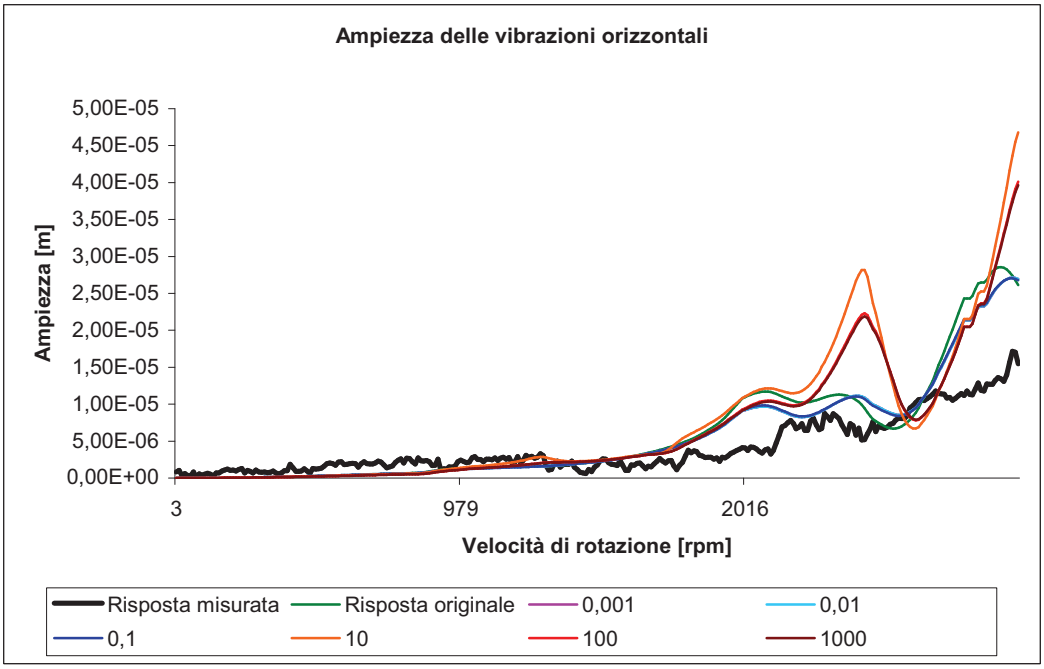
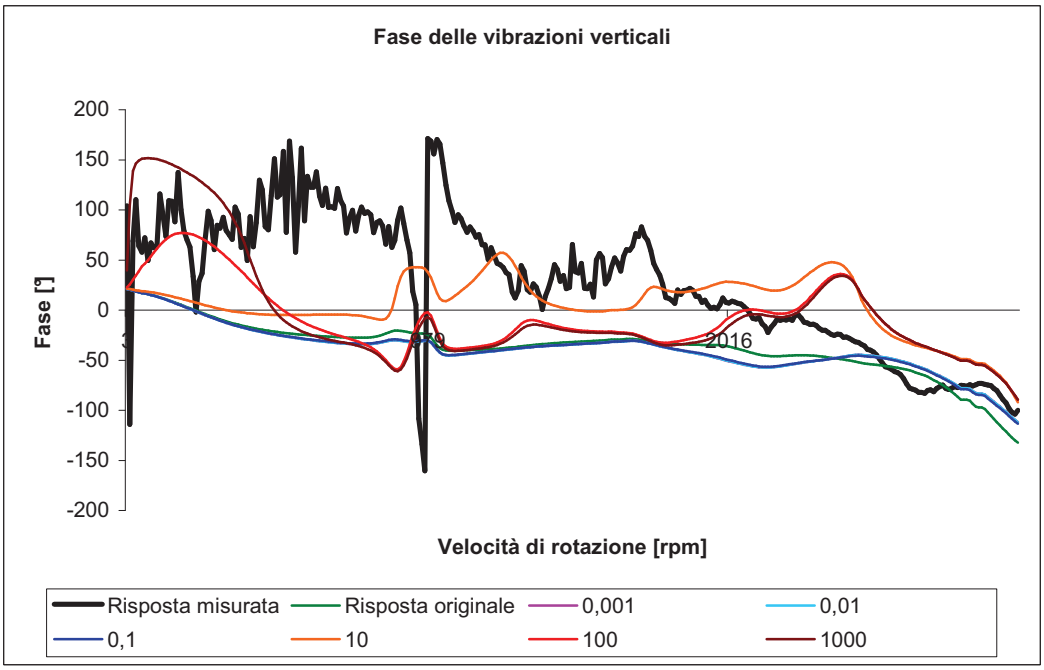


Figura 4.12 – Influenza sull'ampiezza delle vibrazioni verticali e orizzontali del Nodo 125 della variazione di c_{xy}



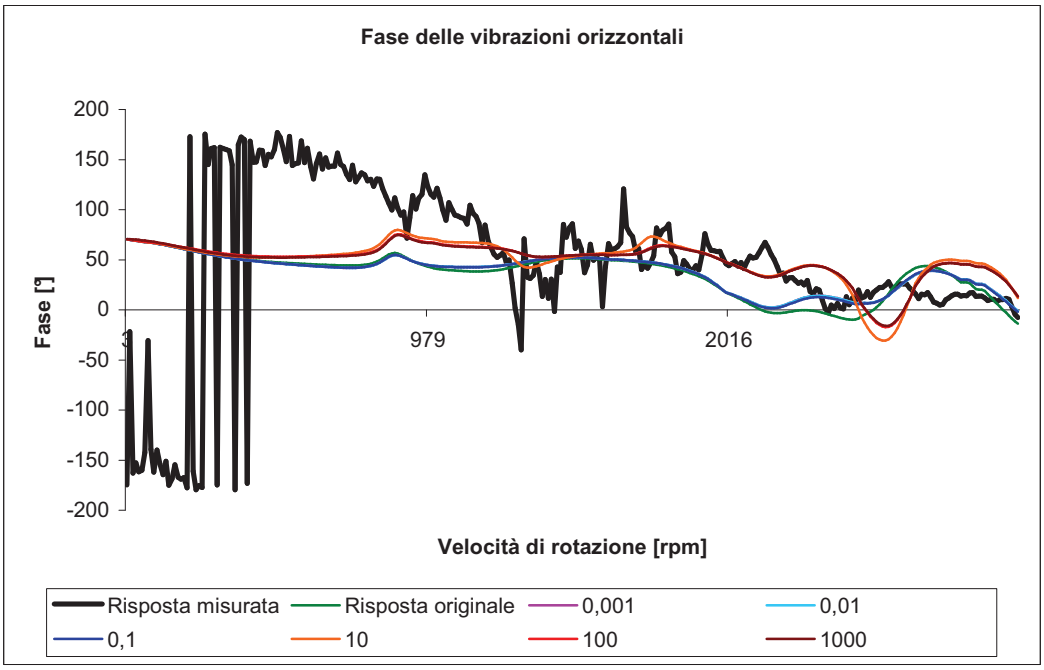
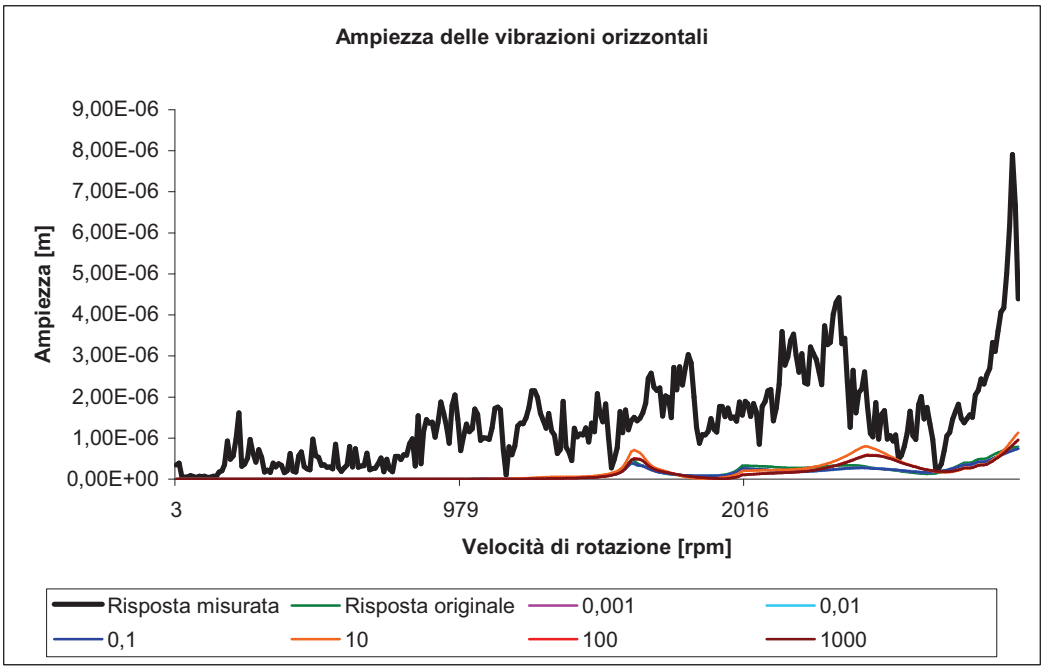


Figura 4.13 – Influenza sulla fase delle vibrazioni verticali e orizzontali del Nodo 125 della variazione di c_{xy}



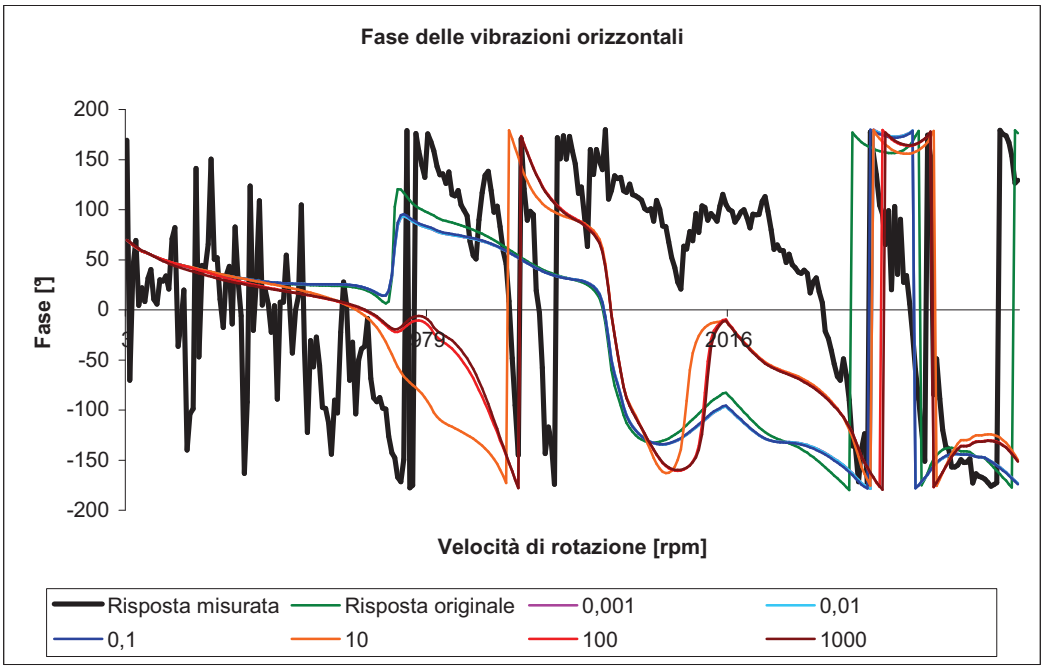


Figura 4.14 – Influenza sull'ampiezza e la fase delle vibrazioni verticali e orizzontali del Nodo 4 della variazione di c_{xy}

4.3 Convergenza dell'algoritmo

Useremo in questo studio i risultati relativi all'algoritmo di ottimizzazione del capitolo 3. Condurremo quindi questo studio con la stessa configurazione. Generiamo 40 padri con una funzione Matlab simile a quella precedente, adattata però a questa modellazione (Appendice 4 : Funzione *GenMR*). Una volta generati questi 40 set di sette cuscinetti padri, utilizzeremo l'algoritmo con i parametri seguenti :

- 1 padre;
- 1 figlio;
- $\sigma = 0.085$.

Sceglieremo un numero di iterazioni di poco inferiore a quello precedente visto che per un miglioramento modesto della convergenza ci sarà un notevole aumento del tempo impiegato. (una grande perdita di tempo.) Una iterazione in una simulazione nel caso del *test_rig* era eseguita in circa 1 secondo. In questo caso, a causa delle dimensioni del modello, della fondazione e del numero più elevato di cuscinetti, il tempo di calcolo sale a 40 secondi per ciascuna iterazione. Facendo 50 iterazioni in meno il tempo di calcolo diminuisce di più di mezz'ora. Visto che dobbiamo effettuare 40 simulazione, risparmiamo più di vent'ore al diminuire il numero di iterazioni. Una volta ottenuti i 40 set di cuscinetti figli, dovremmo normalizzare, come nel caso precedente, rispetto ai cuscinetti originali e fare il confronto tra ripartizione dei padri e ripartizione dei figli. Avremo qua 56 parametri da analizzare per sette cuscinetti diversi. Certi coefficienti sono nulli, l'ottimizzazione non li cambierà visto che agisce come la moltiplicazione. Saranno rappresentati solo i coefficienti non nulli, 44 in totale. Per questo studio saranno utilizzate due funzioni. La prima permette di selezionare, per un parametro scelto, tutti i valori ottenuti sia dei padri sia dei figli e di normalizzarli rispetto al valore originale (Appendice 5 : Funzione *MatPFMR*). La seconda ripartisce i valori secondo una suddivisione dell'intervallo scelta dall'utente (Appendice 6: Funzione *RipartizioneMR*). Questo procedimento ci permette di ottenere i risultati grafici delle pagine seguenti (Figure 4.3 a 4.13).

Dopo un primo sguardo sui risultati vediamo che la convergenza è meno netta per la maggiore parte dei coefficienti in questo caso che nel caso precedente. Un gran numero di coefficienti verifica una dispersione uniforme dei risultati, possiamo prendere come esempio, K_{yy} e C_{yy} del cuscinetto 1 (Figura 4.15), C_{yy} del cuscinetto 3 (Figura 4.18), C_{xx} del cuscinetto 6 (Figura 4.24) e K_{yy} del cuscinetto 7 (Figura 4.25).

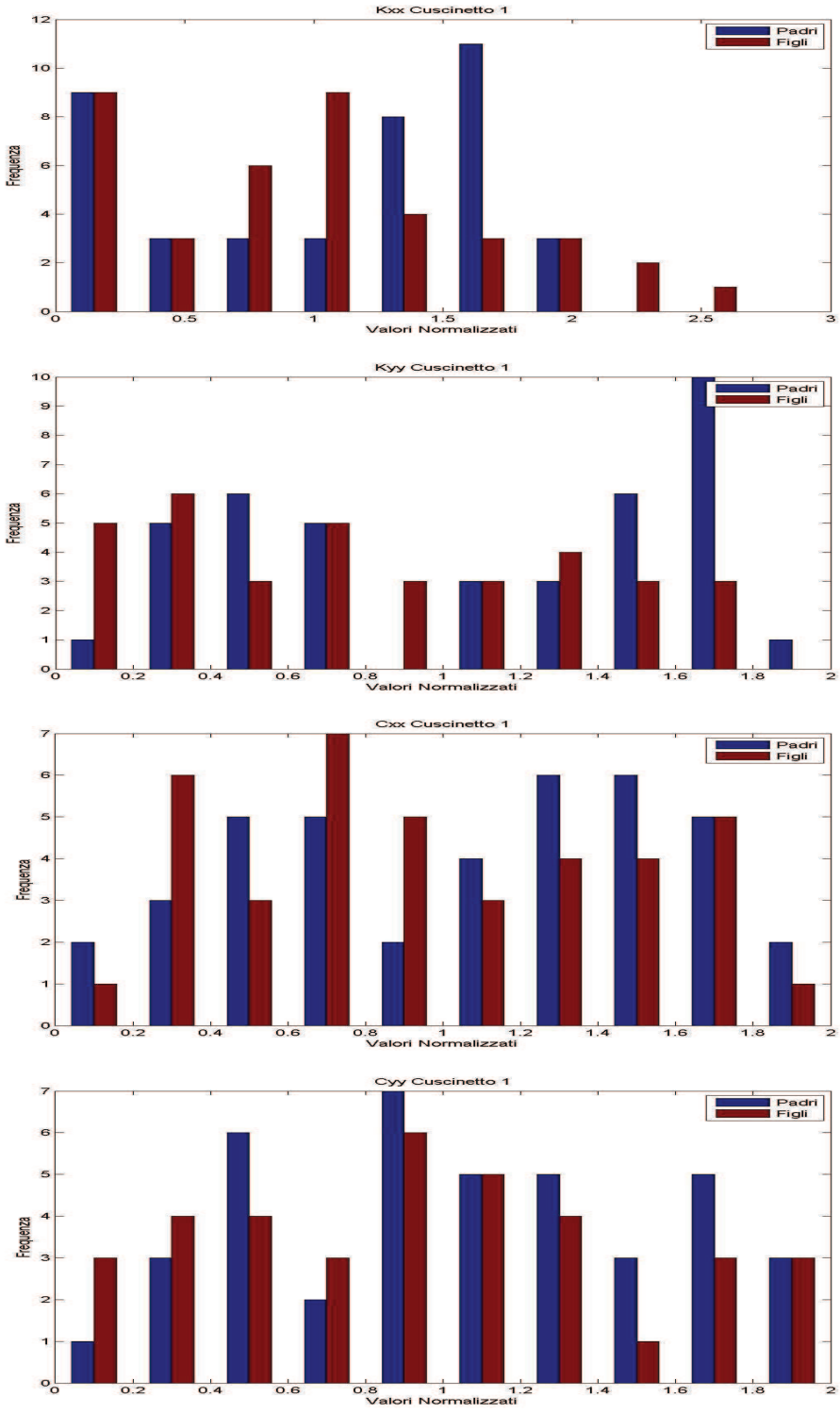


Figura 4.15 – Convergenza per K_{xx}, K_{yy}, C_{xx}, C_{yy} del cuscinetto 1

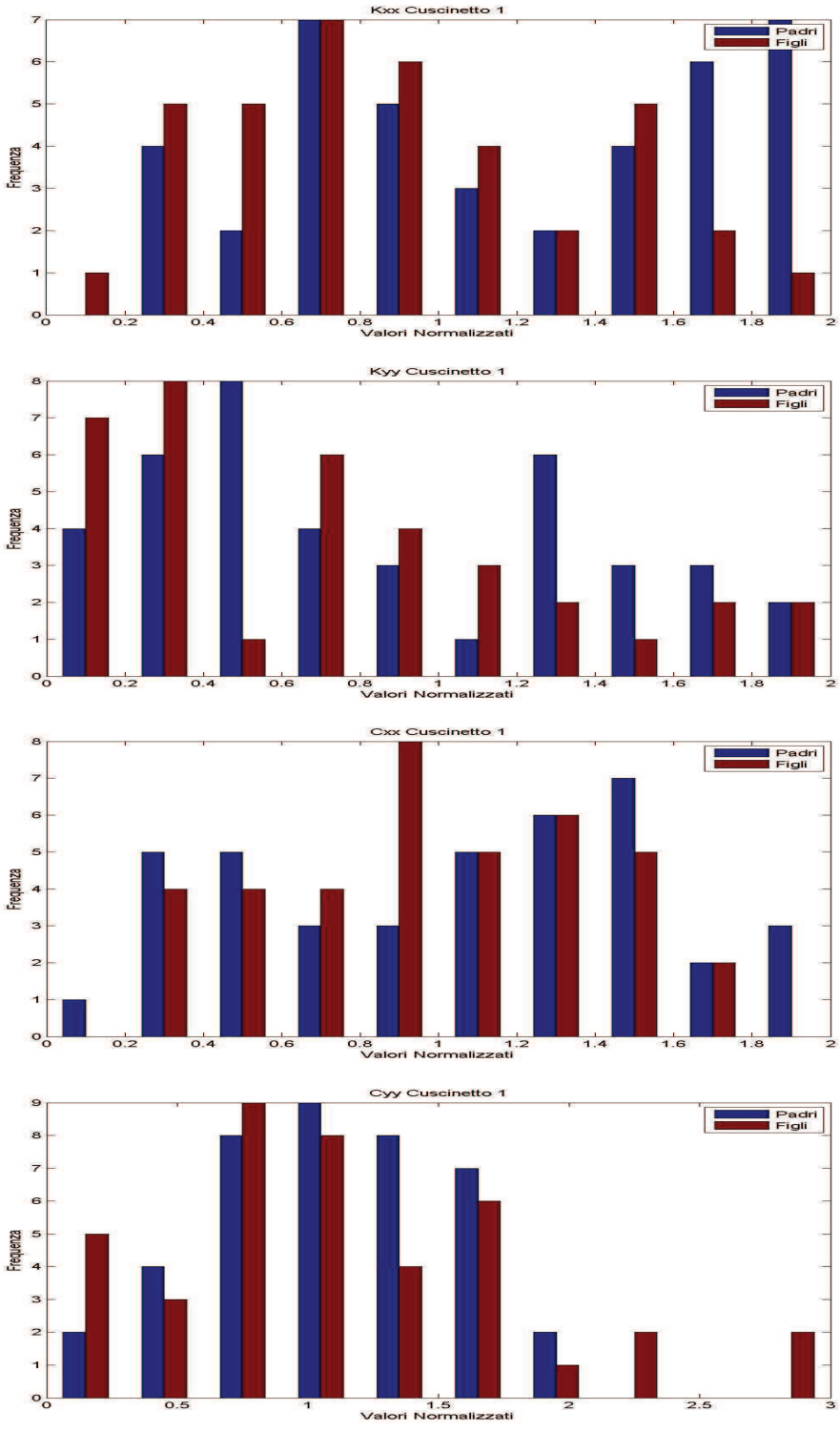


Figura 4.16 – Convergenza per K_{xx}, K_{yy}, C_{xx}, C_{yy} del cuscinetto 2

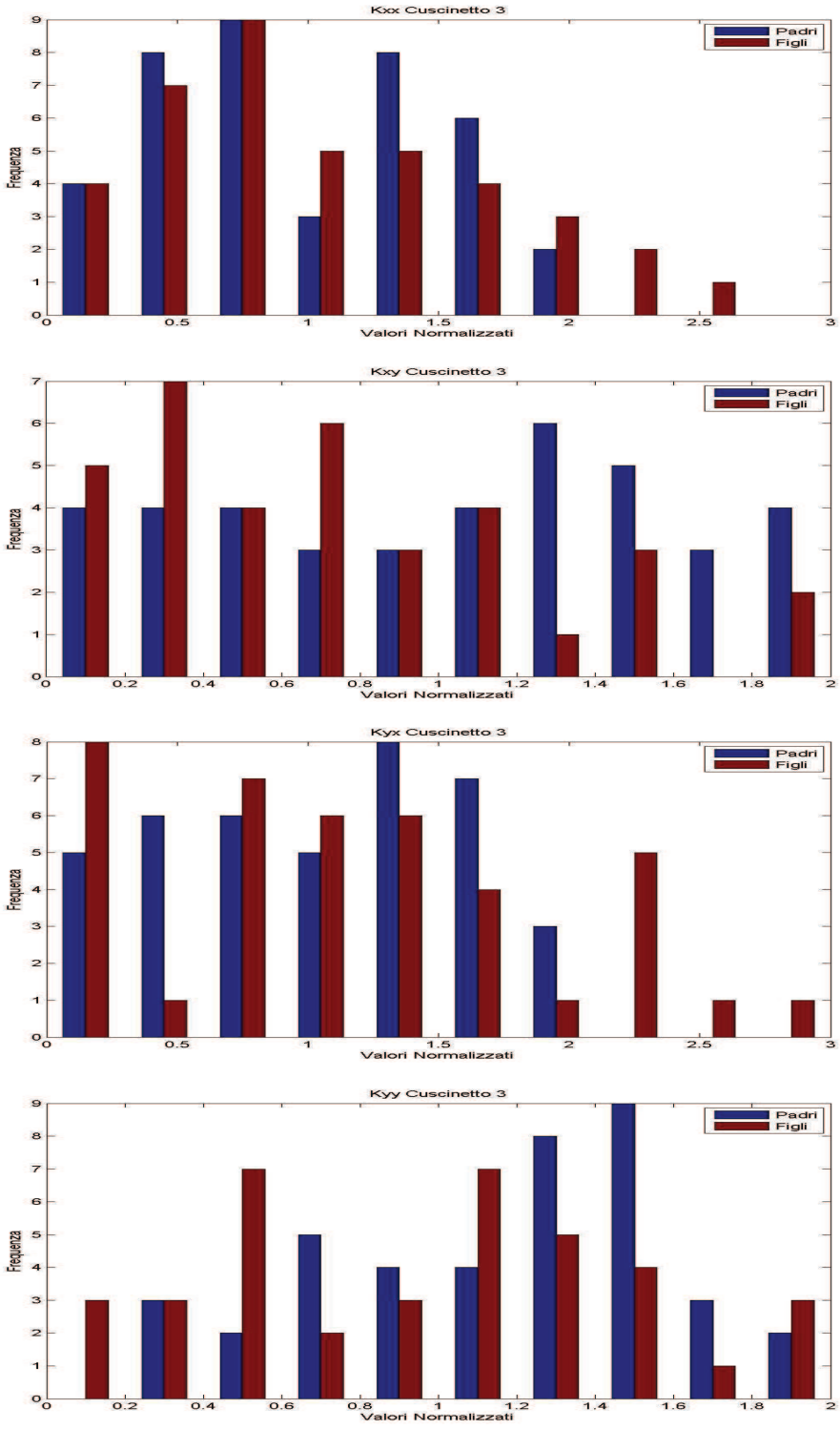


Figura 4.17 – Convergenza per K_{xx}, K_{xy}, K_{yx}, K_{yy} del cuscinetto 3

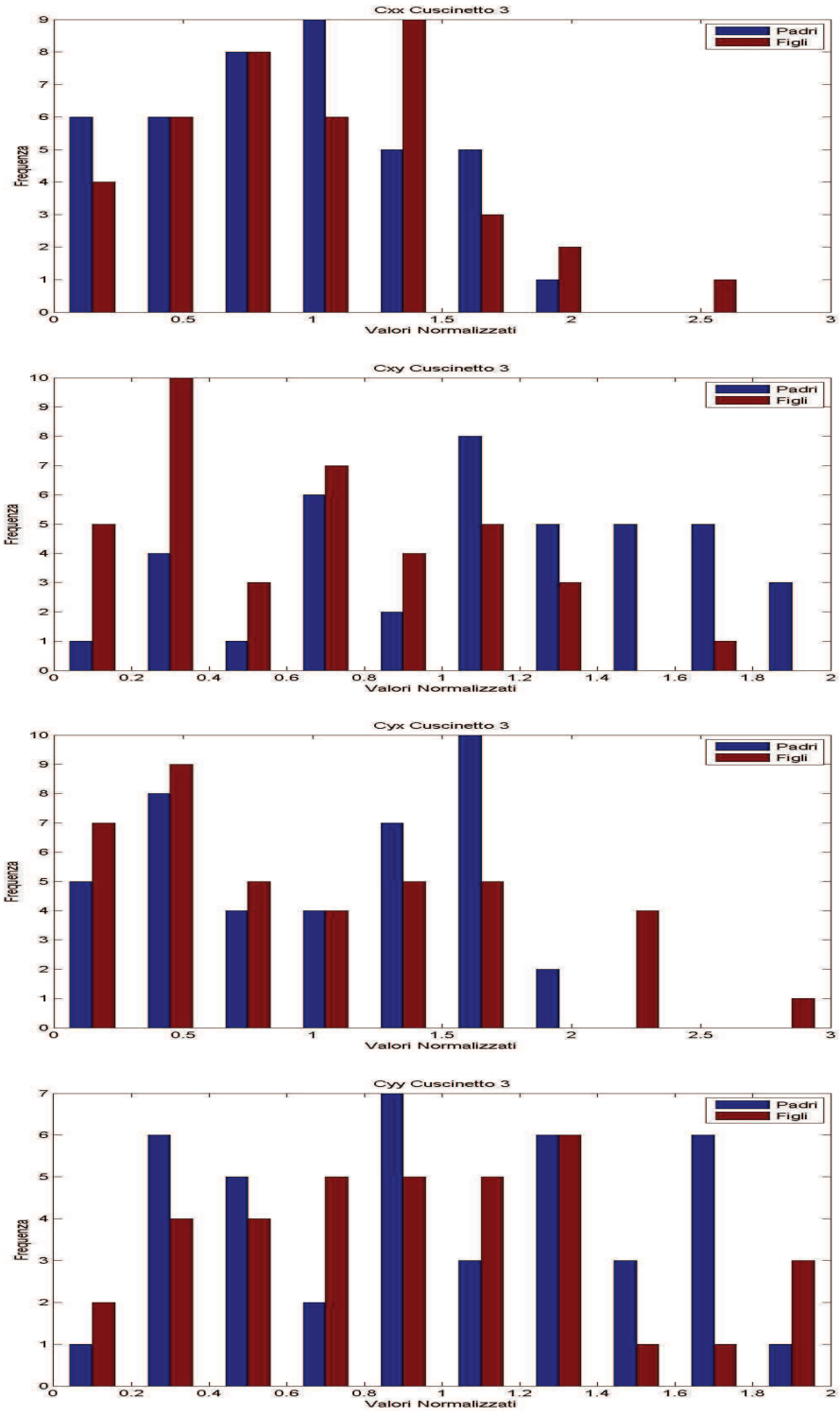


Figura 4.18 – Convergenza per C_{xx}, C_{xy}, C_{yx}, C_{yy} del cuscinetto 3

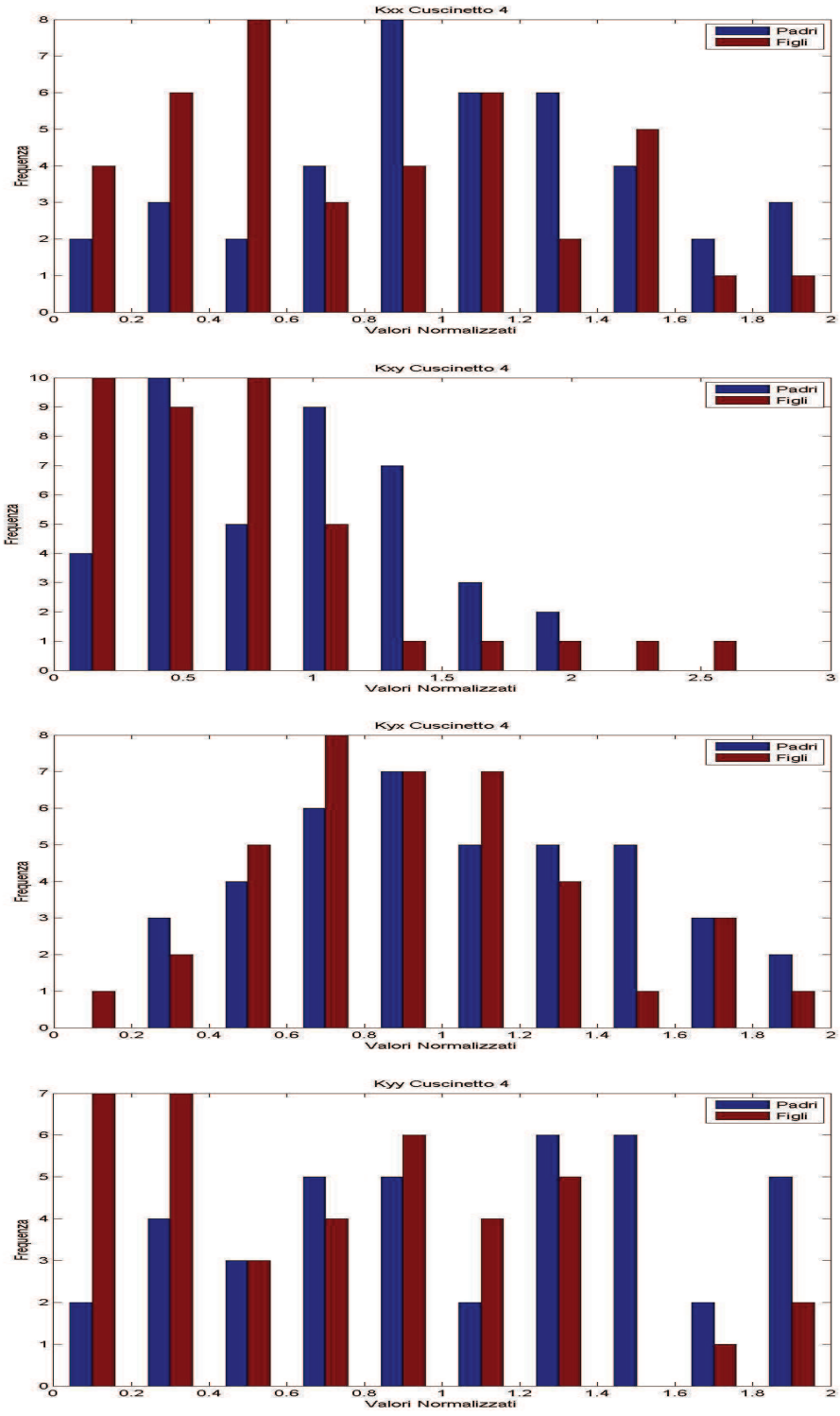


Figura 4.19 – Convergenza per K_{xx}, K_{xy}, K_{yx}, K_{yy} del cuscinetto 4

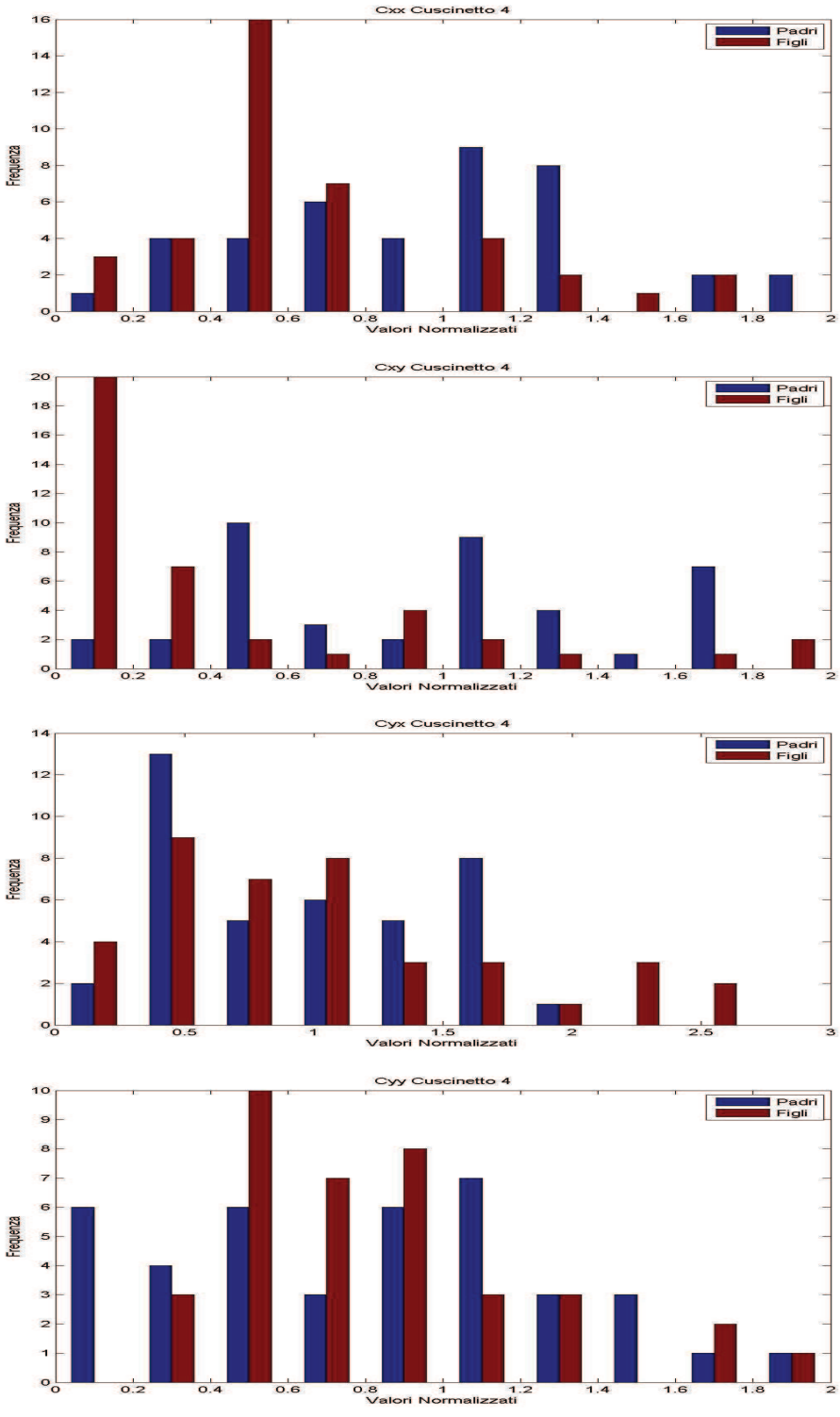


Figura 4.20 – Convergenza per C_{xx}, C_{xy}, C_{yx}, C_{yy} del cuscinetto 4

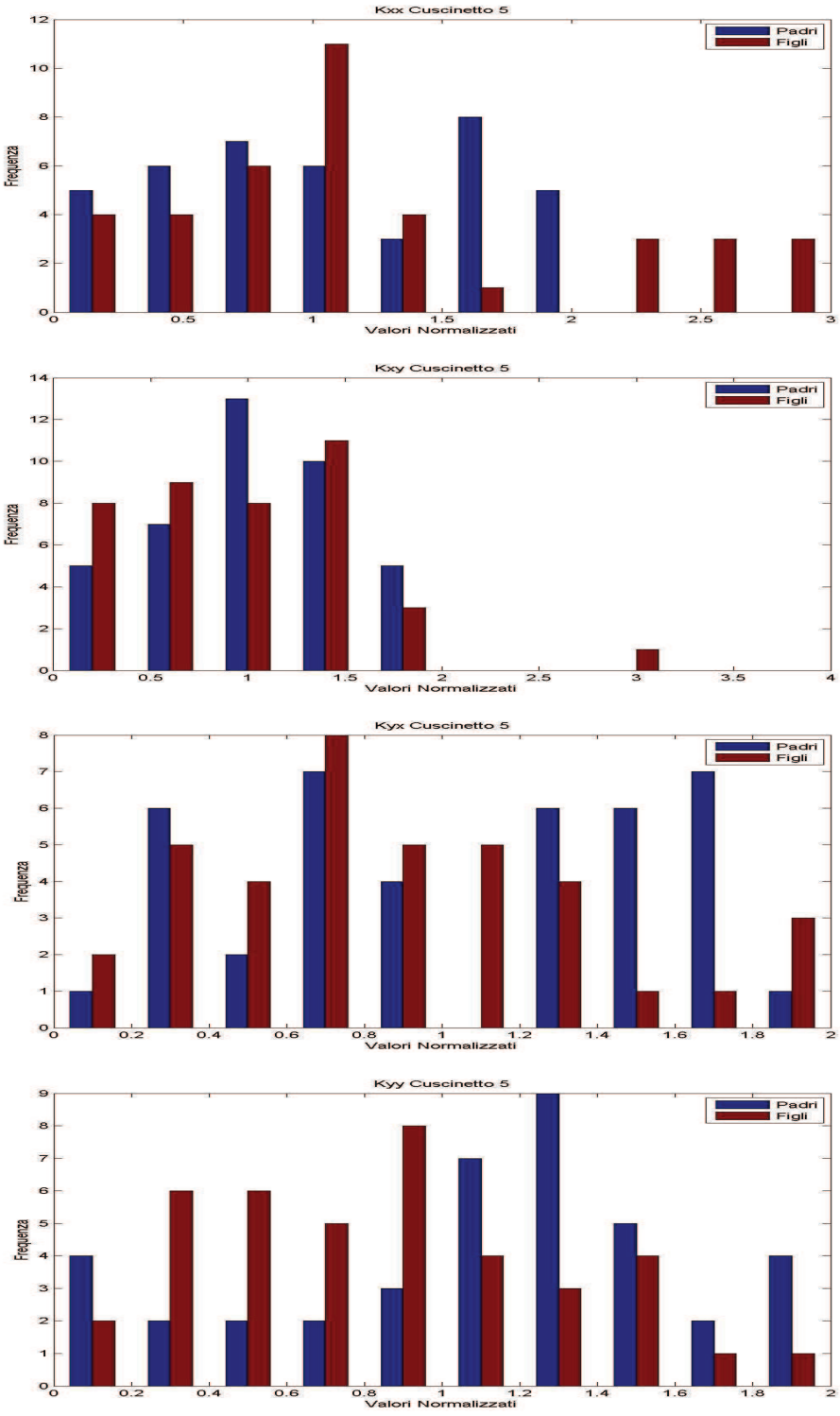


Figura 4.21 – Convergenza per K_{xx}, K_{xy}, K_{yx}, K_{yy} del cuscinetto 5

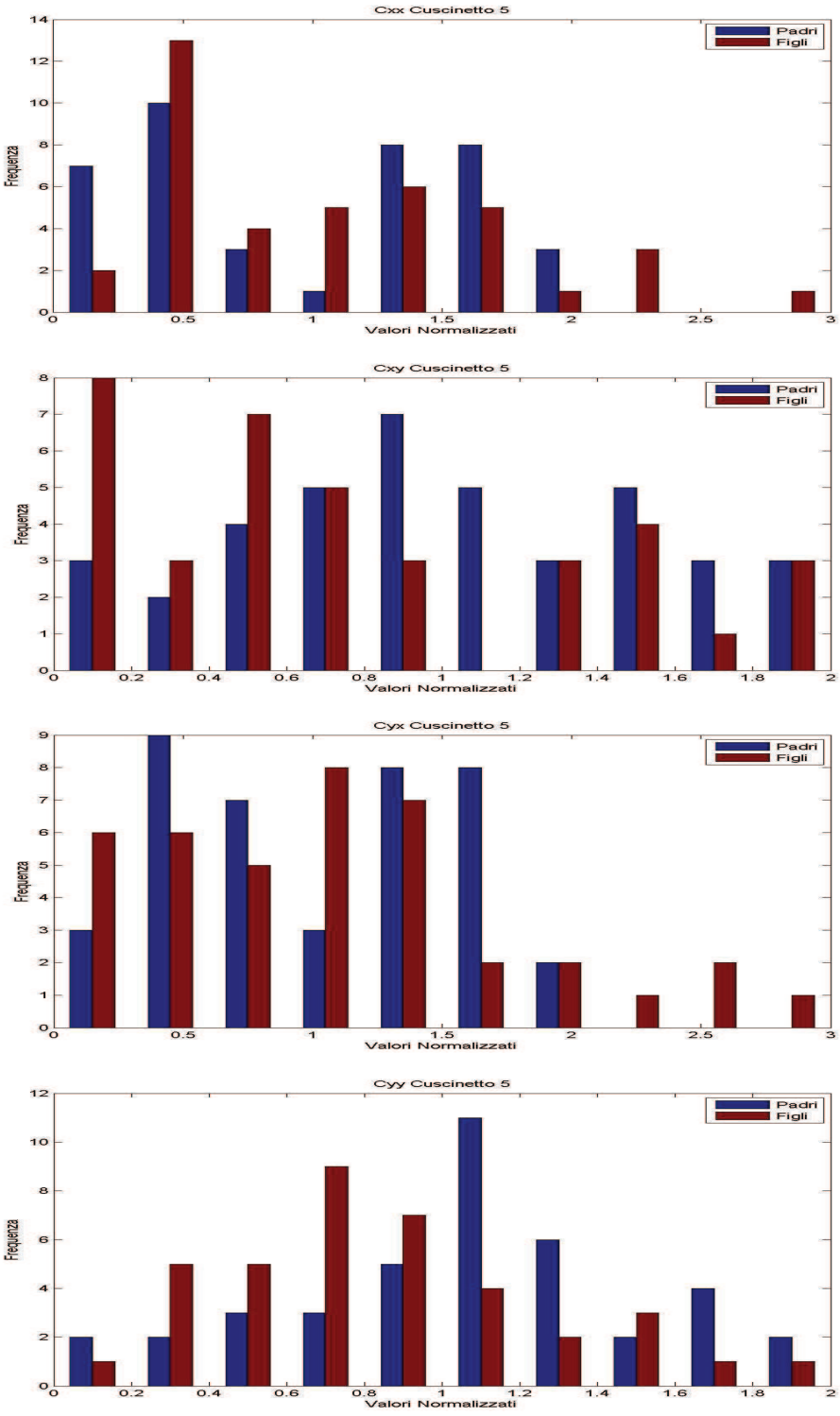


Figura 4.22 – Convergenza per C_{xx}, C_{xy}, C_{yx}, C_{yy} del cuscinetto 5

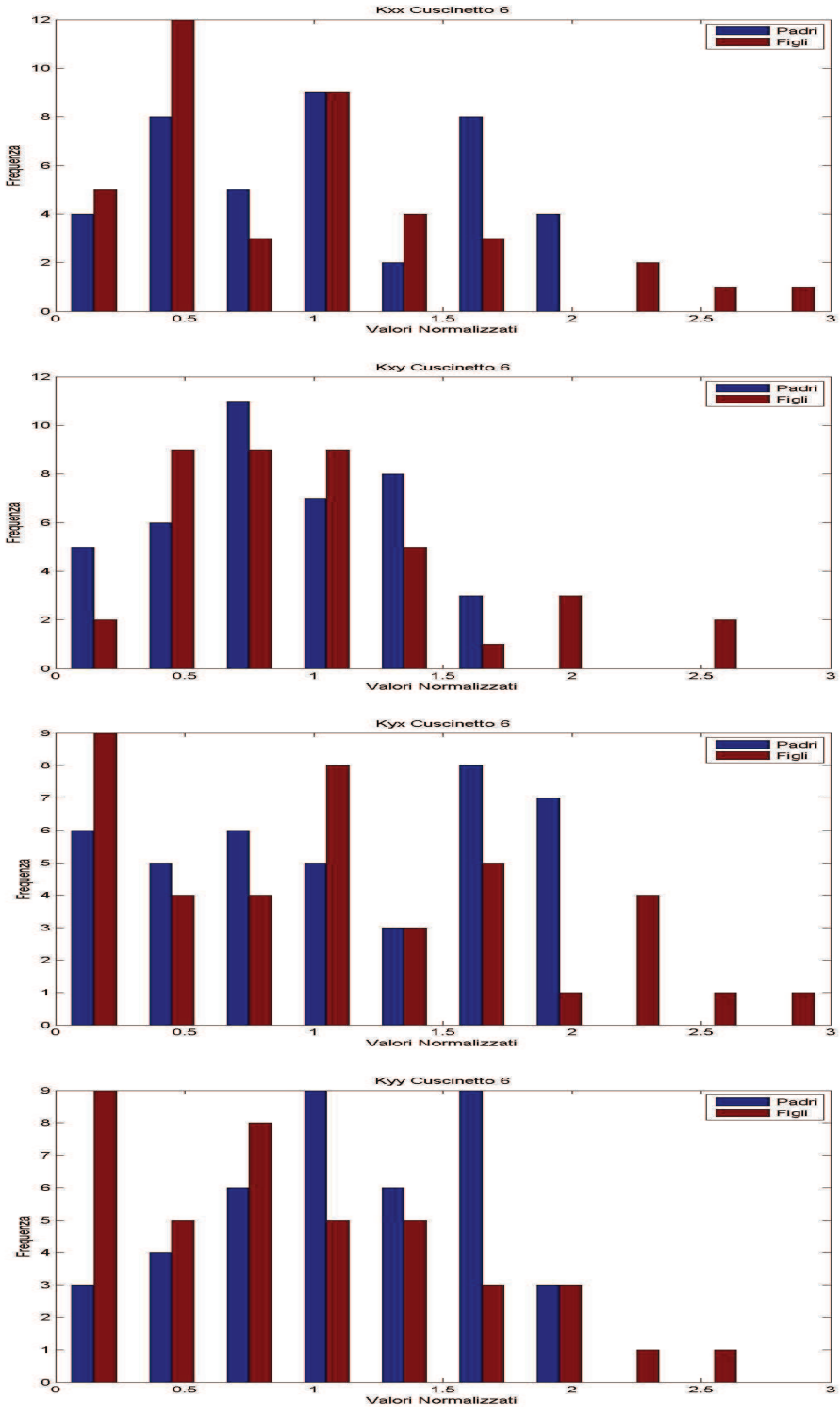


Figura 4.23 – Convergenza per K_{xx}, K_{xy}, K_{yx}, K_{yy} del cuscinetto 6

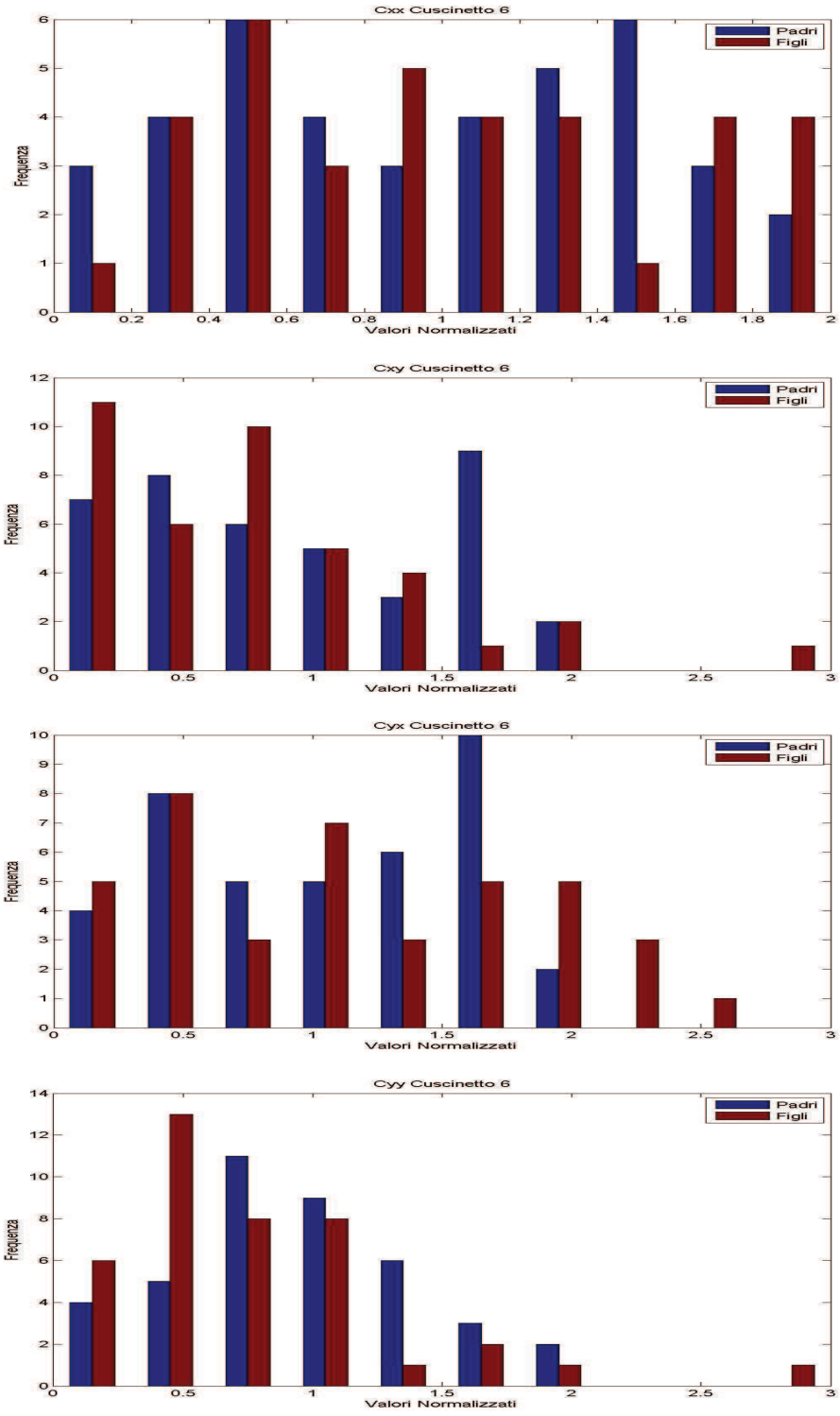


Figura 4.24 – Convergenza per C_{xx}, C_{xy}, C_{yx}, C_{yy} del cuscinetto 6

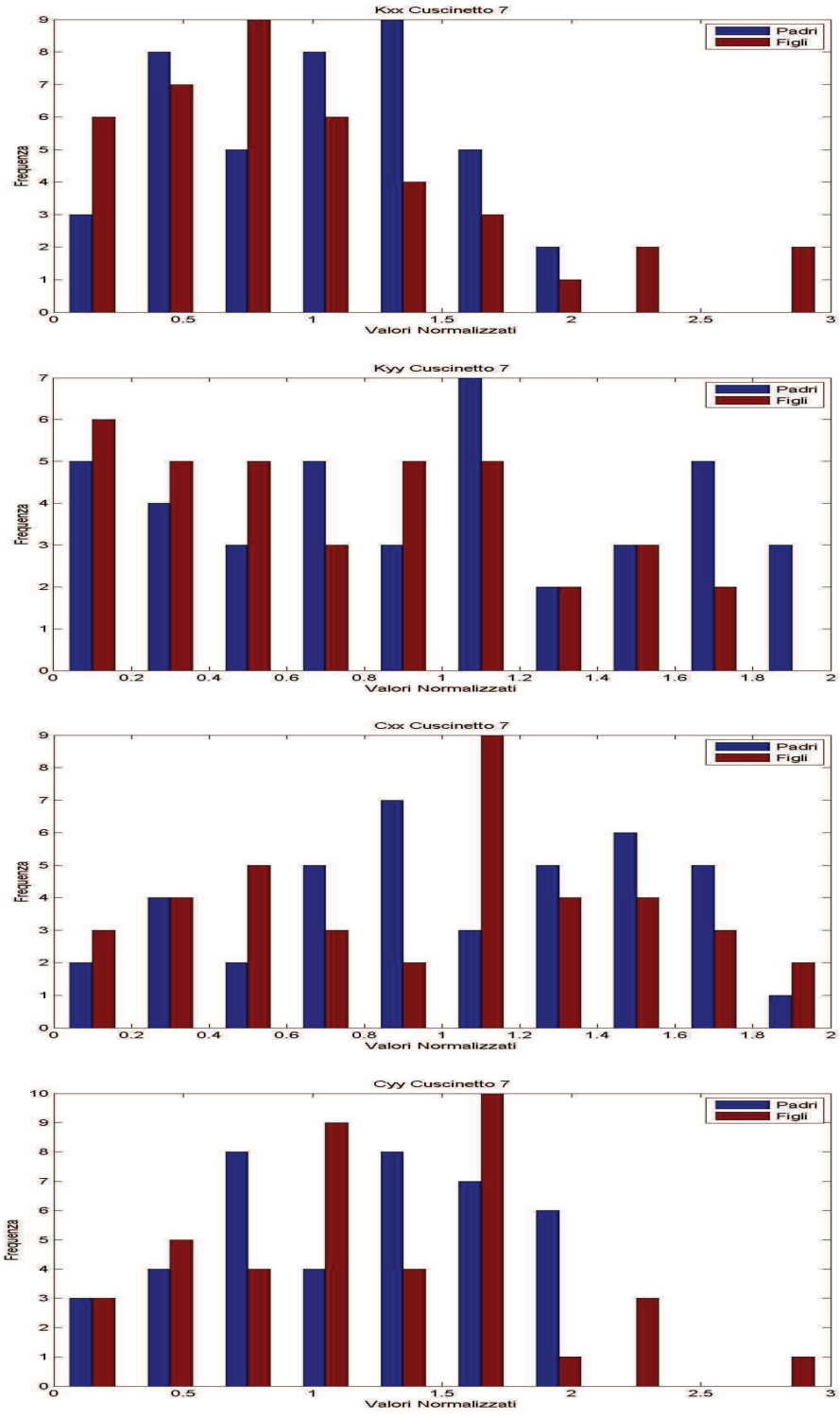


Figura 4.25 – Convergenza per K_{xx}, K_{yy}, C_{xx}, C_{yy} del cuscinetto 7

Certi coefficienti invece dimostrano una forte convergenza, più forte anche di quelle osservate nel caso del *test_rig*. Per esempio, i coefficienti C_{xx} e C_{xy} del cuscinetto 4 (Figura 4.20) si concentrano rispettivamente intorno ai valori 0.5 e 0.1. Possiamo quindi dedurne che essi sono veramente sensibili, che la loro influenza è più importante dei parametri elencati precedentemente. Possiamo notare che anche K_{yy} e C_{yy} del cuscinetto 4 (Figure 4.20 e 4.21) , C_{yy} del cuscinetto 6 (Figura 4.24) e C_{yy} del cuscinetto 7 (Figura 4.25) convergono in maniera notevole con una varianza un po' più forte dei due precedenti. Per verificare la sensibilità di questi coefficienti possiamo fare un test di ottimizzazione agendo solo su questi e annotando i risultati ottenuti, per poi confrontarli con quelli ricavati da un'ottimizzazione effettuata con parametri considerati poco sensibili. Il confronto seguente sarà quindi fatto tra un'ottimizzazione agendo sui coefficienti C_{xx} e C_{xy} del cuscinetto 4 e C_{yy} del cuscinetto 6 e un'ottimizzazione agendo sui coefficienti K_{yy} del cuscinetto $1, C_{xy}$ del cuscinetto 5 e C_{yy} del cuscinetto 6. Il set di cuscinetti iniziale sul quale saranno effettuate le due ottimizzazioni sarà generato casualmente come nello studio della convergenza.

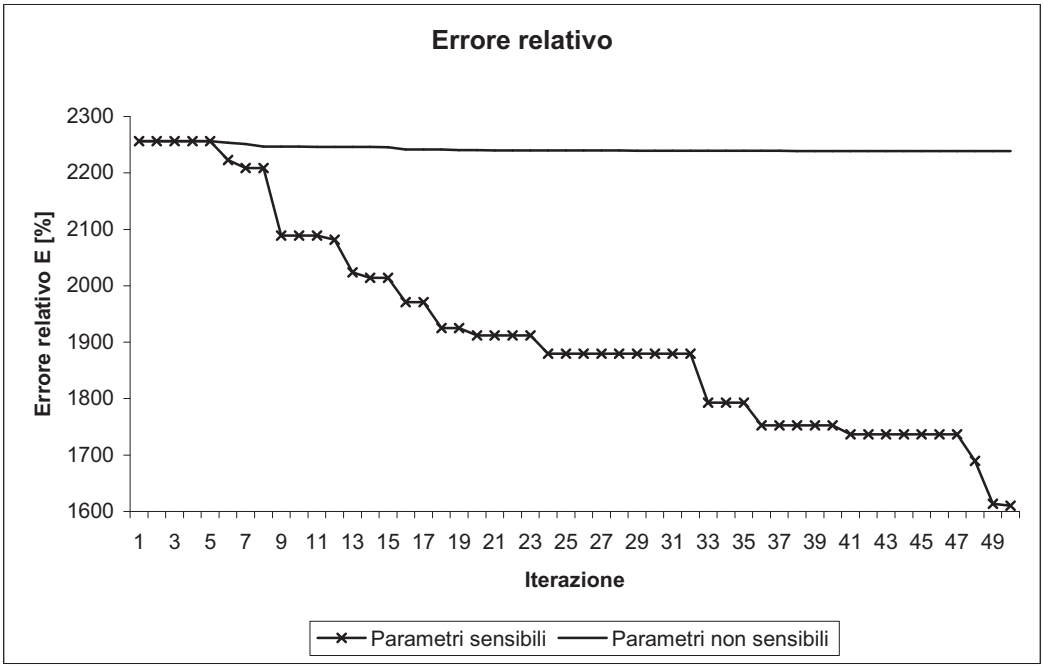


Figura 4.26 – Comparativo di ottimizzazioni con parametri diversi

L'updating eseguito con i parametri non sensibili elencati sopra non porta a nessun miglioramento significativo, mentre quello fatto con i parametri sensibili permette un miglioramento notevole su solo 50 iterazioni (Figura 4.26). Questo fatto conferma l'idea che una convergenza forte del test precedente indica una grande sensibilità per un parametro. Possiamo allora concludere che questo test permette di individuare le zone del modello deboli rispetto alle misure. I coefficienti individuati come sensibili, in questo caso, si discostano notevolmente dal valore del cuscinetto originale. Questo è molto interessante perché si può immaginare un procedimento sperimentale per individuare delle anomalie della struttura del turbogeneratore. In effetti, in questo caso, la modellazione sembra un po' difettosa per il cuscinetto 4 e quindi può essere opportuno per il gestore della macchina in oggetto fare delle analisi approfondite su questa zona della macchina.

4.4 Conclusioni sull'updating della macchina reale

Questo studio sulla macchina reale ci ha permesso di constatare che la natura complessa del modello comporta tempi di calcolo molto lunghi e risultati delicati da analizzare perché numerosi. Le convergenze appaiono un po' meno nette che nel caso del *test_rig*. L'errore relativo diminuisce circa del 20% rispetto all'errore iniziale, quindi diminuisce rispetto al caso del *test_rig* dove si attestava attorno al 40%. In effetti è più difficile ottimizzare contemporaneamente un gran numero di parametri.

Abbiamo visto come l'updating consenta di individuare delle zone del modello più sensibili delle altre e quindi permetta all'utente di cercare la ragione di questa sensibilità. In effetti può essere un difetto della modellazione a priori della macchina però, nel caso si esegua spesso questa procedura sulla stessa macchina con delle misure aggiornate, si può pensare ad un metodo che permetta di determinare le zone danneggiate della macchina. Il significato fisico dei parametri ci consente, dopo aver confrontato i risultati ottenuti con i valori iniziali, di trovare in quali punti il modello della macchina non rispecchia la realtà, cosa che non è possibile con i metodi di aggiornamento diretto.

5. Conclusioni

I confronti effettuati nello stato dell'arte ci hanno permesso di individuare un procedimento per realizzare l'ottimizzazione dei modelli. La superiorità dei metodi iterativi rispetto ai metodi diretti per tutto ciò che riguarda la parametrizzazione e la fisica del modello ci ha spinto a usare questo procedimento nel nostro studio. E' stato scelto un metodo iterativo basato su un algoritmo di tipo genetico che permette di conservare il significato fisico del modello dopo l'aggiornamento e di essere poco soggetto all'intrappolamento dei minimi locali.

L'efficienza dell'algoritmo di ottimizzazione dipende, come l'abbiamo visto nel terzo capitolo, di diversi parametri. La configurazione ottimale richiederebbe un gran numero di padri accoppiato con un numero di figli circa dieci volte maggiore. Il tempo di calcolo con questa configurazione aumenta notevolmente. La forza di mutazione può invece benissimo essere regolata in modo adatto visto che il suo valore ottimale è un numero finito stabilito dall'esperienza a 0.085.

L'analisi di sensibilità ci ha permesso in un primo tempo di capire che l'influenza di un parametro era sopratutto locale e che la natura della sensibilità dipende anche dalla natura fisica del parametro studiato. In un secondo tempo ci ha permesso di individuare quali erano i parametri che agivano maggiormente sulle risposte del modello.

Un'altra osservazione interessante è la possibile introduzione di risonanze nella modellazione dovuta all'updating. E' un problema che deve essere controllato perché può modificare in maniera errata l'approccio che l'utente ha sulla sua macchina. Sappiamo che le risonanze danneggiano le macchine e quindi l'utente può prendere delle precauzioni inutili per evitare risonanze che in realtà non esistono.

Per il test_rig abbiamo visto che erano soprattutto i coefficienti del terzo cuscinetto ad essere i più sensibili. Lo studio della convergenza di questi parametri ci ha mostrato che, dopo avere eseguito l'updating globale, i parametri citati precedentemente si concentravano di più nello spazio delle soluzioni.

Per lo studio della macchina reale osserviamo che l'effetto della variazione di un parametro è quasi esclusivamente locale. Abbiamo anche potuto capire che l'aumento del numero di parametri diminuisce la capacità che ha ogni parametro di modificare le risposte date dal modello.

Tutti questi test sulla sensibilità e la convergenza dei parametri ci hanno permesso di stabilire una correlazione tra la sensibilità di un parametro e la sua convergenza rispetto all'algoritmo. Una convergenza forte del parametro significa una sensibilità forte del modello a questo parametro. L'applicazione del processo di model updating ad un caso reale introduce nuove problematiche come il tempo di calcolo e la complessità del modello. L'osservazione del caso precedente è stata confermata con questo studio, dove alcuni parametri con forte convergenza risultano essere molto sensibili.

Possiamo quindi concludere che il model updating eseguito con un algoritmo genetico possiede le qualità per essere un metodo sfruttabile dal momento che permette all'utente una migliore precisione della modellazione, un'analisi fisica dei risultati ottenuti e un'efficienza teoricamente massima per un tempo di calcolo sufficientemente lungo. Permette di capire quali sono i punti deboli del sistema e di cercare le ragioni di questa debolezza.

I miglioramenti che possono essere introdotti in questo procedimento sono diversi. Si potrebbe per esempio pensare di aumentare la complessità dell'algoritmo utilizzando la ricombinazione, un tempo di sopravvivenza limitato degli individui o un auto-adattamento della forza di mutazione. A livello fisico una possibilità è quella di usare il numero di Sommerfeld per l'updating dei cuscinetti dei turbogeneratori. Usando delle tabelle relative a questo numero si potrebbe immaginare una generazione sempre stocastica però con senso fisico dei parametri aggiornati.

6. Appendice

6.1 Funzione *Generazione*

```
function []=Generazione
D1=load('C:\Users\Manuel\Documents\MATLAB\modello test-
rig\CUSC01STD.bea');
D2=load('C:\Users\Manuel\Documents\MATLAB\modello test-
rig\CUSC02STD.bea');
D3=load('C:\Users\Manuel\Documents\MATLAB\modello test-
rig\CUSC03MED.bea');
D4=load('C:\Users\Manuel\Documents\MATLAB\modello test-
rig\CUSC04MED.bea');
C1=zeros(13,9);
C2=zeros(13,9);
C3=zeros(13,9);
C4=zeros(13,9);
r=(1.8*rand(1,32)+ 0.1*ones(1,32));

    C1(:,1)=D1(:,1);
    C2(:,1)=D2(:,1);
    C3(:,1)=D3(:,1);
    C4(:,1)=D4(:,1);
    for a = 1:8
        C1(:,a+1)=D1(:,a+1)*r(a);
        C2(:,a+1)=D2(:,a+1)*r(a+8);
        C3(:,a+1)=D3(:,a+1)*r(a+16);
        C4(:,a+1)=D4(:,a+1)*r(a+24);
    end

% Scrivo i cuscinetti generati in nuovi file con estensione .bea
[nomefile,percorso]=uiputfile({'.bea'},'Scegli la cartella dove
salvare le figure');

    str_cusc=[nomefile num2str(1),'.bea'];
    fid=fopen(strcat(percorso,str_cusc),'wt');
    for zz=1:1:13
        for jj=1:1:9
            str=C1(zz,jj);
            if jj==1
                fprintf(fid,'%d\t',str);
            elseif jj>1 && jj<9
                fprintf(fid,'%1.5E\t',str);
            elseif jj==9
                fprintf(fid,'%1.5E\n',str);
            end
        end
    end
fclose(fid);
```

```
    str_cusc=[nomefile num2str(2),'.bea'];
    fid=fopen(strcat(percorso,str_cusc),'wt');
    for zz=1:1:13
        for jj=1:1:9
            str=C2(zz,jj);
            if jj==1
                fprintf(fid,'%d\t',str);
            elseif jj>1 && jj<9
                fprintf(fid,'%1.5E\t',str);
            elseif jj==9
                fprintf(fid,'%1.5E\n',str);
            end
        end
    end
    fclose(fid);

    str_cusc=[nomefile num2str(3),'.bea'];
    fid=fopen(strcat(percorso,str_cusc),'wt');
    for zz=1:1:13
        for jj=1:1:9
            str=C3(zz,jj);
            if jj==1
                fprintf(fid,'%d\t',str);
            elseif jj>1 && jj<9
                fprintf(fid,'%1.5E\t',str);
            elseif jj==9
                fprintf(fid,'%1.5E\n',str);
            end
        end
    end
    fclose(fid);

    str_cusc=[nomefile num2str(4),'.bea'];
    fid=fopen(strcat(percorso,str_cusc),'wt');
    for zz=1:1:13
        for jj=1:1:9
            str=C4(zz,jj);
            if jj==1
                fprintf(fid,'%d\t',str);
            elseif jj>1 && jj<9
                fprintf(fid,'%1.5E\t',str);
            elseif jj==9
                fprintf(fid,'%1.5E\n',str);
            end
        end
    end
    fclose(fid);

end
```

6.2 Funzione *MatPF*

```
function [Pad,Fig]=MatPF(cusc,Np,Nf)
D=load(strcat('C:\Users\Manuel\Documents\MATLAB\modello test-
rig\CUSC0',num2str(cusc),'.bea'));
Pad=zeros(Np,8);
Fig=zeros(Nf,8);
    for k=1:Np
        [a,b,c]=uigetfile('*.bea',strcat('Padre-
',num2str(k)),strcat('C:\Users\Manuel\Desktop\Tesi\Cuscinetti
Casuali\1-20'));
        A=load(strcat(b,a));
        Pad(k,:)=A(6,2:9)./D(6,2:9);
    end
    for s=1:Nf
        [a,b,c]=uigetfile('*.bea',strcat('Figlio-
',num2str(s)),strcat('C:\Users\Manuel\Desktop\Tesi\Cuscinetti
Casuali\1-20'));
        B=load(strcat(b,a));
        Fig(s,:)=B(6,2:9)./D(6,2:9);
    end
end
```

6.3 Funzione *Ripartizione*

```
function [Res]=Ripartizione(Pad,Fig,p,m,cusc)
[Np,Npp]=size(Pad);
[Nf,Npf]=size(Fig);
Res=zeros(max([Np,Nf,m]),7);
Res(1:Np,1)=Pad(:,p);
Res(1:Nf,2)=Fig(:,p);
Noms=['Kxx','Kxy','Kyx','Kyy','Cxx','Cxy','Cyx','Cyy'];
    pas=floor(max(max(Res(1:m,1:2)))+1)/m;
    Res(1,3)=0;
    Res(1,4)=Res(1,3)+pas;
    Res(1,5)=(Res(1,3)+Res(1,4))/2;
    for k=2:m
        Res(k,3)=Res(k-1,3)+pas;
        Res(k,4)=Res(k-1,4)+pas;
        Res(k,5)=(Res(k,3)+Res(k,4))/2;
    end
    for w =1:Np
        for v=1:m
            if (Res(w,1)>=Res(v,3)) && (Res(w,1)<Res(v,4))
            Res(v,6)=Res(v,6)+1;
            else
            end
        end
    end
    for y=1:Nf
        for z=1:m
            if (Res(y,2)>=Res(z,3)) && (Res(y,2)<Res(z,4))
            Res(z,7)=Res(z,7)+1;
            else
            end
        end
    end
bar(Res(1:m,5),Res(1:m,6:7),1),legend('Padri','Figli'),xlabel('V
alori
Normalizzati'),ylabel('Frequenza'),title(strcat(Noms(1,3*p-
2:1:3*p),{' '},'Cuscinetto',{' '},num2str(cusc)))
print -djpeg Resultat.jpg;
end
```

6.4 Funzione *GenMR*

```
function []=GenMR
D1=load('C:\Users\Manuel\Documents\MATLAB\Macchina
Reale\modelloMV\Cuscinetto-1.bea');
D2=load('C:\Users\Manuel\Documents\MATLAB\Macchina
Reale\modelloMV\Cuscinetto-2.bea');
D3=load('C:\Users\Manuel\Documents\MATLAB\Macchina
Reale\modelloMV\Cuscinetto-3.bea');
D4=load('C:\Users\Manuel\Documents\MATLAB\Macchina
Reale\modelloMV\Cuscinetto-4.bea');
D5=load('C:\Users\Manuel\Documents\MATLAB\Macchina
Reale\modelloMV\Cuscinetto-5.bea');
D6=load('C:\Users\Manuel\Documents\MATLAB\Macchina
Reale\modelloMV\Cuscinetto-6.bea');
D7=load('C:\Users\Manuel\Documents\MATLAB\Macchina
Reale\modelloMV\Cuscinetto-7.bea');
C1=zeros(7,9);
C2=zeros(7,9);
C3=zeros(7,9);
C4=zeros(7,9);
C5=zeros(7,9);
C6=zeros(7,9);
C7=zeros(7,9);
r=0.1*ones(1,56)+1.8*rand(1,56);

    C1(:,1)=D1(:,1);
    C2(:,1)=D2(:,1);
    C3(:,1)=D3(:,1);
    C4(:,1)=D4(:,1);
    C5(:,1)=D5(:,1);
    C6(:,1)=D6(:,1);
    C7(:,1)=D7(:,1);
    for a = 1:8
        C1(:,a+1)=D1(:,a+1)*r(a);
        C2(:,a+1)=D2(:,a+1)*r(a+8);
        C3(:,a+1)=D3(:,a+1)*r(a+16);
        C4(:,a+1)=D4(:,a+1)*r(a+24);
        C5(:,a+1)=D5(:,a+1)*r(a+32);
        C6(:,a+1)=D6(:,a+1)*r(a+40);
        C7(:,a+1)=D7(:,a+1)*r(a+48);
    end

% Scrivo i cuscinetti ottimizzati in nuovi file con estensione
.bea
[nomefile,percorso]=uiputfile({'.bea'},'Scegli la cartella dove
salvare le figure');

    str_cusc=[nomefile num2str(1),'.bea'];
    fid=fopen(strcat(percorso,str_cusc),'wt');
    for zz=1:1:7
```

```
    for jj=1:1:9
        str=C1(zz,jj);
        if jj==1
            fprintf(fid,'%d\t',str);
        elseif jj>1 && jj<9
            fprintf(fid,'%1.5E\t',str);
        elseif jj==9
            fprintf(fid,'%1.5E\n',str);
        end
    end
end
fclose(fid);

str_cusc=[nomefile num2str(2),'.bea'];
fid=fopen(strcat(percorso,str_cusc),'wt');
for zz=1:1:7
    for jj=1:1:9
        str=C2(zz,jj);
        if jj==1
            fprintf(fid,'%d\t',str);
        elseif jj>1 && jj<9
            fprintf(fid,'%1.5E\t',str);
        elseif jj==9
            fprintf(fid,'%1.5E\n',str);
        end
    end
end
fclose(fid);

str_cusc=[nomefile num2str(3),'.bea'];
fid=fopen(strcat(percorso,str_cusc),'wt');
for zz=1:1:7
    for jj=1:1:9
        str=C3(zz,jj);
        if jj==1
            fprintf(fid,'%d\t',str);
        elseif jj>1 && jj<9
            fprintf(fid,'%1.5E\t',str);
        elseif jj==9
            fprintf(fid,'%1.5E\n',str);
        end
    end
end
fclose(fid);
str_cusc=[nomefile num2str(4),'.bea'];
fid=fopen(strcat(percorso,str_cusc),'wt');
for zz=1:1:7
    for jj=1:1:9
        str=C4(zz,jj);
        if jj==1
            fprintf(fid,'%d\t',str);
        elseif jj>1 && jj<9
```

```
                fprintf(fid,'%1.5E\t',str);
            elseif jj==9
                fprintf(fid,'%1.5E\n',str);
            end
        end
    end
    str_cusc=[nomefile num2str(5),'.bea'];
    fid=fopen(strcat(percorso,str_cusc),'wt');
    for zz=1:1:7
        for jj=1:1:9
            str=C5(zz,jj);
            if jj==1
                fprintf(fid,'%d\t',str);
            elseif jj>1 && jj<9
                fprintf(fid,'%1.5E\t',str);
            elseif jj==9
                fprintf(fid,'%1.5E\n',str);
            end
        end
    end
    str_cusc=[nomefile num2str(6),'.bea'];
    fid=fopen(strcat(percorso,str_cusc),'wt');
    for zz=1:1:7
        for jj=1:1:9
            str=C6(zz,jj);
            if jj==1
                fprintf(fid,'%d\t',str);
            elseif jj>1 && jj<9
                fprintf(fid,'%1.5E\t',str);
            elseif jj==9
                fprintf(fid,'%1.5E\n',str);
            end
        end
    end
    str_cusc=[nomefile num2str(7),'.bea'];
    fid=fopen(strcat(percorso,str_cusc),'wt');
    for zz=1:1:7
        for jj=1:1:9
            str=C7(zz,jj);
            if jj==1
                fprintf(fid,'%d\t',str);
            elseif jj>1 && jj<9
                fprintf(fid,'%1.5E\t',str);
            elseif jj==9
                fprintf(fid,'%1.5E\n',str);
            end
        end
    end
    fclose(fid);

end
```

6.5 Funzione *MatPFMR*

```
function [Pad,Fig]=MatPFMR(cusc,Np,Nf)
D=load(strcat('C:\Users\Manuel\Documents\MATLAB\Macchina
Reale\modelloMV\Cuscinetto-',num2str(cusc),'.bea'));
Pad=zeros(Np,8);
Fig=zeros(Nf,8);
    for k=1:Np
        [a,b,c]=uigetfile('*.bea',strcat('Padre-
',num2str(k)),strcat('C:\Users\Manuel\Desktop\Tesi\Cuscinetti
Casuali\MR\Padri',num2str(cusc)));
        A=load(strcat(b,a));
        Pad(k,:)=A(3,2:9)./D(3,2:9);
    end
    for s=1:Nf
        [a,b,c]=uigetfile('*.bea',strcat('Figlio-
',num2str(s)),strcat('C:\Users\Manuel\Desktop\Tesi\Cuscinetti
Casuali\MR\Figli',num2str(cusc)));
        B=load(strcat(b,a));
        Fig(s,:)=B(3,2:9)./D(3,2:9);
    end
end
```

6.6 Funzione *RipartizioneMR*

```
function [Res]=RipartizioneMR(Pad,Fig,p,m,cusc)
[Np,Npp]=size(Pad);
[Nf,Npf]=size(Fig);
Res=zeros(max([Np,Nf,m]),7);
Res(1:Np,1)=Pad(:,p);
Res(1:Nf,2)=Fig(:,p);
Noms=['Kxx','Kxy','Kyx','Kyy','Cxx','Cxy','Cyx','Cyy'];
    pas=floor(max(max(Res(1:m,1:2)))+1)/m;
    Res(1,3)=0;
    Res(1,4)=Res(1,3)+pas;
    Res(1,5)=(Res(1,3)+Res(1,4))/2;
    for k=2:m
        Res(k,3)=Res(k-1,3)+pas;
        Res(k,4)=Res(k-1,4)+pas;
        Res(k,5)=(Res(k,3)+Res(k,4))/2;
    end
    for w =1:Np
        for v=1:m
            if (Res(w,1)>=Res(v,3)) && (Res(w,1)<Res(v,4))
            Res(v,6)=Res(v,6)+1;
            else
            end
        end
    end
    for y=1:Nf
        for z=1:m
            if (Res(y,2)>=Res(z,3)) && (Res(y,2)<Res(z,4))
            Res(z,7)=Res(z,7)+1;
            else
            end
        end
    end
bar(Res(1:m,5),Res(1:m,6:7),1),legend('Padri','Figli'),xlabel('V
alori
Normalizzati'),ylabel('Frequenza'),title(strcat(Noms(1,3*p-
2:1:3*p),{' '},'Cuscinetto',{' '},num2str(cusc)))
print -djpeg Resultat.jpg;
end
```

7. Bibliografia

[1] - M. I. Friswell, J. E. Mottershead (1993) – *Model Updating in structural dynamics : A survey*

[2] - N. Cottin, J. Reetz (2004) – *Accuracy of multiparameter eigenvalues used for dynamic model updating with measured natural frequency only*

[3] - D. Arnold, H. Beyer (2003) – *A comparison of evolution strategies with other direct search methods in the presence of noise*

[4] - R. I. Levin, N. A. J Lieven (1997) – *Dynamic finte element model updating using simulated annealing and genetic algorithms*

[5] - D. Wiestra, T. Schaul, J. Peter, J Scmidhuber (2008) – *Natural evolution strategies*

[6] - J. Xu, J. Zhang, X, Song (2006) – *Evolutionary Programming : The state of the art*

[7] - Kwang Y. Lee, Mohamed A. El-Sharkawi (2008) – *Modern heuristic optimization technique.*