

MAGNETIC MODELING OF INTERNAL PERMANENT MAGNET MOTORS USING RADIAL BASIS FUNCTION NETWORKS

POTA STEFANO¹

07/04/19

CONTENTS

1	Introduzione	2
2	Background teorico	2
3	Descrizione della rete RBF	4
4	Data Collection	5
5	Risultati della simulazione	6
6	Conclusioni	9
A	Script Matlab Rete RBF	11
B	Script Plecs Data Collection	11

LIST OF FIGURES

Figure 1	Schema usato per il training della rete RBF	2
Figure 2	I punti sono i centri delle funzioni Gaussiane nel piano \mathbf{R}^2	4
Figure 3	Rete RBF	4
Figure 4	Controllo di velocità agente sul modello meccanico del motore.	5
Figure 5	Funzione errore in uscita dall'output layer dopo la convergenza della rete.	6
Figure 6	Confronto delle funzioni flusso stimato e flusso reale dell'asse d.	7
Figure 7	Confronto delle funzioni flusso stimato e flusso reale dell'asse q.	7
Figure 8	Funzione 3D della differenza del flusso stimato e quello reale in percentuale, sia per l'asse d che quello q. Come si può notare si ottenuto un errore massimo pari al 1% sul flusso di asse q e ad 1.7% su flusso dell'asse d.	8
Figure 9	Confronto di slices del flusso stimato e di quello reale di asse d e q, avendo posto un valore costante per Iq ed Id rispettivamente nei due casi.	8

LIST OF TABLES

Table 1	Parametri simulazione Plecs	6
Table 2	Parametri simulazione Matlab	6

ABSTRACT

A causa dell'aumento di popolarità che i motori di tipo *IPM* e *SynR* stanno ricevendo, ne viene richiesto sempre più spesso un modello matematico accurato per poterli controllare. Questa relazione ha lo scopo di estrarre le mappe di flusso di asse d e q, compresi gli effetti di non linearità e di cross coupling, di un motore IPM, utilizzando una rete neurale di tipo Radial Basis, ne vengono poi confrontati i risultati ottenuti con le mappe di flusso reali del motore.

1 INTRODUZIONE

La soluzione studiata, si basa sull'utilizzo di una rete neurale artificiale (*ANN*), per mappare le relazioni non lineari che i flussi di asse d e q hanno rispetto le correnti i_d ed i_q . Per far questo si fa un training offline della rete utilizzando un dataset composto da misure, fatte in condizioni di regime, delle tensioni di asse d e q (u_d, u_q) e delle rispettive correnti e velocità meccanico elettrica imposte (i_d, i_q, ω_{me}). Il principale vantaggio che ne deriva da questa soluzione è quello di riuscire a ricavare, a differenza delle *look at table*, un'equazione continua delle caratteristiche magnetiche della macchina. Lo svantaggio invece è quello del costo computazionale estremamente più elevato rispetto ad altre soluzioni che si possono trovare in letteratura [3] [4], la grande quantità di memoria richiesta dall'algoritmo e il tempo richiesto per fare il training della rete, che poco si presta ad una versione online dove vengono aggiornate le curve di flusso ad ogni periodo di controllo. Ci si ritiene in ogni caso fiduciosi di possibili implementazioni future, anche a fronte di un maggiore studio sull'argomento.

2 BACKGROUND TEORICO

Le equazioni del motore sincrono nel sistema di riferimento (d,q) in condizioni di steady state, sono le seguenti:

$$\begin{cases} U_d = R_s i_d - w_{me} \lambda_q \\ U_q = R_s i_q + w_{me} \lambda_d \end{cases} \quad (1)$$

Dove la dipendenza del flusso di asse d e q, dalle correnti di entrambi gli assi è stata omessa per semplicità. Lo schema per l'identificazione delle funzioni non lineari è invece:

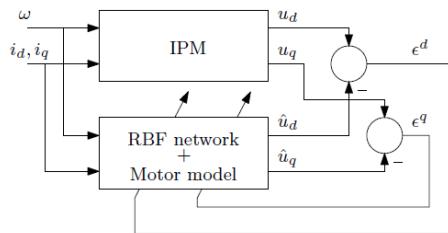


Figure 1: Schema usato per il training della rete RBF

¹ Dipartimento di Ingegneria Elettronica, Università Degli Studi di Udine, 116983, pota.stefano@spes.uniud.it

L'algoritmo di *backpropagation* che si occupa di aggiornare i pesi della rete funziona nel seguente modo:

1. I pesi della rete vengono inizializzati con dei valori scelti in modo casuale ed il motore viene pilotato ad una velocità costante w_{me} da un sistema esterno.
2. Viene inserito in ingresso della rete neurale e del motore, una corrente di riferimento di un determinato punto nel piano dq, viene utilizzata l'equazione (2) per stimare i flussi dei due assi ed successivamente viene utilizzata l'equazione (3) per stimare i valori delle tensioni \hat{u}_d , \hat{u}_q .
3. Viene fatta la differenza tra i valori di tensione misurati e quelli stimati sul motore per ottenere la funzione errore, usando l'equazione (4).
4. Tale procedura viene eseguita M volte, cambiando di volta in volta il riferimento di corrente imposto e coprendo tutta l'area di lavoro descritta dal rettangolo in figura (2). Gli M errori ricavati vengono memorizzati in un unico vettore e la cui dimensione è appunto M .
5. Viene utilizzato infine l'algoritmo di *Levenberg-Marquardt* (5) per aggiornare i pesi della rete nel generico step h della fase di training (vedi più avanti per i dettagli) allo scopo di minimizzare l'errore ϵ in uscita dalla rete.
6. Si rifà la procedura di training partendo dallo step 2 finché i pesi della rete convergono, o come nel nostro caso per un determinato numero di volte pari ad $N_{training}$.

$$\begin{cases} \hat{\lambda}_d = \sum_{k=1}^K a_k w_k^d + w_b^d \\ \hat{\lambda}_q = \sum_{k=1}^K a_k w_k^q + w_b^q \end{cases} \quad (2)$$

$$\begin{cases} \hat{u}_d = R_s i_d - w_{me} \hat{\lambda}_q \\ \hat{u}_q = R_s i_q + w_{me} \hat{\lambda}_d \end{cases} \quad (3)$$

$$\begin{cases} \epsilon^d = u_d - \hat{u}_d = u_d - R_s i_d + w_{me} \hat{\lambda}_q \\ \epsilon^q = u_q - \hat{u}_q = u_q - R_s i_q - w_{me} \hat{\lambda}_d \end{cases} \quad (4)$$

$$\begin{cases} w_{h+1}^d = w_h^d - [J_d^T J_d + \mu_h I]^{-1} J_d^T \epsilon^q \\ w_{h+1}^q = w_h^q - [J_q^T J_q + \mu_h I]^{-1} J_q^T \epsilon^d \end{cases} \quad (5)$$

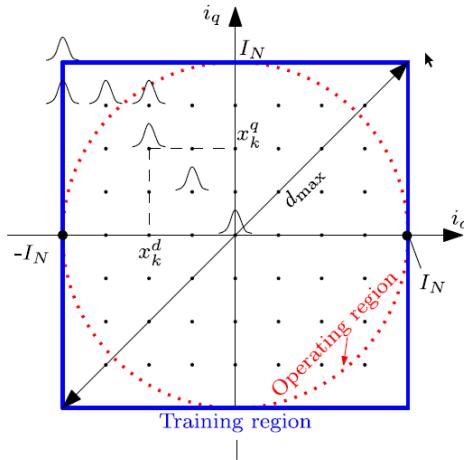


Figure 2: I punti sono i centri delle funzioni Gaussiane nel piano \mathbf{R}^2

3 DESCRIZIONE DELLA RETE RBF

L'area di lavoro di figura (2), viene nuovamente divisa in K punti, tali punti servono alla rete RBF, in particolare al suo primo layer, per computare la distanza Euclidea tra questi K punti fissi nello spazio ed il riferimento di corrente in ingresso della rete, successivamente i valori ottenuti vengono moltiplicati per un *bias* e per delle funzioni gaussiane con centro nell'origine, per sfruttare il cosiddetto *principio di località*². I valori in uscita dall'hidden layer vengono moltiplicati per $2K$ pesi (essendo diversi i pesi per l'asse d e per quello q) ed infine sommati per ricavare i flussi stimati dei due assi. L'ultimo bias della rete è stato posto pari a zero, in modo da tener conto che per corrente nulla il flusso stimato in uscita sull'asse d deve essere pari ad Λ_{mg} , mentre per l'asse q deve essere nullo essendo il motore utilizzato di tipologia IPM.

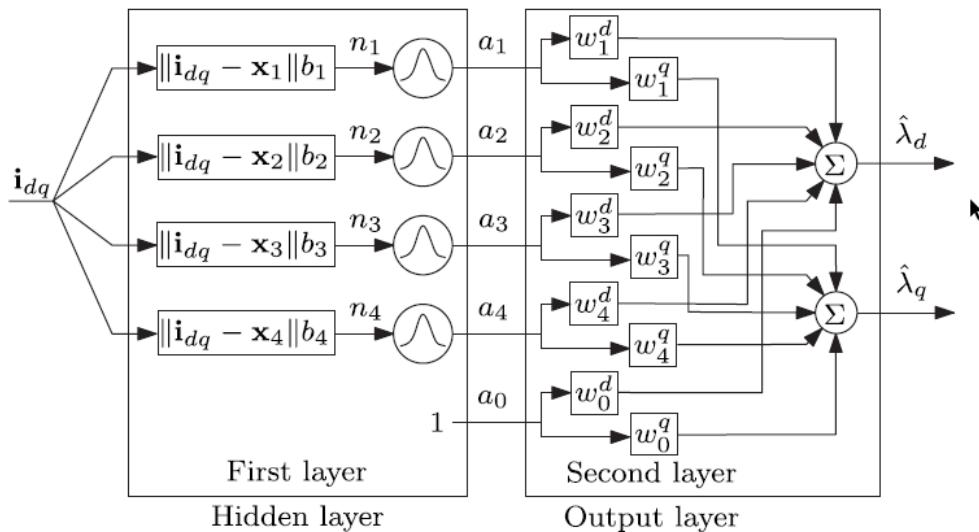


Figure 3: Rete RBF

$$n_k = \|\mathbf{i}_{dq} - \mathbf{x}_k\| b_i \quad k = 1 \dots K \quad (6)$$

$$a_k = e^{-n_k^2} \quad (7)$$

Per b_k si è scelto di utilizzare, secondo quanto citato in [2], un valore costante pari ad

$$b_k = \frac{\sqrt{K}}{2d_{\max}} \quad (8)$$

dove d_{\max} è quello definito nella figura(2).

Seguendo i calcoli svolti in [2][1], per J_d e J_q possiamo usare le matrici:

$$J_d = -w_{me} \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1K} \\ a_{21} & a_{22} & \dots & a_{2K} \\ \vdots & \vdots & \ddots & \vdots \\ a_{M1} & a_{M2} & \dots & a_{MK} \end{pmatrix} \quad (9)$$

$$J_q = +w_{me} \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1K} \\ a_{21} & a_{22} & \dots & a_{2K} \\ \vdots & \vdots & \ddots & \vdots \\ a_{M1} & a_{M2} & \dots & a_{MK} \end{pmatrix} \quad (10)$$

Dove i valori $a_{11} \dots a_{1K}$ sono le uscite dell'hidden layer quando in ingresso ho imposto il primo riferimento di corrente usato, e così via fino all'ultimo riferimento di corrente, ottenendo gli indici $a_{M1} \dots a_{MK}$.

4 DATA COLLECTION

Il dataset utilizzato per il training offline della rete è stato ricavato dal file di simulazione Plecs "Nonlinear_IPMSM_model.plecs". Allo scopo di mantenere la velocità elettrica del rotore costante durante l'acquisizione di M copie di valori di tensione su i due assi (ud, uq), quando in ingresso della rete vengono imposti M valori diversi di riferimenti di corrente id^*, iq^* , si è andato ad aggiungere al file di simulazione già esistente, un controllo di velocità, capace di generare una coppia attiva proporzionale all'errore di velocità ed agente sul modello meccanico del motore.

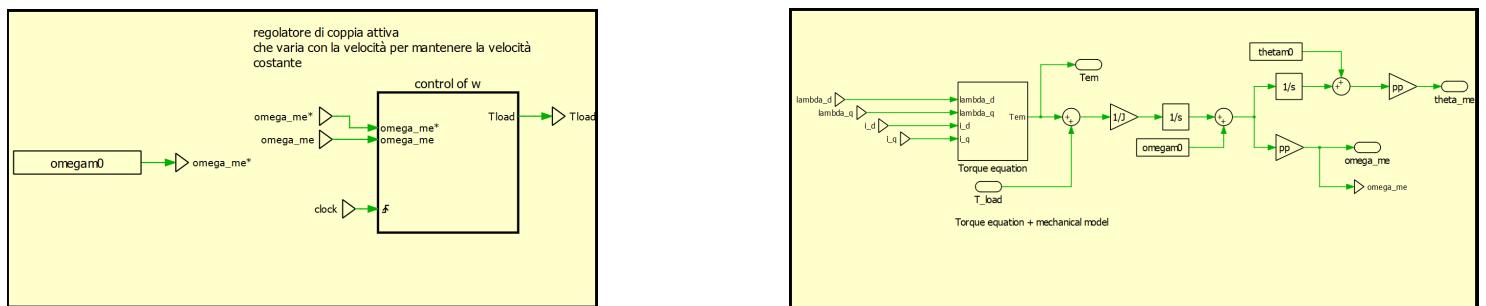


Figure 4: Controllo di velocità agente sul modello meccanico del motore.

Per imporre M valori diversi del riferimento di corrente, si è invece implementato lo script Plecs presente in Appendice B. Tale script, oltre a creare una mesh di M riferimenti di corrente equamente spaziati e ad imporli al sistema, si occupa di andare a leggere i rispettivi valori di tensione dopo un determinato tempo t quando il sistema è andato in convergenza, ad andarli a mediare in un intervallo Δt per ridurre gli effetti dei disturbi per poi salvarli su file "dataIn_Rett.txt", tali dati

² Il principio di località permette, dato un certo ingresso di corrente, di andare a modificare solamente una parte dei pesi nel layer di uscita grazie all'utilizzo delle funzioni gaussiane.

vengono successivamente letti ed elaborati nella fase di training dallo script Matlab "RBF_network.m" il cui codice è stato allegato al seguente documento. Tale script utilizza le classi "RBFn.m" e "olayer.m" per andare ad implementare rispettivamente l'hidden e l'output layer della rete.

Table 1: Parametri simulazione Plecs

	Valore
t	85ms
Δt	5ms

5 RISULTATI DELLA SIMULAZIONE

Table 2: Parametri simulazione Matlab

Parametro	Valore	Significato
K	81	Numero di funzioni Gaussiane
M	451	Numero dei riferimenti di corrente usati
W_{me}	50[rad/s]	Velocità meccanico elettrica
R_s	3,4[Ω]	Resistenza di statore
b	0,136	Bias primo layer
Λ_{mg}	0,22[Vs]	Flusso magnetico
$N_{training}$	10	Numero di cicli per il training

Data l'alta velocità di convergenza dei pesi, è bastato usare un $N_{training} = 10$ utilizzando lo stesso dataset ad ogni ciclo di training. Di seguito vengono fatti vedere i risultati più importanti ottenuti dalla simulazione con i parametri del modello usati.

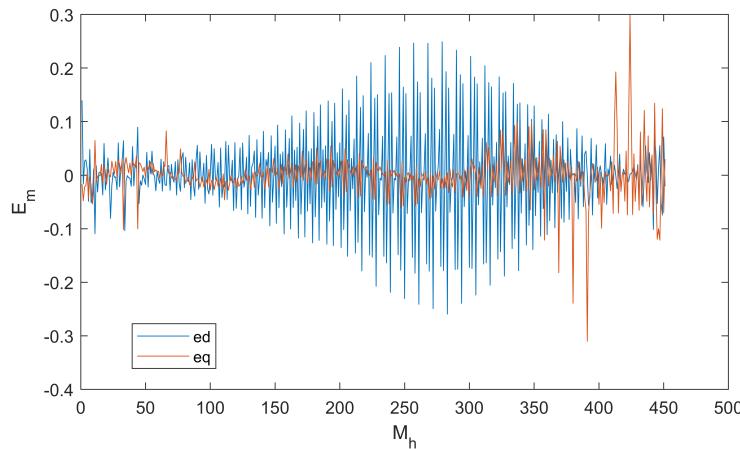


Figure 5: Funzione errore [V] in uscita dall'output layer dopo la convergenza della rete.

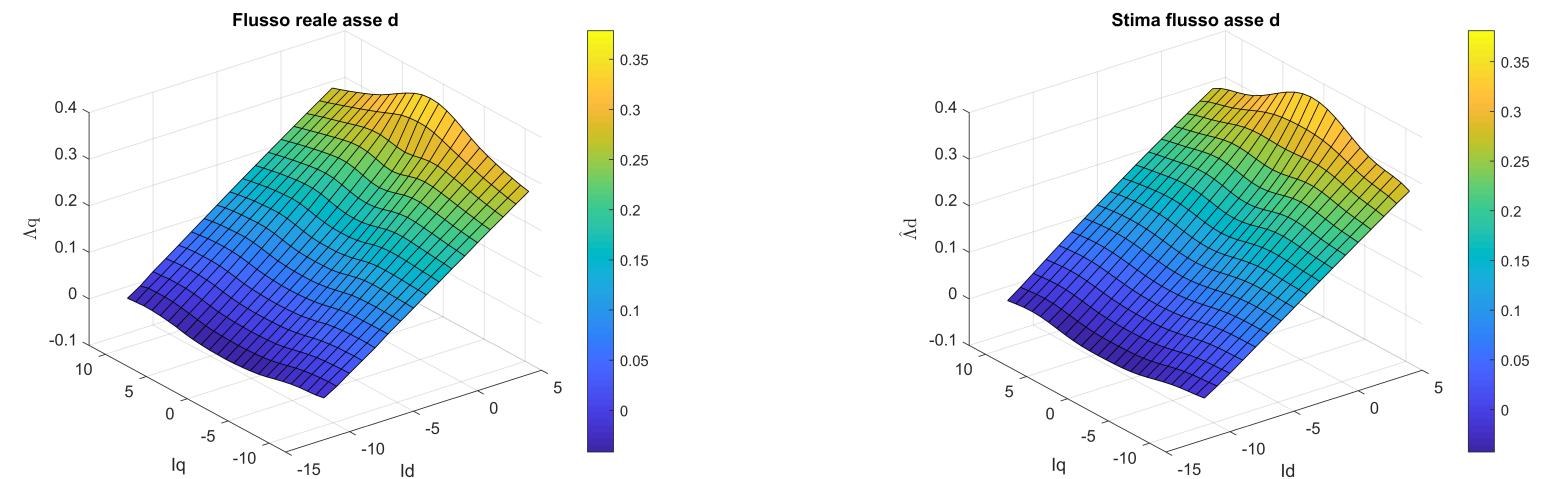


Figure 6: Confronto delle funzioni flusso stimato e flusso reale dell'asse d.

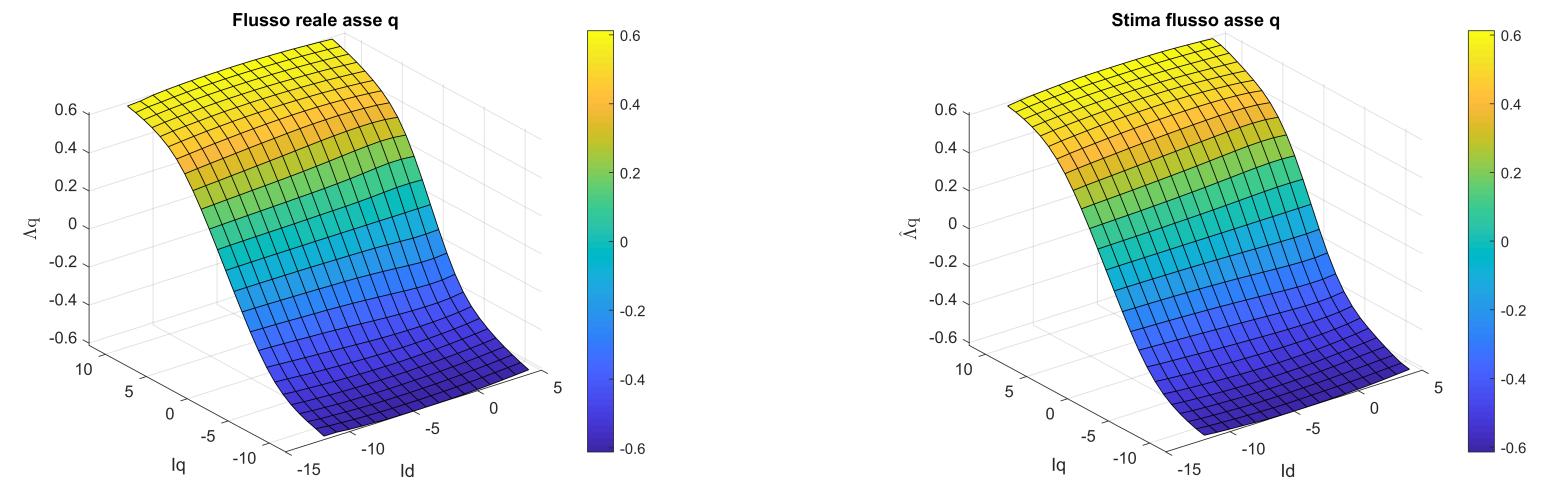


Figure 7: Confronto delle funzioni flusso stimato e flusso reale dell'asse q.

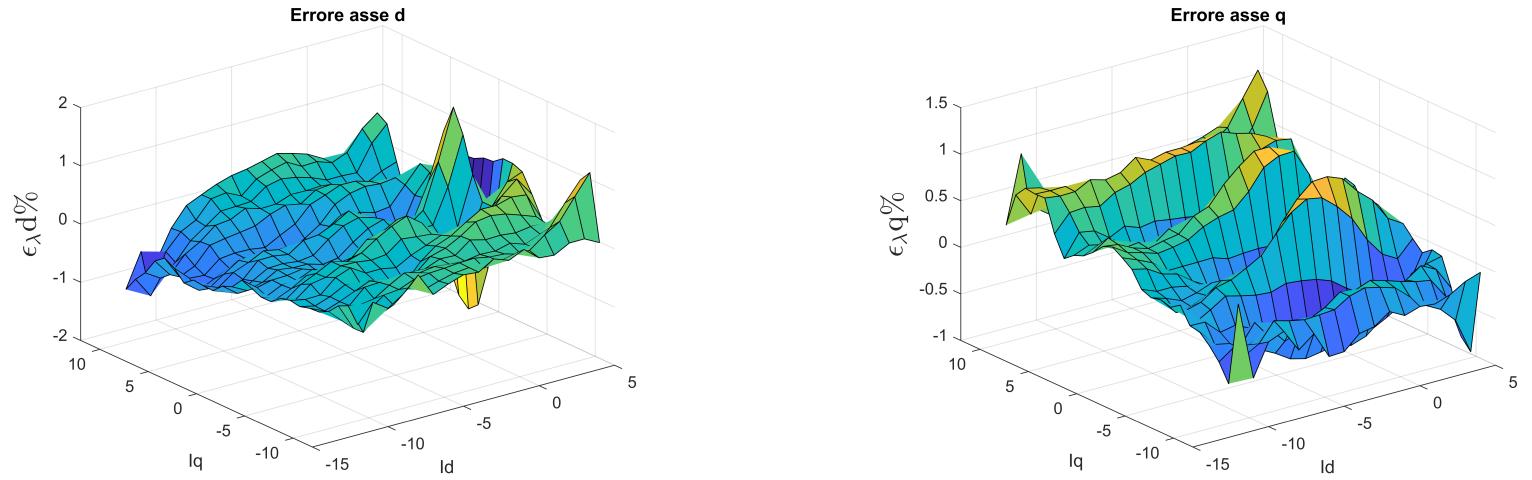


Figure 8: Funzione 3D della differenza del flusso stimato e quello reale in percentuale, sia per l'asse d che quello q. Come si può notare si ottiene un errore massimo pari al 1% sul flusso di asse q e ad 1.7% su flusso dell'asse d.

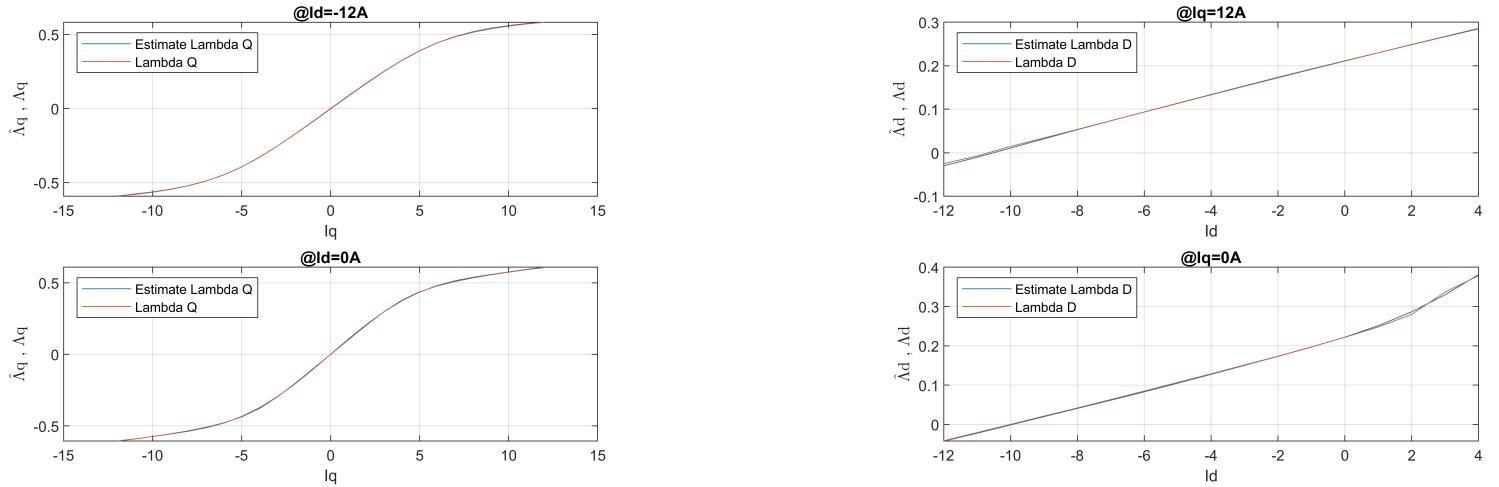


Figure 9: Confronto di slices del flusso stimato e di quello reale di asse d e q, avendo posto un valore costante per Iq ed Id rispettivamente nei due casi.

Infine si è fatta un'analisi del costo computazionale della rete, che come accennato precedentemente risulta essere molto più elevato di altre tecniche presenti nella letteratura [4] [3]. Tale funzione costo, determina quante operazioni bisogna fare, dato un valore di corrente, per determinare i valori dei flussi su i due assi, senza tener conto della fase di training.

Table 3: Costo computazionale, K=81 nella simulazione.

Operatore	Numero occorrenze
*	6K
±	5K
√	K

6 CONCLUSIONI

L'algoritmo usato risulta essere computazionalmente molto oneroso sia dal punto di vista delle risorse di calcolo che di quantità di memoria usata, basti pensare che per l'aggiornamento dei pesi della rete c'è bisogno di moltiplicare e ad invertire matrici di dimensione $N \times K$. Nel complesso ci si ritiene soddisfatti dei risultati ottenuti, avendo avuto modo di implementare quello che risulta essere attualmente un argomento di ricerca poco documentato dalla letteratura. Oltre tutto, l'algoritmo usato può essere utile per mappare qualsiasi tipo di funzione non lineare e dato l'aumento delle potenze di calcolo dei dispositivi, risulta essere una tecnica fruibile anche in altri campi.

REFERENCES

- [1] M. Zigliotto L. Ortombina, F. Tinazzi. Comprehensive magnetic modelling of internal pm synchronous motors through radial basis function networks. *IEEE*, 2016.
- [2] M. Zigliotto L. Ortombina, F. Tinazzi. Magnetic modeling of synchronous reluctance and internal permanent magnet motors using radial basis function networks. *IEEE*, 2018.
- [3] R. Petrella N. Bedetti, S. Calligaro. Stand-still self-identification of flux characteristics for synchronous reluctance machines using novel saturation approximating function and multiple linear regression. *IEEE*, 2016.
- [4] G. Pellegrino P. Pescetto. Automatic tuning for sensorless commissioning of synchronous reluctance machines augmented with high-frequency voltage injection. *IEEE*, 2018.

A SCRIPT MATLAB RETE RBF

Per chiarezza, si è preferito non copiare in questa relazione il codice dello script Matlab che và ad implementare la rete neurale, può essere visualizzato, insieme a tutti i suoi commenti, sul file main "RBF_network.m", il quale usa a sua volta le classi definite in "RBFn.m" ed "olayer.m" le quali sintetizzano rispettivamente l'hidden e l'output layer.

B SCRIPT PLECS DATA COLLECTION

In tale appendice c'è il codice dello script Plecs usato per impostare alla simulazione diversi valori per il riferimento di corrente e a salvare le tensioni misurate su file.

```

1 format short
2
3 mdl = plecs('get', '', 'CurrentCircuit');
4 scopepath = [mdl '/I_Cbus'];
5 scopew = [mdl '/w_theta'];
6 idqStructure = struct('varId', o,'varIq',o);
7 varStructure = struct('ModelVars', idqStructure);
8 plecs('scope', scopepath, 'ClearTraces');
9 plecs('scope', scopew, 'ClearTraces');

10 %Apertura file per scrittura dei dati letti
11 f=fopen("dataIn_Rett.txt",'w');

12 %Generazione mesh di corrente di forma rettangolare
13 x=-12:0.4:4; %id
14 y=-12:2.4:12; %iq
15 [xx,yy]=meshgrid(x,y);
16 i (: ,1)=reshape(xx,[],1);
17 i (: ,2)=reshape(yy,[],1);

18
19
20
21
22 for ix = 1:length(i)
    %Impongo n valori per i riferimenti di corrente e simulo
23     varStructure.ModelVars.varId = i(ix,1); %id
24     varStructure.ModelVars.varIq = i(ix,2); %iq
25     plecs('simulate', varStructure);
26     idd=i(ix,1);
27     iqq=i(ix,2);

28
29
30 %Lettura dei rispettivi valori di tensione e velocita (quest'ultima per
31 %controllare che sia costante nell'intervallo)
32 data=plecs('scope', scopepath, 'GetCursorData', [0.085 0.09], 'mean');
33 dataw=plecs('scope', scopew, 'GetCursorData', [0.085 0.09], 'mean');

34 %Ne viene eseguita inoltre la media per ridurre gli effetti dei disturbi
35 m1=data.cursorData{1}{1}.mean;
36 m2=data.cursorData{2}{1}.mean;
37 mw=dataw.cursorData{2}{1}.mean;
38 dataw.cursorData{2}{1}=

39
40 %Salvataggio su file
41 fprintf(f, "%d,Id:%.3f,Iq:%.3f,Ud:%.3f,Uq:%.3f,wme:%.3f,\n",ix,idd,iqq,m1,m2
42 ,mw);
43 end
44 fclose(f);

```