

POLITECNICO DI TORINO

Corso di Laurea Magistrale in Ingegneria Elettrica

Tesi di Laurea Magistrale

Progetto di Motori Sincroni a Riluttanza in SyR-e e Motor-CAD



Relatore:

Prof. Gianmario PELLEGRINO

Candidato:

Antonio MANDARANO

Correlatore:

Ing. Simone FERRARI

Anno accademico 2019-2020

*"L'unico modo per fare un ottimo lavoro è amare quello che fate.
Se non avete ancora trovato ciò che fa per voi, continuate a cercare."*

Steve Jobs

Ringraziamenti

Prima di procedere con il lavoro di tesi, desidero dedicare qualche riga a tutte le persone che con il loro supporto mi sono state vicine in questi anni di università.

Vorrei innanzitutto ringraziare il Professore Gianmario Pellegrino, relatore di questa tesi, per la sua disponibilità e gentilezza, la porta del suo ufficio è sempre stata aperta quando più ne ho avuto bisogno.

Un ringraziamento particolare va al dottorando Simone Ferrari, grazie per i preziosi consigli e per tutto il tempo dedicatomi in questi mesi.

Vorrei ringraziare infinitamente mamma e papà, per aver permesso che il mio sogno si realizzasse e per avermi supportato in questi anni di studio con innumerevoli sacrifici senza avermi mai fatto mancare nulla.

Un grazie di cuore va alla mia cara nonna Nunziata, per il suo immenso amore.

Grazie a mia sorella Letizia per avermi sempre incoraggiato e per esserci sempre stata. Grazie a Lillo e ai miei nipotini Nunzia, Elisabetta e Lorenzo per il loro affetto.

Grazie a Nuccio, Rossella e Sergio per aver avuto sempre fiducia in me.

Un ringraziamento carico d'amore va a te Alessia che con il tuo sorriso illumini le mie giornate. Grazie per avermi supportato e sopportato con pazienza quando più ne ho avuto bisogno, grazie per aver letto e corretto ogni riga di questa tesi come fosse la tua e per essermi sempre stata accanto dandomi la forza di affrontare qualsiasi cosa. Senza di te tutto sarebbe stato più difficile.

Grazie ai miei amici piemontesi Emanuele, Rossana, Fabrizio e Elena, per tutti i bei momenti passati insieme.

Grazie a Davide, compagno d'avventura, sempre pronto a rispondere alle mie innumerevoli chiamate giornaliera e ad aiutarmi senza chiedere nulla in cambio.

Ringrazio Giorgio e Paolo, per la piacevole compagnia durante questo periodo di tesi.

Infine ringrazio Massimo, Gaetano, Chiara, Giuseppe, Andrea Comino, Francesco e Andrea Nicastro per aver reso più piacevoli questi anni di università.

Sommario

Negli ultimi anni i vari settori industriali ripongono sempre più attenzione verso i motori sincroni a riluttanza (*SyR*), impegnandosi nella ricerca e sviluppo di strumenti adatti che permettano di ridurre i tempi di progettazione.

I motori *SyR* permettono di sostituire in modo vantaggioso le tipologie di motori ad oggi più comuni, come il motore ad induzione, in quanto risultano più efficienti, affidabili e la loro produzione richiede costi minori.

Tuttavia, nonostante gli studi condotti su questa tipologia di motori, non sono ancora state definite delle procedure di progettazione standard. Ad oggi numerosi sono i software adoperati in tale ambito, tra questi vi sono *SyR-e* e *Motor-CAD*, programmi efficienti ma ancora non del tutto completi. Con *SyR-e*, infatti, non è possibile effettuare simulazioni strutturali e termiche oltre che calcolare in modo accurato le perdite nel ferro, mentre in *Motor-CAD* non è presente una funzione che permetta di realizzare un dimensionamento iniziale della geometria di macchina partendo dalle prestazioni desiderate.

L'obiettivo di questa tesi è quello di ovviare a tali mancanze attraverso lo sviluppo di uno strumento di progettazione completo basato sul lavoro sinergico tra *SyR-e* e *Motor-CAD*. Per interfacciare i due software così da sfruttare al massimo le potenzialità di ciascuno di questi nelle fasi di progetto e simulazione, sono stati sviluppati script *Matlab* ed è stata introdotta una nuova finestra di collegamento nell'interfaccia grafica di *SyR-e*.

Infine, il lavoro è stato validato confrontando i dati forniti dalle simulazioni in *Motor-CAD* con quelli sperimentali o ottenuti con altri software.

I risultati finali sono da considerarsi soddisfacenti e pongono le basi per eventuali sviluppi futuri.

Indice

Elenco delle tabelle	IV
Elenco delle figure	V
1 Introduzione	1
1.1 Generalità sui motori sincroni a riluttanza	1
1.1.1 Descrizione del motore <i>SyR</i>	2
1.1.2 Modello del motore <i>SyR</i> in assi <i>dq</i>	4
1.2 <i>SyR-e</i> : funzionalità e caratteristiche	5
1.2.1 <i>Graphical User Interface</i>	7
1.2.2 <i>SyrmDesign</i> : equazioni di progetto	7
1.2.3 Finestre della <i>GUI</i>	9
1.2.4 Altre funzionalità di SyR-e	19
1.3 Motor-CAD: funzionalità e caratteristiche	20
1.4 Confronto tra SyR-e e Motor-CAD	22
1.5 Obiettivi della tesi	24
2 Interfaccia tra SyR-e e Motor-CAD	26
2.1 Nuova finestra Motor-CAD nella GUI SyR-e	26
2.1.1 Export della geometria di macchina	27
Export degli avvolgimenti di statore	30
Export del disegno 2-D	31
2.1.2 Simulazioni magnetiche in Motor-CAD	32
2.1.3 Esportazione delle mappe di flusso in Motor-CAD	34
2.1.4 Simulazioni termiche in Motor-CAD	34

2.2	Esempio: motore RawPower	36
3	Simulazioni magnetiche	39
3.1	Descrizione del modulo EMag e Lab di Motor-CAD	39
3.2	Calcolo e manipolazione delle mappe di flusso	42
3.3	Verifica dell'errore su un singolo punto di lavoro	44
3.4	Modelli di calcolo delle perdite nel ferro	46
3.4.1	Perdite nel ferro in Motor-CAD	46
3.4.2	Perdite nel ferro in SyR-e (MagNet)	47
3.4.3	Stima dei coefficienti	47
3.5	Confronto delle mappe di efficienza e di perdita	49
4	Simulazioni meccaniche	53
4.1	Progettazione meccanica in SyR-e	53
4.2	Descrizione modulo meccanico in Motor-CAD	55
4.3	Validazione del progetto sviluppato in SyR-e	56
4.4	Macchina ad alta velocità	57
5	Simulazioni termiche	62
5.1	Descrizione del modulo termico in Motor-CAD	62
5.2	Modello della carcassa di riferimento	64
5.3	Costruzione del ciclo di lavoro considerando la coppia	65
5.4	Costruzione del ciclo di lavoro considerando le perdite	68
5.5	Confronto dei valori di temperatura simulati e sperimentali	69
6	Conclusioni	71
A	Script per l'esportazione della macchina da SyR-e a Motor-CAD	73
B	Script per l'impostazione delle simulazioni magnetiche singolo punto	82
C	Script per l'importazione in Motor-CAD delle mappe di flusso calcolate in SyR-e	93

D Script per impostare e simulare il modello termico in Motor-CAD 97

Bibliografia 103

Elenco delle tabelle

1.1	Disposizione degli avvolgimenti in <i>SyR-e</i>	14
1.2	Confronto funzionalità tra <i>SyR-e</i> e <i>Motor-CAD</i>	23
2.1	Spedifiche motore <i>RawPower</i>	36
3.1	Confronto dei risultati ottenuti mediante l'utilizzo dei due software.	45
3.2	Coefficienti stimati per il modello di perdita di <i>Steimetz</i> in <i>Motor-CAD</i>	49
3.3	Coefficienti stimati per il modello di perdita in <i>SyR-e (MagNet)</i>	49
3.4	Valori di resistenza forniti dai due software.	51
4.1	Proprietà meccaniche del lamierino <i>M600 - 50A</i>	56
4.2	Dimensioni iniziali e finali dei ponticelli.	58
4.3	Confronto delle grandezze meccaniche ottenute.	60

Elenco delle figure

1.1	Esempio di un motore <i>SyR</i>	2
1.2	Linee di campo dell'induzione magnetica generata iniettando correnti in asse d (a) e in asse q (b) [9].	3
1.3	Andamento delle induttanze (a) e dei flussi (b) in funzione della corrente.	3
1.4	Polo di rotore di una macchina <i>SyR</i>	5
1.5	Scambio di dati tra <i>SyR-e</i> e <i>FEMM</i> [7].	6
1.6	<i>GUI</i> - schermata <i>Main Data</i> [3].	7
1.7	Geometrie di rotore supportate in <i>SyR-e</i> [3].	8
1.8	<i>SyR-e</i> : piano di progettazione parametrica [3].	9
1.9	<i>GUI</i> - schermata <i>Geometry</i> [3].	10
1.10	Definizione dei parametri α e hc [7].	10
1.11	<i>GUI</i> - schermata <i>Options</i> [3].	11
1.12	Stima della temperatura degli avvolgimenti in funzione del parametro k_j	12
1.13	Dimensionamento dei ponticelli radiali.	12
1.14	<i>GUI</i> - schermata <i>Windings</i> [3].	13
1.15	Tipologie di cava di statore.	13
1.16	<i>GUI</i> - schermata <i>Materials</i> [3].	15
1.17	Curva $B-H$ del materiale M600-50A, presente nella libreria di <i>SyR-e</i>	15
1.18	<i>GUI</i> - schermata <i>Optimization</i> [3].	16
1.19	Esempio dei risultati del processo di ottimizzazione: Fronte di <i>Pareto</i> [15].	17
1.20	<i>GUI</i> - schermata <i>Simulation</i> [3].	17

1.21	Esempio dei risultati ottenuti simulando un singolo punto di lavoro.	18
1.22	Esempio dei risultati del calcolo delle mappe di flusso [3].	19
1.23	Fasi di progetto di un motore <i>SyR</i> mediante l'utilizzo integrato di <i>SyR-e</i> e <i>Motor-CAD</i>	24
2.1	<i>GUI</i> - schermata <i>Motor-CAD</i>	27
2.2	Schema a blocchi: comando <i>Export.mot</i>	28
2.3	Struttura del file (<i>.mat</i>).	29
2.4	Risultati ottenuti mediante la funzione <i>Export.mot</i>	30
2.5	Confronto dei disegni prodotti con le funzioni (a) e (b).	31
2.6	Rappresentazione degli assi <i>dq</i> in <i>SyR-e</i> (rosso) e <i>Motor-CAD</i> (blu).	32
2.7	Esempio dei risultati ottenuti utilizzando la funzione <i>Emag sim</i>	33
2.8	Impostazione dei parametri termici nella <i>GUI</i>	34
2.9	Esempio dei risultati ottenuti utilizzando la funzione <i>Therm Sim</i>	35
2.10	Risultati ottenuti con la funzione <i>draw_motor_in_MCAD</i>	37
2.11	Zoom del rotore.	38
2.12	Riconoscimento regioni <i>FEA</i>	38
3.1	<i>Motor-CAD</i> - modulo <i>Lab</i>	41
3.2	Confronto tra le curve di flusso sperimentali e quelle calcolate in <i>Motor-CAD</i>	43
3.3	Confronto tra le curve di flusso sperimentali e quelle calcolate in <i>SyR-e</i>	43
3.4	Andamento <i>MTPA</i>	44
3.5	Confronto dei risultati ottenuti utilizzando <i>SyR-e</i> e <i>Motor-CAD</i>	45
3.6	Confronto delle cifre di perdita.	48
3.7	Confronto delle mappe di efficienza sul piano coppia-velocità.	50
3.8	Confronto delle perdite <i>Joule</i> sul piano coppia-velocità.	50
3.9	Confronto delle perdite nel ferro sul piano coppia-velocità.	51
3.10	Confronto della coppia e delle perdite nel ferro lungo l' <i>MTPA</i> di corrente.	52
4.1	Definizione della geometria delle barriere di flusso [14].	54
4.2	<i>Motor-CAD</i> - modulo <i>Mechanical</i>	55
4.3	<i>Stress Analysis</i> - <i>Motor-CAD</i>	56

4.4	<i>Stress Analysis - SolidWorks.</i>	57
4.5	Simulazione meccanica sul lamierino iniziale.	58
4.6	Modifica del lamierino di rotore.	59
4.7	Simulazione meccanica sul lamierino modificato.	59
4.8	Ciclo iterativo per la progettazione meccanica.	60
4.9	Simulazione meccanica sul lamierino modificato.	61
5.1	<i>Motor-CAD</i> - modulo <i>Thermal.</i>	63
5.2	Modello bidimensionale del motore di riferimento.	64
5.3	Modello tridimensionale del motore di riferimento.	65
5.4	Rappresentazione della variazione di coppia durante un intero ciclo di lavoro.	66
5.5	Andamento della coppia durante una fase del ciclo di lavoro alla velocità di 1222 <i>rpm.</i>	67
5.6	Rappresentazione della variazione delle perdite durante un intero ciclo di lavoro.	68
5.7	Confronto tra temperature simulate e sperimentali.	69

Capitolo 1

Introduzione

1.1 Generalità sui motori sincroni a riluttanza

Il motore sincrono a riluttanza, noto in letteratura come *Synchronous Reluctance Motor* (*SyR*), è un motore molto utilizzato in vari settori industriali in quanto offre numerosi vantaggi in termini di efficienza, affidabilità e basso costo di produzione [8].

Negli ultimi anni, i motori sincroni a riluttanza vengono utilizzati come alternativa al motore a induzione (*IM* - *Induction Motor*) e alle macchine sincrone a magneti permanenti (*PMSM* - *Permanent Magnet Synchronous Motor*), per i seguenti motivi [5]:

- costo di produzione inferiore, in quanto nel rotore non vi sono avvolgimenti e magneti, rispettivamente presenti nel motore *IM* e nella macchina *PMSM*;
- processo di produzione semplificato, poiché la punzonatura dei lamierini risulta più semplice rispetto alla produzione della gabbia rotorica in alluminio o rame del motore *IM*;
- perdite a rotore ridotte, grazie all'assenza della gabbia. Questo gioca un ruolo importante nel raffreddamento della macchina, in quanto nelle macchine *SyR* l'elemento che si surriscalda maggiormente è lo statore, più semplice da raffreddare rispetto al rotore;

- maggiore efficienza rispetto al motore *IM*, data l'assenza del rotore a gabbia e delle relative perdite [8].

1.1.1 Descrizione del motore *SyR*

La figura 1.1 rappresenta un esempio di motore *SyR*.

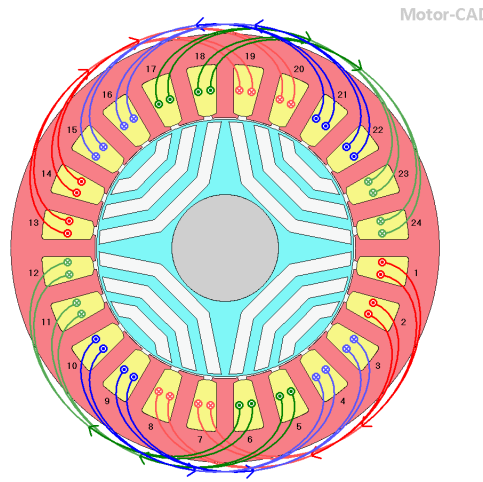


Figura 1.1. Esempio di un motore *SyR*.

Lo statore presenta un avvolgimento trifase distribuito in modo del tutto analogo a quello di un motore a induzione. Gli avvolgimenti di statore sono alimentati da inverter e, in genere, con controllo vettoriale di corrente.

La particolarità delle macchine sincrone a riluttanza è ascrivibile alla geometria di rotore. Il rotore è anisotropo¹, data la presenza delle barriere di flusso visibili (in bianco) in figura 1.1. La produzione di coppia in questo tipo di macchine nasce proprio dalla presenza nel rotore di percorsi magnetici di diversa riluttanza (figura 1.2) che danno origine alla cosiddetta coppia di riluttanza.

Uno dei problemi che si riscontra nello studio di tali macchine è il comportamento magnetico non lineare causato dalla saturazione magnetica del lamierino di statore e di rotore.

¹caratteristiche dipendenti dalla direzione lungo la quale vengono considerate

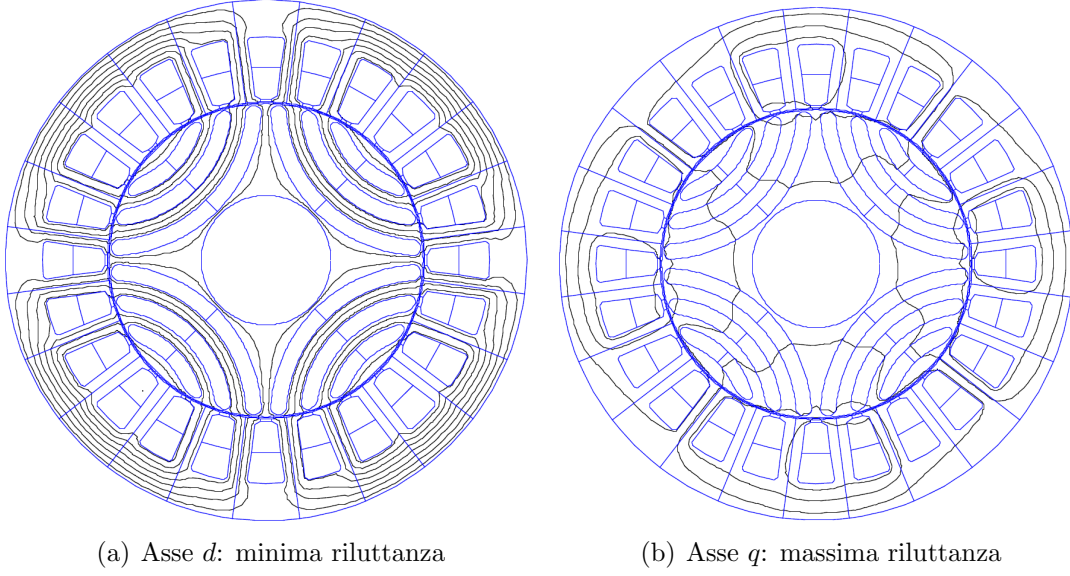


Figura 1.2. Linee di campo dell'induzione magnetica generata iniettando corrente in asse d (a) e in asse q (b) [9].

La presenza di saturazione magnetica non consente di studiare le prestazioni della macchina utilizzando un modello lineare, è necessario dunque un modello non lineare che tenga in considerazione la variazione delle induttanze con la corrente (figura 1.3(a)). In figura 1.3(b) vengono rappresentati i flussi in funzione della corrente. La variazione di pendenza dei flussi è causata dalla non costanza delle induttanze L_d e L_q .

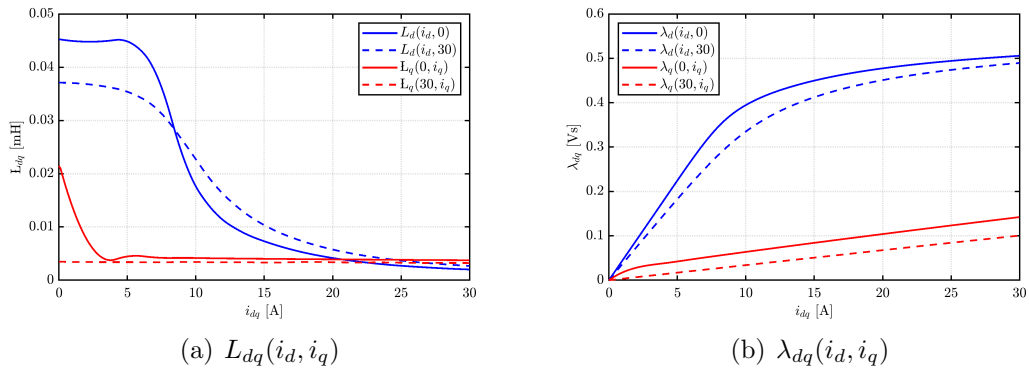


Figura 1.3. Andamento delle induttanze (a) e dei flussi (b) in funzione della corrente.

La presenza di saturazione magnetica implica che lo studio della prestazione della macchina debba basarsi sull'utilizzo dell'analisi agli elementi finiti (*FEA - Finite Element Analysis*). L'utilizzo dell'analisi *FEA* permette di effettuare una stima accurata delle prestazioni, a spese di un elevato tempo di calcolo [8]-[4].

1.1.2 Modello del motore *SyR* in assi dq

Lo studio delle macchine trifase viene effettuato in assi dq , attraverso l'utilizzo della trasformata di *Park* [11]. Nella macchina a riluttanza viene definito come asse d quello che favorisce il passaggio del flusso di macchina come mostrato in figura 1.2. L'asse q invece viene orientato nella direzione in cui la macchina presenta la massima riluttanza.

Definito p come numero di paia poli, la coppia elettromagnetica del motore *SyR* si esprime come (1.1) [8] :

$$T = \frac{3}{2}p(\lambda_d i_q - \lambda_q i_d) \quad (1.1)$$

Esprimendo i flussi come in (1.2) (1.3):

$$\lambda_d = L_d i_d \quad (1.2)$$

$$\lambda_q = L_q i_q \quad (1.3)$$

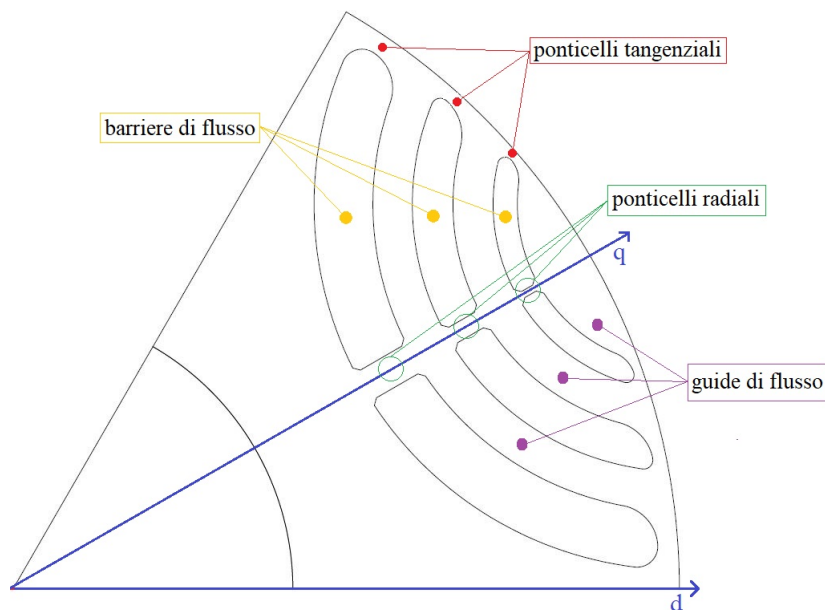
L'espressione di coppia diventa (1.4):

$$T = \frac{3}{2}p(L_d - L_q)i_d i_q \quad (1.4)$$

Dall'equazione di coppia (1.4) si evince che in fase di progetto bisogna estremizzare la differenza $(L_d - L_q)$ così da ottenere la massima coppia a parità di corrente.

In figura 1.4 viene mostrata la geometria di un polo di rotore.

Per massimizzare il più possibile la coppia sarebbe conveniente non avere i ponticelli tangenziali lungo la geometria di rotore. La presenza di questi è però necessaria per garantire la fattibilità meccanica del rotore. Lo spessore dei ponticelli è calcolato in base alla massima velocità operativa della macchina, per garantire che il rotore resista allo sforzo centrifugo. Per le macchine a bassa velocità lo spessore minimo è dettato dalle tolleranze di taglio dei lamierini e di solito non è mai inferiore allo

Figura 1.4. Polo di rotore di una macchina *SyR*.

spessore del lamierino (es. 0.35 - 0.5 *mm*).

Nelle macchine ad alta velocità di rotazione è indispensabile aggiungere i ponticelli radiali (figura 1.4), perché il solo ispessimento dei ponticelli tangenziali non è sufficiente a sostenere la struttura. Questo introduce un peggioramento dal punto di vista magnetico ma un miglioramento per la meccanica del rotore, in quanto i ponticelli radiali sono strutturalmente più efficaci.

Poiché i ponticelli sono molto sottili rispetto agli altri percorsi in ferro della macchina tendono a saturare facilmente, presentano dunque valori di riluttanza incrementale simili a quelli dell'aria (figura 1.3). In figura 1.3(b) è possibile osservare che il flusso in asse q presenta una forte pendenza iniziale causata dalla presenza del lamierino ferromagnetico, man mano che il valore della corrente aumenta il lamierino satura e quindi si ha una drastica diminuzione della pendenza del flusso.

1.2 *SyR-e*: funzionalità e caratteristiche

Sul mercato esistono diversi software per la progettazione delle macchine elettriche, in questo elaborato ne vengono presi in considerazione due: *SyR-e*, descritto

in questo paragrafo, e *Motor-CAD*, che verrà descritto al paragrafo 1.3.

SyR-e, acronimo di *Synchronous Reluctance - evolution*, è un codice open source² nato dalla collaborazione tra il Politecnico di Torino e il Politecnico di Bari.

Il programma si basa sull'utilizzo di *Matlab/Octave* e *FEMM*, per la progettazione di motori *SyR* [7]. *SyR-e* viene utilizzato per progettare macchine sincrone a riluttanza mediante l'utilizzo dell'analisi *FEA* associato ad equazioni di progetto e di algoritmi di ottimizzazione multi-obiettivo.

Il software è sviluppato in ambiente *Matlab* e si appoggia al client gratuito *FEMM* [1] per le simulazioni magnetiche. I risultati vengono importati su *Matlab* e rielaborati, come mostrato in figura 1.5.

In *Matlab*, attivando la funzione di *Parallel Computing*, è possibile eseguire più simulazioni *FEA* in contemporanea. Il numero di simulazioni che si riescono ad eseguire in parallelo dipende dal numero di core del processore. Grazie a questa funzione è possibile ridurre notevolmente il tempo di calcolo di alcune simulazioni, come nel caso di ottimizzazioni o identificazioni magnetiche.

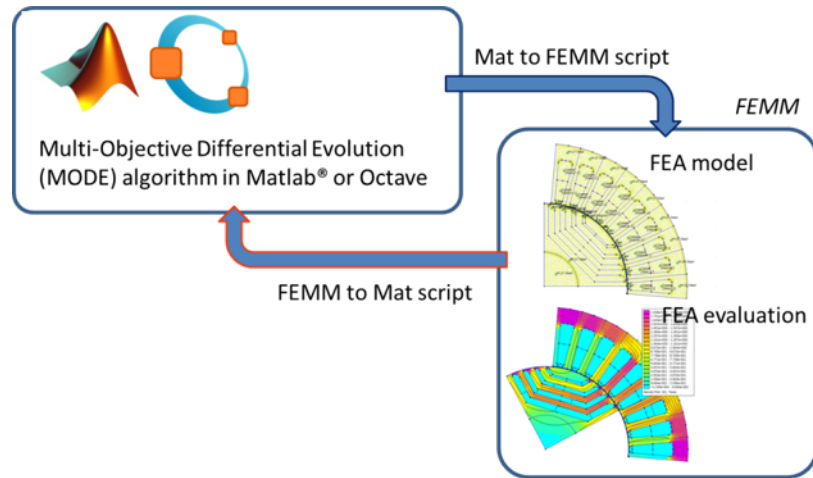


Figura 1.5. Scambio di dati tra *SyR-e* e *FEMM* [7].

²software dotato di una licenza attraverso cui i possessori dei diritti ne permettono la modifica e l'utilizzo.

1.2.1 Graphical User Interface

Il software si presenta all'utente mediante un'interfaccia grafica *GUI*, acronimo di *Graphical User Interface*, mostrata in figura 1.6. Questa interfaccia non è supportata in *Octave*.

La *GUI* permette all'utente di modificare rapidamente i diversi parametri relativi alla fase di progetto e simulazione.

All'avvio dell'interfaccia grafica viene caricata di default una geometria di tipo *Circular*, ovvero con rotore *SyR* con barriere a profilo circolare. Agendo sul menù a tendina *Type of rotor* è possibile selezionare la tipologia di rotore desiderata tra quelle presenti in figura 1.7. Oltre al rotore *SyR Circular*, sono presenti i rotor *SyR* con barriere a segmenti (*Seg* e *I-Seg*) e fluide, e i motori a magneti permanenti superficiali (*SPM*) ed interni (*V-Type*) (figura 1.7).

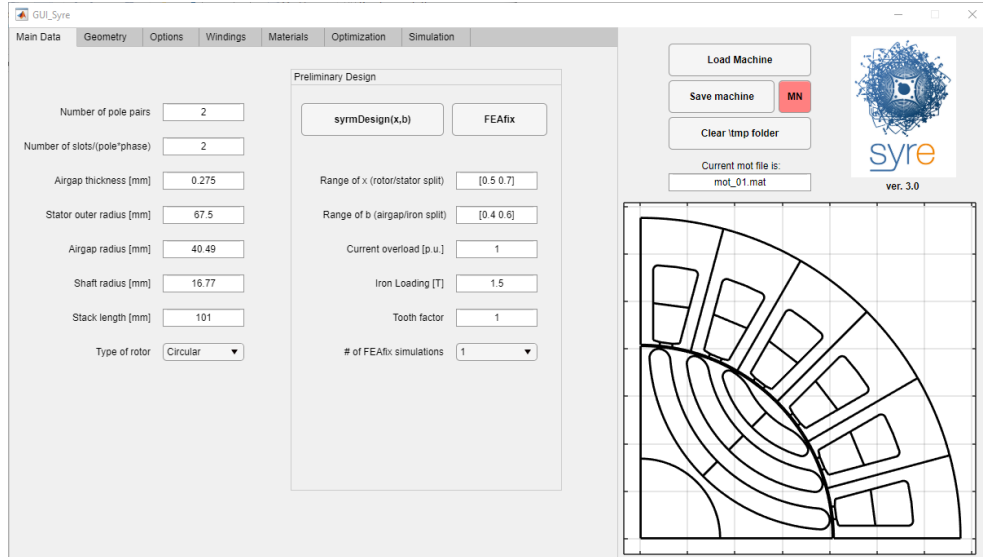


Figura 1.6. *GUI* - schermata *Main Data* [3].

1.2.2 SyrmDesign: equazioni di progetto

SyR-e è dotato di uno strumento di *Preliminary Design* molto utile per effettuare un dimensionamento iniziale della macchina. Utilizzando il comando *syrmDesign(x,b)*, viene visualizzato il piano di progettazione parametrica (x,b) con le curve di livello relative alla coppia e al fattore di potenza come mostrato in figura

1.8 [8].

I parametri sugli assi del piano di progetto sono:

- $x = \frac{r}{R}$ rapporto tra raggio esterno di rotore e statore;
- $b = \frac{B_g}{B_{Fe}}$ rapporto tra la densità di flusso nel gioco di statore e la densità di flusso dispersa in aria.

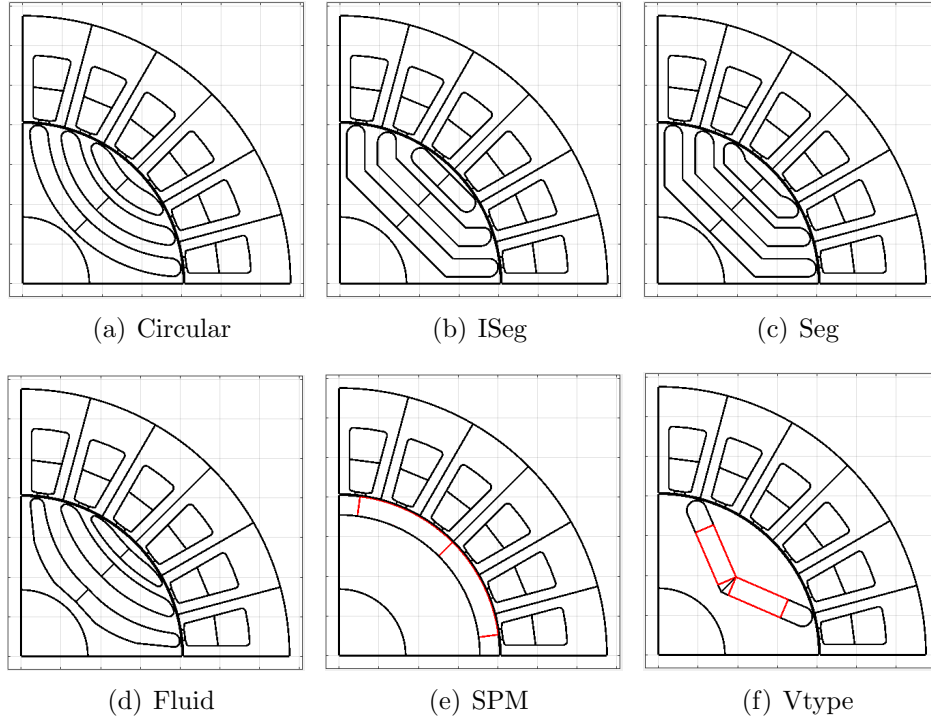


Figura 1.7. Geometrie di rotore supportate in *SyR-e* [3].

Ciascun punto del piano rappresenta una diversa geometria di macchina in funzione di (x, b) , con prestazioni differenti in termini di coppia e fattore di potenza.

Gli errori di approssimazione del modello analitico si possono correggere utilizzando il comando *FEAfix* basato sull'utilizzo dell'analisi *FEA* [8]. Per correggere l'errore vengono calcolati dei fattori correttivi relativi al flusso in assi dq , successivamente applicati al piano. La geometria della macchina resta invariata [8].

È possibile definire inoltre il numero di punti da simulare in *FEA*, le scelte consentite sono: 1,4,5,8.

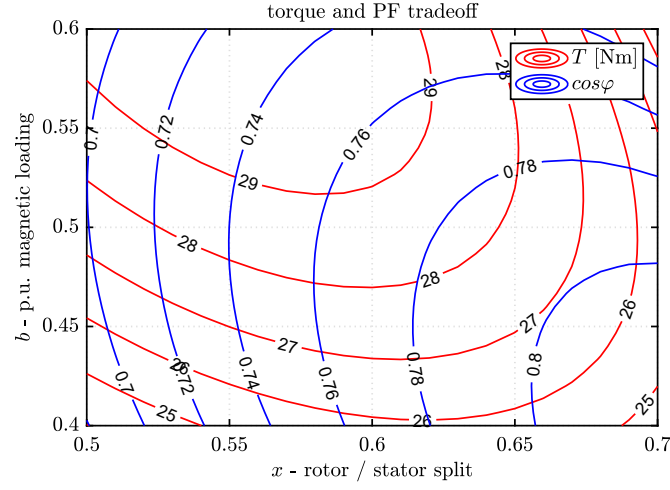


Figura 1.8. *SyR-e*: piano di progettazione parametrica [3].

1.2.3 Finestre della *GUI*

Di seguito viene riportata una breve descrizione delle finestre presenti nella *GUI* [7]:

- Main Data

In questa sezione è possibile impostare il numero di coppie polari, il numero di cave di statore per polo per fase q , lo spessore del traferro, il raggio esterno dello statore, il raggio esterno del rotore, il raggio dell'albero, la lunghezza dell'albero e il tipo di geometria del rotore (Circular, Seg, ISeg, Fluid, SPM e Vtype).

- Geometry

La finestra si presenta divisa in due sezioni in cui inserire rispettivamente i parametri dello statore e quelli del rotore (figura 1.9).

Per quanto riguarda lo statore è possibile modificarne la geometria impostando la lunghezza e la larghezza del dente, il tipo di cava, l'apertura di cava e altri parametri. Nell'altra sezione è possibile definire il numero e la geometria delle barriere di flusso inserendo i seguenti dati rotorici: numero di barriere di flusso, larghezza dei ponticelli radiali, larghezza delle barriere di flusso,

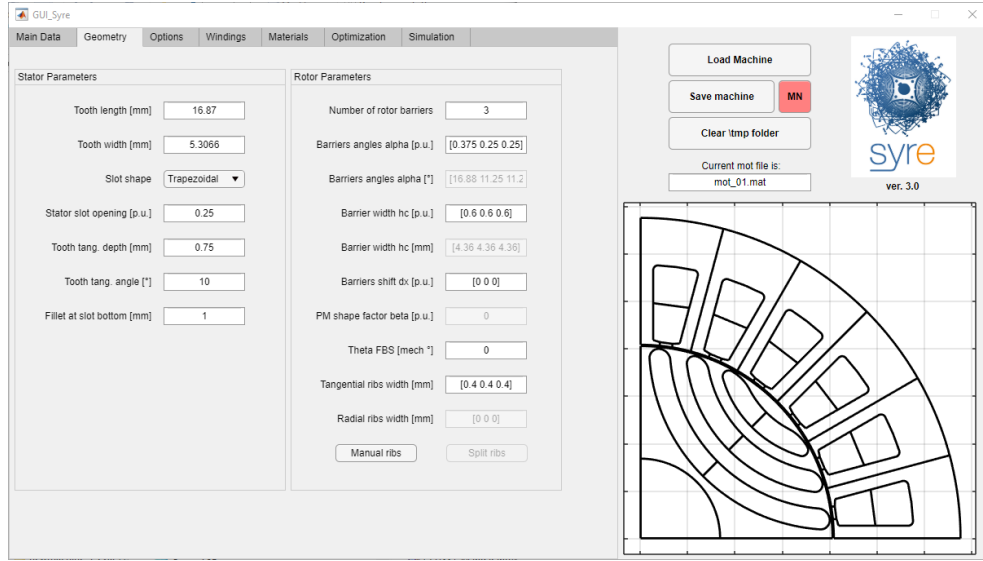


Figura 1.9. GUI - schermata *Geometry* [3].

traslazione delle barriere di flusso rispetto all'asse q (dx) e angolo di apertura delle barriere di flusso α (figura 1.10).

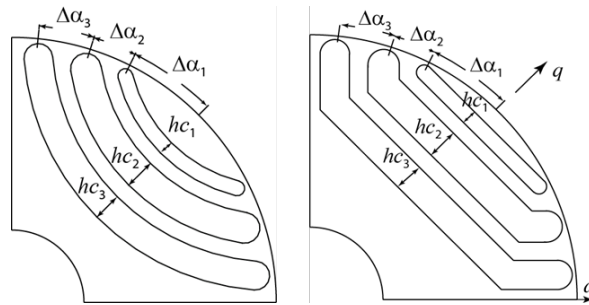


Figura 1.10. Definizione dei parametri α e hc [7].

- Options

In questa finestra (figura 1.11) è possibile impostare parametri di diversa natura.

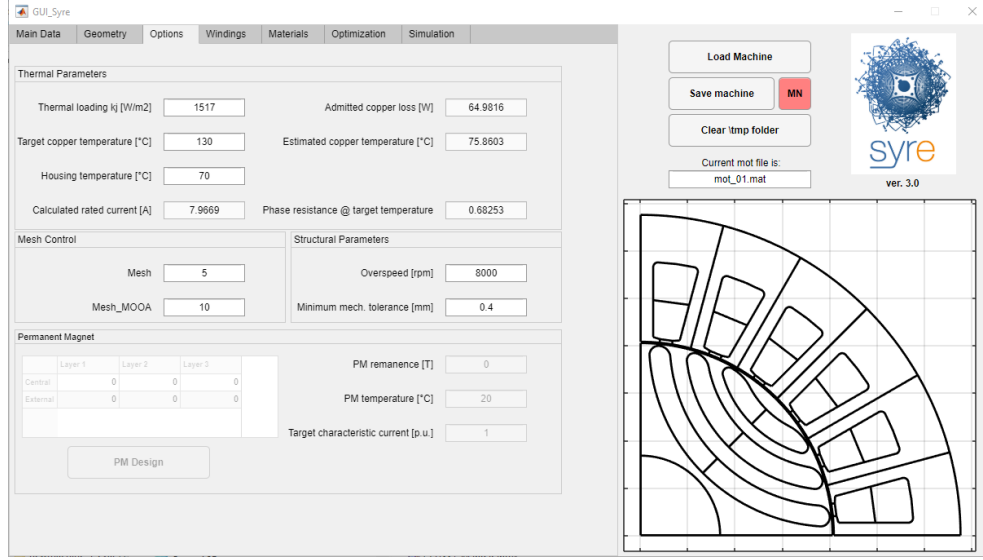


Figura 1.11. GUI - schermata *Options* [3].

Definendo il fattore di carico termico k_j , attraverso dei calcoli interni, il programma è in grado di definire le perdite a rotore bloccato P_{Cu} . La valutazione della corrente nominale richiede la stima della resistenza di fase, basata sulle dimensioni geometriche dello statore, e la disposizione degli avvolgimenti, come spiegato in [10]. La resistenza di fase viene riportata alla temperatura indicata nel riquadro *target copper temperature*.

La corrente nominale viene valutata come illustrato in (1.5).

$$i_0 = \sqrt{\frac{P_{Cu}}{3R}} \quad (1.5)$$

Il modello termico permette, inoltre, di stimare la temperatura degli avvolgimenti avendo nota la temperatura della carcassa. In figura 1.12 viene mostrato come *SyR-e* stima la temperatura degli avvolgimenti in funzione del parametro k_j .

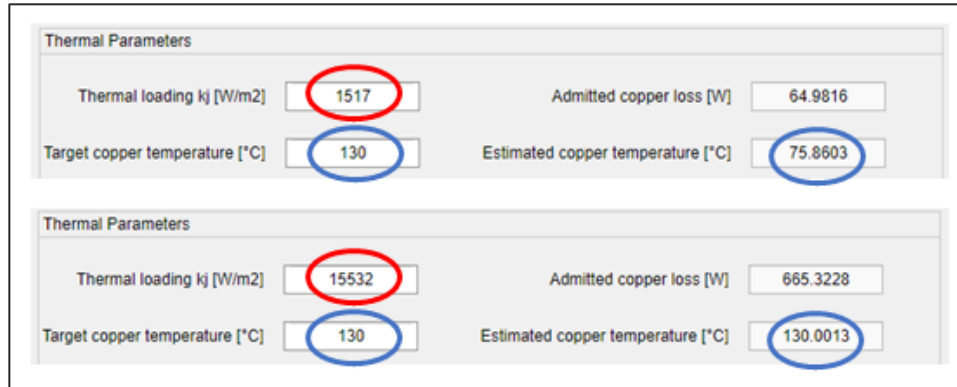


Figura 1.12. Stima della temperatura degli avvolgimenti in funzione del parametro k_j .

La velocità massima di rotazione, indicata nel riquadro *overspeed*, viene utilizzata per dimensionare i ponticelli radiali tenendo in considerazione solo gli sforzi di tipo centrifugo. Il parametro *Minimum mech. tolerance* consente di settare lo spessore dei ponticelli tangenziali. Come mostrato in figura 1.13, lo spessore dei ponticelli radiali aumenta in seguito all'incremento della velocità di rotazione.

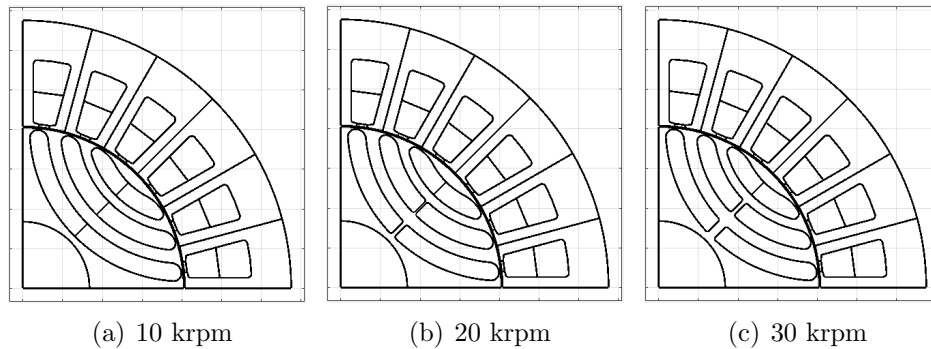


Figura 1.13. Dimensionamento dei ponticelli radiali.

La sezione *Permanent Magnet* è utilizzabile solo nel caso di macchine con magneti ed è possibile specificare i dati relativi ai magneti. Infine, attraverso i parametri *Mesh* e *mesh_MOOA*, è possibile specificare rispettivamente la risoluzione della mesh in fase di ottimizzazione e di post elaborazione.

- Windings

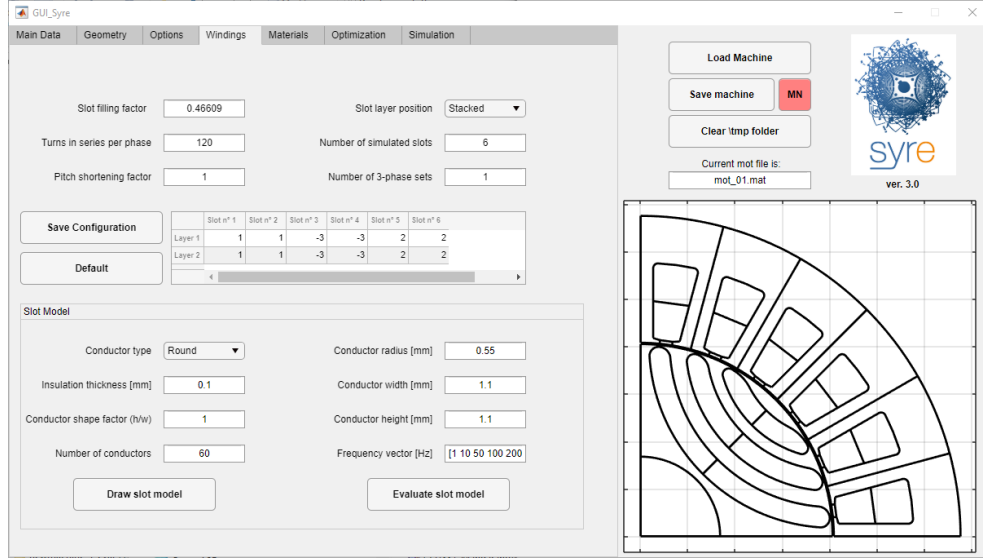


Figura 1.14. GUI - schermata *Windings* [3].

Nella finestra *Winding* (figura 1.14) è possibile impostare i parametri che definiscono l'avvolgimento della macchina, tra i quali: fattore di riempimento, numero di spire in serie per fase, fattore di raccorciamento K_{racc} , divisione della cava e il numero di cave da simulare.

Come mostra la figura 1.15, modificando il parametro *Slot layer position*, è possibile selezionare il tipo di divisione di cava desiderata.

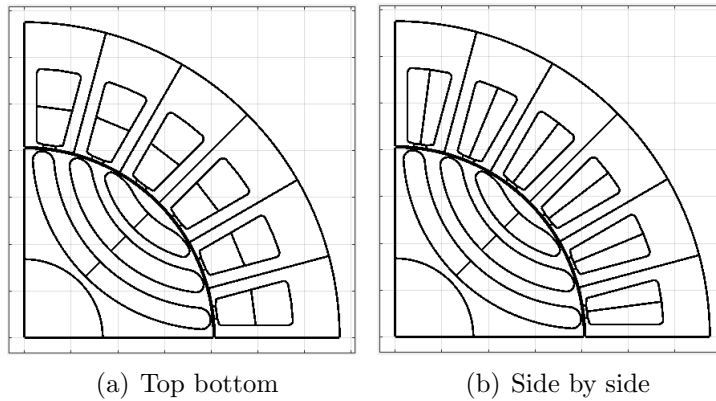


Figura 1.15. Tipologie di cava di statore.

La topologia dell'avvolgimento viene mostrata all'utente mediante una tabella di dimensioni $2 \cdot n^\circ \text{ slot da simulare}$, un esempio è mostrato in tabella 1.1.

	Slot n° 1	Slot n° 2	Slot n° 3	Slot n° 4	Slot n° 5	Slot n° 6
Layer 1	1	1	-3	-3	2	2
Layer 2	1	1	-3	-3	2	2

Tabella 1.1. Disposizione degli avvolgimenti in *SyR-e*.

In *SyR-e* sono supportati avvolgimenti a singolo e doppio strato, infatti, la prima riga della tabella fa riferimento alla posizione dei conduttori disposti nella parte interna della cava mentre la seconda riga si riferisce allo strato esterno. Gli avvolgimenti a singolo strato vengono rappresentati indicando lo stesso numero nella prima e seconda riga della matrice.

La prima colonna si riferisce alla prima cava rappresentata sull'asse orizzontale, continua poi in senso antiorario. La posizione dei conduttori nelle tre fasi viene indicata utilizzando i numeri (1,2,3, -1, -2, -3), il segno indica il verso della corrente di fase negli avvolgimenti.

La sequenza degli avvolgimenti viene calcolata in maniera automatica attraverso il software *Koil*, ogni volta che vengono modificati i parametri q e K_{racc} . Resta comunque possibile modificare la disposizione degli avvolgimenti manualmente.

- Materials

Nella scheda *Materials* (figura 1.16) è possibile impostare i materiali del nucleo dello statore, nucleo del rotore, albero, avvolgimenti e, se presenti, dei magneti. Inoltre vi è la possibilità di aggiungere nuovi materiali. La figura 1.17 mostra un esempio di curva magnetica $B-H$ presente nella libreria dei materiali a cui attinge *SyR-e*. Tale libreria è più limitata rispetto a quelle presenti nei software commerciali come lo stesso *Motor-CAD*, in quanto conta una quantità più ridotta di materiali. I dati delle curve $B-H$ fornite dai costruttori spesso riportano valori di campo magnetico limitati, per cui risulta necessario estendere la curva effettuando delle approssimazioni numeriche.

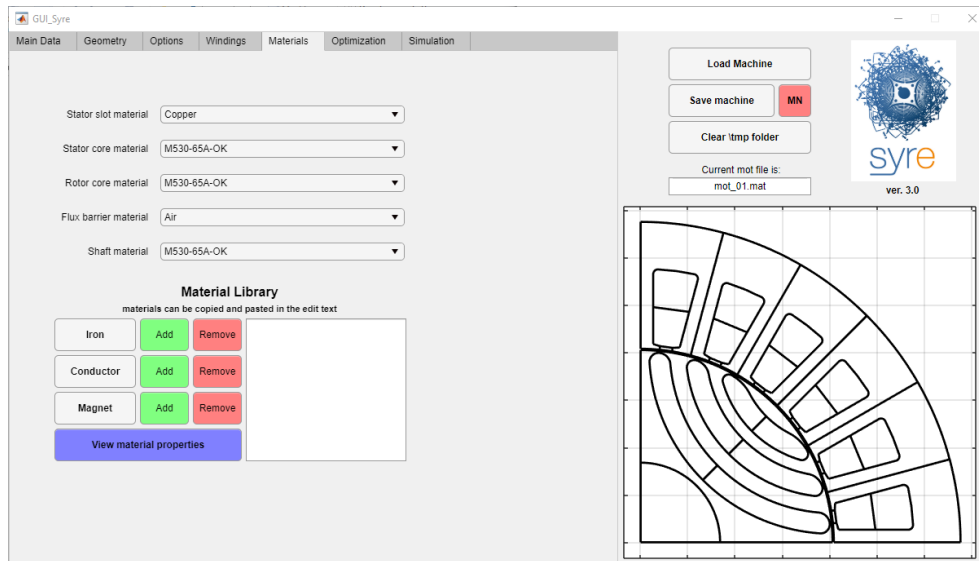


Figura 1.16. *GUI* - schermata *Materials* [3].

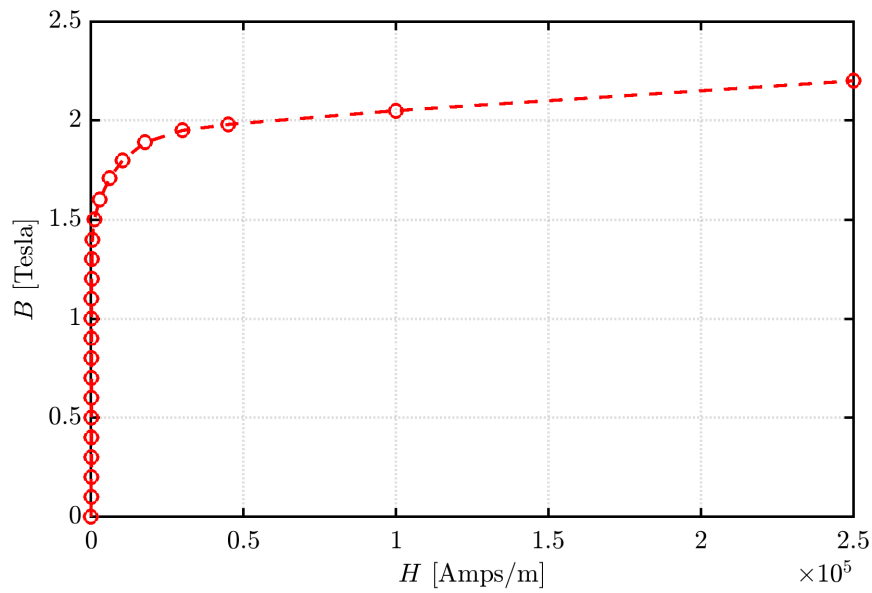


Figura 1.17. Curva B - H del materiale M600-50A, presente nella libreria di *SyR-e*.

- Optimization

La finestra *Optimization* (figura 1.18) permette di effettuare l'ottimizzazione della macchina presa in esame utilizzando un algoritmo di ottimizzazione

multi-obiettivo, basato sull'evoluzione differenziale.

Durante la procedura di ottimizzazione, alcuni parametri di rotore e statore vengono modificati automaticamente, così da raggiungere l'obiettivo prefissato. Per avviare la fase di ottimizzazione, è prima necessario impostare i parametri dell'algoritmo di ottimizzazione, le variabili da ottimizzare e gli obiettivi. Lo spazio di ricerca delle variabili da ottimizzare deve essere inserito in modo ragionevole, così da non avere delle macchine irrealizzabili.

È possibile definire i seguenti obiettivi di ottimizzazione, selezionabili singolarmente:

- massimizzazione della coppia;
- minimizzazione dell'ondulazione di coppia;
- minimizzazione della massa di rame;
- minimizzazione della massa di magnete.

L'algoritmo di ottimizzazione fornisce come risultati dei file *.mat* e *.fem* contenenti i dati relativi alla macchina ottimizzata.

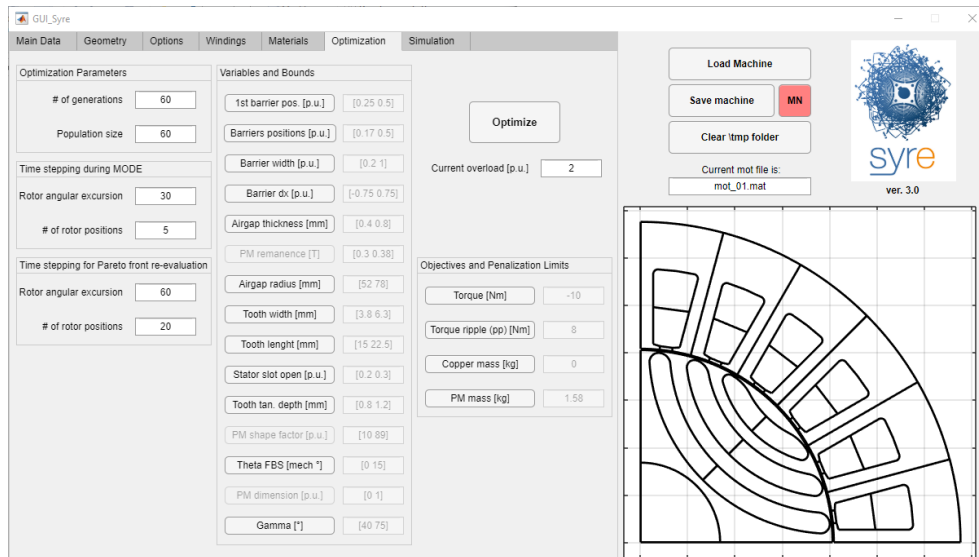


Figura 1.18. GUI - schermata *Optimization* [3].

In figura 1.19 è riportato un esempio di fronte di *Pareto* ottenuto in seguito a un processo di ottimizzazione, con l'obiettivo di minimizzare il ripple di coppia.

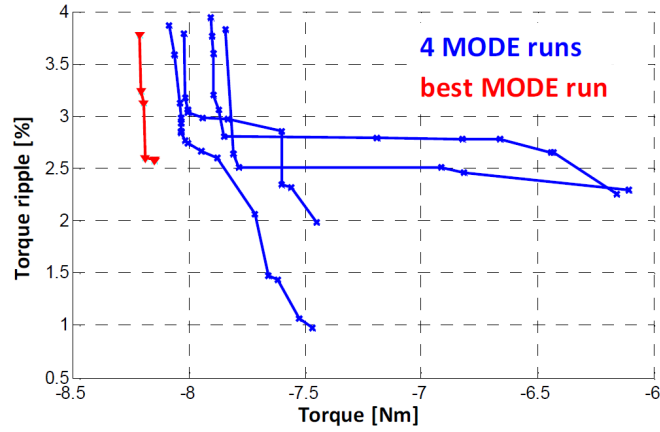


Figura 1.19. Esempio dei risultati del processo di ottimizzazione: Fronte di *Pareto* [15].

- Simulation

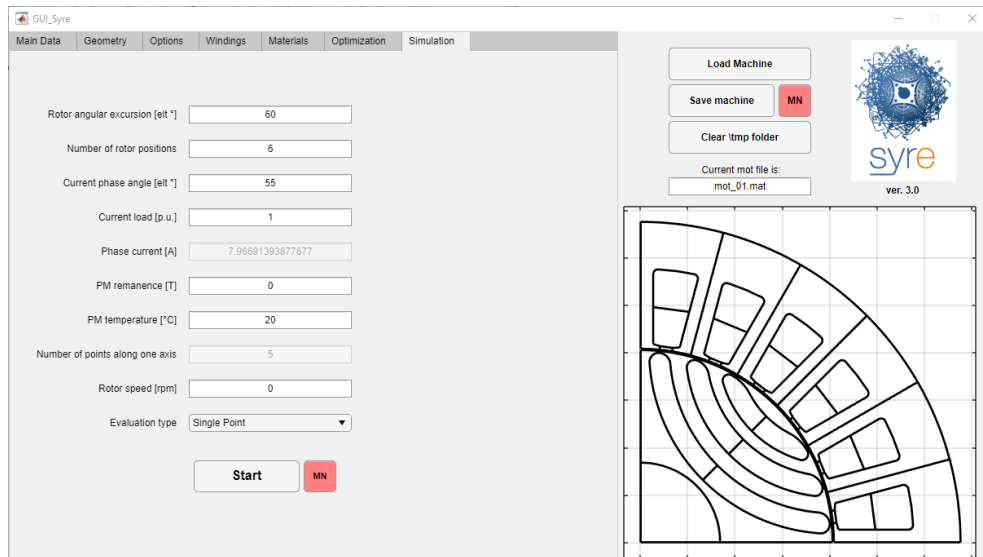


Figura 1.20. GUI - schermata *Simulation* [3].

Dopo aver definito il progetto della macchina, *SyR-e* permette di effettuare diversi tipi di simulazioni magnetiche attraverso la finestra di *Simulation* (figura 1.20). I principali parametri da impostare sono: escursione angolare del rotore, angolo di corrente, carico di corrente in *p.u.* e numero di posizioni di rotore da simulare.

Il comando *Evaluation Type* permette di scegliere il tipo di simulazioni da eseguire. Tra i risultati ottenuti con le varie simulazioni vi sono: prestazioni della macchina in termini di coppia, fattore di potenza e flussi per un dato punto di lavoro (figura 1.21), mappe di flusso in funzione di i_d e i_q (figura 1.22), analisi dell'induzione al traferro e nel ferro per dati punti di lavoro.

I calcoli vengono eseguiti di default utilizzando *FEMM*, che esegue simulazioni statiche. Se si vuole condurre una valutazione accurata delle perdite nel ferro, è possibile eseguire la simulazione utilizzando *MagNet* [2]. Quest'ultimo, a differenza di *FEMM*, esegue simulazioni di tipo transitorio con movimento di rotore.

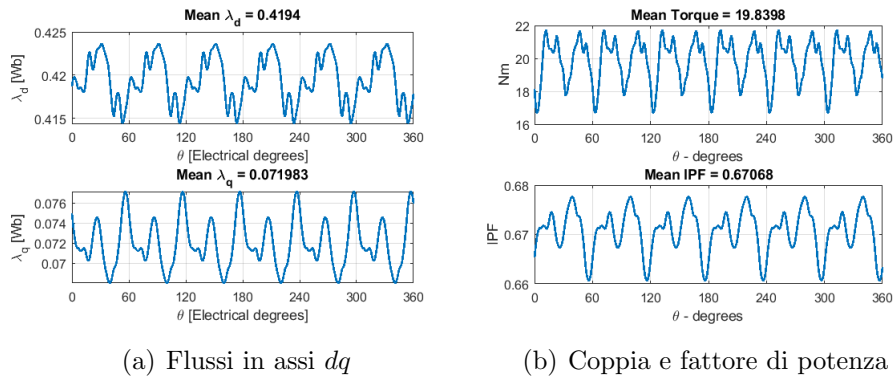


Figura 1.21. Esempio dei risultati ottenuti simulando un singolo punto di lavoro.

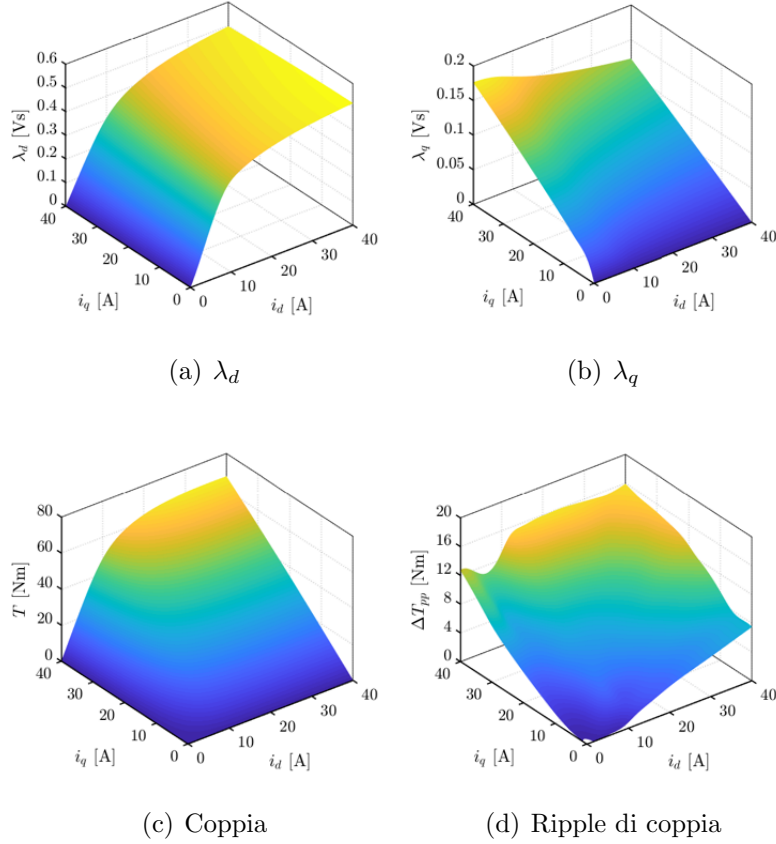


Figura 1.22. Esempio dei risultati del calcolo delle mappe di flusso [3].

1.2.4 Altre funzionalità di SyR-e

Oltre alle funzionalità nella *GUI*, *SyR-e* è dotato di funzionalità aggiuntive eseguibili tramite script. Attraverso queste è possibile effettuare una manipolazione dei dati prodotti in fase di simulazione. Di seguito vengono elencate le diverse funzioni presenti nella cartella *syreManipulateMM*.

- *C_MMLut*: permette di calcolare nuovamente le mappe di flusso dopo aver apportato alcune modifiche. Ad esempio è possibile aggiungere delle induttanze extra sugli assi dq (come le induttanze di testata), variare la lunghezza del pacco attivo e il numero di spire;

- *C_InverseModel*: fornendo in ingresso i dati contenenti il flusso in assi dq in funzione della corrente, fornisce le correnti in assi dq in funzione dei flussi;
- *C_MtpaMtpvLut*: permette di calcolare l' *MTPA* e l' *MTPV* - *Maximum Torque per Voltage*;
- *C_OperatingLimits*: calcola le curve coppia-velocità e potenza-velocità con limite di tensione e corrente impostati, trascurando la resistenza e le perdite nel ferro;
- *C_torqueVsGamma*: fornisce in uscita le curve di coppia in funzione dell'angolo di corrente γ a corrente costante;
- *E_eval_dqtMap*: crea il modello *dqt* che esprime flussi e coppia in funzione delle correnti dq e della posizione di rotore;
- *E_invert_dqtMap*: permette di invertire il modello *dqt*;
- *E_skew_dqtMap*: calcola le mappe *dqt* della macchina *skewata*, senza rieseguire nessuna simulazione;
- *MaxTw*: permette di calcolare le prestazioni e le mappe di perdita sul piano coppia-velocità considerando i limiti di tensione e corrente, la resistenza di statore (con l'effetto pelle e l'effetto della temperatura), le perdite nel ferro e le perdite meccaniche.

In aggiunta, la funzione *syreToDxf* permette di effettuare l'esportazione della geometria di macchina in formato *.dxf*.

1.3 Motor-CAD: funzionalità e caratteristiche

Motor-CAD [13] è un software per la progettazione e simulazione multi-fisica di motori elettrici, sviluppato dalla *Motor Design Ltd.* Nel software sono presenti quattro moduli integrati, elencati di seguito, che consentono di eseguire calcoli multi-fisici in modo rapido e iterativo, questo permette di valutare le prestazioni in un'ampia gamma operativa [13].

- Modelo E-Magnetic

Per l'analisi elettromagnetica vengono utilizzati metodi numerici basati sull'utilizzo dell'analisi *FEA* basata su modelli di tipo bidimensionale, in quanto i modelli tridimensionali seppur più precisi, richiederebbero un onere computazionale elevato. In questa sezione è possibile settare i parametri geometrici di macchina, tipologia di avvolgimenti e materiali. Dopo aver definito tali parametri è possibile effettuare le simulazioni *FEA* con le quali vengono calcolate ad esempio la coppia, potenza, perdite, tensioni, flussi e induttanze. Attraverso l'utilizzo del comando *Saturation Map* è possibile calcolare la mappe di flusso.

- Modulo Thermal

Il modulo termico permette di calcolare le temperature dei componenti del motore in condizioni operative stazionarie e transitorie. Il modello si basa sull'analisi di modelli a parametri concentrati, questo consente un'analisi molto veloce rispetto all'utilizzo di modelli numerici. Le resistenze termiche e le capacità termiche del modello sono calcolate in modo automatico dal software, avendo come riferimento i dati geometrici della macchina e le proprietà dei materiali utilizzati. Inoltre è possibile impostare i parametri relativi alla carcassa e la tipologia del sistema di raffreddamento.

- Modulo Lab

Il modulo *Lab* è un utile strumento per la modellazione elettromagnetica e termica ed è il punto di congiunzione tra i due mondi fisici. Permette di realizzare l'ottimizzazione del progetto su un intero intervallo operativo, consentendo di calcolare le mappe di efficienza oppure le caratteristiche coppia/velocità. I calcoli del modulo *Lab* vengono effettuati utilizzando il risolutore del modello magnetico di *Motor-CAD*.

- Modulo Mechanical

In questa sezione è possibile effettuare l'analisi strutturale delle macchina presa in esame, ovvero si va a verificare se la macchina è in grado di supportare le sollecitazioni causate dalla forza centrifuga.

Oltre alle funzionalità descritte, *Motor-CAD* permette di importare geometrie di macchina personalizzate in formato *.dxf*. Questa funzionalità risulta utile per effettuare simulazioni magnetiche e termiche di geometrie non incluse nella libreria di *Motor-CAD*.

1.4 Confronto tra SyR-e e Motor-CAD

Mettendo a confronto i due software di progettazione precedentemente descritti, la prima sostanziale differenza è che *SyR-e* è un codice sviluppato in ambiente *Matlab* scaricabile gratuitamente online, mentre *Motor-CAD* è un software commerciale, ottenibile attraverso un abbonamento.

Si può, inoltre, osservare che in *SyR-e* è presente un modulo di *Preliminary Design* utile per effettuare un dimensionamento iniziale della macchina partendo dalle caratteristiche di coppia e fattore di potenza desiderate come spiegato al paragrafo 1.2, mentre in *Motor-CAD* questa funzionalità non è presente ed è dunque necessario fare riferimento ad un progetto già esistente su cui apportare modifiche per ottenere la geometria di macchina desiderata. *SyR-e*, inoltre, in fase di progettazione iniziale, suggerisce l'inserimento di eventuali ponticelli radiali, in relazione alla massima velocità di rotazione desiderata.

SyR-e, utilizzando un algoritmo di ottimizzazione multi-obiettivo (*MODE - multi-objective differential evolution*), fornisce un quadro completo di macchina ottimizzata [17] - [16]. *Motor-CAD*, invece, permette di ottenere un progetto ottimizzato attraverso l'esecuzione di analisi di sensitività dei singoli parametri [13].

Per quanto riguarda le simulazioni magnetiche, attraverso entrambi i software è possibile condurre la simulazione di singoli punti di lavoro e il calcolo delle mappe di flusso e di efficienza. In aggiunta *Motor-CAD* fornisce autonomamente il calcolo delle perdite nel ferro, mentre *SyR-e* per tale funzione si appoggia a *MagNet*, software commerciale per le simulazioni elettromagnetiche.

Motor-CAD possiede tutti gli strumenti per ottenere una verifica strutturale della macchina, in *SyR-e* invece, per sopperire a questa mancanza, è necessario utilizzare parallelamente *SolidWorks*, programma specificatamente dedicato alla progettazione di apparati meccanici.

SyR-e non è fornito di appositi strumenti per ottenere la simulazione termica della macchina, mentre utilizzando *Motor-CAD* è possibile effettuare simulazioni termiche sia a regime che in transitorio.

La tabella 1.2 riassume le caratteristiche di confronto tra i due software.

Funzioni	<i>SyR-e</i>	<i>Motor-CAD</i>
Dimensionamento preliminare della geometria di macchina	Si	No
Dimensionamento strutturale iniziale	Si	No
Ottimizzazione	Multi-obiettivo	Analisi di sensitività
Simulazioni magnetiche	Si	Si
Calcolo delle perdite nel ferro	No \implies <i>MagNet</i>	Si
Verifica strutturale	No \implies <i>SolidWorks</i>	Si
Simulazioni termiche	No	Si

Tabella 1.2. Confronto funzionalità tra *SyR-e* e *Motor-CAD*.

1.5 Obiettivi della tesi

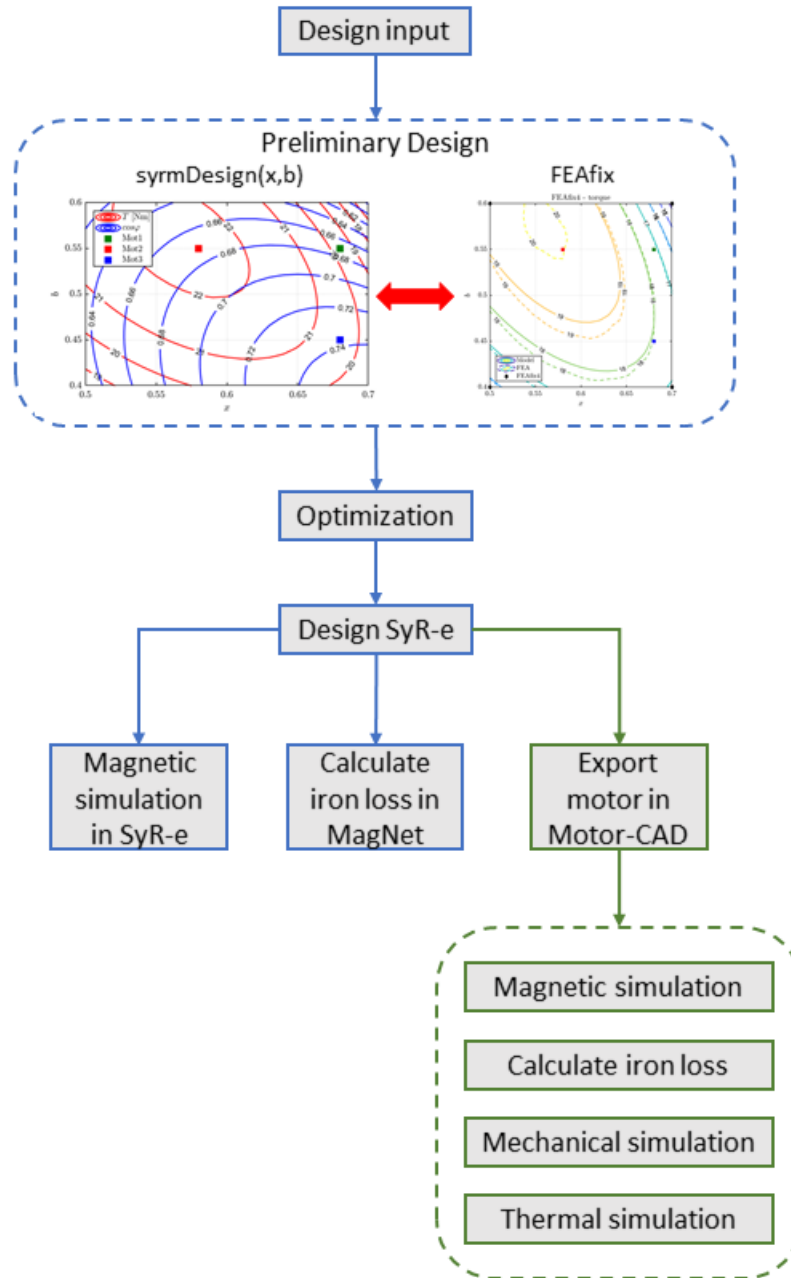


Figura 1.23. Fasi di progetto di un motore *SyR* mediante l'utilizzo integrato di *SyR-e* e *Motor-CAD*.

L'analisi preliminare ha messo in luce le possibili sinergie tra i due ambienti software, evidenziando che *SyR-e* è uno strumento completo di progettazione magnetica della macchina *SyR*, che si appoggia su altri software per la verifica strutturale e il calcolo delle perdite nel ferro. *Motor-CAD* viceversa è una suite di modellazione piuttosto completa, ma non ha strumenti di dimensionamento della macchina a partire dalle specifiche di progettazione iniziale. I due software si prestano ad essere interfacciati con facilità attraverso script *Matlab*.

In questa tesi viene condotta la progettazione e programmazione degli strumenti di scambio di dati tra i due ambienti software in modo da sfruttare questa sinergia. Il risultato è una procedura di progettazione completa, che include equazioni di dimensionamento, eventuale uso dell'ottimizzazione numerica, modelli numerici magnetici, strutturali e termici e calcolo delle curve di prestazione della macchina. Il lavoro prevede inoltre la validazione dei risultati ottenuti con i nuovi script mediante il confronto di questi con quelli forniti dalle simulazioni magnetiche, meccaniche e termiche condotte singolarmente attraverso gli appositi software.

Nella figura 1.23 vengono indicate in blu le funzionalità già presenti, mentre in verde le funzionalità implementate nel lavoro di tesi.

Capitolo 2

Interfaccia tra SyR-e e Motor-CAD

Per sfruttare a pieno le potenzialità di *SyR-e* e *Motor-CAD* si vuole sviluppare una procedura di progettazione ibrida che permetta di studiare la stessa macchina in entrambi i software. Attraverso l'implementazione di tale procedura è possibile avviare la progettazione del motore in *SyR-e* sfruttando la presenza dello strumento di *Preliminary Design*, per poi eseguire le simulazioni in ambito magnetico, termico e strutturale in *Motor-CAD*.

2.1 Nuova finestra Motor-CAD nella GUI SyR-e

La figura 2.1 riassume le modifiche apportate a *SyR-e* ed il contributo della tesi al progetto open-source.

Per effettuare l'esportazione dei parametri di progetto e di simulazione di un motore da *SyR-e* a *Motor-CAD* è necessario tenere in considerazione che i due software utilizzano grandezze e riferimenti differenti.

Per poter condurre con maggiore facilità l'esportazione dei parametri è stata inserita una finestra apposita nell'interfaccia grafica di *SyR-e*.

Nella nuova *GUI* sono stati inseriti i comandi di seguito descritti:

- *Export.mot*: permette di eseguire l'esportazione di una macchina definita in *SyR-e* in *Motor-CAD*;
- *Emag sim*: permettere di eseguire le simulazioni magnetiche di un singolo punto di lavoro;
- *Export maps*: permette di esportare in *Motor-CAD* le mappe di flusso ottenute con *SyR-e*;
- *Therm Export*: permette di impostare i parametri termici presenti in *Motor-CAD*;
- *Therm Sim*: permette di eseguire in *Motor-CAD* simulazioni termiche di tipo transitorio e a regime.

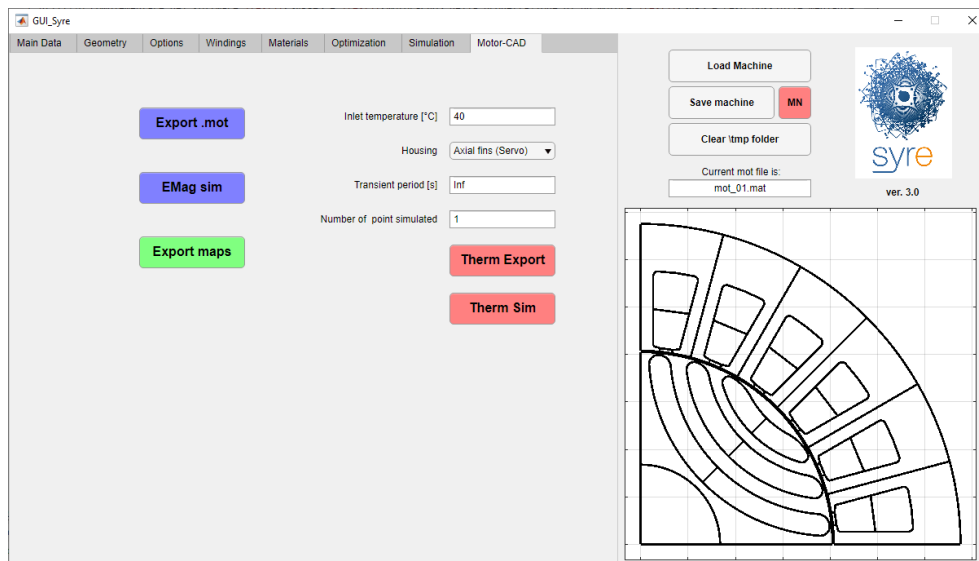


Figura 2.1. GUI - schermata *Motor-CAD*.

2.1.1 Export della geometria di macchina

Per realizzare l'esportazione dei dati di progetto da *SyR-e* a *Motor-CAD* è stata creata in *Matlab* la funzione *draw_motor_in_MCAD* (appendice A), eseguibile con il comando *Export .mot* della *GUI* (figura 2.1). Tale funzione permette di definire

la geometria del rotore e dello statore di una qualunque macchina *SyR* con rotore di tipo *Circular* e salvare il progetto in formato *Motor-CAD* (.mot).

La fase di esportazione si articola in diversi passaggi schematizzati in figura 2.2.

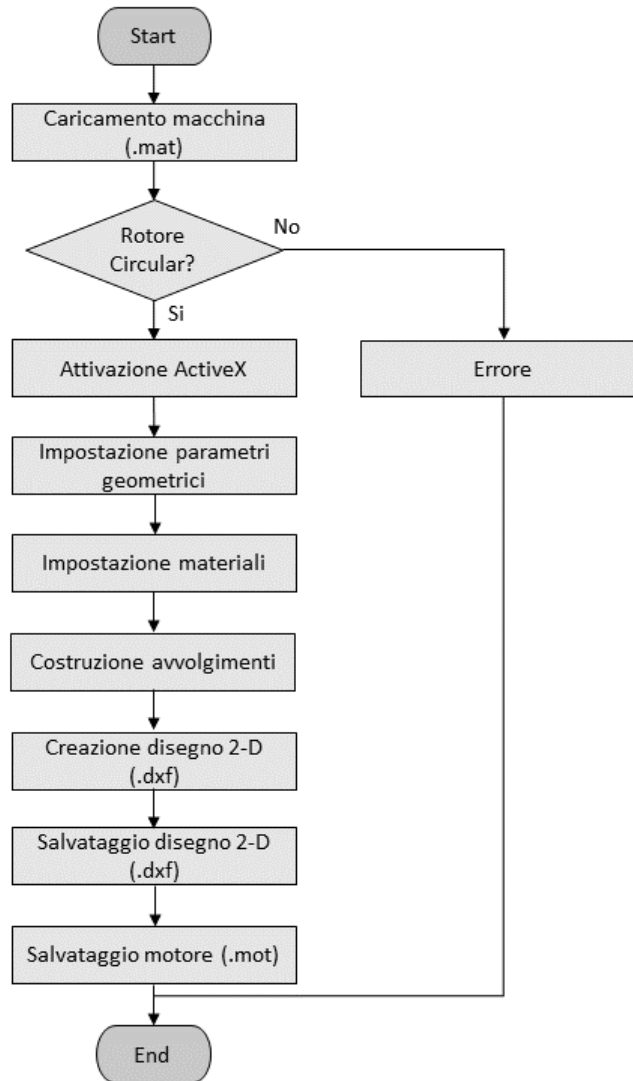


Figura 2.2. Schema a blocchi: comando *Export.mot*.

Il codice realizzato per l'esportazione del modello di macchina si basa sulla lettura del file .mat contenente i dati del progetto definito in *SyR-e*.

Come mostrato in figura 2.3, il file .mat è costituito da quattro strutture di dati,

quelle utilizzate in tale contesto sono *geo* e *dataSet*. La struttura *geo* racchiude i dati geometrici della macchina e dei materiali utilizzati, la struttura *dataSet* raccoglie altri importanti parametri di progetto.

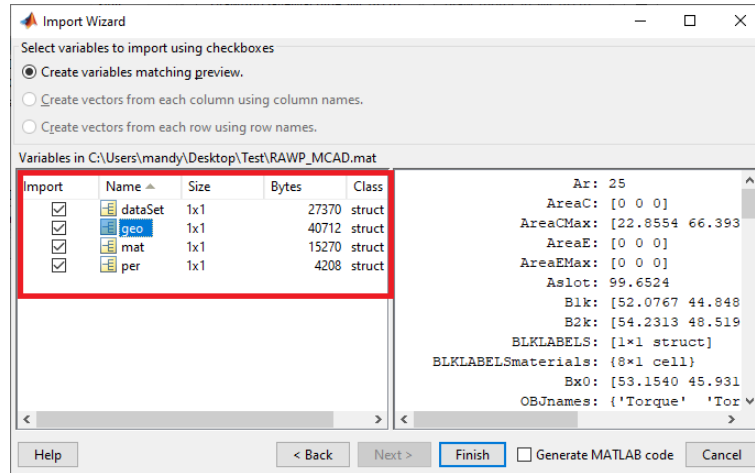


Figura 2.3. Struttura del file (.mat).

Prima di iniziare la fase di esportazione viene verificato che la geometria di rotore sia di tipo *Circular*, in caso di esito negativo l'esecuzione verrà arrestata.

L'utilizzo complementare dei software *SyR-e* e *Motor-CAD* è reso possibile mediante l'utilizzo del comando *ActiveX*, presente tra gli strumenti di *Motor-CAD*. Più nello specifico, *ActiveX* permette di interfacciare *Motor-CAD* con diversi software di calcolo, tra cui *Excel* e *Matlab*. Si procede, dunque, con l'attivazione della funzione *ActiveX* per avviare la comunicazione tramite script tra *Matlab* e *Motor-CAD*. Il passo successivo consiste nella definizione delle variabili geometriche di statore e di rotore. Dal momento che i due software utilizzano parametri e convenzioni differenti, per ogni grandezza si procede inserendo come variabile temporanea il dato proveniente da *SyR-e* preceduto da un eventuale parametro di correzione, per poi effettuare la trasformazione dei numeri in una stringa di caratteri.

Poiché *SyR-e* e *Motor-CAD* utilizzano come separatore decimale rispettivamente il punto e la virgola, è necessario effettuare la sostituzione dei due caratteri per permettere la lettura corretta dei dati in *Motor-CAD*. Il processo si conclude assegnando la variabile temporanea al parametro *Motor-CAD* di interesse.

Definite le variabili geometriche, vengono esportati in *Motor-CAD* i materiali e,

tramite la funzione *windingSyreToMCAD*(appendiceA), gli avvolgimenti. Segue infine l'inserimento del disegno 2-D del rotore e di una cava di statore in formato *.dxf* e il salvataggio del modello in formato *.mot*.

Per realizzare con entrambi i software delle macchine che risultino essere quanto più possibile simili, bisogna verificare che le caratteristiche (elettriche, ferromagnetiche e meccaniche) dei materiali siano uguali. A tal proposito è necessario controllare manualmente le specifiche librerie.

In figura 2.4 viene riportato un esempio del risultato dell'esportazione dei dati tramite il comando *Export.mot*.

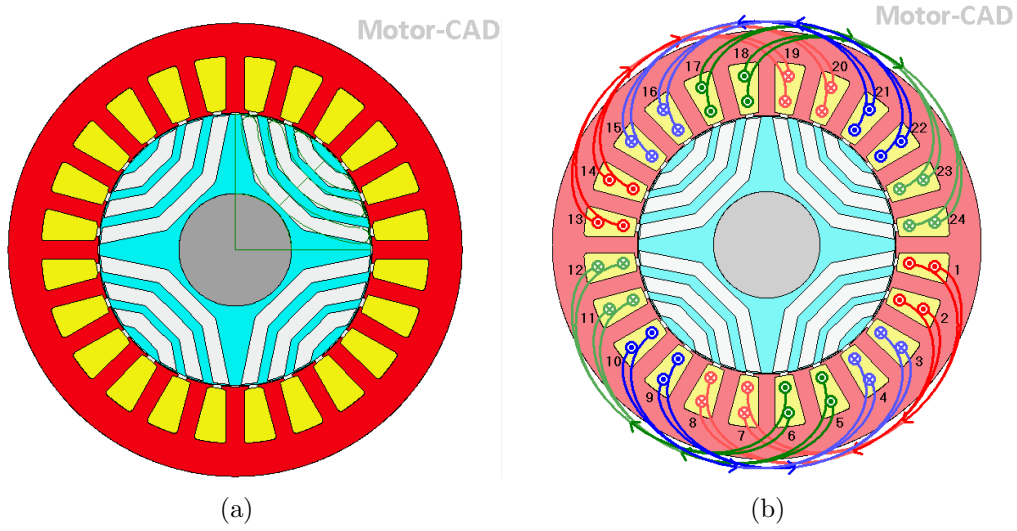


Figura 2.4. Risultati ottenuti mediante la funzione *Export.mot*.

Export degli avvolgimenti di statore

SyR-e fornisce tra i dati progetto una matrice in cui viene riportata la disposizione degli avvolgimenti nelle cave per una singola espansione polare. *Motor-CAD* richiede in input la disposizione degli avvolgimenti per l'intera geometria di macchina. A tal fine è stata sviluppata la funzione *windingSyreToMCAD* (appendice A) che permette di definire in maniera automatica in *Motor-CAD* la disposizione degli avvolgimenti partendo dalla matrice definita in *SyR-e*.

La procedura di esportazione è eseguita con la seguente modalità:

- creazione della matrice di avvolgimento per tutte le espansioni polari della macchina, partendo dalla matrice per una singola espansione polare, seguendo lo schema di avvolgimento mostrato in (2.1);

$$A A \quad \overline{C C} \quad B B \quad \overline{A A} \quad C C \quad \overline{B B} \quad (2.1)$$

- ricerca della cava di partenza e arrivo della bobina per ognuna delle tre fasi;
- assegnazione della disposizione delle cave in *Motor-CAD*.

Export del disegno 2-D

In *Motor-CAD* è possibile importare il disegno in formato *.dxf* del lamierino di statore e di rotore che si vuole realizzare. Il software richiede il disegno di un polo di rotore.

In *SyR-e* è presente la funzione *syreToDxf* che permette di realizzare lo schizzo 2-D di un polo del lamierino di rotore e di statore (figura 2.5(a)). La funzione *syreToDxfMCAD* (appendice A) è stata implementata partendo da quella già esistente, così da ottenere il disegno come richiesto da *Motor-CAD* (figura 2.5(b)).

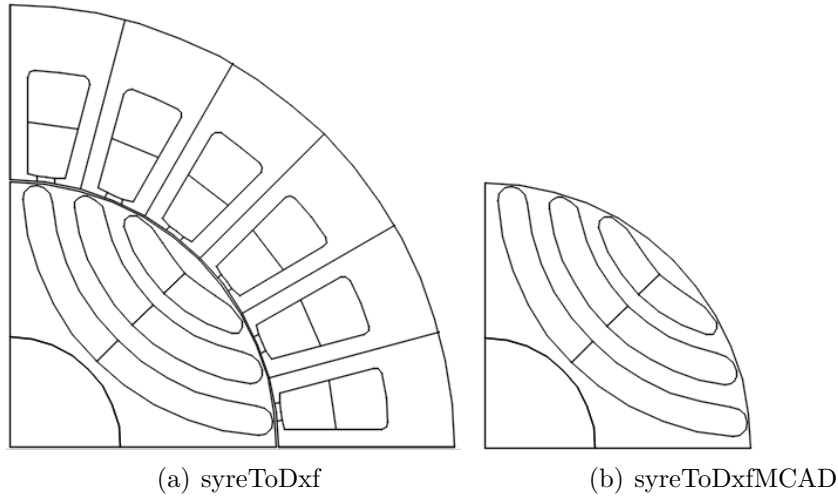


Figura 2.5. Confronto dei disegni prodotti con le funzioni (a) e (b).

2.1.2 Simulazioni magnetiche in Motor-CAD

Per effettuare in *Motor-CAD* le simulazioni di un singolo punto di lavoro è stata creata in *Matlab* la funzione *eval_operatingPointMCAD* (appendice B), eseguibile utilizzando il comando *Emag sim* della *GUI* (figura 2.1).

Prima di iniziare la simulazione viene verificato che nella cartella di lavoro sia presente il file *.mot*, in caso contrario l'esecuzione verrà arrestata.

Le simulazioni magnetiche vengono condotte seguendo i passaggi successivi:

- caricamento della macchina da simulare in *Motor-CAD*;
- settaggio di alcuni parametri necessari per eseguire le simulazioni (corrente, angolo di corrente γ , tipologia di simulazione, ecc.);
- esecuzione delle simulazioni magnetiche di un singolo punto di lavoro;
- acquisizione dei dati relativi al flusso in assi *dq*, alla coppia e al fattore di potenza;
- salvataggio e stampa dei risultati ottenuti.

Poiché *SyR-e* e *Motor-CAD* utilizzano grandezze e riferimenti differenti, è necessario soffermarsi sul diverso orientamento nello spazio degli assi *dq* (figura 2.6).

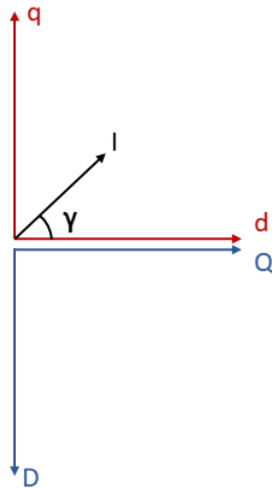


Figura 2.6. Rappresentazione degli assi *dq* in *SyR-e* (rosso) e *Motor-CAD* (blu).

In *SyR-e* gli assi dq vengono definiti nel seguente modo:

- asse d : orientato lungo la direzione di minima riluttanza, quindi massima induttanza;
- asse q : orientato lungo la direzione di massima riluttanza, quindi minima induttanza.

In *Motor-CAD* tale convenzione non viene utilizzata. A tal proposito, durante il confronto dei risultati delle simulazioni magnetiche, è necessario considerare che le condizioni di anisotropia lungo i due assi risultano invertite (2.2). L'angolo di corrente γ rimane invariato per entrambi i software.

$$\begin{cases} L_d > L_q \rightarrow \text{SyR-e} \\ L_Q > L_D \rightarrow \text{Motor-CAD} \end{cases} \quad (2.2)$$

Un esempio dei risultati ottenuti attraverso l'esecuzione del comando *Emag sim* viene mostrato in figura 2.7.

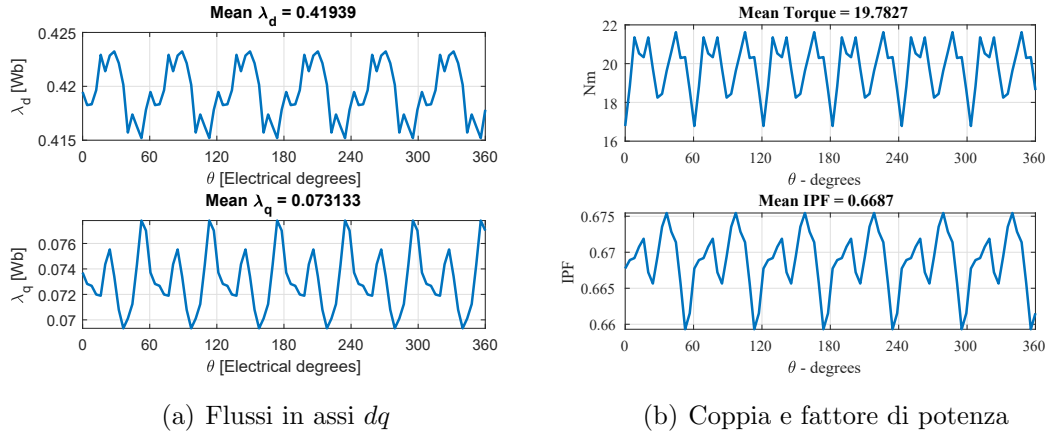


Figura 2.7. Esempio dei risultati ottenuti utilizzando la funzione *Emag sim*.

2.1.3 Esportazione delle mappe di flusso in Motor-CAD

Motor-CAD permetta di ottenere in maniera autonoma le mappe di flusso, nonostante ciò, in tale contesto risulta utile esportare quelle prodotte in *SyR-e*. Tale passaggio viene effettuato attraverso l'apposito comando *Export maps* inserito nella *GUI* (figura 2.1). *Export maps* effettua una rielaborazione del file prodotto in *SyR-e* restituendolo in formato *.txt* leggibile in *Motor-CAD*. I dati vengono inseriti in *Motor-CAD* tramite script.

2.1.4 Simulazioni termiche in Motor-CAD

In *SyR-e* non è possibile condurre l'analisi termica della macchina progettata, è stato dunque implementato un codice per eseguire le simulazioni termiche direttamente in *Motor-CAD*.

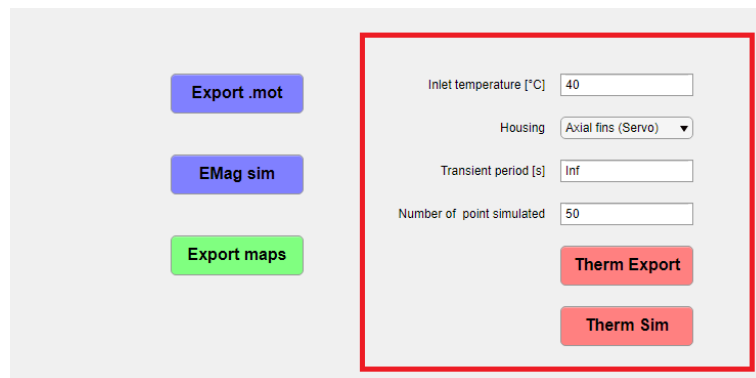


Figura 2.8. Impostazione dei parametri termici nella *GUI*.

Nell'interfaccia grafica di *SyR-e* dedicata alla parte termica (figura 2.8) si possono definire i seguenti parametri, necessari per la costruzione del modello termico:

- *Inlet temperature*: temperatura ambiente in caso di macchina con raffreddamento ad aria, oppure temperatura di ingresso del liquido in caso di macchina con raffreddamento a liquido;
- *Housing*: scelta del sistema di raffreddamento;

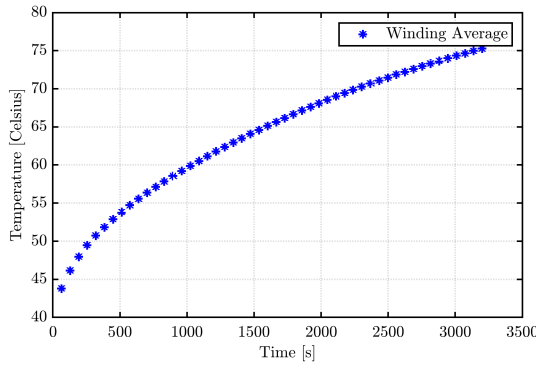
- *Transient period*: tempo di simulazione per eseguire delle simulazioni di tipo transitorio. Digitando *Inf* si eseguono invece delle simulazioni termiche a regime;
- *Number of points simulated*: numero di punti che si vogliono simulare.

Questi parametri rappresentano solo una parte di quelli necessari per eseguire le simulazioni termiche in *Motor-CAD*, ma consentono di analizzare in linee generali il comportamento termico della macchina.

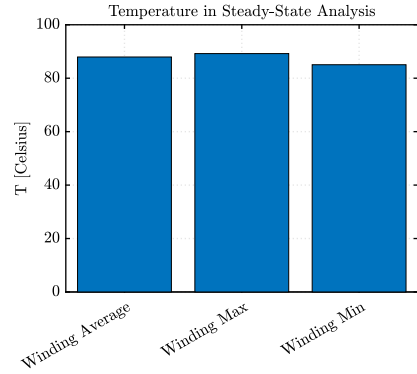
Attraverso il comando *Therm Export* vengono impostati e salvati in *Motor-CAD* i parametri in questione.

Per eseguire un'analisi termica più dettagliata, è necessario aggiungere nel progetto ulteriori parametri che definiscono il sistema di raffreddamento.

Il comando *Therm Sim* esegue le simulazione termiche a regime o in transitorio, in funzione del parametro impostato nella casella *Transient period*, restituendo dei grafici riportati come da esempio in figura 2.9. I risultati relativi alle temperature di ogni singola parte della macchina vengono salvati in *Excel*.



(a) Simulazione Transient, $\theta_a = 40^\circ C$, 1 ora



(b) Simulazione Steady-State, $\theta_a = 40^\circ C$

Figura 2.9. Esempio dei risultati ottenuti utilizzando la funzione *Therm Sim*.

2.2 Esempio: motore RawPower

Per valutare il grado di precisione del codice sviluppato, è stata condotta una verifica su un motore *SyR* definito in *SyR-e*. In tabella 2.1 vengono riportate le specifiche del motore preso in esame ottenute con *SyR-e*.

Parametro	Valore	Unità di misura
Materiale ferromagnetico	M600-50A	-
Spessore traferro	0.325	<i>mm</i>
Raggio esterno dello statore	87.5	<i>mm</i>
Raggio esterno del rotore	59.5	<i>mm</i>
Raggio dell'albero	25	<i>mm</i>
Lunghezza dell'albero	110	<i>mm</i>
Barriere di flusso	3	-
Coppie polari	3	-
Numero di cave polo/fase	2	-
Avvolgimenti in serie per fase	120	-
Fattore di avvolgimento	0.45	-
Carico termico	2800	W/m^2
Corrente nominale	15.77	<i>A</i>
Coppia nominale	21	<i>Nm</i>

Tabella 2.1. Spedifiche motore *RawPower*.

Il comando *Export .mot* (figura 2.8), noti i dati di progetto della macchina mostrata in (figura 2.10(a)), permette di definire la medesima macchina in *Motor-CAD*. Vengono inoltre esportati gli avvolgimenti di statore (figura 2.10(c)). Si ottiene in output un file *.mot* contenente il progetto definito in *Motor-CAD*, come mostrato in figura 2.10(b).

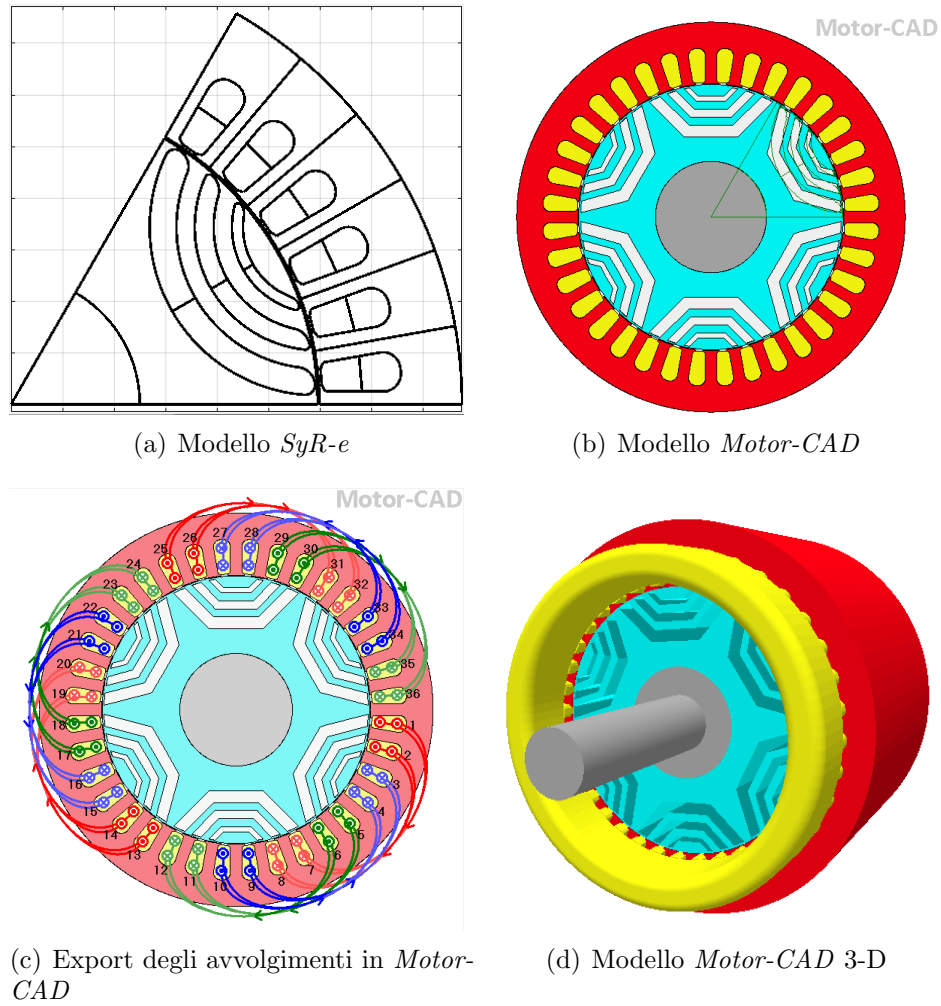


Figura 2.10. Risultati ottenuti con la funzione *draw_motor_in_MCAD*.

Prima di effettuare il salvataggio del modello *Motor-CAD* viene caricato il file *.dxf* contenente il disegno 2-D della geometria del rotore. Questo passaggio risulta utile in quanto permette:

- il confronto visivo della geometria realizzata con quella da realizzare, rappresentata dal modello 2-D;
- l'esecuzione di simulazioni magnetiche e meccaniche del modello personalizzato, a condizione che le due geometrie non differiscano notevolmente.

In figura 2.11, si può osservare che i due modelli non corrispondono perfettamente data l'assenza in *Motor-CAD* di un rotore di tipo *Circular*. Per sopperire a questa

manca si è scelto di utilizzare un rotore di tipo *Seg* rendendolo quanto più simile ad un rotore *Circular*.

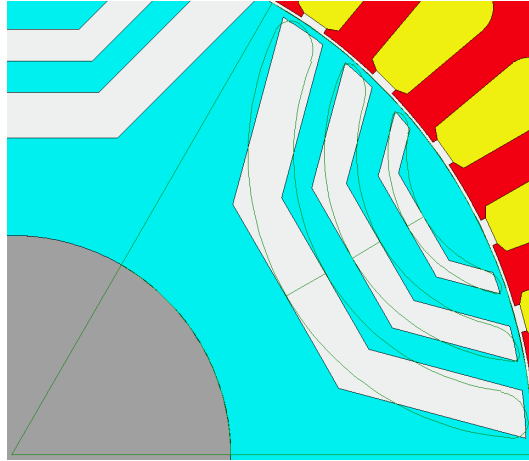


Figura 2.11. Zoom del rotore.

Dopo aver realizzato il modello di macchina, è importante verificare che le regioni *FEA* vengano riconosciute in maniera corretta (figura 2.12). Questa verifica è necessaria per eseguire delle simulazioni magnetiche e meccaniche utilizzando come geometria di riferimento quella importata in formato *.dxf*. Le aree non riconosciute vengono identificate dal simulatore come aria.

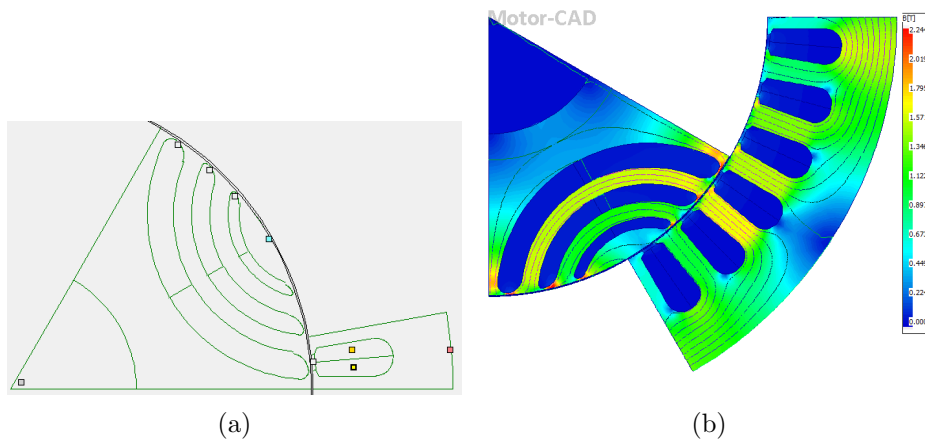


Figura 2.12. Riconoscimento regioni *FEA*.

Capitolo 3

Simulazioni magnetiche

3.1 Descrizione del modulo EMag e Lab di Motor-CAD

Il modulo *Lab*, presente in *Motor-CAD*, è un modulo multi-fisico che consente l'analisi combinata del modello elettromagnetico e termico. In questo modulo è possibile eseguire delle analisi utili in fase di progetto, come il calcolo delle prestazioni massime della macchina e il calcolo delle mappe di efficienza e di perdita [13]. Per la costruzione del modello elettromagnetico, il modulo *Lab* richiede come dati di partenza i flussi, le induttanze e le perdite. Questi dati vengono forniti in maniera automatica dal modulo *E-Magnetic* presente in *Motor-CAD*, oppure, vi è la possibilità di importarli in formato *.txt* utilizzando altri risolutori *FEA*. Nella finestra *E-Magnetic* vengono richiesti i parametri geometrici e le caratteristiche dei materiali che influenzano il calcolo delle prestazioni magnetiche della macchina in diverse condizioni elencate di seguito:

- Singolo punto di lavoro:
 - *Open Circuit*: misura del flusso a circuito aperto;
 - *Q axis current only*: misura del flusso iniettando corrente solo in asse q , così da considerare la saturazione del circuito magnetico;
 - *On Load*: calcolo delle prestazioni a carico come ad esempio flusso, induttanze e coppia.

- Circuito aperto:
 - *Back EMF*: misura, a corrente nulla, della tensione e delle perdite indotte nella macchina ruotando il rotore di un giro meccanico;
 - *Cogging Torque*: misura del ripple di coppia;
 - *Electromagnetic Forces*: misura della forza elettromagnetica.
- A carico:
 - *Torque*: calcolo della coppia, delle perdite e altre grandezze elettriche;
 - *Torque Speed Curve*: misura delle prestazioni della macchina per diversi angoli di fase della corrente;
 - *Demagnetization*: calcolo dei limiti operativi dei magneti per fare in modo che non si smagnetizzino;
 - *Electromagnetic Forces*: calcolo della forza elettromagnetica.

Nel modulo *E-Magnetic* è inoltre possibile misurare le induttanze in funzione dei flussi e simulare un corto circuito in tutte le fasi della macchina per visualizzare graficamente le variazioni di coppia, velocità, corrente e avanzamento di fase rispetto al tempo.

In figura 3.1 viene mostrata la schermata iniziale della sezione *Lab* in cui è possibile impostare le caratteristiche desiderate per la costruzione del modello elettromagnetico, tra cui:

- *Saturation Model*: scelta del modello di saturazione, tra quelli elencati di seguito:
 - *Fixed inductance*: considera i valori di L_d e L_q costanti, dunque non dipendenti dalla corrente;
 - *Saturation Model (Single Step)*: considera la variazione del flusso con la corrente in un'unica posizione di rotore tenendo conto dell'effetto della saturazione ferromagnetica dei lamierini;
 - *Saturation Model (Full cycle)*: considera la variazione del flusso con la corrente e l'angolo di rotore. I valori di interesse vengono mediati sull'intero ciclo elettrico, tenendo così in considerazione l'effetto delle armoniche.

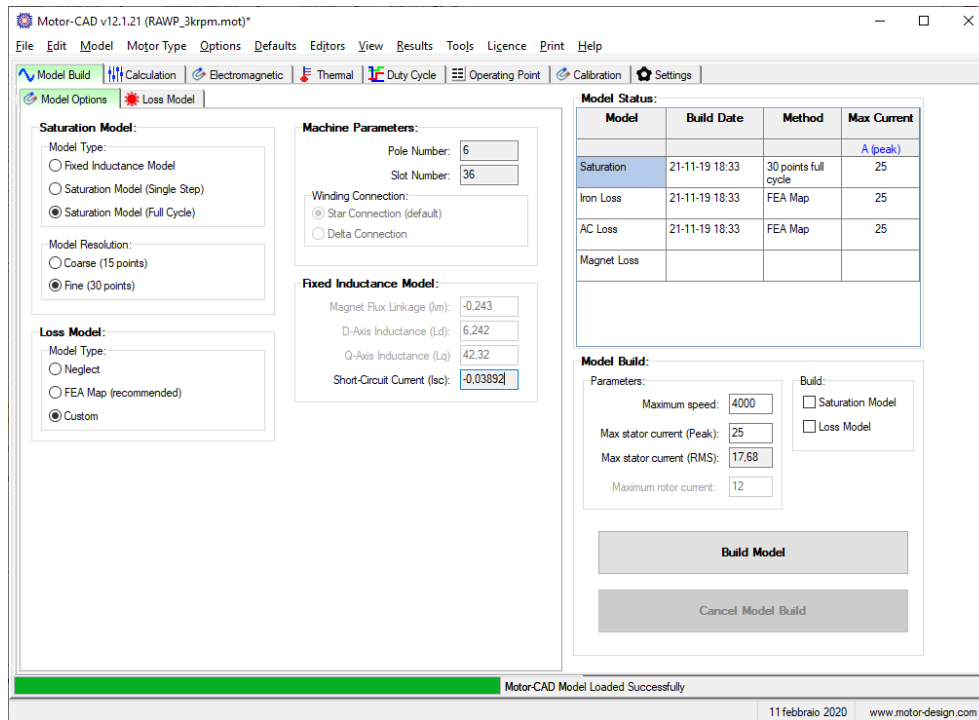


Figura 3.1. *Motor-CAD* - modulo *Lab*.

- *Model Resolution*: scelta della risoluzione della griglia di corrente;
- *Loss Model*: scelta del modello di perdita, tra quelli elencati di seguito:
 - *Neglect*: considera solo le perdite per effetto *Joule* causate dalla componente continua della corrente;
 - *FEA Map*: considera le perdite nel ferro e le perdite dovute alla circolazione di correnti parassite, ovvero le perdite *Joule* causate dalla componente continua e dalla componente alternata della corrente;
 - *Custom*: permette la costruzione di un modello di perdita personalizzato.

Attraverso la funzione *Saturation Map*, *Motor-CAD* permette di calcolare ed esportare le mappe di flusso in *Matlab*. Questo passaggio risulta utile per condurre delle analisi di sistema, ma anche per confrontare i risultati ottenuti con quelli prodotti da altri software, come ad esempio *SyR-e*.

Il calcolo delle mappe di flusso può essere condotto seguendo due modalità:

- interpolando i dati ottenuti mediante la sezione *Lab*;
- conducendo un'analisi *FEA* dedicata utilizzando una singola posizione di rotore o l'intero periodo.

Attraverso la seconda modalità è possibile ottenere dei risultati più accurati a spese, però, di un tempo di calcolo maggiore.

3.2 Calcolo e manipolazione delle mappe di flusso

Per descrivere in modo quanto più corretto il comportamento magnetico di un motore soggetto a fenomeni di saturazione significativi, come ad esempio i motori *SyR*, è necessario utilizzare delle mappe di flusso che descrivono il legame tra corrente e flusso in assi *dq*. A tal proposito, per validare lo script realizzato per condurre l'esportazione di un motore da *SyR-e* a *Motor-CAD*, è stata condotta un'analisi di confronto tra le mappe di flusso sperimentali e quelle ottenute dalle simulazioni magnetiche eseguite con i due software. Le prove sono state condotte sul motore *RawPower*, le cui specifiche sono riportate in tabella 2.1.

In figura 3.2 e 3.3 sono mostrate le mappe di flusso in forma bidimensionale. Per mettere in evidenza il fenomeno della *cross saturation*, le curve di flusso sono state tracciate considerando due diversi livelli di corrente in assi *dq*.

Dal confronto si evince che *SyR-e* descrive meglio il comportamento magnetico fino al ginocchio, mentre in saturazione i flussi vengono meglio descritti in *Motor-CAD*.

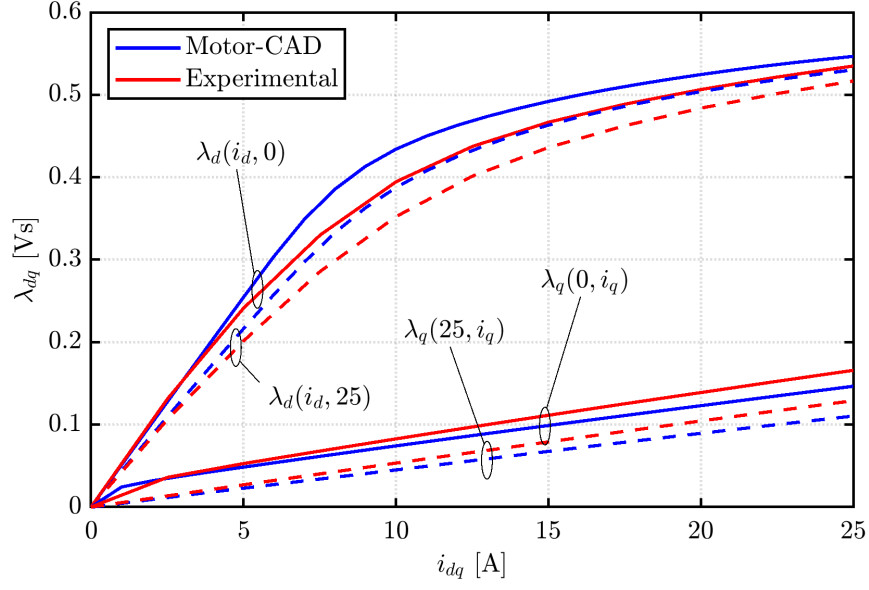


Figura 3.2. Confronto tra le curve di flusso sperimentali e quelle calcolate in *Motor-CAD*.

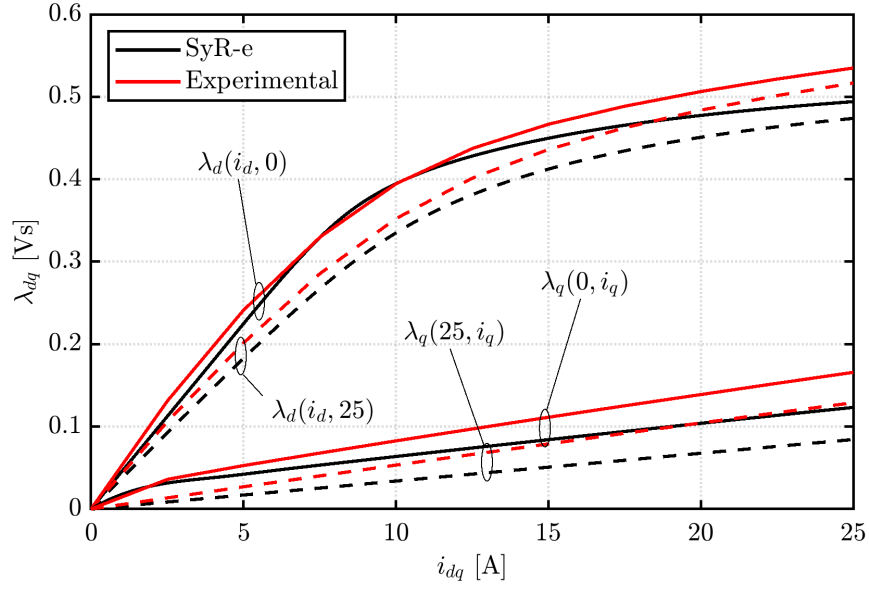


Figura 3.3. Confronto tra le curve di flusso sperimentali e quelle calcolate in *SyR-e*.

3.3 Verifica dell'errore su un singolo punto di lavoro

Per valutare le prestazioni elettriche del modello esportato sono state eseguite delle simulazioni magnetiche *Single Point*. L'analisi è stata condotta utilizzando entrambi i software, così da poter confrontare i risultati ottenuti.

L'esempio preso in esame valuta le prestazioni alla corrente nominale lungo il luogo dei punti (*MTPA-Maximum Torque per Ampere*). L'angolo di corrente è stato ricavato utilizzando la funzione *C_MtpaMtpvLut* presente in *SyR-e*. Il luogo dei punti descritto in figura 3.4 permette di sfruttare al meglio la macchina in quanto consente di ottenere la massima coppia con la minima corrente.

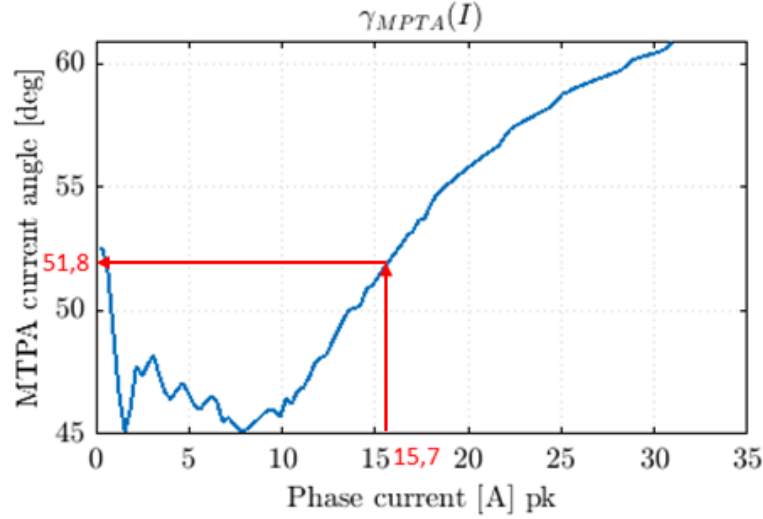


Figura 3.4. Andamento *MTPA*.

Le simulazioni *FEA* in entrambi i software sono state condotte nel modo seguente:

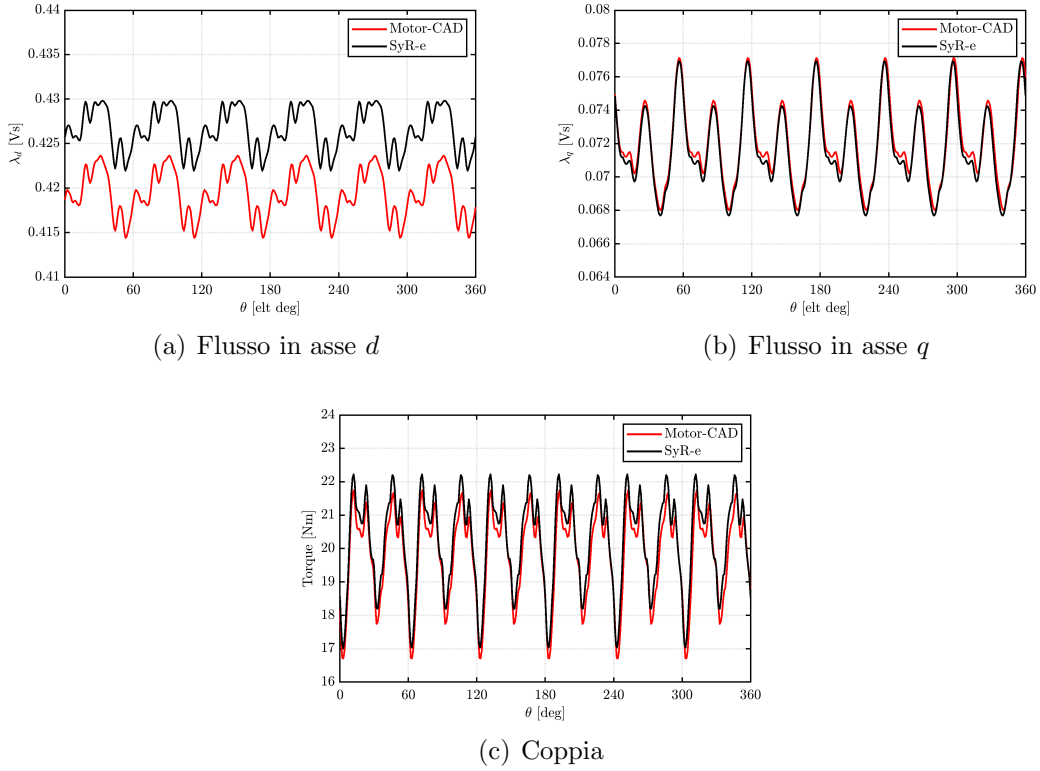
- simulazione di 60 punti ogni 60 gradi elettrici;
- settaggio della griglia *mesh* con 1440 punti interni e superficiali.

In tabella 3.1 vengono riportati i valori di flusso, corrente e coppia ottenuti in *SyR-e* e *Motor-CAD* e i rispettivi errori percentuali.

Parametro	<i>SyR-e</i>	<i>Motor-CAD</i>	Unità di misura	Errore percentuale
i_d	9.5427	9.5427	A	—
i_q	12.152	12.152	A	—
λ_d	0.4265	0.4194	Vs	0.84 %
λ_q	0.0714	0.0720	Vs	0.42 %
T	20.261	19.839	Nm	1.05 %

Tabella 3.1. Confronto dei risultati ottenuti mediante l'utilizzo dei due software.

In figura 3.5 vengono riportate e confrontate le forme d'onda ottenute simulando il medesimo motore con entrambi i software.


 Figura 3.5. Confronto dei risultati ottenuti utilizzando *SyR-e* e *Motor-CAD*.

3.4 Modelli di calcolo delle perdite nel ferro

I dati relativi alle cifre di perdita nel ferro vengono forniti dai costruttori dei lamierini ferromagnetici e, solitamente, fanno riferimento ad un'eccitazione di tipo sinusoidale. Le perdite nel ferro, per tale motivo, possono essere parametrizzate utilizzando il modello di *Bertotti* o il modello di *Steinmetz*.

Motor-CAD e *SyR-e* utilizzano delle formulazioni matematiche differenti per il calcolo delle perdite nel ferro. In entrambi i software viene utilizzato il modello di *Steinmetz*, ma vi è una diversa parametrizzazione dei coefficienti.

3.4.1 Perdite nel ferro in Motor-CAD

In *Motor-CAD* le perdite nel ferro possono essere modellizzate in due diversi modi [13]:

- Modello di Bertotti:

$$W_{Fe}[W/kg] = K_h f B^\alpha + \frac{(\text{spessore laminazione})^2}{12 \cdot \text{densità} \cdot \text{resistività elettrica}} f^2 B^2 + K_{exc} f^{3/2} B^{3/2} \quad (3.1)$$

$$W_{Fe}[W/kg] = K_h f B^\alpha + K_{eddy} f^2 B^2 + K_{exc} f^{3/2} B^{3/2} \quad (3.2)$$

Dove:

- f frequenza;
- B induzione di picco;
- K_h , α , K_{eddy} e K_{exc} considerano le perdite per isteresi, per correnti parassite e le perdite aggiuntive. I coefficienti vengono ricavati in *Motor-CAD* utilizzando una o due curve di perdita.

- Modello di Steinmetz:

$$W_{Fe}[W/kg] = K_h f B^{(\alpha+\beta B)} + 2 \cdot \pi^2 \cdot K_{eddy} f^2 B^2 \quad (3.3)$$

Dove:

- f frequenza;
- B induzione di picco;
- K_h , α e β caratterizzano le perdite per isteresi, mentre K_{eddy} tiene conto delle perdite per correnti parassite. I coefficienti vengono ricavati in Motor-CAD utilizzando tre o più curve di perdita.

3.4.2 Perdite nel ferro in SyR-e (MagNet)

Il modello di perdita utilizzato in *SyR-e (MagNet)* corrisponde al modello di *Steinmetz*, ma utilizza dei coefficienti diversi rispetto a Motor-CAD per caratterizzare le perdite, come mostrato nell'equazione 3.4 [12].

$$W_{Fe}[W/kg] = K_h f^\alpha B^\beta + K_e f^2 B^2 \quad (3.4)$$

Dove:

- f frequenza;
- B induzione di picco;
- k_h, α, β coefficienti relativi alle perdite per isteresi;
- k_e coefficiente relativo alle perdite per correnti parassite.

3.4.3 Stima dei coefficienti

Per valutare i coefficienti del modello di *Steinmetz*, sono stati utilizzati dei dati relativi alle perdite nel ferro ottenuti su lamierini ferromagnetici *M600-50A* prodotti da *ArcelorMittal*. Utilizzando la funzione *cftool* in *Matlab*, è stata condotta una stima dei coefficienti considerando dei valori di induzione compresi tra $0.1 T$ e $1.8 T$ per diversi valori di frequenza.

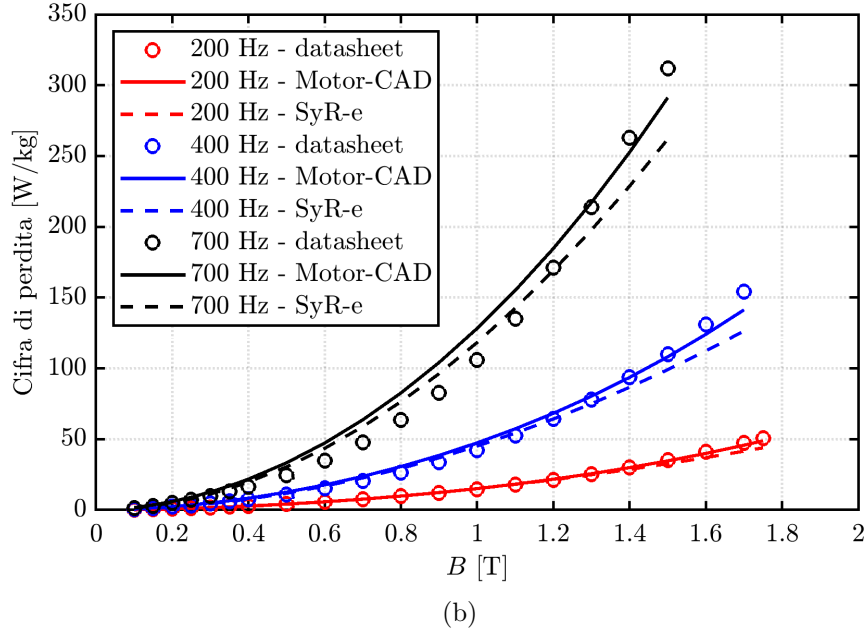
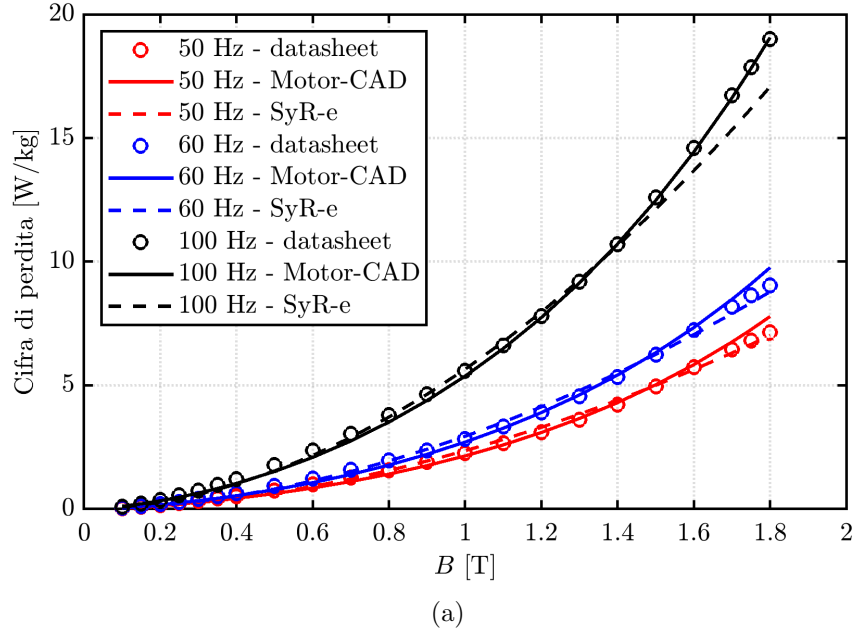


Figura 3.6. Confronto delle cifre di perdita.

In figura 3.6 viene mostrato il confronto tra le cifre di perdita fornite dal costruttore e quelle ottenute utilizzando le formule (3.3) (3.4).

In tabella 3.2 e 3.3 vengono riportati i coefficienti calcolati per *Motor-CAD* e *SyR-e*.

Coefficiente	Valore
α	1.511
β	0.4105
k_h	0.03218
k_{eddy}	$1.093 \cdot 10^{-5}$

Tabella 3.2. Coefficienti stimati per il modello di perdita di *Steimetz* in *Motor-CAD*.

Coefficiente	Valore
α	1.001
β	1.818
k_h	0.03763
k_e	$1.873 \cdot 10^{-4}$

Tabella 3.3. Coefficienti stimati per il modello di perdita in *SyR-e* (*MagNet*).

3.5 Confronto delle mappe di efficienza e di perdita

Le mappe di efficienza costituiscono un utile strumento in ambito industriale per la comparazione di differenti motori, in quanto forniscono la massima efficienza raggiunta dal motore per ogni punto di lavoro sul piano coppia-velocità.

In figura 3.7 viene riportato il confronto delle mappe di efficienza ottenute con *Motor-CAD* e *SyR-e* imponendo in entrambi i software le seguenti condizioni:

- stesse proprietà e caratteristiche del materiale ferromagnetico;
- stesso limite di corrente e tensione;
- perdite aggiuntive per effetto pelle trascurate;
- perdite meccaniche per attrito e ventilazione trascurate.

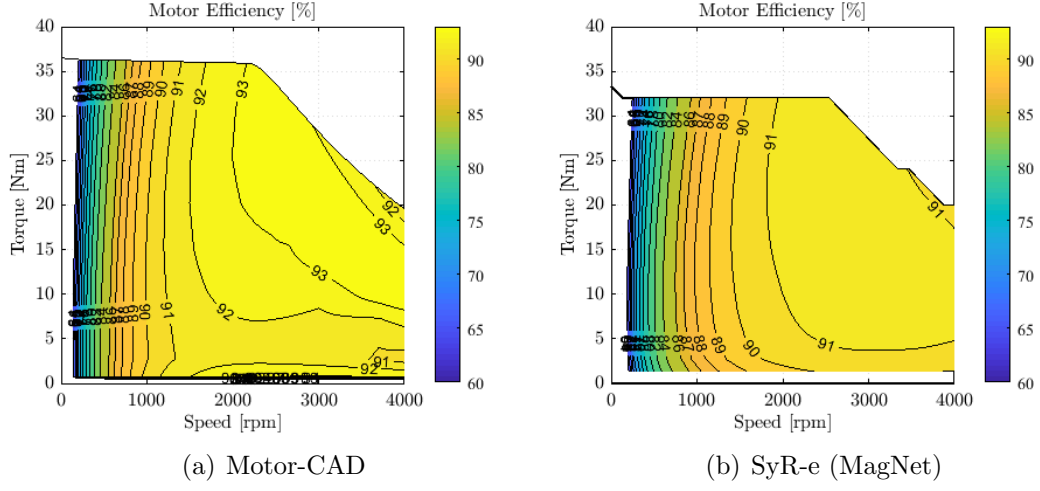


Figura 3.7. Confronto delle mappe di efficienza sul piano coppia-velocità.

Motor-CAD fornisce una sovrastima dei valori di efficienza rispetto a quelli forniti da *SyR-e*, questo è dovuto principalmente ad una sottostima delle perdite nel rame e nel ferro da parte di *Motor-CAD*, come mostrato in figura 3.8 e 3.9.

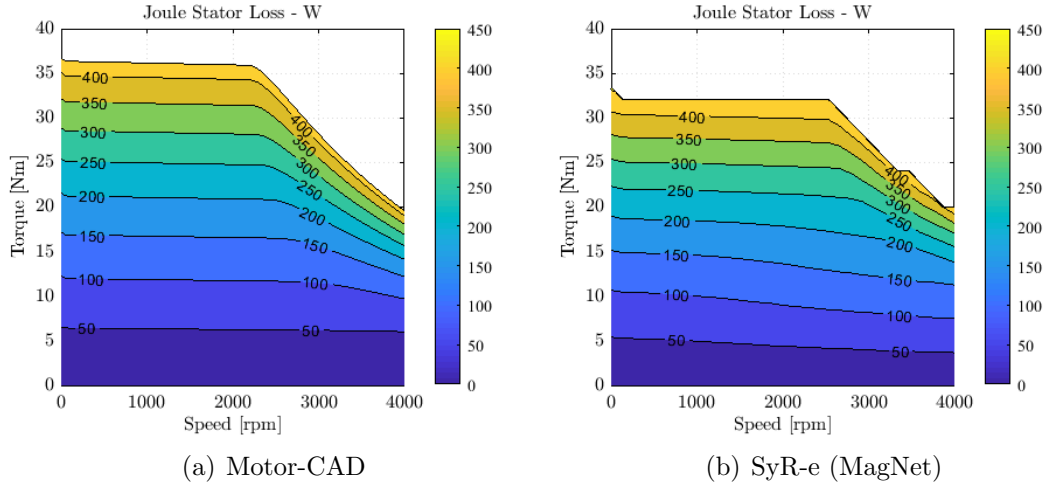


Figura 3.8. Confronto delle perdite *Joule* sul piano coppia-velocità.

I differenti valori di perdita nel rame sono in parte influenzati dalla diversa stima della resistenza degli avvolgimenti di statore (tabella 3.4) e dall'utilizzo di una diversa strategia di controllo.

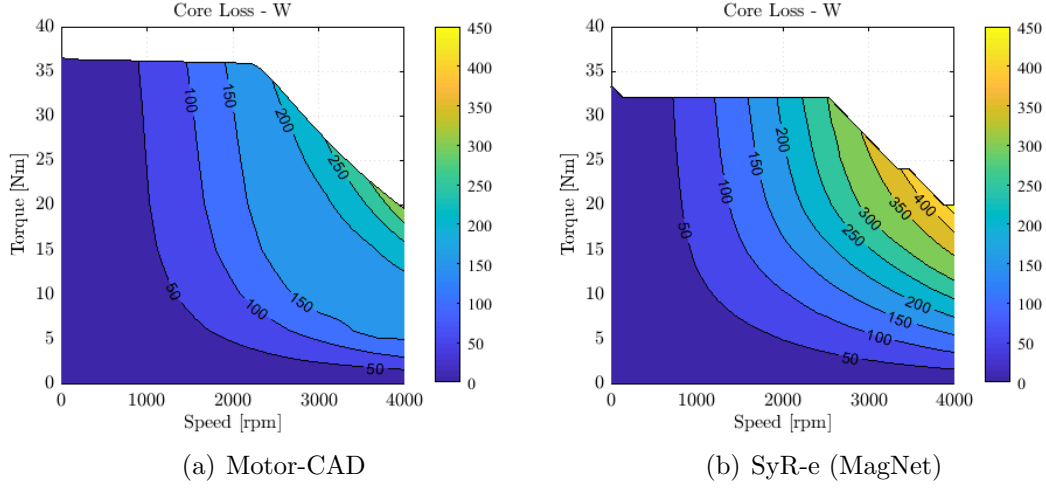


Figura 3.9. Confronto delle perdite nel ferro sul piano coppia-velocità.

Software	Valore	Unità
<i>Motor-CAD</i>	0.4566	Ω
<i>SyR-e</i>	0.4541	Ω

Tabella 3.4. Valori di resistenza forniti dai due software.

La diversa stima delle perdite nel ferro è attribuibile all'utilizzo di una differente strategia di controllo, ma anche ad una diversa formulazione delle stesse perdite nei due software, come illustrato al paragrafo 3.4. È bene fare un confronto delle perdite nel ferro lungo i punti di lavoro di *MTPA* facendo in modo che l'ampiezza e l'angolo di corrente siano gli stessi per i due modelli e per le prove sperimentali. In figura 3.10 (b) risulta evidente che *SyR-e (MagNet)* esegue una stima più accurata delle perdite nel ferro. Dal confronto delle mappe di efficienza e di perdita si evincono tre sostanziali differenze:

- diverso limite di coppia, causato da una maggiore stima di coppia in *Motor-CAD* a parità di corrente (figura 3.10 (a));
- inizio del deflussaggio a velocità inferiori in *Motor-CAD* rispetto a *SyR-e*, in quanto, a parità di tensione, in *Motor-CAD* la macchina è dotata di un flusso maggiore;

- sottostima delle perdite in *Motor-CAD*, maggiormente evidente a carico elevato.

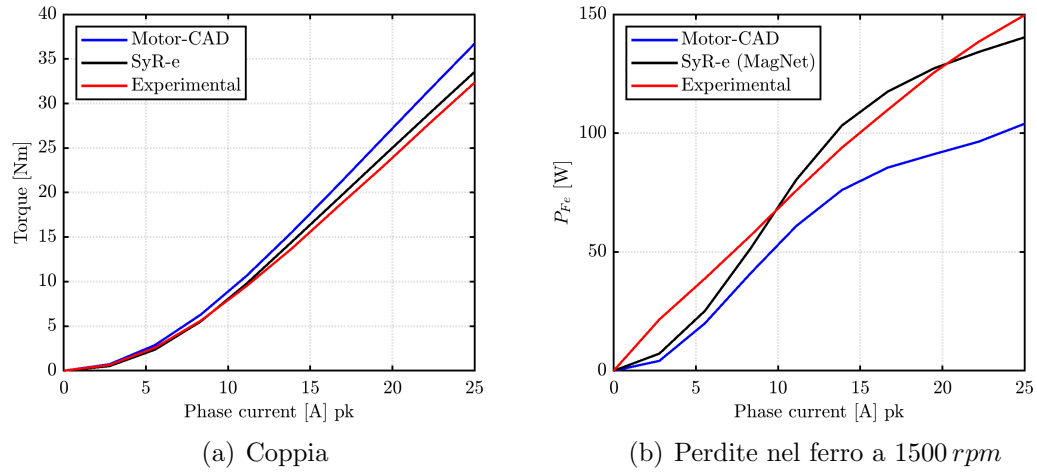


Figura 3.10. Confronto della coppia e delle perdite nel ferro lungo l'*MTPA* di corrente.

Capitolo 4

Simulazioni meccaniche

4.1 Progettazione meccanica in SyR-e

In *SyR-e* non vi è la possibilità di eseguire le verifiche strutturali sulla macchina progettata. Tuttavia, lo spessore dei ponticelli viene dimensionato in maniera automatica partendo dalla velocità massima del motore e dalle caratteristiche del materiale. Il dimensionamento tiene conto solo degli sforzi di tipo centrifugo. Lo spessore di un generico ponticello radiale è calcolato in *SyR-e* come descritto in (4.1) [14].

$$w_{rj} = \frac{F_j}{(L \cdot K \cdot \sigma_{max})} \quad (4.1)$$

Dove:

- F_j è la forza centrifuga supportata da ciascun ponticello radiale;
- L è la lunghezza assiale della macchina;
- K è un fattore di sicurezza e può assumere valori compresi nell'intervallo $[0.6 - 1]$;
- σ_{max} è la tensione di snervamento massima supportata dal materiale ferromagnetico.

La forza centrifuga supportata da ciascuna barriera di flusso viene calcolata come descritto in (4.2), assumendo che la massa M_j sia concentrata nel punto di gravità

G_j di coordinate $[r_j, \pi/(2n_p)]$, dove n_p è il numero di coppie polari.

$$F_j = M_j \cdot r_j \cdot (\omega_{max})^2 \quad (4.2)$$

Dove r_j è il raggio del centro di gravità e ω_{max} indica la massima velocità di rotazione espressa in rad/s . La massa dell'area di interesse (area verde in figura 4.1) può essere calcolata come mostrato in (4.3):

$$M_j = \rho \cdot L \cdot \Sigma_j \quad (4.3)$$

Dove ρ è la densità di massa del materiale e Σ_j è la sezione trasversale della parte di laminazione che deve essere supportata dal ponticello radiale di ciascuna barriera di flusso.

Lo spessore di ciascun ponticello radiale viene ricalcolato ogni qualvolta si modifica in *SyR-e* la massima velocità che la macchina deve supportare. Se il valore di w_{rj} risultante dal calcolo è inferiore allo spessore minimo di taglio del lamierino, lo sforzo centrifugo viene supportato solo dai ponticelli tangenziali. In caso contrario il software inserisce nel lamierino di rotore i ponticelli radiali definendone in automatico lo spessore.

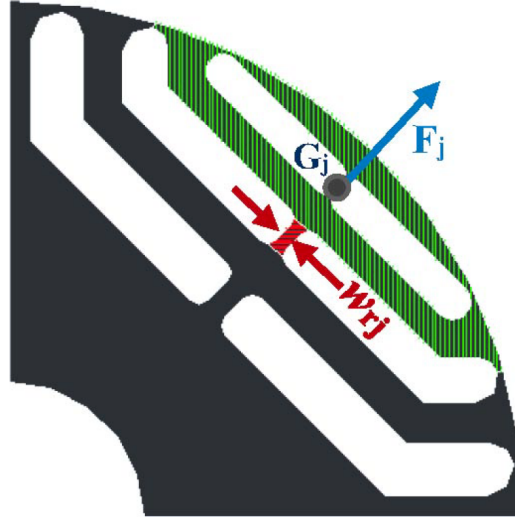


Figura 4.1. Definizione della geometria delle barriere di flusso [14].

4.2 Descrizione modulo meccanico in Motor-CAD

Motor-CAD permette di condurre l'analisi strutturale della macchina modellata utilizzando il modulo *Mechanical* (figura 4.2). I calcoli delle sollecitazioni meccaniche vengono condotti considerando solo l'effetto delle forze centrifughe, in quanto le forze magnetiche assumono normalmente dei valori bassi che risultano trascurabili [13]. La schermata *Calculation* consente di selezionare la velocità con la quale si vuole condurre l'analisi strutturale e, nel caso di macchine sincrone con magneti, se si vuole considerare la presenza di questi.

L'analisi strutturale può essere direttamente condotta sulla macchina parametrizzata in *Motor-CAD* oppure su un lamierino di rotore importato in formato *.dxf*, purché i lamierini di rotore (*Motor-CAD - Custom*) della macchina presa in considerazione risultino abbastanza simili.

La possibilità di condurre le analisi FEA sul lamierino in formato *.dxf*, risulta particolarmente utile in quanto permette di realizzare l'analisi strutturale di una macchina progettata in *SyR-e*.

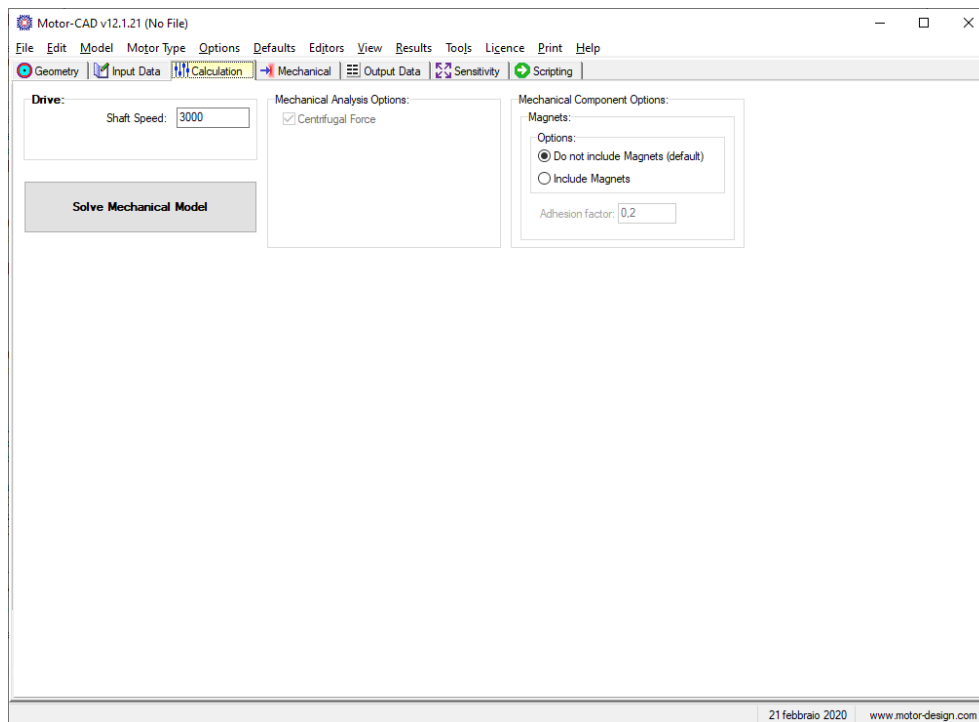


Figura 4.2. *Motor-CAD* - modulo *Mechanical*.

4.3 Validazione del progetto sviluppato in SyR-e

Per verificare la validità del dimensionamento strutturale del rotore eseguito in *SyR-e* in fase di ottimizzazione, sono state condotte delle analisi strutturali utilizzando due diversi software: *Motor-CAD* e *SolidWorks*.

In tabella 4.1 vengono riportati i dati meccanici relativi al lamierino utilizzato.

Grandezza	Valore	Unità
Modulo di Young	210000	<i>MPa</i>
Limite di snervamento	285	<i>MPa</i>
Coefficiente di Poisson	0.3	-

Tabella 4.1. Proprietà meccaniche del lamierino *M600 - 50A*.

In figura 4.3 e 4.4 vengono riportati i risultati delle simulazioni meccaniche condotte a 4000 *rpm*, in particolare viene mostrato lo stress calcolato con il metodo di *Von Mises* (SVM) e lo spostamento (U), calcolati in *Motor-CAD* e *SyR-e*. Dall'analisi risulta evidente che il limite di snervamento non viene raggiunto in nessun punto, dunque, i ponticelli tangenziali riescono a supportare lo sforzo centrifugo al quale vengono sottoposti.

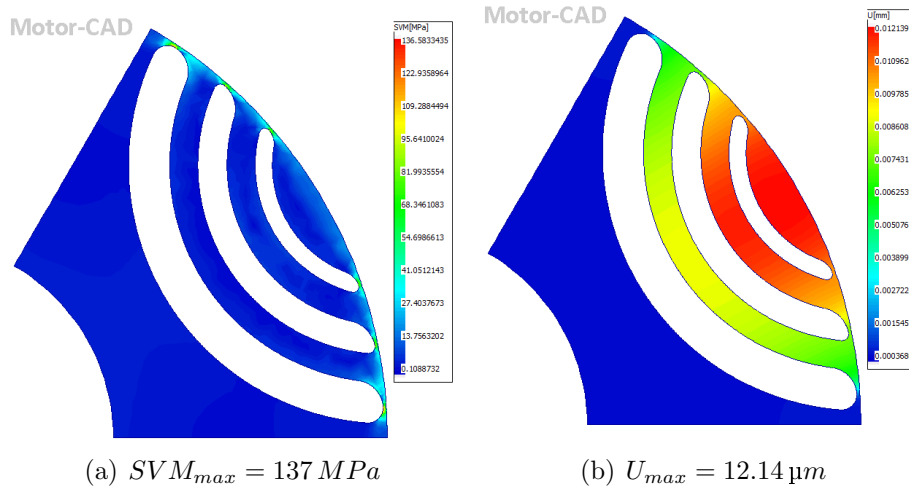


Figura 4.3. *Stress Analysis - Motor-CAD*.

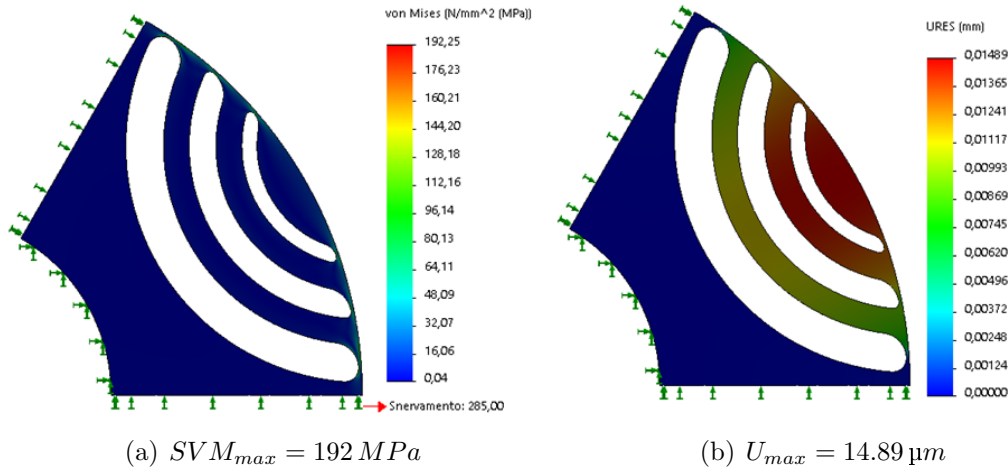


Figura 4.4. *Stress Analysis - SolidWorks.*

4.4 Macchina ad alta velocità

Per valutare in maniera più dettagliata gli effetti della forza centrifuga sulla geometria di rotore, è stato riprogettato il rotore della macchina a 4000 *rpm* (paragrafo 4.3), rendendolo in grado di sostenere una velocità operativa di 20000 *rpm*.

La progettazione della macchina ad alta velocità è stata condotta utilizzando in maniera congiunta *SyR-e* e *Motor-CAD*. In particolare, il dimensionamento dei ponticelli radiali e tangenziali è stato eseguito in *SyR-e* mentre la verifica strutturale è stata effettuata utilizzando *Motor-CAD*.

Lo studio è stato condotto sul medesimo lamierino di rotore sfruttando la nuova funzionalità di esportazione in *Motor-CAD* della geometria di macchina introdotta in *SyR-e*, come illustrato al capitolo 2.

Con l'aumentare della velocità di rotore, è necessario inserire dei ponticelli radiali che permettano di assorbire parte del carico centrifugo.

Per iniziare, è stata eseguita l'analisi statica sul lamierino dimensionato in *SyR-e*. In figura 4.5 è possibile osservare che le zone sottoposte a maggiore stress sono quelle in corrispondenza dei ponticelli radiali e tangenziali (aree in rosso), dove si riscontra uno stress massimo di 620 *MPa*. Per garantire l'integrità meccanica del lamierino, lo sforzo massimo deve essere in ogni punto inferiore al limite di snervamento, corrispondente in questo caso a 285 *MPa*.

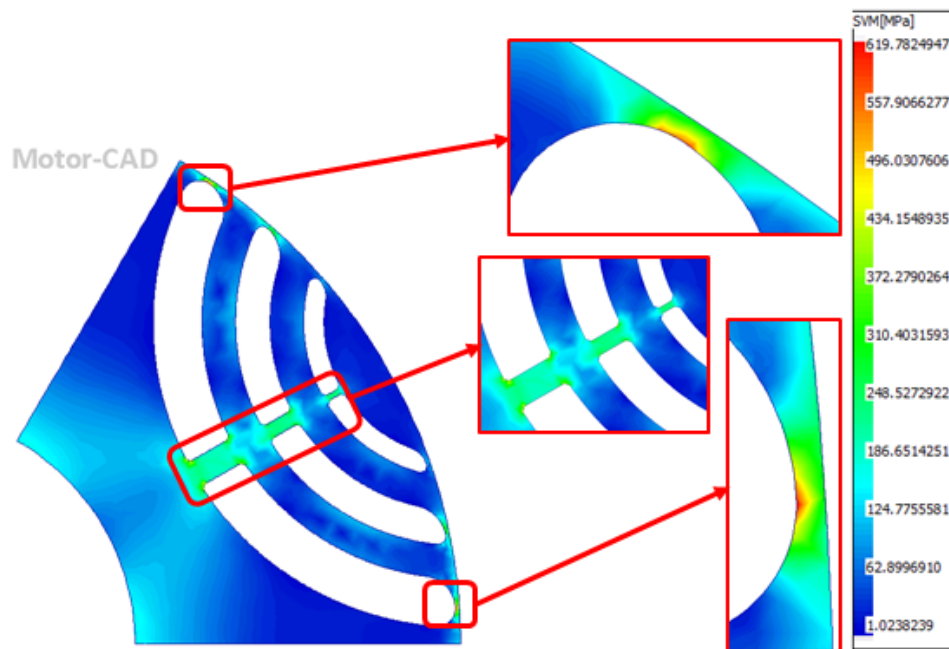


Figura 4.5. Simulazione meccanica sul lamierino iniziale.

Per ridurre lo stress meccanico in corrispondenza dei ponticelli tangenziali è stato necessario aumentare lo spessore degli stessi, mentre per i ponticelli radiali è stato sufficiente arrotondare gli spigoli utilizzando *AutoCAD*.

In tabella 4.2 vengono riportate le dimensioni iniziali e finali dei ponticelli radiali e tangenziali.

Barriere di flusso	Ponticelli radiali		Ponticelli tangenziali		Unità
	Iniziale	Finale	Iniziale	Finale	
1°	0.62	0.62	0.4	0.8	mm
2°	1.58	1.58	0.4	1.2	mm
3°	2.86	2.86	0.4	1.8	mm

Tabella 4.2. Dimensioni iniziali e finali dei ponticelli.

Le modifiche apportate al lamierino di rotore vengono mostrate in figura 4.6. Poiché le prestazioni in termini di coppia e potenza sono influenzate dallo spessore dei ponticelli risulta conveniente ridurre al minimo il loro spessore, in modo tale

da ottenere una macchina meccanicamente valida e dotata di buone prestazioni. Il ridimensionamento viene effettuato sul file *.dxf* utilizzando *SyR-e*. Non può essere effettuato direttamente su *Motor-CAD* a causa dell'assenza di una geometria di rotore di tipo *Circular*.

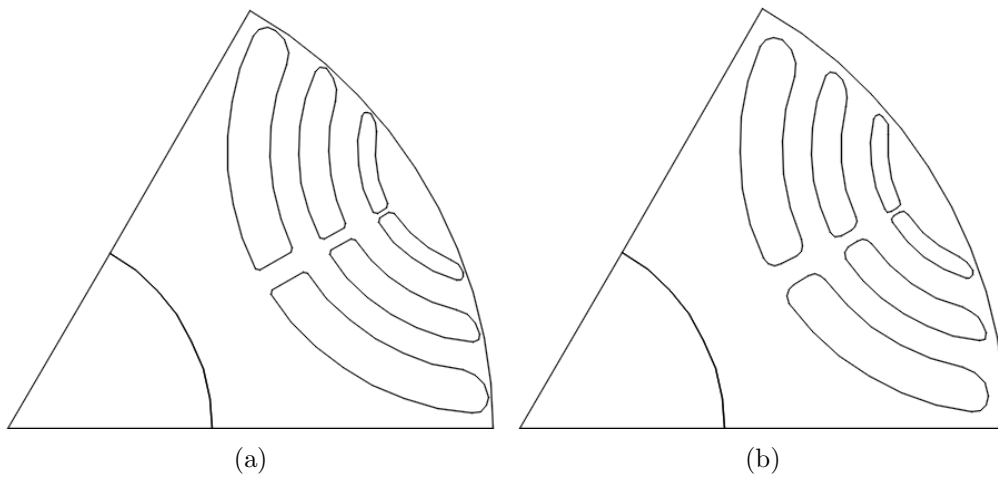


Figura 4.6. Modifica del lamierino di rotore.

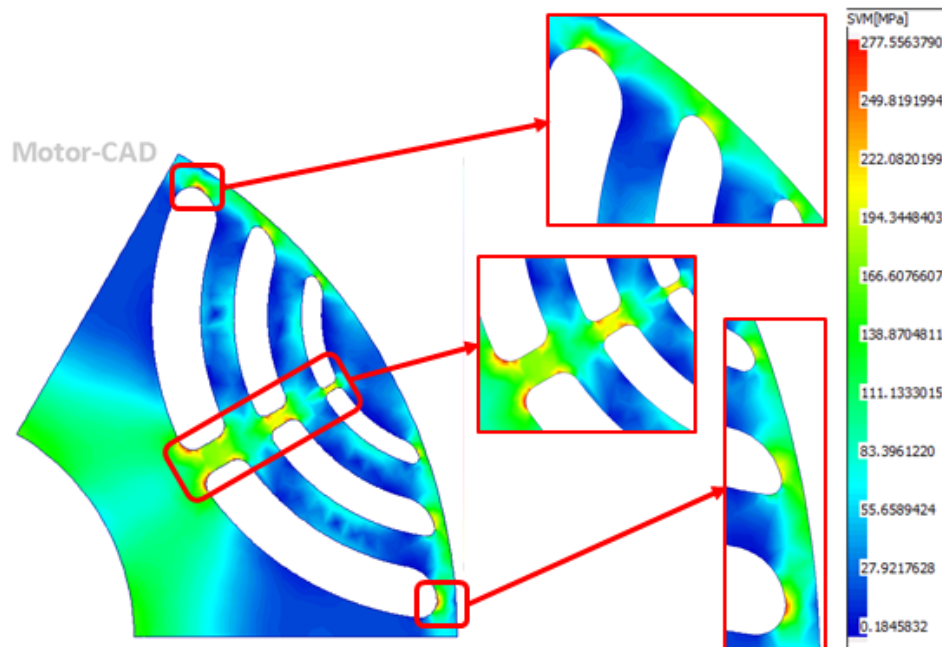


Figura 4.7. Simulazione meccanica sul lamierino modificato.

Dopo aver attuato tutte le opportune modifiche, lo stress in ogni punto della macchina risulta essere inferiore al limite di snervamento, come riportato in figura 4.7.

Il processo iterativo utilizzato viene mostrato in figura 4.8.

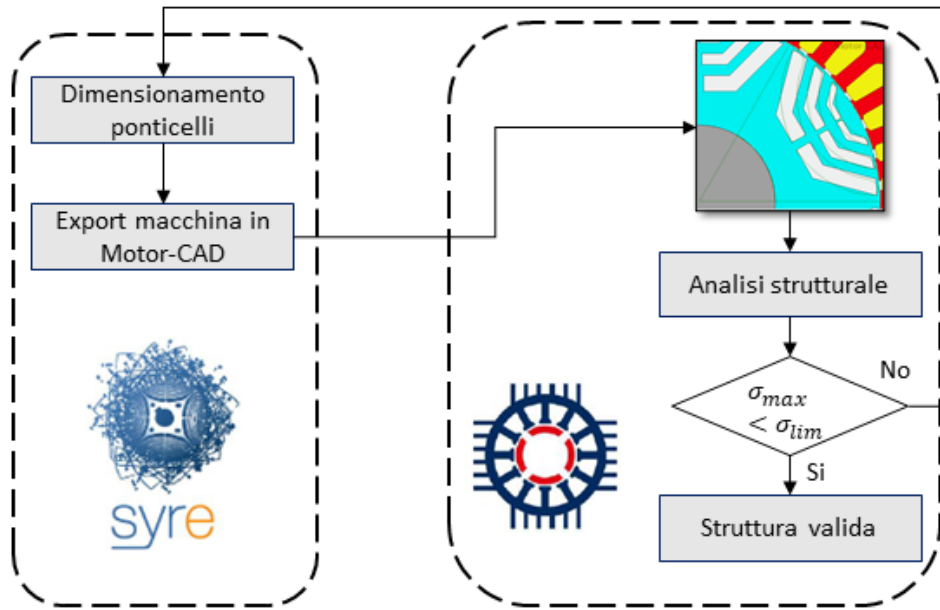


Figura 4.8. Ciclo iterativo per la progettazione meccanica.

La tabella 4.3 riassume i risultati delle simulazioni meccaniche in termini di stress massimo e spostamento prima e dopo le modifiche apportate. In entrambe le geometrie si è riscontrato uno spostamento inferiore al 12% rispetto allo spessore del traferro, ciò non causa particolari problematiche.

Grandezza	Valore iniziale	Valore finale	Unità
Stress massimo	620	278	MPa
Spostamento massimo	38.60	28.87	μm

Tabella 4.3. Confronto delle grandezze meccaniche ottenute.

È stata eseguita un'ulteriore verifica della macchina finale utilizzando *Solid-Works*.

Dalla figura 4.9 si evince che il limite di snervamento viene superato in corrispondenza delle curvature dei ponticelli radiali. La struttura, tuttavia, viene considerata meccanicamente valida in quanto un errore di questo tipo può essere attribuito ad una diversa *mesh* nei due software. Questo problema è risolvibile arrotondando ulteriormente i raggi di curvatura.

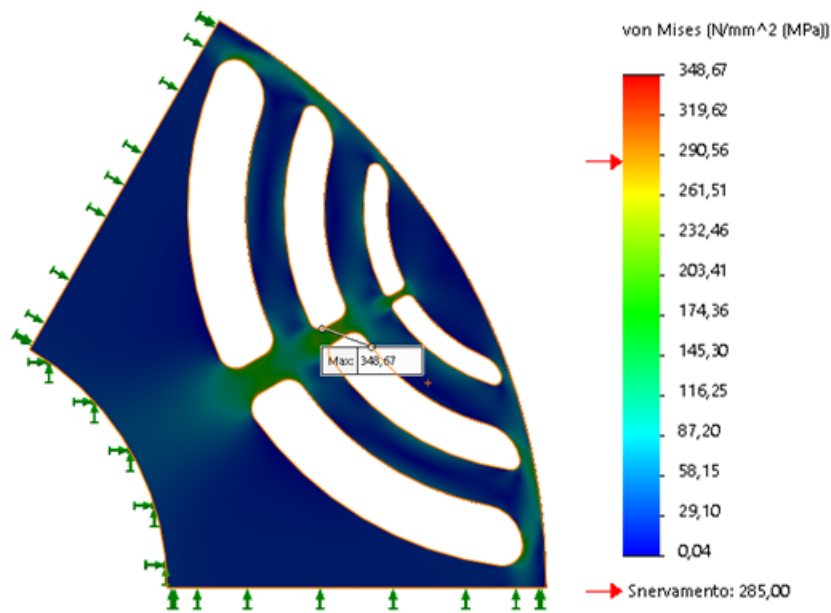


Figura 4.9. Simulazione meccanica sul lamierino modificato.

Capitolo 5

Simulazioni termiche

5.1 Descrizione del modulo termico in Motor-CAD

Il modulo *Thermal* (figura 5.1), presente in *Motor-CAD*, consente di valutare le prestazioni termiche della macchina elettrica, eseguendo analisi a regime o in transitorio [13]. Per accorciare i tempi di calcolo è possibile ridurre il numero di nodi della rete termica, è importante tuttavia non eliminare i nodi in cui le perdite risultano essere sostanziali.

Motor-CAD utilizza di default un modello termico completo 3D che tiene conto delle superficie di contatto e degli avvolgimenti di testata. Per poter confrontare il modello con i risultati dell'analisi *FEA* della singola cava risulta necessario selezionare il modello 2D. Quest'ultimo è un modello più semplificato che permette di condurre in modo più accurato la calibrazione *FEA* in quanto:

- considera solo le parti attive della macchina, trascurando gli avvolgimenti di testata;
- trascura le perdite nel ferro;
- non considera alcun sistema di raffreddamento.

Per condurre un'analisi termica è necessario conoscere tutte le perdite della macchina, *Motor-CAD* permette di importarle direttamente dal modello elettromagnetico

o di impostarle manualmente.

Nella finestra *Cooling* è possibile impostare le opzioni di raffreddamento. In particolare in un primo momento viene richiesto di scegliere tra il raffreddamento a convezione naturale e quello a convezione forzata, per poi selezionare uno specifico sistema di raffreddamento. *Motor-CAD* consente di condurre analisi termiche effettuando un singolo calcolo transitorio oppure impostando dei cicli di lavoro più o meno complessi. Nel singolo calcolo transitorio le perdite vengono considerate costanti, mentre nei cicli di lavoro è possibile definirne la variabilità in funzione del tempo. In entrambi i casi le capacità termiche sono calcolate automaticamente in *Motor-CAD* in relazione alle caratteristiche dei materiali e alla geometria della macchina.

Per impostare un ciclo di lavoro è indispensabile definirne la durata e la variazione di coppia e velocità in funzione del tempo. L'opzione *Transient Start Point* consente di specificare la temperatura iniziale del calcolo transitorio che generalmente corrisponde alla temperatura ambiente o a quella di regime.

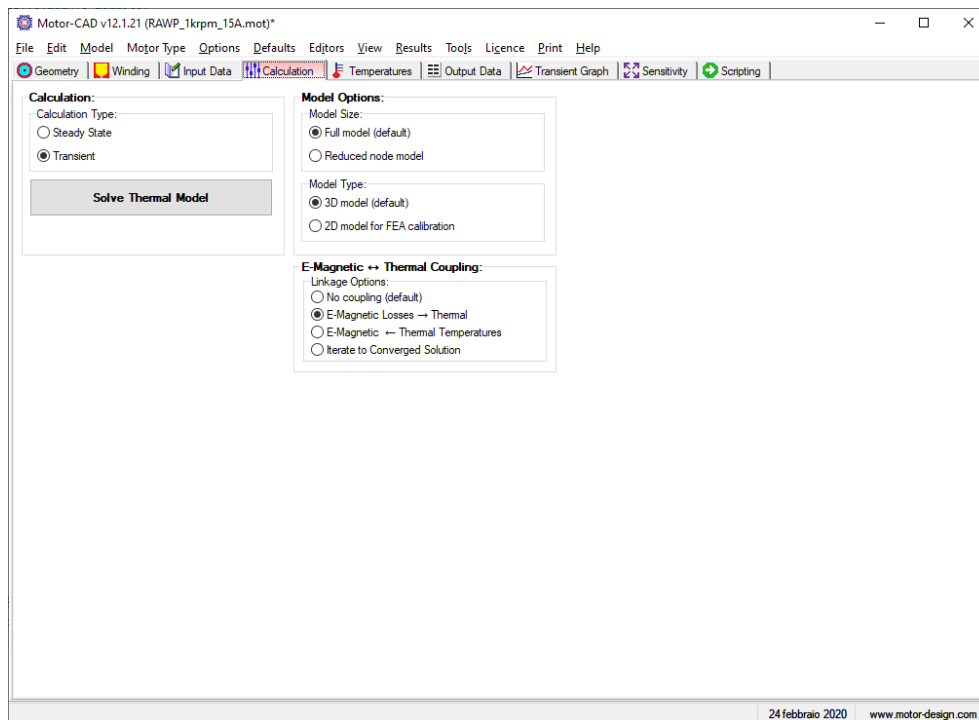


Figura 5.1. *Motor-CAD* - modulo *Thermal*.

5.2 Modello della carcassa di riferimento

Per poter riprodurre in *Motor-CAD* il modello della carcassa del motore di riferimento è stato necessario attuare in laboratorio un rilevamento dettagliato delle misure di alcune componenti. In particolare sono state definite le seguenti grandezze:

- lunghezza assiale della carcassa;
- spessore e diametro della carcassa;
- dimensione delle alette;
- dimensione della ventola;
- dimensione encoder;
- sporgenza degli avvolgimenti di testata;
- dimensione del coperchio della ventola.

Nella finestra *Geometry* presente nel modulo termico di *Motor-CAD* sono state riportate tutte le misure rilevate. Inoltre è stato possibile impostare la tipologia di carcassa desiderata tra quelle disponibili che, in questo caso, corrisponde alla tipologia *Axial Fins (RoundSt)*.

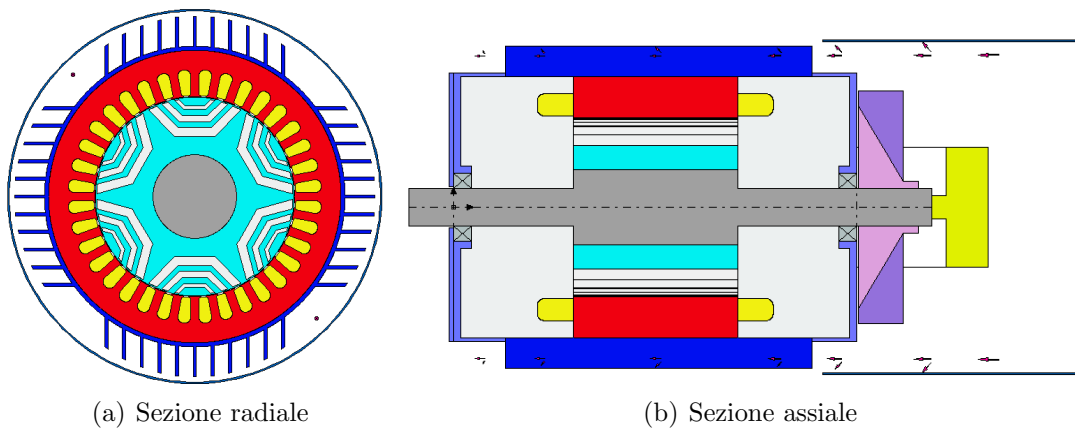


Figura 5.2. Modello bidimensionale del motore di riferimento.

In figura 5.2 viene mostrata la sezione radiale (a) e la sezione assiale della macchina in questione, completa del sistema di raffreddamento. Motor-CAD restituisce anche il modello 3D del motore, rappresentato in figura 5.3. In figura 5.3 (a) è possibile osservare la direzione del flusso d'aria (freccia rossa) generato dalla ventola.

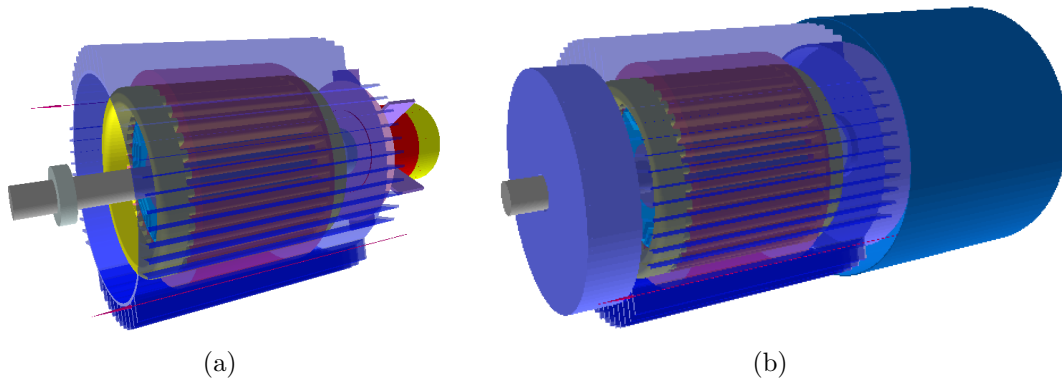


Figura 5.3. Modello tridimensionale del motore di riferimento.

5.3 Costruzione del ciclo di lavoro considerando la coppia

Il modello termico dinamico è un modello ad elevata complessità, per cui è stato necessario far ricorso a dati ottenuti attraverso prove sperimentali per poter condurre un'attenta verifica del modello sviluppato in *Motor-CAD*. Il ciclo di lavoro utilizzato è quello legato alla misura della mappa di efficienza della macchina, come descritto in [6].

L'implementazione di un ciclo di lavoro in *Motor-CAD* richiede che vengano definite le variazioni di coppia e velocità in funzione del tempo, queste vengono fornite al software in formato *.txt*. Il ciclo di lavoro è stato costruito tramite script *Matlab* avendo noti i seguenti dati:

- coppia di riferimento;
- incremento del gradino di coppia;

- variazione della velocità per ogni *interrupt*;
- variazione della coppia per ogni *interrupt*;
- periodo di *interrupt*;
- durata della misura di resistenza.

In figura 5.4 viene rappresentato l'intero ciclo di lavoro articolato in nove livelli di velocità.

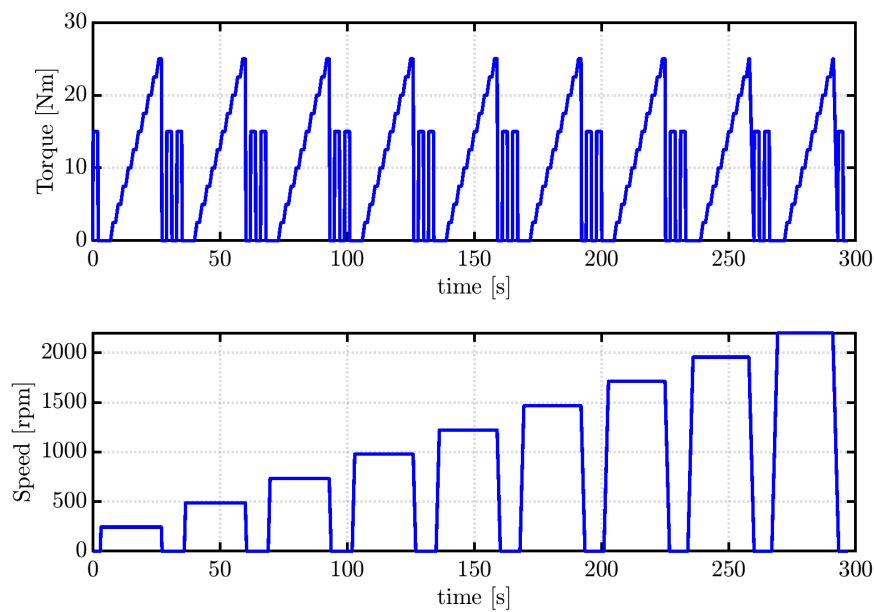


Figura 5.4. Rappresentazione della variazione di coppia durante un intero ciclo di lavoro.

In figura 5.5 viene rappresentata una fase dell'intero ciclo di lavoro in cui, impostando una velocità costante, si induce una variazione a gradino della coppia. Prima e dopo la variazione della coppia, quando la velocità risulta nulla e la coppia costante, viene eseguita la misura della resistenza degli avvolgimenti di statore.

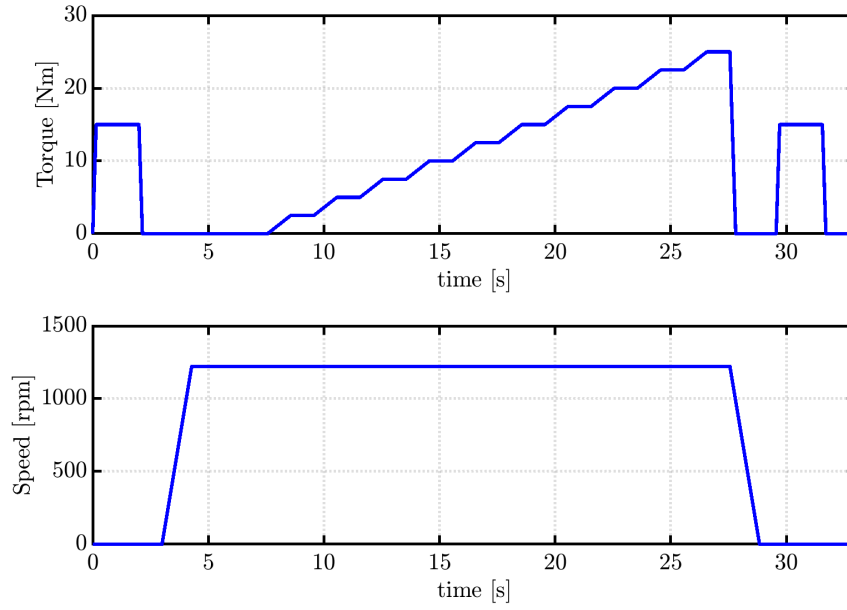


Figura 5.5. Andamento della coppia durante una fase del ciclo di lavoro alla velocità di 1222 rpm.

In tale contesto le misure di resistenza dello statore sono state sfruttate per ottenere una stima delle temperature tramite l'equazione (5.1).

$$\theta = \frac{R}{R_0} \cdot (235 + \theta_a) - 235 \quad (5.1)$$

Dove:

- θ è la temperatura stimata;
- R è la resistenza misurata negli avvolgimenti di statore;
- R_0 è la resistenza misurata a θ_a ;
- θ_a è la temperatura ambiente.

5.4 Costruzione del ciclo di lavoro considerando le perdite

In *Motor-CAD* è possibile costruire un ciclo di lavoro considerando la variazione delle perdite in funzione del tempo. Questo permette di ottenere dei risultati più accurati in fase di confronto tra dati simulati e sperimentali.

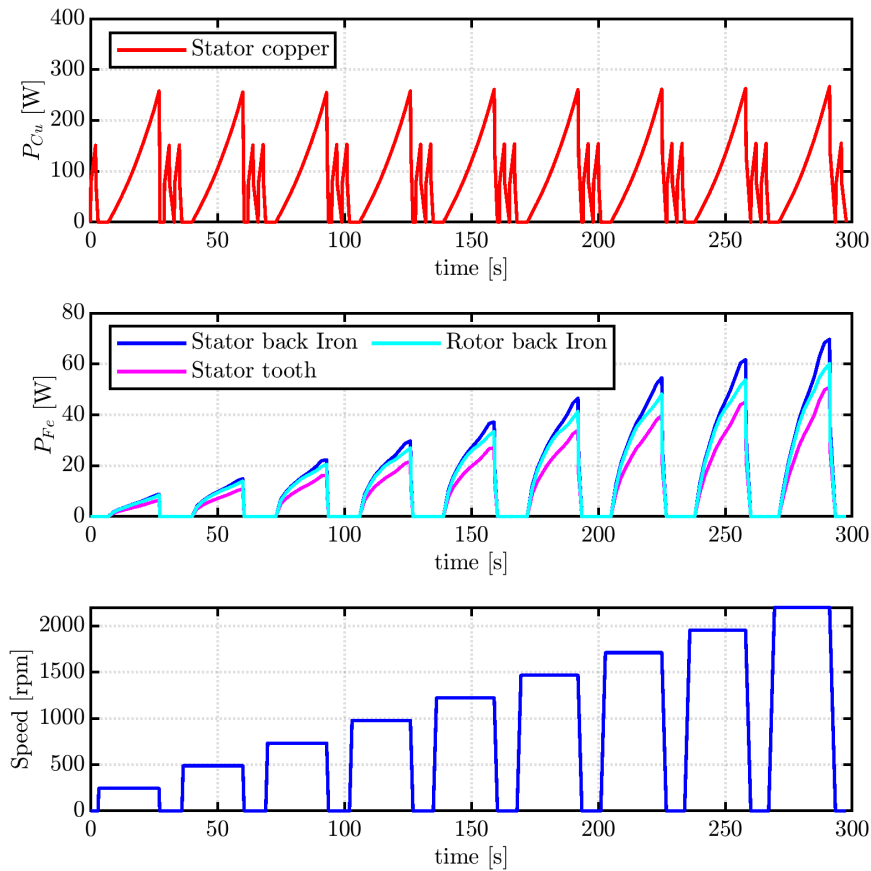


Figura 5.6. Rappresentazione della variazione delle perdite durante un intero ciclo di lavoro.

Per la costruzione del ciclo di lavoro, è necessario conoscere i seguenti dati:

- perdite nel ferro nel gioco e nel dente di statore;
- perdite nel ferro nel gioco di rotore;

- perdite nel rame negli avvolgimenti di statore.

Le misure sperimentali permettono di ottenere dei valori complessivi di perdita che devono essere suddivisi nelle specifiche componenti. Per ottenere le perdite nel ferro di statore e rotore è stato ricavato un coefficiente partendo dal calcolo delle perdite in *MagNet*. Per la successiva suddivisione tra dente e gioco di statore si è fatto ricorso ai risultati forniti dalla simulazione magnetica condotta in *Motor-CAD*.

In figura 5.6 viene mostrato come variano le perdite durante un intero ciclo di lavoro.

5.5 Confronto dei valori di temperatura simulati e sperimentali

I valori di temperatura sperimentali vengono posti a confronto con quelli forniti dalle simulazioni termiche condotte in *Motor-CAD* (figura 5.7 (a)). Le simulazioni sono state eseguite adoperando entrambi i cicli di lavoro precedentemente descritti. I valori di temperatura sperimentali risultavano eccessivamente dispersivi, per cui si è ritenuto opportuno effettuarne un fit.

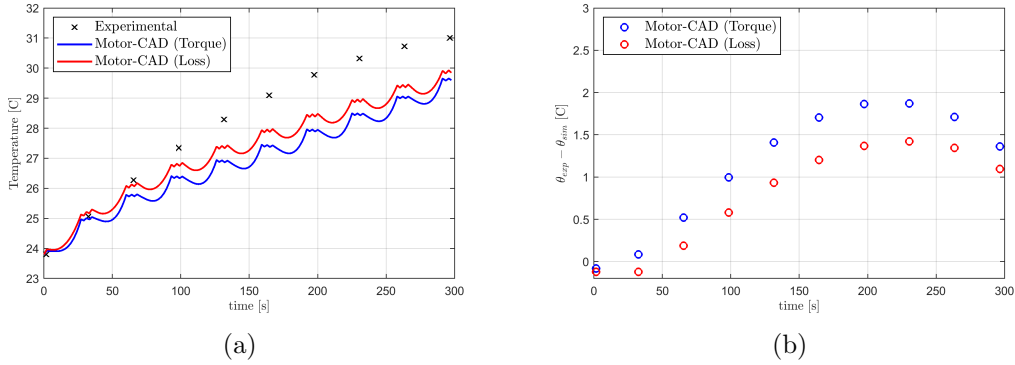


Figura 5.7. Confronto tra temperature simulate e sperimentali.

Come mostrato in figura 5.7 (b) vi è una maggiore differenza tra i valori di temperatura sperimentali e quelli simulati attraverso un ciclo di lavoro che considera la coppia (in blu) piuttosto che tra le misure sperimentali e quelle simulate considerando le perdite (in rosso).

Il modello termico potrebbe essere ulteriormente tarato, tuttavia si può già ritenere valido in quanto i valori di temperatura stimati e simulati si discostano al massimo di $2^{\circ}C$.

Capitolo 6

Conclusioni

Questo lavoro ha voluto fornire un contributo nella progettazione di motori sincroni a riluttanza mediante lo sviluppo di apposite funzioni nel software *SyR-e*. I risvolti chiave raggiunti sono i seguenti:

- Sviluppo di script *Matlab* per l'utilizzo complementare di *SyR-e* e *Motor-CAD* in fase di progettazione di un motore *SyR*, permettendo così di sfruttare al massimo le potenzialità dei due software.
- Inserimento nella *GUI SyR-e* di una nuova finestra dedicata che permette di interfacciare con semplicità i due software di progettazione, mediando l'esportazione dei dati di progetto e la messa a punto di simulazioni magnetiche e termiche in *Motor-CAD*.
- Validazione della nuova procedura di progettazione mediante analisi di confronto di tipo magnetico, termico e strutturale.

Possibili sviluppi futuri potrebbero essere:

- Permettere l'esportazione in *Motor-CAD* di tutte le geometrie di macchina presenti in *SyR-e*.
- Definire in modo più dettagliato nella *GUI SyR-e* il sistema di raffreddamento adoperato nelle analisi termiche in *Motor-CAD*.

- Rendere possibile l'importazione in *SyR-e* dei risultati del modello termico sviluppato in *Motor-CAD*, al fine di inserire l'informazione della temperatura nelle mappe di efficienza e effettuare il calcolo delle prestazioni nominali e della durata ammessa del sovraccarico.

Appendice A

Script per l'esportazione della macchina da SyR-e a Motor-CAD

```
1
2 function dataSet = DrawAndSaveMachine_MCAD(dataSet,filename,pathname)
3
4 filename=dataSet.currentfilename;
5 pathname=dataSet.currentpathname;
6 load([pathname filename]);
7
8 if strcmp(dataSet.TypeOfRotor,'Circular')
9     draw_motor_in_MCAD(filename, pathname);
10 else
11     error('Error: Export of the motor is only possible for circular geometries...')
12 end
13
14 dataSet.currentpathname = [pathname];
15 dataSet.currentfilename = filename;
16 end
```

Listing A.1. DrawAndSaveMachine_MCAD.m

```
1 function draw_motor_in_MCAD(filename, pathname)
2
3 load([pathname filename])
4 mcad=actxserver('MotorCAD.AppAutomation');
5
6 % stator parameters
7 slot=geo.parallel_slot;
```

```

8  if slot==0
9      invoke(mcad,'SetVariable','SlotType',0);      %slot type (ParallelTooth)
10 else
11     invoke(mcad,'SetVariable','SlotType',2);      %slot type (ParallelSlot)
12 end
13
14 Q=6*geo.p*geo.q;
15 tmp=num2str(Q);
16 tmp(tmp=='.')=',';
17 invoke(mcad,'SetVariable','Slot_number',tmp);
18
19 tmp=2*dataSet.StatorOuterRadius;
20 tmp=num2str(tmp);
21 tmp(tmp=='.')=',';
22 invoke(mcad,'SetVariable','Stator_Lam_Dia',tmp);
23
24 tmp=geo.l;
25 tmp=num2str(tmp);
26 tmp(tmp=='.')=',';
27 invoke(mcad,'SetVariable','Stator_Lam_Length',tmp);
28 invoke(mcad,'SetVariable','Rotor_Lam_Length',tmp);
29 invoke(mcad,'SetVariable','Magnet_Length',tmp);
30
31 tmp=2.5*dataSet.StatorOuterRadius;
32 tmp=num2str(tmp);
33 tmp(tmp=='.')=',';
34 invoke(mcad,'SetVariable','Housing_Dia',tmp);
35
36 tmp=2*(geo.r+geo.g);
37 tmp=num2str(tmp);
38 tmp(tmp=='.')=',';
39 invoke(mcad,'SetVariable','Stator_Bore',tmp);
40
41 tmp=geo.g;
42 tmp=num2str(tmp);
43 tmp(tmp=='.')=',';
44 invoke(mcad,'SetVariable','Airgap',tmp);
45
46 if slot==0
47     tmp=num2str(geo.wt);
48     tmp(tmp=='.')=',';
49     invoke(mcad,'SetVariable','Tooth_Width',tmp);      %ParallelTooth
50 else
51     tmp=(geo.r+geo.g+geo.lt/15)*sin(pi/geo.p/geo.Qs)-geo.wt;
52     tmp=num2str(tmp);
53     tmp(tmp=='.')=',';
54     invoke(mcad,'SetVariable','Slot_Width',tmp);      %ParallelSlot
55 end

```

```

56
57 tmp=geo.lt;
58 tmp=num2str(tmp);
59 tmp(tmp=='.')=',';
60 invoke(mcad,'SetVariable','Slot_Depth',tmp);
61 if slot==0
62     tmp=num2str(geo.SFR);
63     tmp(tmp=='.')=',';
64     invoke(mcad,'SetVariable','Slot_Corner_Radius',tmp);%ParallelTooth
65 end
66
67 tmp=num2str(geo.ttd);
68 tmp(tmp=='.')=',';
69 invoke(mcad,'SetVariable','Tooth_Tip_Depth',tmp);
70 tmp=(geo.acs)*((geo.r+geo.g)*2*pi/Q);
71 tmp=num2str(tmp);
72 tmp(tmp=='.')=',';
73 invoke(mcad,'SetVariable','Slot_Opening',tmp);
74
75 tmp=num2str(geo.tta);
76 tmp(tmp=='.')=',';
77 invoke(mcad,'SetVariable','Tooth_Tip_Angle',tmp);
78 invoke(mcad,'SetVariable','Sleeve_Thickness','0');
79
80 tmp=num2str(geo.win.Nbob);
81 tmp(tmp=='.')=',';
82 invoke(mcad,'SetVariable','MagTurnsConductor',tmp);
83
84 %%%%%%%%%%%
85 if geo.nmax>9999
86     invoke(mcad,'SetVariable','Wedge_Model','0'); %% with wedge
87 else
88     invoke(mcad,'SetVariable','Wedge_Model','1'); %% without wedge
89 end
90 %%%%%%%%%%%
91
92 % rotor parameters
93 tmp=geo.Ar*2;
94 tmp=num2str(tmp);
95 tmp(tmp=='.')=',';
96 invoke(mcad,'SetVariable','Shaft_Dia',tmp); %rotor diameter
97 invoke(mcad,'SetVariable','Pole_number',geo.p*2);
98
99 %Stator and Rotor angle
100 invoke(mcad,'SetVariable','StatorRotation',0);
101 invoke(mcad,'SetVariable','RotorRotation',(90/geo.p));
102 invoke(mcad,'SetVariable','BPM_Rotor','13')
103

```



```
104 %number of magnetic layers
105 invoke(mcad, 'SetVariable', 'Magnet_Layers', int2str(geo.nlay));
106
107 %centre posts
108 tmp=flip1r(geo.pontR);
109 tmp=mat2str(tmp);
110 tmp=tmp(2:end-1);
111 tmp(tmp==' ')=':';
112 invoke(mcad, 'SetVariable', 'UShape_CentrePost_Array', tmp);
113
114 nlay=geo.nlay;
115 mg_leng=zeros(1,nlay);
116 mg_leng=mat2str(mg_leng);
117 mg_leng=mg_leng(2:end-1);
118 mg_leng(mg_leng==' ')=':';
119 invoke(mcad, 'SetVariable', 'UMagnet_Length_Inner_Array', mg_leng);
120 invoke(mcad, 'SetVariable', 'UMagnet_Length_Outer_Array', mg_leng);
121
122 alpha=cumsum(geo.dalpha);
123 delta = [alpha(1) diff(alpha) (90/geo.p)-alpha(end)];
124
125 if geo.p==2
126     tmp=(-delta(2)).*ones(1,nlay);
127 elseif geo.p==3
128     tmp=(-delta(2)*2).*ones(1,nlay);
129 elseif geo.p==4
130     tmp=(-delta(2)*3.8).*ones(1,nlay);
131 else
132     tmp=(-delta(2)).*ones(1,nlay);
133 end
134
135 tmp=mat2str(tmp);
136 tmp=tmp(2:end-1);
137 tmp(tmp==' ')=':';
138 invoke(mcad, 'SetVariable', 'UShape_OuterAngleOffset_Array', tmp);
139
140 tmp=flip1r(geo.hc);
141 tmp=mat2str(tmp);
142 tmp=tmp(2:end-1);
143 tmp(tmp==' ')=':';
144 invoke(mcad, 'SetVariable', 'UShape_Thickness_Inner_Array', tmp);
145 invoke(mcad, 'SetVariable', 'UShape_Thickness_Outer_Array', tmp);
146
147 hc=geo.hc(1);
148 tmp=0;
149 for i=1:1:nlay
150     tmp(i)=2*geo.B1k(nlay-i+1);
151 end
```

```

152 tmp=mat2str(tmp);
153 tmp=tmp(2:end-1);
154 tmp(tmp==' ')=':';
155 invoke(mcad,'SetVariable','UShape_InnerRadius_Array',tmp);
156
157 tmp=fliplr(geo.pontT);
158 tmp=mat2str(tmp);
159 tmp=tmp(2:end-1);
160 tmp(tmp==' ')=':';
161 invoke(mcad,'SetVariable','UShape_BridgeThickness_Array',tmp);
162
163 if geo.p==2
164     tmp=sqrt(2)/2*geo.xxD1k-sqrt(2)/2*geo.yyD1k;
165 else
166     m=tan(pi/(2*geo.p));
167     tmp=(abs(geo.yyD1k-m*geo.xxD1k))/(sqrt(1+m^2));
168 end
169     tmp=2*fliplr(tmp);
170     tmp=mat2str(tmp);
171     tmp=tmp(2:end-1);
172     tmp(tmp==' ')=':';
173     invoke(mcad,'SetVariable','UShape_WebThickness_Array',tmp);
174     tmpPont=geo.pontT;
175
176 %%%%%%%%% Materials
177 tmp=geo.BKLABELS.materials(4);
178 tmp = convertCharsToStrings(tmp);
179 invoke(mcad,'SetComponentMaterial','Stator Lam (Back Iron)',tmp);
180 invoke(mcad,'SetComponentMaterial','Stator Lam (Tooth)',tmp); %stator
181 invoke(mcad,'SetComponentMaterial','Rotor Lam (Back Iron)',tmp); %rotor
182 invoke(mcad,'SetComponentMaterial','Shaft [Active]',tmp); %shaft
183
184 %%%%%%%%%Lamination stacking factor
185 invoke(mcad,'SetVariable','Stacking_Factor_[Stator]',1);
186 invoke(mcad,'SetVariable','Stacking_Factor_[Rotor]',1);
187
188 file_mot=strrep(filename,'.mat','.mot');
189 invoke(mcad,'SaveToFile',[pathname file_mot]);
190
191 %export winding to MotorCAD
192 windingSyreToMCAD(mcad,pathname,filename,file_mot)
193
194 %create a proper .dxf for MotorCAD with 1 slot and 1 pole
195 syreToDxfMCAD(pathname,filename)
196
197 %.dxf MCAD settings
198 invoke(mcad,'LoadDXFFile',[pathname filename(1:end-4),'.dxf']);
199 invoke(mcad,'SetVariable','UseDXFImportForFEA_Magnetic', true);

```

```
200 invoke(mcad, 'SetVariable', 'UseDXFImportForFEA_Mechanical', true);
201 invoke(mcad, 'SetVariable', 'DXFImportType', 1);
202
203 %Save MCAD model
204 invoke(mcad, 'SaveToFile', [pathname file_mot]);
205
206 invoke(mcad, 'Quit');
207
208 %Save workspace
209 save([pathname, filename], 'dataSet', 'geo', 'per', 'mat');
210
211 disp('Motor-CAD file saved in:')
212 disp([pathname file_mot])
213 disp(' ')
214 disp('Syr-e file saved in:')
215 disp([pathname filename])
216 disp(' ')
217 end
```

Listing A.2. draw_motor_in_MCAD.m

```
1 function windingSyreToMCAD(mcad, pathname, filename, file_mot)
2 load([pathname filename])
3
4 geo.avvtot=geo.win.avv;
5 cyclew=1;
6 for k=2:1:(geo.p*2)
7     if cyclew==1;
8         geo.avvtot=[geo.avvtot (-geo.win.avv)];
9         cyclew=0;
10    else geo.avvtot=[geo.avvtot geo.win.avv];
11        cyclew=1;
12    end
13 end
14
15 i=1; m=1; e=1; n=1; j=1; o=1;
16 for k=1:1:(geo.Qs*2*geo.p)
17     value=geo.avvtot(1,k);
18
19     if value==1;
20         ph1go(i)=k; i=i+1;
21     end
22
23     if value==-1;
24         ph1ret(m)=k; m=m+1;
25     end
26
27     if value==2;
```

```
28     ph2go(n)=k; n=n+1;
29 end
30
31 if value==2;
32     ph2ret(o)=k; o=o+1;
33 end
34
35 if value==3;
36     ph3go(j)=k; j=j+1;
37 end
38
39 if value==4;
40     ph3ret(e)=k; e=e+1;
41 end
42
43 end
44
45 nbob=(e-1)*2;
46 geo.NbobInteger=round(gco.win.Nbob);
47
48 invoke(mcad,'LoadFromFile',[pathname file_mot]);
49
50 invoke(mcad,'SetVariable','MagWindingType', 1);
51 invoke(mcad,'SetVariable','MagPathType', 1);
52 invoke(mcad,'SetVariable','NumberOfCoils', nbob);
53 invoke(mcad,'SetVariable','Coil_Divider_Width',0);
54
55 a=1;
56 for i=0:2:(nbob-1)
57     invoke(mcad,'SetVariable',['Phase_1_Go1', num2str(i), ''], 0);
58     invoke(mcad,'SetVariable',['Phase_1_Go2', num2str(i), ''], ph1go(a));
59     invoke(mcad,'SetVariable',['Phase_1_Return1', num2str(i), ''], ph1ret(a));
60     invoke(mcad,'SetVariable',['Phase_1_Return2', num2str(i), ''], 0);
61     invoke(mcad,'SetVariable',['Phase_1_Turns', num2str(i), ''], geo.NbobInteger
62         );
63
64     i=i+1;
65
66     invoke(mcad,'SetVariable',['Phase_1_Go1', num2str(i), ''], ph1go(a));
67     invoke(mcad,'SetVariable',['Phase_1_Go2', num2str(i), ''], 0);
68     invoke(mcad,'SetVariable',['Phase_1_Return1', num2str(i), ''], 0);
69     invoke(mcad,'SetVariable',['Phase_1_Return2', num2str(i), ''], ph1ret(a));
70     invoke(mcad,'SetVariable',['Phase_1_Turns', num2str(i), ''], geo.NbobInteger
71         );
72
73     a=a+1;
74 end
```

```
74
75 a=1;
76 for i=0:2:(nbob-1)
77
78     invoke(mcad,'SetVariable',['Phase_2_Go1[' , num2str(i), ']' ], 0);
79     invoke(mcad,'SetVariable',['Phase_2_Go2[' , num2str(i), ']' ], ph2go(a));
80     invoke(mcad,'SetVariable',['Phase_2_Return1[' , num2str(i), ']' ], ph2ret(a));
81     invoke(mcad,'SetVariable',['Phase_2_Return2[' , num2str(i), ']' ], 0);
82     invoke(mcad,'SetVariable',['Phase_2_Turns[' , num2str(i), ']' ], geo.NbobInteger
83         );
84
85     i=i+1;
86
87     invoke(mcad,'SetVariable',['Phase_2_Go1[' , num2str(i), ']' ], ph2go(a));
88     invoke(mcad,'SetVariable',['Phase_2_Go2[' , num2str(i), ']' ], 0);
89     invoke(mcad,'SetVariable',['Phase_2_Return1[' , num2str(i), ']' ], 0);
90     invoke(mcad,'SetVariable',['Phase_2_Return2[' , num2str(i), ']' ], ph2ret(a));
91     invoke(mcad,'SetVariable',['Phase_2_Turns[' , num2str(i), ']' ], geo.NbobInteger
92         );
93
94     a=a+1;
95 end
96
97 a=1;
98 for i=0:2:(nbob-1)
99
100     invoke(mcad,'SetVariable',['Phase_3_Go1[' , num2str(i), ']' ], 0);
101     invoke(mcad,'SetVariable',['Phase_3_Go2[' , num2str(i), ']' ], ph3go(a));
102     invoke(mcad,'SetVariable',['Phase_3_Return1[' , num2str(i), ']' ], ph3ret(a));
103     invoke(mcad,'SetVariable',['Phase_3_Return2[' , num2str(i), ']' ], 0);
104     invoke(mcad,'SetVariable',['Phase_3_Turns[' , num2str(i), ']' ], geo.NbobInteger
105         );
106
107     i=i+1;
108
109     invoke(mcad,'SetVariable',['Phase_3_Go1[' , num2str(i), ']' ], ph3go(a));
110     invoke(mcad,'SetVariable',['Phase_3_Go2[' , num2str(i), ']' ], 0);
111     invoke(mcad,'SetVariable',['Phase_3_Return1[' , num2str(i), ']' ], 0);
112     invoke(mcad,'SetVariable',['Phase_3_Return2[' , num2str(i), ']' ], ph3ret(a));
113     invoke(mcad,'SetVariable',['Phase_3_Turns[' , num2str(i), ']' ], geo.NbobInteger
114         );
115
116     a=a+1;
117 end
118
119 end
```

Listing A.3. windingSyreToMCAD.m

```
2 function syreToDxfMCAD(pathname,filename)
3
4 % syreToDxf.m - exports a fem model created by syre to dxf
5 % input: motorname.mat (created by syre along with motorname.fem)
6 % output: motorname.dxf, into the folder motorname
7 load([pathname filename]);
8
9 stator = geo.stator(1:17,:);
10 stator(:,1:6)=0;
11 rotor = geo.rotor;
12
13 % export to dxf
14 pathname_DXF=pathname;
15
16 if not(isfolder(pathname_DXF))
17     mkdir(pathname_DXF);
18 end
19
20 raggi=[];
21 avvolgimento=[];
22 magneti=[];
23
24 DXFconv(raggi,avvolgimento,rotor,stator,magneti,[pathname_DXF filename(1:end-4)','.
    dxf']);
```

Listing A.4. syreToDxfMCAD.m

Appendice B

Script per l'impostazione delle simulazioni magnetiche singolo punto

```
1
2 function eval_operatingPointMCAD(dataIn)
3
4 % simulates single or multiple (id,iq) conditions
5 % Key INPUTs: CurrLoPP: current to be simulated
6 %             GammaPP: current phase angle
7 %             BrPP: remanence of all barriers magnets
8 %             NumOfRotPosPP: # simulated positions
9 %             AngularSpanPP: angular span of simulation
10 %=====
11
12 filemot = strrep(dataIn.currentfilename, '.mat', '.mot');
13 tmp = exist([dataIn.currentpathname filemot], 'file');
14
15 if tmp == 2
16
17     pathname=dataIn.currentpathname;
18     filemot = strrep(dataIn.currentfilename, '.mat', '.fem');
19     load([dataIn.currentpathname dataIn.currentfilename]);
20
21     CurrLoPP = dataIn.CurrLoPP;
22     GammaPP = dataIn.GammaPP;
23     BrPP = dataIn.BrPP;
24     NumOfRotPosPP = dataIn.NumOfRotPosPP;
```

```
25     AngularSpanPP = dataIn.AngularSpanPP;
26     NumGrid = dataIn.NumGrid;
27
28     clc;
29
30     eval_type = dataIn.EvalType;
31
32     per.overload=CurrLoPP;
33     per.BrPP=BrPP;
34     geo.nsim_singt = NumOfRotPosPP;           % simulated positions
35     geo.delta_sim_singt = AngularSpanPP;      % angular span of simulation
36
37     iAmp = dataIn.SimulatedCurrent;
38
39     % single point or array of points simulation
40     performance = cell(1,length(CurrLoPP));
41     output = cell(1,length(CurrLoPP));
42     geometry = cell(1,length(CurrLoPP));
43     tempDirName = cell(1,length(CurrLoPP));
44     for ii = 1:length(CurrLoPP)
45         performance{ii} = per;
46         performance{ii}.overload = CurrLoPP(ii);
47         performance{ii}.gamma=GammaPP(ii);
48     end
49     geo.RemoveTMPfile = 'OFF';
50
51     % check parallel computing
52     ppState=parallelComputingCheck();
53     if (ppState==0 && length(CurrLoPP)>4)
54         parpool();
55         ppState=parallelComputingCheck();
56     end
57
58     fileMotWithPath=[pathname filemot];
59     geo0=geo;
60     mat0=mat;
61
62     % evaluation
63     dataIn.MCADFEMM=0;
64
65     if dataIn.MCADFEMM==0
66         if ppState<1
67             for ii = 1:length(CurrLoPP)
68                 geoTmp = geo0;
69                 perTmp = performance{ii};
70                 matTmp = mat0;
71                 [~,geometry{ii},~,output{ii},tempDirName{ii}] = MCADfitness([],
                    geoTmp,perTmp,matTmp,eval_type,fileMotWithPath);
```



```

72         end
73     else
74         parfor ii = 1:length(CurrLoPP) %%%
75             geoTmp = geo0;
76             perTmp = performance{ii};
77             matTmp = mat0;
78             [~,geometry{ii},~,output{ii},tempDirName{ii}] = MCADfitness([],
79                 geoTmp,perTmp,matTmp,eval_type,fileMotWithPath);
80         end
81     end
82
83     % save output into individual folders
84     for ii = 1:length(CurrLoPP)
85
86         geo = geometry{ii};
87         out = output{ii};
88         per = performance{ii};
89         dirName = tempDirName{ii};
90
91         iStr=num2str(iAmp(ii),3); iStr = strrep(iStr, '.', 'A');
92         gammaStr=num2str(GammaPP(ii),4); gammaStr = strrep(gammaStr, '.', 'd');
93         if ~contains(gammaStr, 'd')
94             gammaStr = [gammaStr 'd'];
95         end
96
97         FILENAME = [filemot(1:end-4) '_T_eval_',iStr,'_',gammaStr,'_MCAD'];
98
99         mkdir(pathname,FILENAME);
100         newDir=[pathname,FILENAME,'\'];
101
102         if isoctave() %OCT
103             file_name1= strcat(newDir,FILENAME,'.mat');
104             save('-mat7-binary', file_name1,'geo','per','out');
105             dirIn=strcat(dirName, ['\' filemot]);
106             dirDest=strcat(newDir, FILENAME, '.fem');
107             movefile(dirIn, dirDest);
108             clear file_name1 dirIn dirDest
109         else
110             save([newDir,FILENAME,'.mat'],'geo','per','out');
111         end
112
113         % plot and save figs
114         klength = 1; kturns = 1; delta_sim_singt = geo.delta_sim_singt;
115
116         if dataIn.MCADFEMM==0
117             plot_singtMCAD(out,klength,kturns,delta_sim_singt,newDir,filemot);
118         end

```

```

119
120     end
121
122     % extra figs, if input current is array
123     if length(CurrLoPP)>1
124
125         id = zeros(1,length(CurrLoPP));
126         iq = zeros(1,length(CurrLoPP));
127         T = zeros(1,length(CurrLoPP));
128         dTpu = zeros(1,length(CurrLoPP));
129         dTpp = zeros(1,length(CurrLoPP));
130         fd = zeros(1,length(CurrLoPP));
131         fq = zeros(1,length(CurrLoPP));
132
133         for ii = 1:length(CurrLoPP)
134             id(ii) = output{ii}.id;
135             iq(ii) = output{ii}.iq;
136             T(ii) = output{ii}.T;
137             dTpu(ii) = output{ii}.dTpu;
138             dTpp(ii) = output{ii}.dTpp;
139             fd(ii) = output{ii}.fd;
140             fq(ii) = output{ii}.fq;
141         end
142         dirPower=[pathname,filemot(1:end-4),'_singT\'];
143         mkdir(dirPower);
144
145         x = 1:length(CurrLoPP);
146         figure();
147         if ~isoctave()
148             figSetting();
149         end
150         subplot(2,1,1)
151         plot(x,T,'-x',x,T+0.5*dTpp,'r',x,T-0.5*dTpp,'r'), grid on, ylabel('$T$ [Nm
152             ]')
153         subplot(2,1,2)
154         plot(x,dTpp,'-x'), grid on, ylabel('$\Delta T_{pp}$ [Nm]')
155         xlabel('simulation \#')
156         h=gcf();
157         if isoctave() %OCT
158             fig_name=strcat(dirPower, filemot(1:end-4), '_torque_sens');
159             hgsave(h,[fig_name]);
160         else
161             saveas(gcf,[dirPower,filemot(1:end-4),'_torque_sens.fig'])
162         end
163
164         figure()
165         if ~isoctave()
166             figSetting();

```

```

166     end
167     subplot(2,1,1)
168     plot(x,fd,'-x',x,fq,'-x'), grid on, ylabel(' [Vs] '), legend('$\lambda_d$', '
        $\lambda_q$'),
169     subplot(2,1,2)
170     plot(x,abs(sin(atan(iq./id)-atan(fq./fd))),'-x'), grid on, ylabel('$\cos \
        \varphi$')
171     xlabel('simulation \#'),
172     h=gcf();
173     if isoctave() %OCT
174         fig_name=strcat(dirPower, filemot(1:end-4), '_fdq_IPF_sens');
175         hgsave(h,[fig_name]);
176     else
177         saveas(gcf,[dirPower,filemot(1:end-4),'_fdq_IPF_sens.fig'])
178     end
179
180     figure()
181     if ~isoctave()
182         figSetting();
183     end
184     subplot(2,1,1)
185     plot(x,fd,'-x','DisplayName','$\lambda_d$');
186     plot(x,fq,'-x','DisplayName','$\lambda_q$');
187     ylabel(' [Vs] ')
188     legend('show');
189     subplot(2,1,2)
190     plot(x,id,'-x','DisplayName','$i_d$');
191     plot(x,iq,'-x','DisplayName','$i_q$');
192     xlabel('simulation \#')
193     ylabel(' [A] ')
194     legend('show');
195     h=gcf();
196     if isoctave() %OCT
197         fig_name=strcat(dirPower, filemot(1:end-4), '_fdq_idiq_sens');
198         hgsave(h,[fig_name]);
199     else
200         saveas(gcf,[dirPower,filemot(1:end-4),'_fdq_idiq_sens.fig'])
201     end
202     senseOut.id = id;
203     senseOut.iq = iq;
204     senseOut.fd = fd;
205     senseOut.fq = fq;
206     senseOut.T = T;
207     senseOut.dTpp = dTpp;
208     senseOut.PF = abs(sin(atan(iq./id)-atan(fq./fd)));
209     save([dirPower,filemot(1:end-4),'_senseResults.mat'],'senseOut');
210
211 end

```

```
212
213 else
214     error('Error: File .mot not found...')
215 end
216
217 end
```

Listing B.1. eval_operatingPointMCAD.m

```
1 function [cost,geo,mat,out,pathname]=MCADfitness (RQ,geo,per,mat,eval_type,
    filenameIn)
2
3 [pathname,filename,ext] = fileparts(filenameIn);
4 filename = [filename ext]; % fem file name
5 pathname=[pathname '\'];
6
7 %load Syr-e and MCAD model
8 mcad=actxserver('MotorCAD.AppAutomation');
9 file_mot=[filename(1:(end-4)) '.mot'];
10 invoke(mcad,'LoadFromFile',[pathname file_mot]);
11
12 %MCAD mesh
13 invoke(mcad,'SetVariable','AirgapMeshPoints_layers',1440);
14 invoke(mcad,'SetVariable','AirgapMeshPoints_mesh',1440);
15
16 [SOL]=simulate_xdegMCAD(geo,per,mat,eval_type,pathname,filename);
17
18 %save outputs
19 out.id = mean(SOL.id); %const
20 out.iq = mean(SOL.iq); %const
21 out.fd = mean(SOL.fd); %waveform
22 out.fq = mean(SOL.fq); %waveform
23 out.T = mean(SOL.T); %waveform
24 out.dT = std(SOL.T);
25 out.dTpu = std(SOL.T)/out.T;
26 out.dTpp = max(SOL.T)-min(SOL.T);
27 out.IPF = SOL.IPF;
28 out.SOL = SOL;
29
30 %check Torque sign
31 if sign(out.T)~=sign(out.fd*out.iq-out.fq*out.id)
32     out.T = -out.T;
33     out.SOL.T = -out.SOL.T;
34 end
35
36 %save losses
37 out.Pfes_h=SOL.Pfes_h;
38 out.Pfes_c=SOL.Pfes_c;
```

```
39 out.Pfer_h=SOL.Pfer_h;
40 out.Pfer_c=SOL.Pfer_c;
41 out.Pfe_total=out.Pfes_h+out.Pfes_c+out.Pfer_h+out.Pfer_c;
42
43 %unused output
44 cost=0;
45
46 %save MCAD model
47 invoke(mcad,'SaveToFile',[pathname file_mot]);
48
49 end
```

Listing B.2. MCADfitness.m

```
1 function [SOL]=simulate_xdegMCAD(geo,per,mat,eval_type,pathname,filename)
2
3 filename=[filename(1:end-4) '.mat'];
4 load([pathname filename])
5 mcad=actxserver('MotorCAD.AppAutomation');
6 invoke(mcad,'SetVariable','PhaseAdvance',dataSet.GammaPP); %phase advance
7 io=calc_io(geo,per);
8 invoke(mcad,'SetVariable','PeakCurrent',dataSet.CurrLoPP*io);%peak current
9 invoke(mcad,'SetVariable','DCBusVoltage',565);
10
11 if dataSet.EvalSpeed~=0
12     invoke(mcad,'SetVariable','Shaft_Speed_Ref',dataSet.EvalSpeed);
13 else invoke(mcad,'SetVariable','Shaft_Speed_[RPM]',1000);
14     disp('simulation runs with a default value of 1000 rpm - No input speed from
        Syr-e')
15 end
16 invoke(mcad,'SetVariable','ArmatureConductor_Temperature',per.tempcu);
17
18 %Simulation settings
19 invoke(mcad,'SetVariable','BackEMFCalculation','False');
20 invoke(mcad,'SetVariable','CoggingTorqueCalculation','False');
21 invoke(mcad,'SetVariable','TorqueSpeedCalculation','False');
22 invoke(mcad,'SetVariable','DemagnetizationCalc','False');
23 invoke(mcad,'SetVariable','TorqueCalculation','True');
24 nPoints=dataSet.NumOfRotPosPP*6; %over 360 eltDeg
25 invoke(mcad,'SetVariable','TorquePointsPerCycle',int2str(nPoints));
26 %multi-static magnetic solver
27 magnetic_solver=1;
28 invoke(mcad,'SetVariable','MagneticSolver',magnetic_solver);
29 invoke(mcad,'SetVariable','ArmatureEWdgMLT_Multiplier',0); %no end-windings
30 %single or multiple threads (0 or 1)
31 invoke(mcad,'SetVariable','MagThreads_Option',0);
32
33 % disp('Magnetic simulation in progress...')
```

```
34 success=invoke(mcad,'DoMagneticCalculation');
35 if success==0
36     disp('Magnetic calculation successfully completed')
37 else
38     disp('Magnetic calculation failed')
39 end
40
41 %save losses
42 if magnetic_solver==0
43     [tmp,Pfes_h_BackIron]=invoke(mcad,'GetVariable','StatorBackIronLoss_Hys');
44     [tmp,Pfes_h_Tooth]=invoke(mcad,'GetVariable','StatorToothLoss_Hys');
45     [tmp,Pfes_exc_backiron]=invoke(mcad,'GetVariable','StatorBackIronLoss_Excess');
46     [tmp,Pfes_exc_tooth]=invoke(mcad,'GetVariable','StatorToothLoss_Excess');
47     SOL.Pfes_h=Pfes_h_BackIron+Pfes_h_Tooth+Pfes_exc_backiron+Pfes_exc_tooth;
48
49     [tmp,Pfes_c_BackIron]=invoke(mcad,'GetVariable','StatorBackIronLoss_Eddy');
50     [tmp,Pfes_c_Tooth]=invoke(mcad,'GetVariable','StatorToothLoss_Eddy');
51     SOL.Pfes_c=Pfes_c_BackIron+Pfes_c_Tooth;
52
53     [tmp,Pfer_h_BackIron]=invoke(mcad,'GetVariable','RotorBackIronLoss_Hys');
54     [tmp,Pfer_h_Tooth]=invoke(mcad,'GetVariable','RotorMagnetPoleLoss_Hys');
55     [tmp,Pfer_exc_backiron]=invoke(mcad,'GetVariable','RotorBackIronLoss_Excess');
56     [tmp,Pfer_exc_tooth]=invoke(mcad,'GetVariable','RotorMagnetPoleLoss_Excess');
57     SOL.Pfer_h=Pfer_h_BackIron+Pfer_h_Tooth+Pfer_exc_tooth+Pfer_exc_backiron;
58
59     [tmp,Pfer_c_BackIron]=invoke(mcad,'GetVariable','RotorBackIronLoss_Eddy');
60     [tmp,Pfer_c_Tooth]=invoke(mcad,'GetVariable','RotorMagnetPoleLoss_Eddy');
61     SOL.Pfer_c=Pfer_c_BackIron+Pfer_c_Tooth;
62 end
63
64 if magnetic_solver==1
65     [tmp,Pfes_h_BackIron]=invoke(mcad,'GetVariable','StatorBackIronLoss_Hys_Static');
66     [tmp,Pfes_h_Tooth]=invoke(mcad,'GetVariable','StatorToothLoss_Hys_Static');
67     [tmp,Pfes_exc_backiron]=invoke(mcad,'GetVariable','StatorBackIronLoss_Exc_Static')
68     ;
69     [tmp,Pfes_exc_tooth]=invoke(mcad,'GetVariable','StatorToothLoss_Exc_Static');
70     SOL.Pfes_h=Pfes_h_BackIron+Pfes_h_Tooth+Pfes_exc_backiron+Pfes_exc_tooth;
71
72     [tmp,Pfes_c_BackIron]=invoke(mcad,'GetVariable','StatorBackIronLoss_Eddy_Static');
73     [tmp,Pfes_c_Tooth]=invoke(mcad,'GetVariable','StatorToothLoss_Eddy_Static');
74     SOL.Pfes_c=Pfes_c_BackIron+Pfes_c_Tooth;
75
76     SOL.Pfer_c=0;
77     SOL.Pfer_h=0;
78 end
79 %save current dq (dq axis Syr-e)
80 [tmp,SOL.id]=invoke(mcad,'GetVariable','CurrentLoad_Q');
81 [tmp,SOL.iq]=invoke(mcad,'GetVariable','CurrentLoad_D');
```

```
81 SOL.iq=-sqrt(2)*SOL.iq;      SOL.id=sqrt(2)*SOL.id;
82
83 %save IPF
84 [tmp,SOL.IPF]=invoke(mcad,'GetVariable','WaveformPowerFactor');
85
86 %save Torque
87 RotorPosition = linspace(0,360,nPoints);
88 SLOT.T=zeros(nPoints,1);
89 for loop=1:nPoints
90     [success,x,y]=invoke(mcad,'GetMagneticGraphPoint','TorqueVW',loop);
91     if success == 0
92         RotorPosition(loop)=x;
93         SOL.T(loop)=y;
94     end
95 end
96
97 %save flux dq
98 for loop=1:nPoints
99     [success,x,y]=invoke(mcad,'GetMagneticGraphPoint','FluxLinkageLoadTotalD',loop
100     );
101     if success == 0
102         RotorPosition(loop)=x;
103         SOL.fq(loop)=-y;
104     end
105 end
106 for loop1=1:nPoints
107     [success,x,y]=invoke(mcad,'GetMagneticGraphPoint','FluxLinkageLoadTotalQ',
108     loop1);
109     if success == 0
110         RotorPosition(loop1)=x;
111         SOL.fd(loop1)=y;
112     end
113 end
114 end
```

Listing B.3. simulate_xdegMCAD.m

```
1
2 function plot_singtMCAD(out,klength,kturns,delta_sim_singt,newDir,filemot)
3
4 % single working point has been simulated
5 t = out.SOL.T'* klength;
6 fd = out.SOL.fd'*klength*kturns;
7 fq = out.SOL.fq'*klength*kturns;
8 gamma = mean(atan2(-out.SOL.id',out.SOL.iq')) * 180/pi;
9 delta = atan2(fq,fd) * 180/pi;
10 IPF = cosd(delta-gamma);
11 th = linspace(0,360,length(t));
```

```
12
13 % plots
14 FontSize = 12;
15 FontName = 'TimesNewRoman';
16
17 figure()
18 subplot(2,1,1)
19 pl = plot(th,abs(t)); grid on
20 title(['Mean Torque = ' num2str(mean(t))]);
21 set(pl,'LineWidth',[2]);
22 xlim([0 360]), %ylim([ymin ymax]),
23 set(gca,'FontName',FontName,'FontSize',FontSize);
24 ti = 0:60:360; set(gca,'XTick',ti);
25 xl = xlabel('\theta - degrees'); set(xl,'Rotation',[0],'FontSize',FontSize);
26 yl = ylabel('Nm');
27 set(yl,'Rotation',[90],'FontName',FontName,'FontSize',FontSize);
28
29 %figure();
30 subplot(2,1,2)
31 pl = plot(th,IPF);
32 title(['Mean IPF = ' num2str(mean(IPF))]);
33 grid on
34 set(pl,'LineWidth',2);
35 xl = xlabel('\theta - degrees'); set(xl,'Rotation',0,'FontName',FontName,'FontSize',FontSize);
36 yl=ylabel('IPF');
37 set(yl,'Rotation',90,'FontName',FontName,'FontSize',FontSize);
38 xlim([0 360]);
39 set(gca,'FontName',FontName,'FontSize',FontSize);
40 ti = 0:60:360;
41 set(gca,'XTick',ti);
42
43 h=gcf(); %OCT
44 if isoctave()
45     fig_name=strcat(newDir, filemot(1:end-4), '_T_gamma');
46     hgsave(h,[fig_name]);
47 else
48     saveas(h,[newDir filemot(1:end-4) '_T_gamma']);
49 end
50
51 ymin = round(( min(min(fd),min(fq))*1000)/1000;
52 ymax = round(( max(max(fd),max(fq))*1000)/1000;
53
54 if ymax>ymin
55     ymax=1.2*ymax;
56     ymin=0.8*ymin;
57 else
58     ymax=1.2*ymin;
```



```
59     ymin=0.8*ymax;
60 end
61
62 hdq=figure();
63 subplot(2,1,1);
64 hd1=plot(th,fd);
65 title(['Mean \lambda_d = ' num2str(mean(fd))]);
66 grid on
67 set(hd1,'LineWidth',2);
68 xl_hd=xlabel('\theta [Electrical degrees]');
69 set(xl_hd,'Rotation',0,'FontName',FontName,'FontSize',FontSize); %,'FontWeight','
    Bold');
70 yl_hd=ylabel('\lambda_d [Wb]');
71 set(yl_hd,'Rotation',90,'FontName',FontName,'FontSize',FontSize); %,'FontWeight','
    Bold');
72 xlim([0 360]);
73 set(gca,'FontSize',FontSize); %,'FontWeight','Bold');
74 ti = 0:60:360;
75 set(gca,'XTick',ti);
76
77 hq = subplot(2,1,2);
78 hq1 = plot(th,fq);
79 title(['Mean \lambda_q = ' num2str(mean(fq))]);
80 grid on
81 set(hq1,'LineWidth',2);
82 xl_hq=xlabel('\theta [Electrical degrees]');
83 set(xl_hq,'Rotation',0,'FontName',FontName,'FontSize',FontSize); %,'FontWeight','
    Bold');
84 yl_hq=ylabel('\lambda_q [Wb]');
85 set(yl_hq,'Rotation',90,'FontName',FontName,'FontSize',FontSize); %,'FontWeight','
    Bold');
86 xlim([0 360]);
87 set(gca,'FontSize',FontSize); %,'FontWeight','Bold');
88 ti = 0:60:360;
89 set(gca,'XTick',ti);
90
91 if isoctave() %OCT
92     fig_name1=strcat(newDir, filemot(1:end-4), '_plot_Flux');
93     hgsave(hdq,[fig_name1]);
94 else
95     saveas(hdq,[newDir filemot(1:end-4) '_plot_Flux']);
96 end
```

Listing B.4. plot_singtMCAD.m

Appendice C

Script per l'importazione in Motor-CAD delle mappe di flusso calcolate in SyR-e

```
1
2
3 function dataSet = Export_FdFq_tables_in_MCAD(dataSet,filename,pathname)
4
5 filemot = strrep(dataSet.currentfilename, '.mat', '.mot');
6 tmp = exist([dataSet.currentpathname filemot], 'file');
7
8 if tmp == 2
9     if nargin() < 2
10         load LastPath
11         [filename, pathname, ~] = uigetfile([pathname '/*_n*.mat'], 'LOAD DATA');
12         load([pathname filename])
13         save LastPath pathname
14     end
15
16     F_map_MCAD=zeros(31,7);
17
18     %first row
19     F_map_MCAD1=["Is","Current Angle","Flux Linkage D","Flux Linkage Q","
        Hysteresis Iron Loss","Eddy Iron Loss","Magnet Loss"];
20     F_map_MCAD=[F_map_MCAD1;F_map_MCAD(2:31,:)];
21
22     %first column (current amplitude)
23     Imax=Id(1,end);
```

```
24     for k=2:6:31
25         for i=2:1:7
26             if i==2
27                 F_map_MCAD(k,1)=0;
28             else
29                 F_map_MCAD(k,1)=Imax/5*(i-2);
30             end
31             k=k+1;
32         end
33     end
34
35     %second column (phase advance)
36     p=1;
37     for k=2:6:31
38         for i=1:1:6
39             if k==2
40                 F_map_MCAD(k+i-1,2)=0;
41             else
42                 F_map_MCAD(k+i-1,2)=90/4*(p-1);
43             end
44         end
45         p=p+1;
46     end
47
48     %3rd and 4th columns (Fd and Fq)
49     Id=round(Id,4);
50     Iq=round(Iq,4);
51     for k=2:1:31
52         %dq components
53         id_mcad=str2num(F_map_MCAD(k,1))*cosd(str2num(F_map_MCAD(k,2)));
54         iq_mcad=str2num(F_map_MCAD(k,1))*sind(str2num(F_map_MCAD(k,2)));
55
56         %find position and interpolation index
57         [deltD c] = min(abs(Id(1,:)-id_mcad));
58         if Id(1,c) < id_mcad
59             indexD=1+deltD/id_mcad;
60         else
61             if Id(1,c)==0
62                 indexD=1;
63
64             else
65                 indexD=1-deltD/id_mcad;
66             end
67         end
68
69         [deltQ r] = min(abs(Iq(:,1)-iq_mcad));
70         if Iq(r,1) < iq_mcad
71             indexQ=1+deltQ/iq_mcad;
```

```
72         else
73             if Iq(r,1)==0
74                 indexQ=1;
75             else
76                 indexQ=1-deltQ/iq_mcad;
77             end
78         end
79
80         %save Fq and Fq
81         F_map_MCAD(k,3)=Fd(r,c)*indexD*indexQ;
82         F_map_MCAD(k,4)=Fq(r,c)*indexD*indexQ;
83     end
84
85     % create and write the .txt for Motor-CAD
86     tmp=[pathname filename(1:end-4) 'MCAD.txt'];
87     fid = fopen(tmp,'wt');
88     fprintf(fid, 'Is      Current Angle      Flux Linkage D  Flux Linkage Q  Hysteresis
89                Iron Loss      Eddy Iron Loss  Magnet Loss\n');
90     for ii = 2:size(F_map_MCAD,1)
91         fprintf(fid,'%g\t',F_map_MCAD(ii,:));
92         fprintf(fid,'\n');
93     end
94     fclose(fid)
95
96     % Load Flux Map to Motor-CAD
97     mcad=actxserver('MotorCAD.AppAutomation');
98
99     % Load file .mot
100     file_mot=[dataSet.currentfilename(1:(end-4)) '.mot'];
101     invoke(mcad,'LoadFromFile',[dataSet.currentpathname file_mot]);
102
103     % Show Lab context
104     invoke(mcad,'SetMotorLABContext');
105
106     % Motor-CAD commands to import the built flux maps
107     invoke(mcad,'SetVariable','ElectroLink_MotorLAB',"Custom (Advanced)");
108
109     % Load file .txt
110     tmp=[pathname filename(1:end-4) 'MCAD.txt'];
111     invoke(mcad,'SetVariable','BPM_FilePath_MotorLAB',tmp);
112
113     % Save file .mot
114     invoke(mcad,'SaveToFile',[dataSet.currentpathname file_mot]);
115
116     disp('Motor-CAD Flux Map file saved in:')
117     disp([dataSet.currentpathname file_mot])
118     disp(' ')
```

```
118     disp('To display the custom flux maps in Motor-CAD perform the following  
        actions:')  
119     disp('Defaults -> Motor-CAD Lab link -> Custom (Advanced)')  
120 else  
121     error('Error: File .mot not found...')  
122 end  
123 end
```

Listing C.1. Export_FdFq_tables_in_MCAD.m

Appendice D

Script per impostare e simulare il modello termico in Motor-CAD

```
1
2 function dataSet = ExportThermalParameters_MCAD(dataSet)
3
4 filemot = strrep(dataSet.currentfilename, '.mat', '.mot');
5 tmp = exist([dataSet.currentpathname filemot], 'file');
6
7 if tmp == 2
8     mcad=actxserver('MotorCAD.AppAutomation');
9
10    if nargin < 2
11        %load Syr-e and MCAD model
12        file_mot=[dataSet.currentfilename(1:(end-4)) '.mot'];
13        invoke(mcad, 'LoadFromFile', [dataSet.currentpathname file_mot]);
14    end
15
16    %Setting type Housing
17    switch dataSet.HousingType
18        case 'Axial fins (Servo)'
19            invoke(mcad, 'SetVariable', 'HousingType', 9 );
20        case 'Water Jacket (Axial)'
21            invoke(mcad, 'SetVariable', 'HousingType', 11);
22        case 'Water Jacket (Spiral)'
23            invoke(mcad, 'SetVariable', 'HousingType', 12);
24        case 'None'
25            invoke(mcad, 'SetVariable', 'HousingType', 13);
26    end
27
28    % Setting inlet temperature
```

```
29     switch    dataSet.HousingType
30         case 'Axial fins (Servo)'
31             tmp=dataSet.InletTemperature;
32             tmp=num2str(tmp);
33             tmp(tmp=='.')=',';
34             invoke(mcad,'SetVariable','Ambient_Temperature',tmp);
35         case 'Water Jacket (Axial)'
36             tmp=dataSet.InletTemperature;
37             tmp=num2str(tmp);
38             tmp(tmp=='.')=',';
39             invoke(mcad,'SetVariable','WJ_Fluid_Inlet_Temperature',tmp);
40         case 'Water Jacket (Spiral)'
41             tmp=dataSet.InletTemperature;
42             tmp=num2str(tmp);
43             tmp(tmp=='.')=',';
44             invoke(mcad,'SetVariable','WJ_Fluid_Inlet_Temperature',tmp);
45         case 'None'
46             tmp=dataSet.InletTemperature;
47             tmp=num2str(tmp);
48             tmp(tmp=='.')=',';
49             invoke(mcad,'SetVariable','Ambient_Temperature',tmp);
50     end
51
52     %Setting calculation type
53     invoke(mcad,'SetVariable','Transient_Calculation_Type', 0);
54
55     %Setting transient period
56     if dataSet.TransientPeriod ~= inf
57         tmp=dataSet.TransientPeriod;
58         tmp=num2str(tmp);
59         tmp(tmp=='.')=',';
60         invoke(mcad,'SetVariable','Transient_Time_Period',tmp);
61     %Setting number of Point
62     tmp=dataSet.TransientTimeStep;
63     tmp=num2str(tmp);
64     tmp(tmp=='.')=',';
65     invoke(mcad,'SetVariable','Number_Transient_Points',tmp);
66 end
67
68 %Show thermal context
69 invoke(mcad,'ShowThermalContext');
70
71 %save MCAD model
72 %invoke(mcad,'SaveToFile',[pathname filename]);
73 invoke(mcad,'SaveToFile',[dataSet.currentpathname file_mot]);
74
75 %Display save
76 disp('Motor-CAD Thermal Model file saved in:')
```

```
77     disp([dataSet.currentpathname file_mot])
78     disp(' ')
79
80     %Close Motor-CAD
81     invoke(mcad,'Quit');
82 else
83     error('Error: File .mot not found...')
84 end
85 end
```

Listing D.1. ExportThermalParameters_MCAD.m

```
1
2 function dataSet = ThermalSimulation_MCAD(dataSet)
3
4 filemot = strrep(dataSet.currentfilename, '.mat', '.mot');
5 tmp = exist([dataSet.currentpathname filemot], 'file');
6
7 if tmp == 2
8
9     mcad=actxserver('MotorCAD.AppAutomation');
10
11     if nargin < 2
12         %load Syr-e and MCAD model
13         file_mot=[dataSet.currentfilename(1:(end-4)) '.mot'];
14         invoke(mcad,'LoadFromFile',[dataSet.currentpathname file_mot]);
15     end
16
17     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
18
19     %Setting type Housing
20     switch dataSet.HousingType
21         case 'Axial fins (Servo)'
22             invoke(mcad,'SetVariable','HousingType', 9 );
23         case 'Water Jacket (Axial)'
24             invoke(mcad,'SetVariable','HousingType', 11);
25         case 'Water Jacket (Spiral)'
26             invoke(mcad,'SetVariable','HousingType', 12);
27         case 'None'
28             invoke(mcad,'SetVariable','HousingType', 13);
29     end
30
31     % Setting inlet temperature
32     switch dataSet.HousingType
33         case 'Axial fins (Servo)'
34             tmp=dataSet.InletTemperature;
35             tmp=num2str(tmp);
36             tmp(tmp=='.')=',';
```



```

37         invoke(mcad, 'SetVariable', 'Ambient_Temperature', tmp);
38     case 'Water Jacket (Axial)'
39         tmp=dataset.InletTemperature;
40         tmp=num2str(tmp);
41         tmp(tmp=='.')=',';
42         invoke(mcad, 'SetVariable', 'WJ_Fluid_Inlet_Temperature', tmp);
43     case 'Water Jacket (Spiral)'
44         tmp=dataset.InletTemperature;
45         tmp=num2str(tmp);
46         tmp(tmp=='.')=',';
47         invoke(mcad, 'SetVariable', 'WJ_Fluid_Inlet_Temperature', tmp);
48     case 'None'
49         tmp=dataset.InletTemperature;
50         tmp=num2str(tmp);
51         tmp(tmp=='.')=',';
52         invoke(mcad, 'SetVariable', 'Ambient_Temperature', tmp);
53     end
54
55     %Setting calculation type
56     invoke(mcad, 'SetVariable', 'Transient_Calculation_Type', 0);
57
58     %Setting transient period
59     if dataset.TransientPeriod ~= inf
60         tmp=dataset.TransientPeriod;
61         tmp=num2str(tmp);
62         tmp(tmp=='.')=',';
63         invoke(mcad, 'SetVariable', 'Transient_Time_Period', tmp);
64     %Setting number of Point
65         tmp=dataset.TransientTimeStep;
66         tmp=num2str(tmp);
67         tmp(tmp=='.')=',';
68         invoke(mcad, 'SetVariable', 'Number_Transient_Points', tmp);
69     end
70
71     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
72
73     % save output into individual folders
74     FILENAME = [filemot(1:end-4), '_ThermalSimulation_', num2str(dataset.
75         TransientPeriod) 's_', num2str(dataset.InletTemperature) 'C', '_MCAD'];
76     mkdir(dataset.currentpathname, FILENAME);
77     newDir=[dataset.currentpathname, FILENAME, '\'];
78
79     %Show thermal context
80     invoke(mcad, 'ShowThermalContext');
81
82     %Check type of calculation
83     if dataset.TransientPeriod==inf
84         invoke(mcad, 'SetVariable', 'ThermalCalcType', 0);

```

```
84     disp('Thermal Steady State Analysis in progress...')
85     success=invoke(mcad,'DoSteadyStateAnalysis');
86     if success==0
87         disp('Thermal calculation successfully completed')
88     else
89         disp('Thermal calculation failed')
90     end
91
92     % Plot steadystate temperature
93     [tmp, WindingTemperature_Min]=invoke(mcad,'GetVariable','T_[Winding_Min]')
94     ;
95     [tmp, WindingTemperature_Max]=invoke(mcad,'GetVariable','T_[Winding_Max]')
96     ;
97     [tmp, WindingTemperature_Average]=invoke(mcad,'GetVariable','T_[
98         Winding_Average]');
99     T=[WindingTemperature_Min,WindingTemperature_Max,
100        WindingTemperature_Average];
101     c=categorical({'Winding Min','Winding Max','Winding Average'});
102     figure()
103     figSetting()
104     bar(c,T)
105     title('Temperature in Steady-State Analysis')
106     ylabel('T [Celsius]')
107     savefig([newDir, 'SteadyState_temperature.fig']);
108     outTherm.WindingTempMeanSteadyState=WindingTemperature_Average;
109 else
110     invoke(mcad,'SetVariable','ThermalCalcType', 1);
111     disp('Thermal Transient Analysis in progress...')
112     success=invoke(mcad,'DoTransientAnalysis');
113     if success==0
114         disp('Thermal calculation successfully completed')
115     else
116         disp('Thermal calculation failed')
117     end
118
119     %Save Transient Solution .csv
120     tmp=[newDir 'TransientTemperatures.csv'];
121     invoke(mcad,'SaveTransientTemperatures',tmp);
122
123     %Set Number Points
124     tmp=dataSet.TransientTimeStep;
125     tmp=num2str(tmp);
126     tmp(tmp=='.')=',';
127     invoke(mcad,'SetVariable','Number_Transient_Points',tmp);
128
129     %Save winding average
130     nPoints = dataSet.TransientTimeStep;
131     WindingTemp_Average_Transient = zeros(nPoints,1);
```

```
128     Time = zeros(nPoints,1);
129     for timestep=1: 1 : nPoints
130         [success,x,y]=invoke(mcad,'GetTemperatureGraphPoint','Winding (Average
131             )',timestep);
132         if success ==0
133             Time(timestep)=x;
134             WindingTemp_Average_Transient(timestep)=y;
135         end
136     end
137
138     outTherm.timeTransient=Time;
139     outTherm.WindingTempTransient=WindingTemp_Average_Transient;
140
141     % Plot winding Average
142     figure
143     figSetting
144     figSetting(16,10)
145     plot(Time,WindingTemp_Average_Transient,'b*','DisplayName','Winding
146         Average')
147     xlabel('Time [s]')
148     ylabel('Temperature [Celsius]')
149     legend show
150     grid on
151     saveas(gcf,[newDir 'TransientTemperature'])
152     print(gcf,[newDir 'TransientTemperature.png'],'-dpng','-r300')
153 end
154
155 %save MCAD model
156 %invoke(mcad,'SaveToFile',[pathname filename]);
157 invoke(mcad,'SaveToFile',[dataSet.currentpathname file_mot]);
158
159 %Display save
160 disp('Motor-CAD Thermal Simulation file saved in:')
161 disp([dataSet.currentpathname file_mot])
162 disp(' ')
163
164 %Save variable
165 save([newDir,FILENAME,'.mat'],'outTherm','dataSet');
166
167 %Close Motor-CAD
168 invoke(mcad,'Quit');
169 else
170     error('Error: File .mot not found...')
171 end
172 end
```

Listing D.2. ThermalSimulation_MCAD.m

Bibliografia

- [1] Software FEMM. <http://www.femm.info/wiki/HomePage>.
- [2] Software MagNet. <https://www.mentor.com/products/mechanical/magnet/magnet/>.
- [3] Software SyR-e. <https://sourceforge.net/projects/syr-e/>.
- [4] Nicola Bianchi. *Electrical Machine Analysis Using Finite Elements*. Power Electronics and Applications Series. CRC Press.
- [5] Aldo Boglietti and Michele Pastorelli. Induction and synchronous reluctance motors comparison. In *2008 34th Annual Conference of IEEE Industrial Electronics*, pages 2041–2044. ISSN: 1553-572X.
- [6] R. Bojoi, E. Armando, M. Pastorelli, and K. Lang. Efficiency and loss mapping of AC motors using advanced testing tools. In *2016 XXII International Conference on Electrical Machines (ICEM)*, pages 1043–1049. ISSN: null.
- [7] Francesco Cupertino and Gianmario Pellegrino. User’s guide SyR-e. <https://sourceforge.net/p/syr-e/src/HEAD/tree/Readme/>.
- [8] Simone Ferrari and Gianmario Pellegrino. FEAfix: FEA refinement of design equations for synchronous reluctance machines. 56(1):256–266.
- [9] Simone Ferrari, Gianmario Pellegrino, and Elvio Bonisoli. Magnetic and structural co-design of synchronous reluctance electric machines in an open-source framework. 17(1):33–40.
- [10] Matteo Gamba, Gianmario Pellegrino, Eric Armando, and Simone Ferrari. Synchronous reluctance motor with concentrated windings for IE4 efficiency. In *2017 IEEE Energy Conversion Congress and Exposition (ECCE)*, pages 3905–3912. ISSN: null.
- [11] T.A. Lipo. *Analysis of Synchronous Machines*. CRC Press, second edition.

- [12] Amin Mahmoudi, Wen L. Soong, Gianmario Pellegrino, and Eric Armando. Efficiency maps of electrical machines. In *2015 IEEE Energy Conversion Congress and Exposition (ECCE)*, pages 2791–2799. ISSN: 2329-3748.
- [13] Ltd Motor Design. Motor-CAD help. <https://www.motor-design.com/>.
- [14] M. Palmieri, M. Perta, F. Cupertino, and G. Pellegrino. High-speed scalability of synchronous reluctance machines considering different lamination materials. In *IECON 2014 - 40th Annual Conference of the IEEE Industrial Electronics Society*, pages 614–620. ISSN: 1553-572X.
- [15] Marco Palmieri, Maurizio Perta, Francesco Cupertino, and Gianmario Pellegrino. Effect of the numbers of slots and barriers on the optimal design of synchronous reluctance machines. In *2014 International Conference on Optimization of Electrical and Electronic Equipment (OPTIM)*, pages 260–267. ISSN: 1842-0133.
- [16] Rainer Storn and Kenneth Price. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. 11(4):341–359.
- [17] F. Xue, A.C. Sanderson, and R.J. Graves. Pareto-based multi-objective differential evolution. In *The 2003 Congress on Evolutionary Computation, 2003. CEC '03.*, volume 2, pages 862–869 Vol.2. ISSN: null.