POLITECNICO DI TORINO

Corso di Laurea Magistrale in Ingegneria Elettrica

Tesi di Laurea

Progettazione e Simulazione di motori elettrici con software MagNet in ambiente open-source SyR-e

Relatori
Prof. Gianmario Pellegrino
Ing. Simone Ferrari

Candidato
Alessandro Loreto

Anno accademico 2018 - 2019

Alla mia famiglia,
che alimenta la passione che è in me.

Getting Better Day by Day

Migliorare Giorno per Giorno

Ringraziamenti

Voglio ringraziare innanzitutto il professore Pellegrino che, sin dal primo momento mi ha dato la possibilità di svolgere questa attività di tesi per la quale ho iniziato a nutrire sempre maggiore interesse: l'ambito di progettazione e simulazione sono stati sempre la mia passione.

Un ringraziamento particolare va anche al dottorando Ferrari Simone, colui che mi ha sempre sostenuto e consigliato durante la mia attività di tesi.

Un ringraziamento speciale va alla mia famiglia che mi ha sempre supportato in questi anni di studio presso il Politecnico, dandomi tutti i confort per affrontare al meglio la vita universitaria, non di certo facile e lineare.

Un pensiero va ai miei migliori amici, Alessandro, Tommaso e Andrea, che nonostante la lontananza abbiamo mantenuto il rapporto di fratellanza che ci ha sempre uniti, la mia ragazza Fiorella che con il suo sorriso e bontà mi illumina le giornate regalandomi felicità, e infine Pamina, amica e supporto ad ogni problema.

Un ultimo ringraziamento va ai miei nonni ai quali ho promesso la visione del raggiungimento di questo traguardo.

Indice

Elenco delle figure

Elenco delle tabelle

Sommario

L'obiettivo della mia tesi verte sul miglioramento delle funzionalità di *SyR-e* acronimo di *Synchronous Reluctance – evolution*, programma sviluppato dal dottorato di ricerca guidato dal Prof. Pellegrino. Al funzionamento della *GUI_Syre* è stata integrata la possibilità di creare, salvare e simulare un motore elettrico con *MagNet*, programma che permette di ottenere gli stessi risultati di *FEMM*, ma grazie alla 'motion', simulazione dinamica del rotore, è stato possibile calcolare le perdite della macchina con minore errore e in modo semplificato rispetto alla versione precedente. Prendendo spunto dagli script già esistenti, ho ricreato la logica di funzionamento per il salvataggio macchina e per il 'Post Processing', già usata con FEMM, che garantisce ottima affidabilità e velocità di calcolo. Infine sono stati confrontati e convalidati i risultati ottenuti.

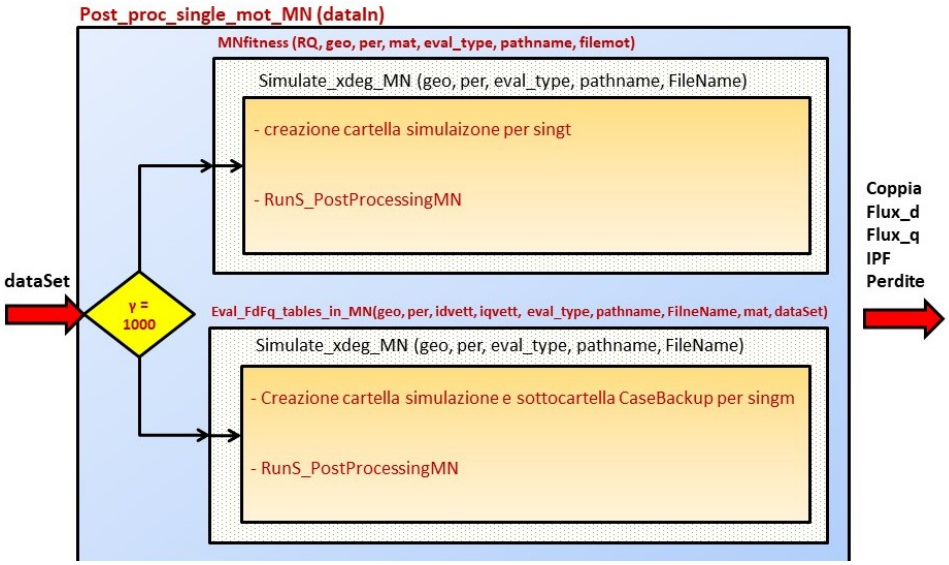


Figura 1 Flow Chart Post Processing SyR-e utilizzando MagNet

Capitolo 1

Introduzione

[7]

Durante la progettazione via software di una macchina elettrica, la fase di modellizzazione e simulazione ha un ruolo fondamentale. Sono essenziali le possibilità di effettuare in prima analisi delle valutazioni sulle soluzioni migliori in base alle specifiche progettuali per poter proseguire con le successive fasi di implementazione.
Motivo per cui sul mercato esistono svariati software che ci assistono durante la nostra progettazione e ci permettono di ottenere informazioni ben precise riguardo le grandezze elettromagnetiche, termiche e meccaniche che caratterizzano la macchina in relazione ai parametri usati, quali dimensioni, materiali, avvolgimenti, velocità di rotazione, capacità di raffreddamento, geometrie rotoriche e statoriche.

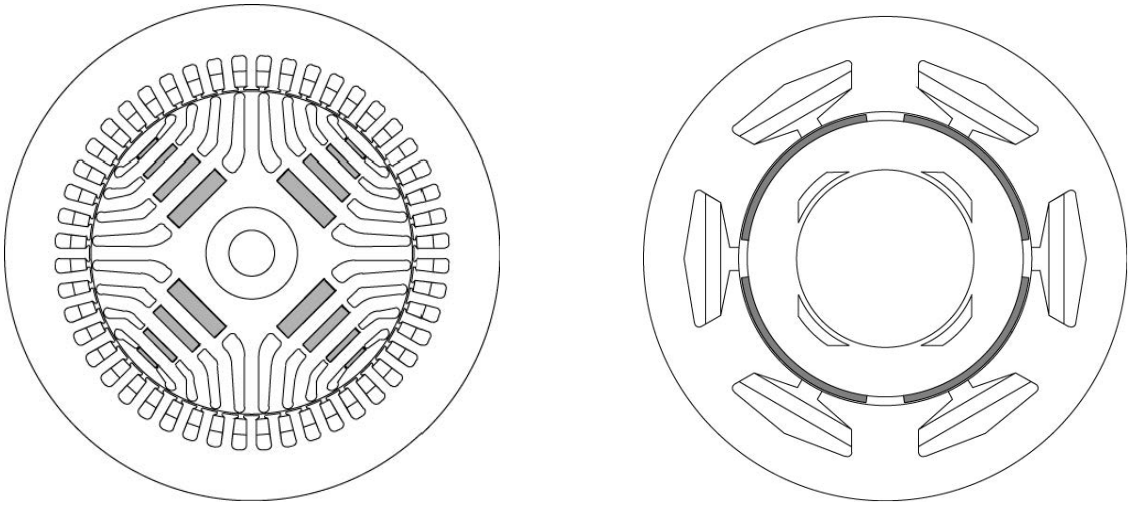
Il lavoro da me svolto è stato incentrato sull'ottimizzazione del codice di *SyR-e*, acronimo di *Synchronous Reluctance – evolution*, software open source sviluppato in collaborazione dal Politecnico di Torino con il Politecnico di Bari per la progettazione delle macchine sincrone attraverso l'utilizzo di analisi agli elementi finiti ed algoritmi di ottimizzazione multi-obiettivo. Precisamente, ci si è occupati della compatibilità del software *SyR-e*, sviluppato in ambiente *Matlab*, di avviare simulazion icon *MagNet* in aggiunta a *FEMM* [5] : quest'ultimo, oltre ai risultati attuali, permette di calcolare in maniera accurata le perdite statoriche e rotoriche della macchina che in *FEMM* non era possibile.

Ciò permette di ampliare le informazioni ottunute da *SyR-e* durante la simulazione.

Nel seguito di questo capitolo verranno descritti gli obiettivi e le caratteristiche principali del lavoro svolto, ponendo particolare attenzione sulle funzionalità del software SyR-e e sulle differenze fra simulazione con *FEMM* e *MagNet* .

1.1 Cos'è SyR-e

Prima di concentrarsi nella descrizione delle funzionalità e caratteristiche di SyR-e è necessario presentare le categorie di macchine elettriche di cui si occupa il software: i motori sincroni a magneti permanenti, tipicamente chiamati *brushless*, fanno parte della famiglia delle macchine in corrente alternata, presentano generalmente a statore un avvolgimento trifase e vengono classificate in base alla configurazione magnetica del rotore 1.1



(a) IPM (b) SPM

Figura 1.1 Macchine Brushless [1]

Di un'altra grande categoria fanno parte tutte quelle macchine il cui rotore presenta una forte anisotropia magnetica: quest'ultima infatti viene sagomata allo scopo di convogliare il flusso magnetico ed essere anche in grado di ospitare eventuali magneti. Di questa

famiglia fanno parte tutte le macchine brushless a magneti interni, a cui viene assegnata la sigla IPM, acronimo di *Interior Permanent Magnet*, ed i motori a riluttanza.

La struttura di riferimento di questi ultimi si caratterizza in particolare per il fatto che il rotore è costituito unicamente da lamierini di ferro, tranciati in modo tale da presentare sezioni d'aria (comunemente chiamati *barriere di flusso*) che indirizzano il flusso magnetico lungo direzioni ben precise; sfruttando questi vani d'aria si è arrivati ai motori a riluttanza assistiti (*Permanent Magnet Assisted Synchronous Reluctance*, nei quali all'interno delle barriere di flusso vengono collocati i magneti opportunamente dimensionati per migliorare le proprietà del flusso magnetico.

Il vantaggio più rilevante di queste macchine è l'assenza di contatti striscianti che risultano degradarsi facilmente e contribuiscono alla diminuzione della vita utile, inoltre rappresentano un elemento di pericolo per l'ambiente esterno, rendendo di fatto i motori a spazzole non adatti per applicazioni in ambiente speciali a rischi incendio. L'uso di magneti che richiedono particolari processi di lavorazione giustificano l'alto costo di queste macchine, a cui si aggiunge una maggior complessità dei sistemi di controllo real-time, i quali richiedono encoder e/o accelerometri dal costo elevato.

1.2 Interfaccia verso FEMM

Il lavoro di tesi svolto è stato focalizzato sull'ampliamento delle funzionalità di progettazione del programma open source *SyR-e*: tale strumento si affianca al software gratuito *FEMM* per l'analisi FEA *Finite Element Analysis* per poi elaborare i risultati all'interno dell'ambiente *Matlab*, in cui è stato scritto e sviluppato; in figura 1.2 è riportata la schematizzazione del flusso di dati scambiati fra i due applicativi.

La scelta di utilizzare Matlab è data dalla necessità di poter creare un'interfaccia grafica in maniera semplificata tramite l'uso del tool di sviluppo dedicato *GUIDE*. Nel tempo è stata implementata una *GUI (Graphic User Interface)* che dispone in maniera organizzata di tutte le funzioni in finestre differenti e semplifica di fatto l'interazione con

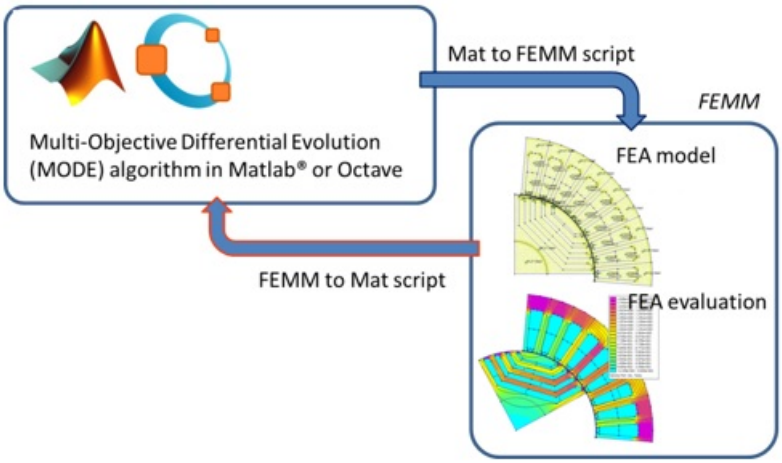


Figura 1.2 Flusso dati tra *SyR-e e FEMM [2]*

il software stesso, sia per l'impostazione dei parametri di input sia per l'acquisizione dei dati elaborati.

Lo script dell'ultima release *GUI_Syre.m* avvia l'interfaccia grafica che si presenta con la schermata principale come in figura 1.3, in cui viene caricata una macchina di default. Nella finestra principale denominata *Main Data* viene riportato il disegno della sezione (in questo caso i primi 60°) e i parametri principali; inoltre è anche possibile caricare un progetto già esistente utilizzando il tasto situato in alto a sinistra, presente in ogni schermata dell'interfaccia *GUI_Syre.m*

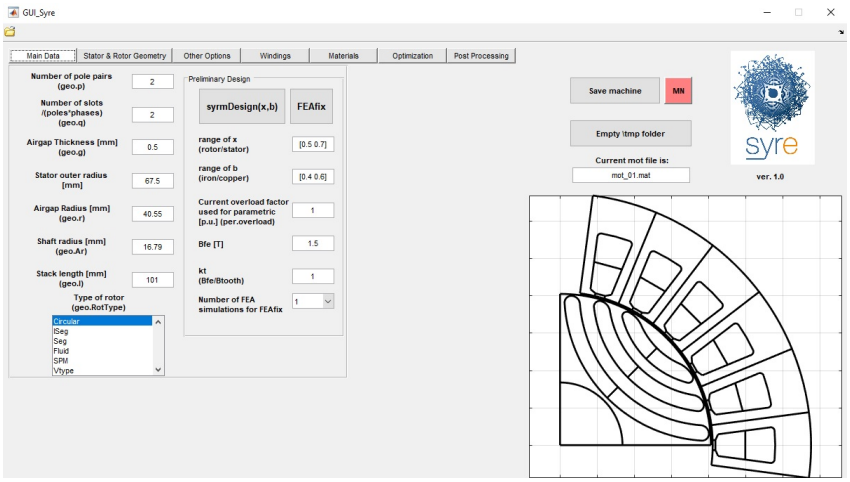
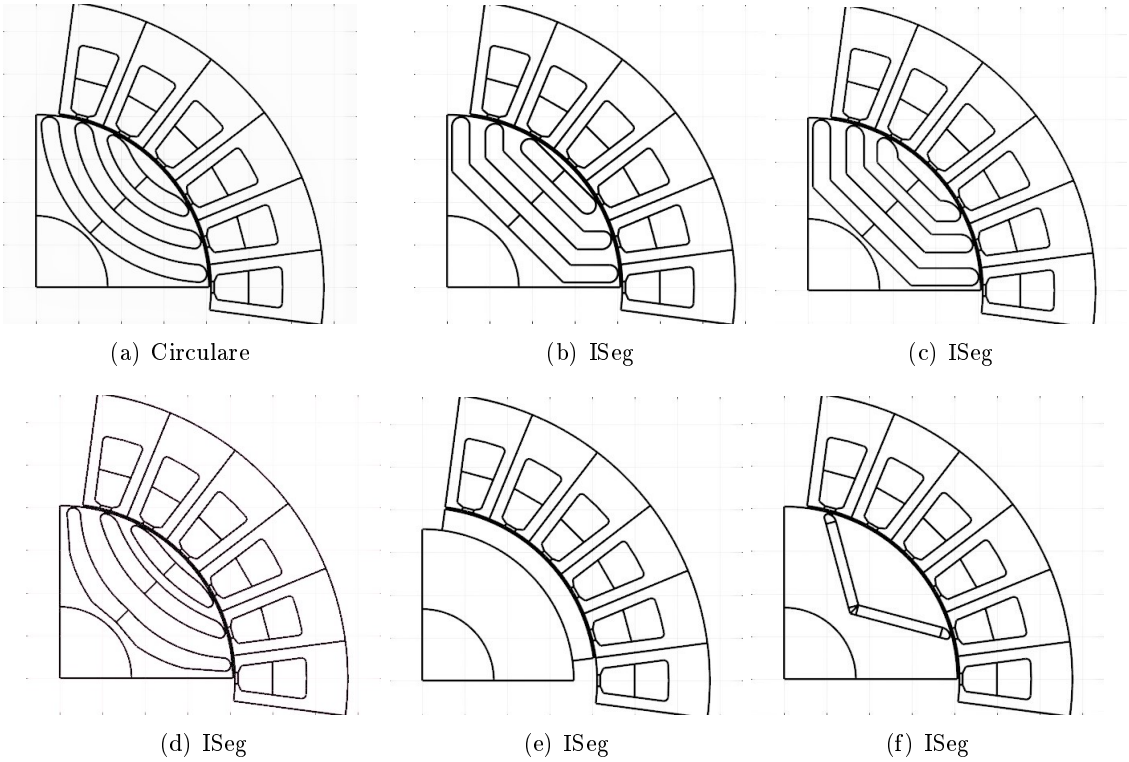


Figura 1.3 *GUI_SyR-e* finestra *Main Data* [3]

Esaminando la finestra principale, si può notare la possibilità di impostare la tipologia di rotore della macchina: questa rappresenta una delle caratteristiche più interessanti di questo programma. Nonostante sia nato come strumento di progettazione per macchine sincrone a riluttanza, al momento supporta differenti geometrie (*Circular, ISeg, Seg, Fluid, SPM, Vtype* mostrati in figura 1.4



(a) Circulare (b) ISeg (c) ISeg

(d) ISeg (e) ISeg (f) ISeg

Figura 1.4 Tipologie di rotore disponibili [3]

Un'altra funzionalità che l'applicativo offre è la possibilità di un'analisi parametrica di coppia e fattore di potenza sulla base dei dati inseriti pigiando il bottone *syrmDesign*. Variando il rapporto tra raggio di rotore e quello di statore (x) ed il rapporto fra i volumi occupati da rame e ferro dello statore (b) i risultati vengono riassunti in un grafico che mette in evidenzia le curve di isocoppia (rossa in figura 1.5) e le curve a fattore di potenza costante (blu), infine è possibile creare e caricare la macchina con le caratteristiche scelte.

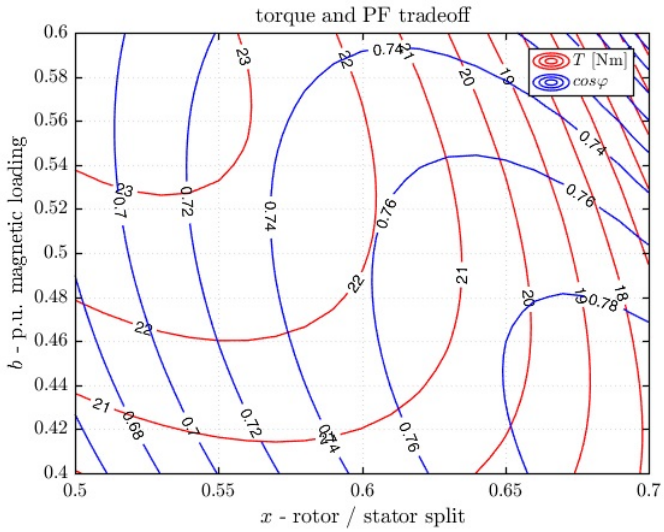


Figura 1.5 *syrmDesign: Grafico x/b e curve di livello* [3]

Proseguendo troviamo la seconda finestra *Stator & Rotor Geometry*, riportata in figura 1.6, ivi si ha accesso alla modifca dei parametri caratterizzanti rotore e statore: è possibile definire accuratamente la geometria delle cave di statore, modificando appositamente la casella della lunghezza *geo.lt*, la larghezza *geo.wt*, l'apertura cava *geo.acs* e altri parametri per modificare i denti di statore.

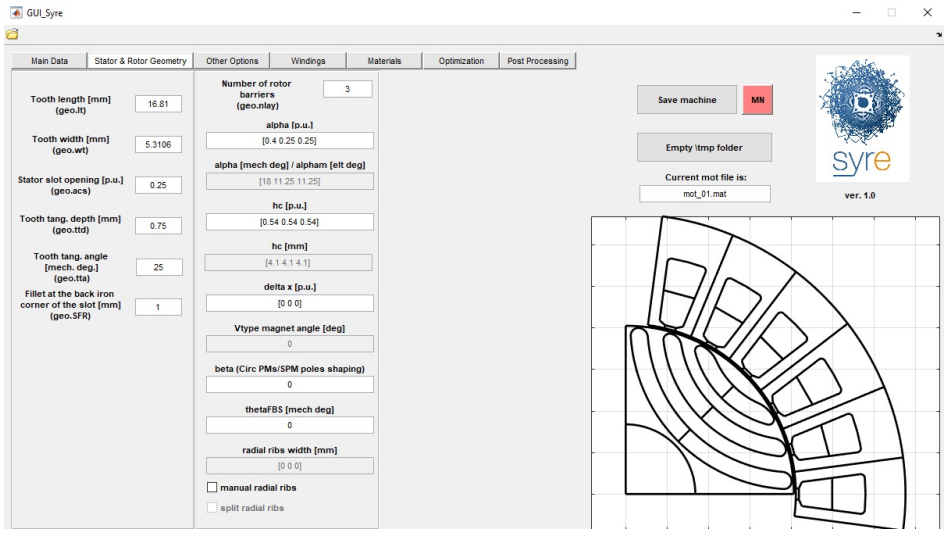


Figura 1.6 *GUI_SyR-e* finestra *Stator & Rotor Geometry* [3]

Mentre per il rotore, in base alla tipologia selezionata, si può scegliere di variare il numero di barriere di flusso e la loro geometria attraverso tre parametri:

alpha rappresenta l'apertura angolare delle estremità di barriera, la prima viene calcolata a partire dalla semiampiezza del polo (asse q in figura 1.7), mentre le altre vengono determinate come differenza dall'angolo della barriera precedente;

dx definisce lo spostamento delle barriere lungo l'asse posto sulla semiampiezza del polo rispetto alle posizioni standard;

hc determina l'ampiezza della barriera di flusso in senso radiale.

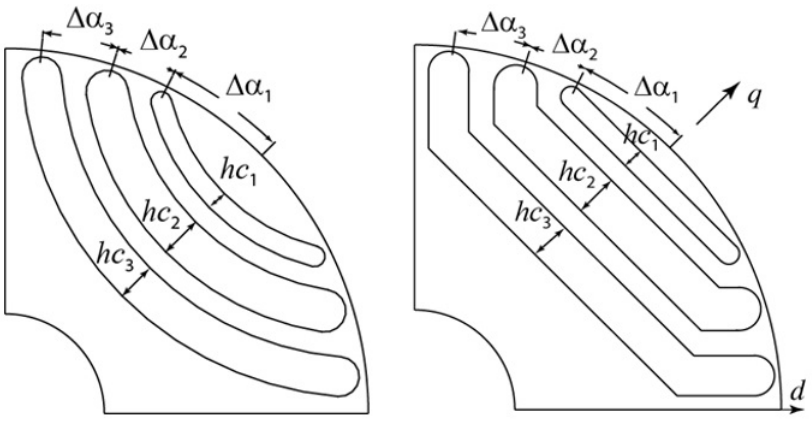


Figura 1.7 Parametri geometri delle barriere di flusso [3]

Inoltre nell'ultima release è stato inserito un parametro *deg* che ci permette di modificare l'angolo di aperture delle barriere per la geometria *Vtype*

La terza finestra, *Other Options* 1.8: il primo input (*per.Loss*) si riferisce alle perdite della macchina, che insieme alla temperatura di progetto dell'avvolgimento (*per.tempcu*) permette di calcolare la corrente nominale della macchina *geo.i0* in accordo alla formula: $per.loss = 3Ri_0^2$. Il calcolo della i0 richiede la stima della resistenza di fase in base alle caratteristiche geometriche dell'avvolgimento stesso. A partire dal modello termico della macchina viene stimata la temperatura di housing (*per.temphous*). La velocità massima attesa dalla macchina (*geo.namx*) è usata per il dimensionamento dei ponticelli radiali

delle barriere di flusso, i quali hanno funzione unicamente strutturale: questi sono calcolati considerando la forza centrifuga costante, alla quale è sottoposta la macchina alla massima velocità; si ignora la forza magnetica e l'effetto strutturale delle nervature tangenziali in corrispondenza dell'traferro, in relazione alle tolleranze meccaniche di realizzabilità del lamierino (parametro *geo.pont0*).

Sono inoltre modificabili i parametri relativi alla mesh utilizzata in FEMM per l'analisi agli elementi finiti: è infatti possibile impostare mesh a densità più o meno elevata in maniera indipendente per le operazioni di ottimizzazione (parametro *geo.K_mesh_MOOA*) e di post processing (*geo.K_mesh*). Nella parte inferiore della finestra compare invece la sezione relativa al dimensionamento dei magneti all'interno delle barriere di flusso (fino ad un massimo di 4): è possibile infatti selezionare la tipologia di magnete da utilizzare e, in base ai dati immessi, il software restituirà le dimensioni della sezione degli stessi.

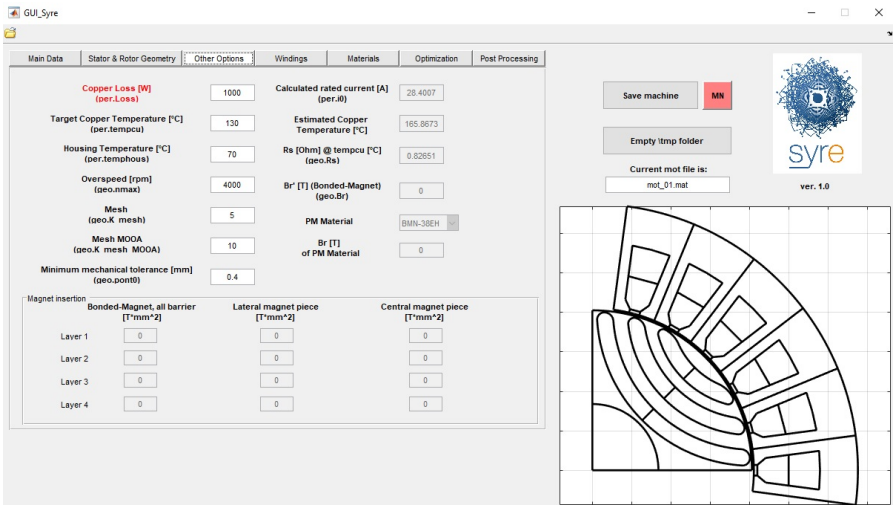


Figura 1.8 *GUI_SyR-e* finestra *Other Options* [3]

Nella finestra *Winding* (in figura 1.9 sono riportate le caratteristiche principali relative all'avvolgimento: in primis il fattore di riempimento delle cave (*geo.Kcu*), il numero di spire in serie per fase (*geo.Ns*) e il fattore di raccorciamento (*geo.Kracc*). Selezionando inoltre il numero di cave da simulare *geo.Qs*, nella parte inferiore della finestra, viene visualizzato lo schema di avvolgimento della macchina all'interno di una tabella di dimensioni

$2 \times geo.Kracc$: la riga superiore corrisponde allo strato esterno del layer, mentre la prima colonna corrisponde alla prima cava, il cui asse di simmetria coincide con l'asse orizzontale.

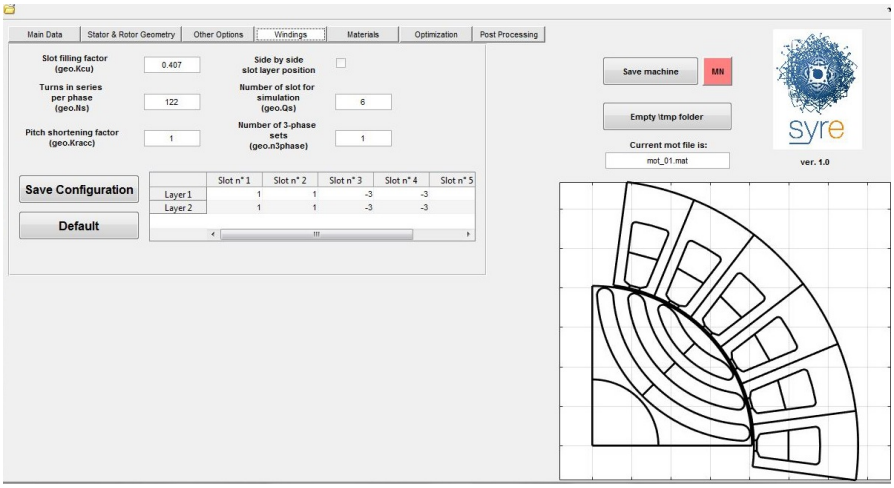


Figura 1.9 *GUI_SyR-e* finestra *Windings* [3]

I valori dello schema d'avvolgimento vengono calcolati da una versione modificata del software open source *Koil*, che si occupa di verificare la fattibilità dell'avvolgimento e genera l'output corretto ogniqualvolta vengano modificati i dati precedenti; un esempio di matrice ottenuta impostando $geo.Qs = 2$ e $geo.Kracc = 5/6$ è riportato nella tabella 1.1.

$$\begin{bmatrix} 1 & 1 & -3 & -3 & 2 & 2 \\ -2 & 1 & 1 & -3 & -3 & 2 \end{bmatrix}$$

Tabella 1.1 Esempio matrice di avvolgimento

Esiste anche la possibilità di progettare avvolgimenti a bobina di dente, che presentano numero di cave/polo/fase non intero: ciò è possibile selezionando l'opzione *Side by side slot layer position*.

Continuando abbiamo la finestra *Materials* (figura 1.10) nella quale è possibile impostare i materiali per cave, statore, rotore, le proprietà delle barriere di flusso e il materiale dell'albero.

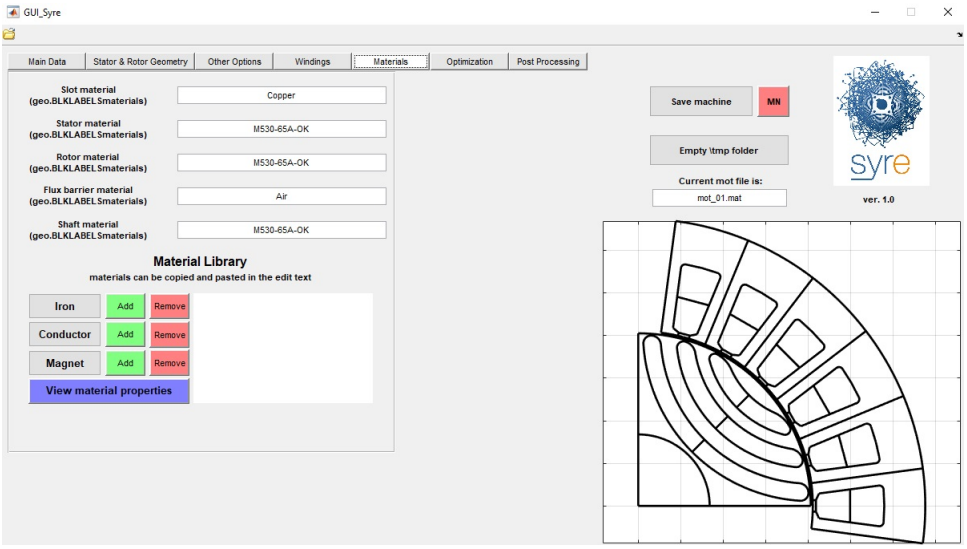


Figura 1.10 *GUI_SyR-e* finestra *Materials* [3]

Nella parte bassa della finestra è possibile scegliere nello specifico la famiglia di materiali di interesse interagendo in *Material Library*: qui troviamo la lista completa dei materiali per ogni tipo di componente (Iron, Conductor, Magnet) e attraverso gli appositi bottoni *Add e Remove* è possibile aggiungere un nuovo materiale inserendo le relative caratteristiche. Inoltre un ultimo bottone *View material proprieties* permette di visualizzare le proprietà relative al materiale inserito.
È importante ricordare che ogni qualvolta un parametro è modificato l'anteprima della macchina appare a destra della finestra di SyR-e; per mantenere le modifiche è necessario salvare la nuova macchina attraverso il bottone *Save Machine* che consente di salvare la configurazione della macchina corrente. Precisamente se si vuole creare e salvare una macchina con FEMM utilizzo il primo bottone, diversamente se si vule creare e salvare la macchina attraverso MagNet utilizzo il secondo bottone appositamente creato (più piccolo e in rosso).

Le ultime due schermate della *GUI_Syre* risultano tra le funzionalità più importanti del software: dalla finestra 1.11 è possibile avviare il processo di ottimizzazione con i parametri geometrici impostati nelle finestre precedente.

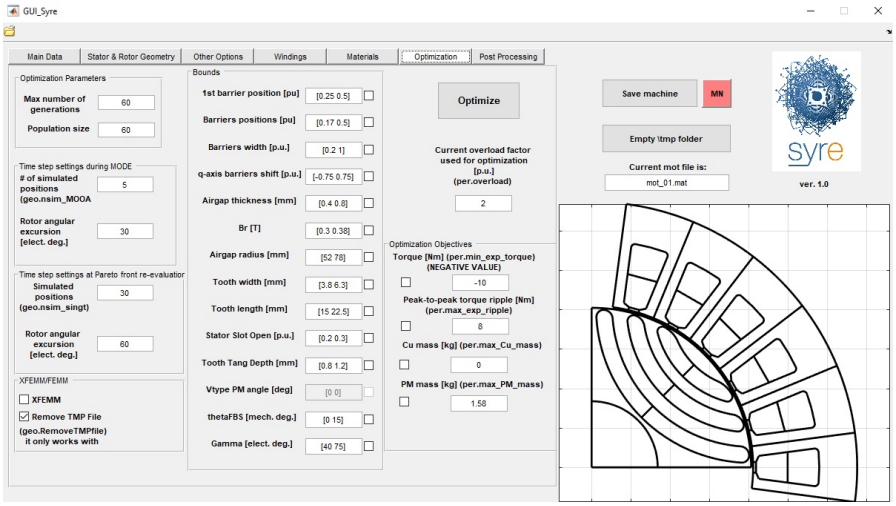


Figura 1.11 *GUI_SyR-e* finestra *Optimization* [3]

Durante l'ottimizzazione, alcuni parametri di rotore e statore sono automaticamente modificati in modo da trovare la miglior soluzione dello stesso obiettivo. Il software è stato implementato con un processo di ottimizzazione basato su due stadi: prima viene eseguito un algoritmo di ottimizzazione multi-obiettivo basato sul concetto di *evoluzione differenziale*, successivamente vengono nuovamente riesaminate le soluzioni migliori con maggior precisione. [6]

Nella schermata è possibile modificare tre tipologie di parametri:

- Parametri di Ottimizzazione: nella parte sinistra della schermata è possibile impostare i parametri di ottimizzazione, quali il numero di generazioni del processo evolutivo e il numero delle popolazioni per ogni singola generazione; inoltre deve essere impostato l'angolo di rotazione massimo e il numero di posizioni rotoriche da simulare in maniera indipendente per i due stadi del processo di ottimizzazione. La corrente da utilizzare deve essere modificata nella parte destra della schermata in *per unit* rispetto a quella nominale.

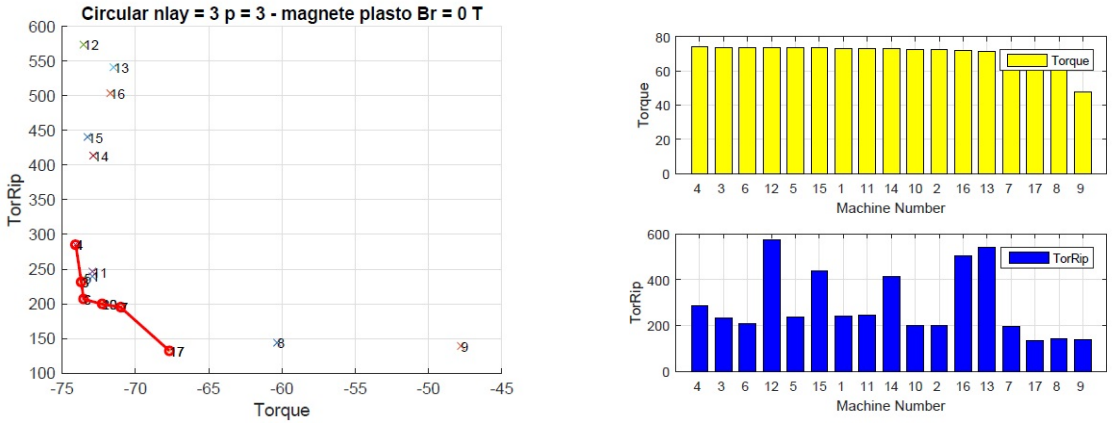
- Variabili Geometriche: nella parte centrale della schermata sono riportati i parametri geometrici che l'algoritmo di ottimizzazione cambia durante il processo evolutivo;

in base al tipo di macchina scelta si può accedere selezionando il checkbox e modificando il range di valori nel box corrispondente. Possono essere selezionati svariati parametri geometrici, quali l'angolo delle barriere di flusso, larghezza barriere di flusso, il raggio e le dimensioni del traferro, i parametri delle cave, l'angolo dei magneti nei motori Vtype (qualora si è selezionata la suddetta macchina) e infine il γ, angolo tra il vettore corrente e l'asse di minima riluttanza (comunemente chiamato *asse d*).

- Quattro Obiettivi di Ottimizzazione: nella parte a destra in basso sono selezionabili i box interessati; è possibile impostare il valore minimo di coppia accettato, il massimo valore del ripple picco-picco accettato, la massa di rame e magneti.

Alla fine dell'ottimizzazione i risultati appaiono a video, un file *.mat* e una cartella sono creati in *\result*, ivi vengono salvati automaticamente le figure di riepilogo dei risultati, che ripotano il Fronte di Pareto trovato, gli obiettivi ed i valori dei paramenti di ottimizzazione associati alle singole macchine. Una coppia di file .fem e .mat sono salvati nella cartella creata per essere poi ricaricata nella GUI oppure aprire il progetto esternamente in FEMM.

Nelle seguenti immagini in figura 1.12i si riportano i risultati di un'ottimizzazione eseguita impostando il numero di generazioni, una dimensione di popolazioni pari a 15 e come parametri di ottimizzazione la posizione di estremità di barriera e lo spessore delle stesse.



(a) Fronte di Pareto

(b) Riepilogo obiettivi

Figura 1.12 Esempio risultati del processo di ottimizzazione [3]

Nell'ultima finestra di SyR-e, relativa al *Post Processing*(in figura 1.13) ci permette di analizzare le grandezze e le performance della singola macchina caricata, opportunamente modificata, in una o più condizioni di funzionamento. Nella parte sinistra si impostano i parametri principali della simulazione:

- *Escursione dell'angolo di Rotore*, questo è la porzione di angolo (in gradi elettrici) di simulazione della macchina (di solito 60 o 360).
- *Angolo di fase della corrente* in coordinate dq
- *Numero di posizioni rotore* cioè il numero delle posizioni di rotore spaziate ugualmente che sarà simulata rispetto all'escursione angolare precedentemente definita.
- *Corrente di carico* cioè il valore della corrente i_0 in *per unit* definita sulla base delle perdite joule ammissibili.
- *Induzione magnetica "Br"* che può essere inclusa nelle barriere di rotore tramite gli appositi magneti
- *Numero dei punti in [0 Imax]* cioè il numero dei punti della suddivisione della griglia nella creazione della mappa dei flussi.

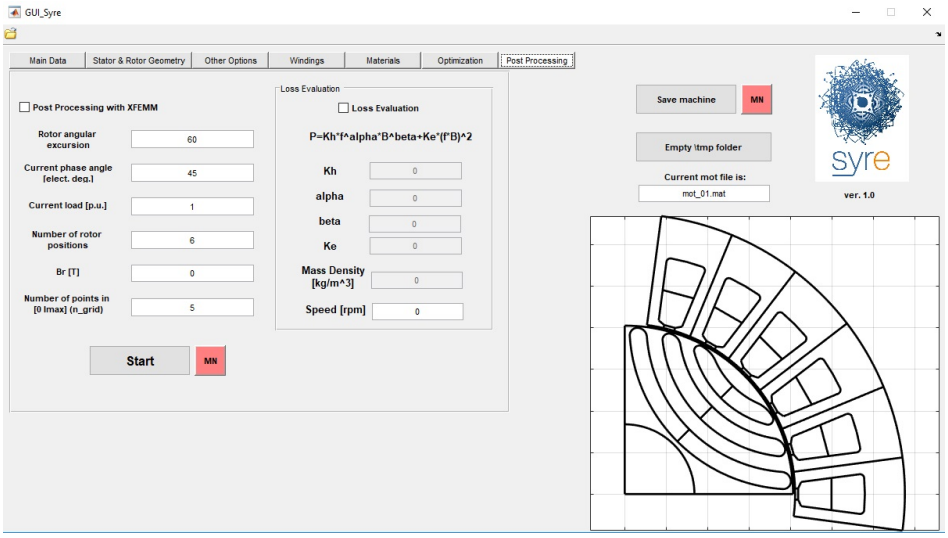


Figura 1.13 *GUI_SyR-e* finestra *Post Processing* [3]

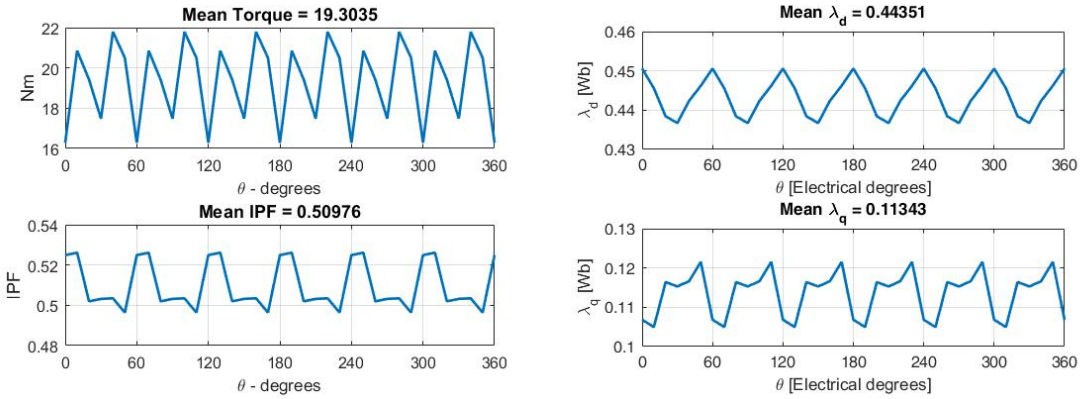
Inoltre, nella parte destra della schermata è presente una parte in cui è possibile abilitare e modificare i parametri utili per la valutazione del calcolo delle perdite nel ferro sulla base della seguente formula $P_{loss} = k_h f^{\alpha} B^{\beta} + k_e (fB)^2$.

Invece simulando tramite *MagNet* otteniamo la valutazione delle perdite direttamente impostando l'escursione dell'angolo di Rotore pari a 360°, condizione per cui viene simulata la macchina per intero e ci permette di ottenere risultati più accurati rispetto a FEMM.

Inoltre esiste la possibilità di accedere a tre differenti tipologie di simulazioni in base all'impostazione dei valori corrente/fase:

- inserendo un'unica coppia di valori di corrente/fase viene simulato un singolo punto di lavoro, ottenendo gli andamenti in figura 1.14, in funzione della posizione angolare, coppia fattore di potenza (IPF) e flussi in asse dq.

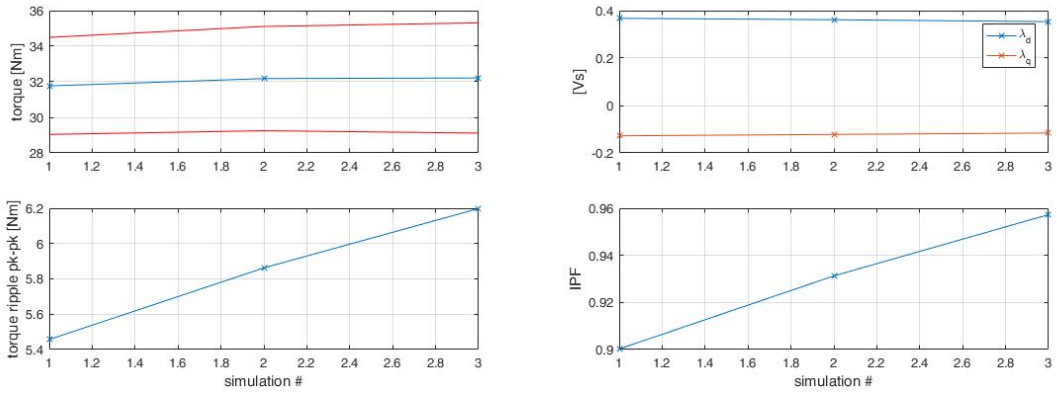
- inserendo più coppie di valori di corrente/fase vengono simulati più punti di lavori (vedi figura 1.15). La differenza sostanziale sta nel fatto che la corrente e la fase sono dei vettori, di ugual dimensioni e ciascun elemento definisce un singolo punto di calcolo: i risultati sono gli stessi per i singoli punti. A fine simulazione coppia, ripple di coppia e flusso in funzione delle simulazioni verranno stampati.



(a) Coppia e fattore di potenza

(b) Flussi dq

Figura 1.14 Risultati finali del Post Processing per un singolo punto di lavoro [3]

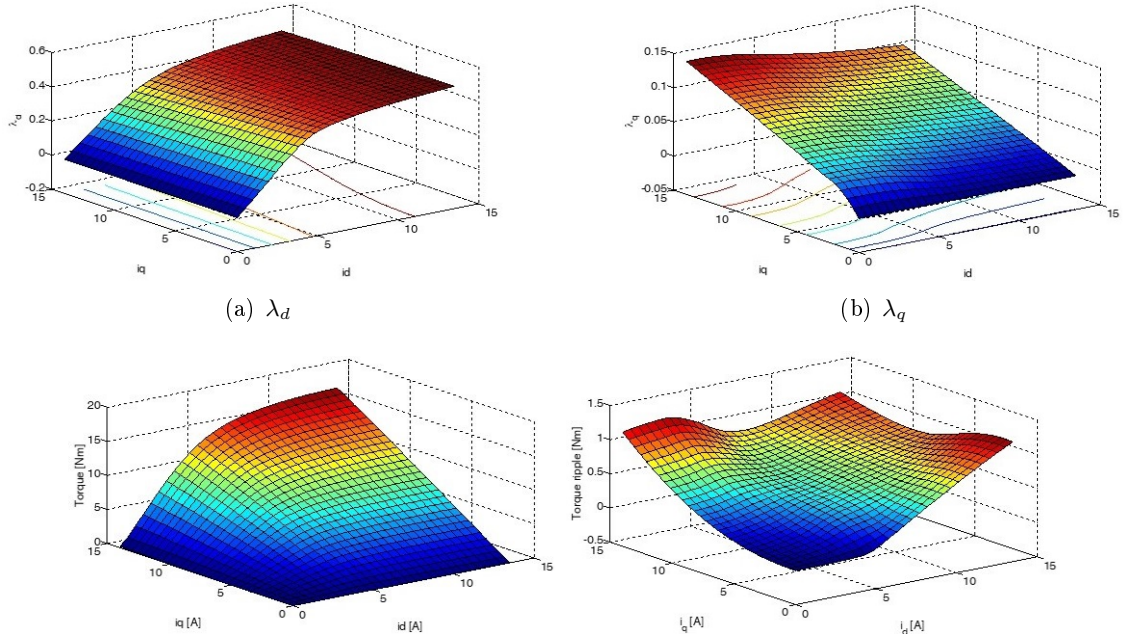


(a) Coppia e Coppia pk-pk

(b) Flussi dq e IPF

Figura 1.15 Risultati finali del Post Processing per un vettore d icorrente e fase [3]

- inserendo invece il valore convenzionale *1000* alla fase della corrente si accede al calcolo della mappa di flusso e coppia (vedi figura 1.16). È usata una griglia quadrata in cui il massimo valore (di i_d e i_q) è la corrente di sovraccarico e il numero di punti in ciascun lato è definito dal parametro precedentemente elencato ("numero di punti in [0 Imax]")



(a) λ_d (b) λ_q

(c) Coppia (d) Ripple di Coppia

Figura 1.16 Risultati finali per mappa dei flussi [3]

1.3 Obiettivi della Tesi

Dopo una spiegazione sugli aspetti fondamentali del software *SyR-e* l'obiettivo della tesi è stato quello di aggiungere alle funzionalità del programma la possibilità di creare una nuova macchina e poterla simularla tramite *MagNet* verificando di ottenere gli stessi risultati che avremmo ottenuto simulando con FEMM: ossia creare un percorso in parallelo che segua la stessa logica di simulazione già esistente in FEMM integrandola nella *GUI SyR-e* attraverso un apposito bottone per il salvataggio e per la simulazione.

Il vero e proprio motivo dell'esigenza di poter simulare attraverso *MagNet* è la facilità nel poter calcolare le perdite *Rotore* - *Statore* (per isteresi e per correnti parassite) e le perdite dei *Magneti*, qualora presenti. In realtà il calcolo delle perdite è già presente attraverso la simulazione con *FEMM*, utilizzando la formula $P_{loss} = k_h f^{\alpha} B^{\beta} + k_e (fB)^2$ che non garantisce un risultato preciso e richiede la conoscenza di tutti i parametri.

Capitolo 2

Presentazione del Problema

La prima versione del software che mi è stata consegnata per la simulazione con *MagNet* non comprendeva l'integrazione con *Syr-e*, ma prevedeva una versione più grossolana di script esterni che bisognava eseguire manualmente, seguendo un preciso ordine, per completare la simulazione. Ciò risultava parecchio scomodo e difficile da gestire considerando l'enorme quantità di file presenti.

Da qui si è proceduti analizzando le differenze e le compatibilità tra *FEMM e MagNet*, in modo da poter integrare, senza troppi cambiamenti, la simulazione con la *GUI_Syre*

2.1 Situazione Iniziale

Il software di partenza prevedeva alcune funzioni di export della macchina appositamente create al di fuori della GUI *SyR-e*: all'interno della cartella *syreExport* è possibile accedere allo script che permette di creare il file *.dxf*, utile per esportare il disegno della macchina in altri software *CAD o FEA* come *MagNet, SolidWorks, AutoCAD ecc.*

Nel nostro caso è di interesse la creazione di un modello .dxf del progetto della macchina in modo da poterlo poi importare in *MagNet* e poter costruire il file .mn che servirà per la simulazione.

In figura 2.1 viene riportato lo schema di funzionamento della simulazione con *MagNet* della prima versione a me consegnata:
si noti subito che una volta creato il file .mat della macchina tramite la GUI SyR-e bisogna richiamare tre script (*syreTodxf.m , syreToMagnet.m e RUN_SIM.m*), che lavorano indipendentemente tra loro, per avviare la simulazione; questo risulta parecchio scomodo e laborioso ad utenti che lo eseguono per la prima volta.

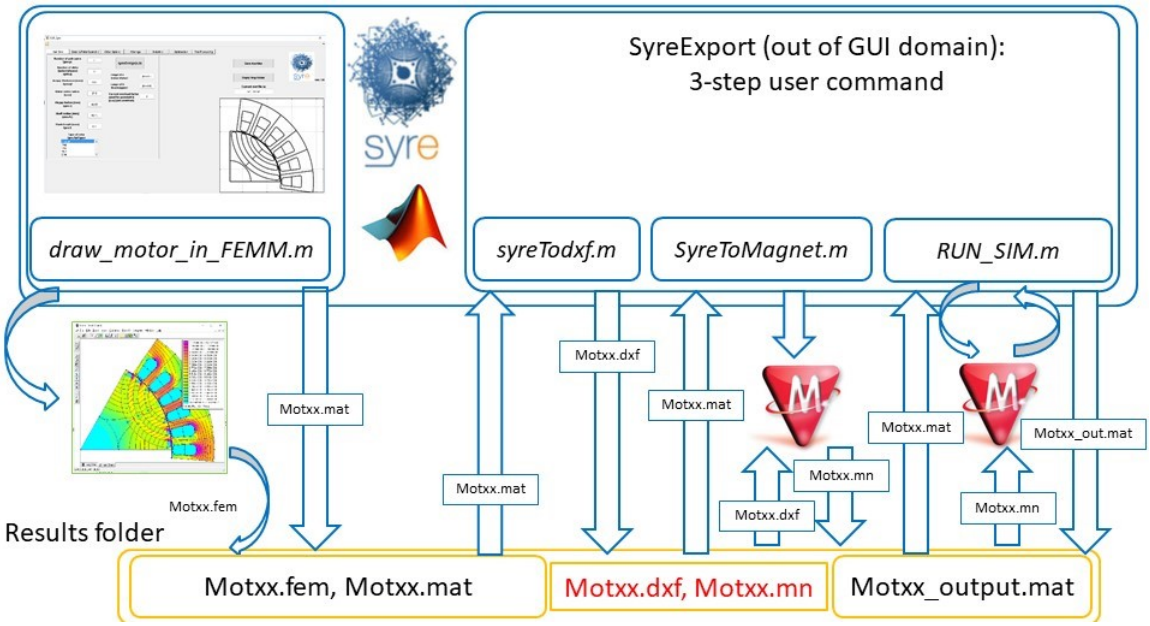


Figura 2.1 Funzionamento prima versione di MagNet [3]

Difatti aprendo da *Matlab* la cartella *syreExport* notiamo parecchi file e cartelle che serviranno per l'esecuzione degli script sopra citati; si noti subito in figura 2.2 l'enorme quantità di file, spesso anche ripetuti tra loro che, se non avviati nella giusta sequenza o non lanciando la giusta Path, portavano ad errori e all'interruzione della simulazione. Motivo per cui si è deciso di integrare l'export in *MagNet* nella GUI di SyR-e e rendere tutto il processo automatizzato. A causa dell'enorme quantità di file presenti nella cartella si è dovuto dedicare buona parte del tempo di lavoro: si è proceduti con la piena comprensione del codice e del suo funzionamento, si sono analizzati i punti critici del programma e le differenze sostanziali rispetto allo schema logico seguito in SyR-e;

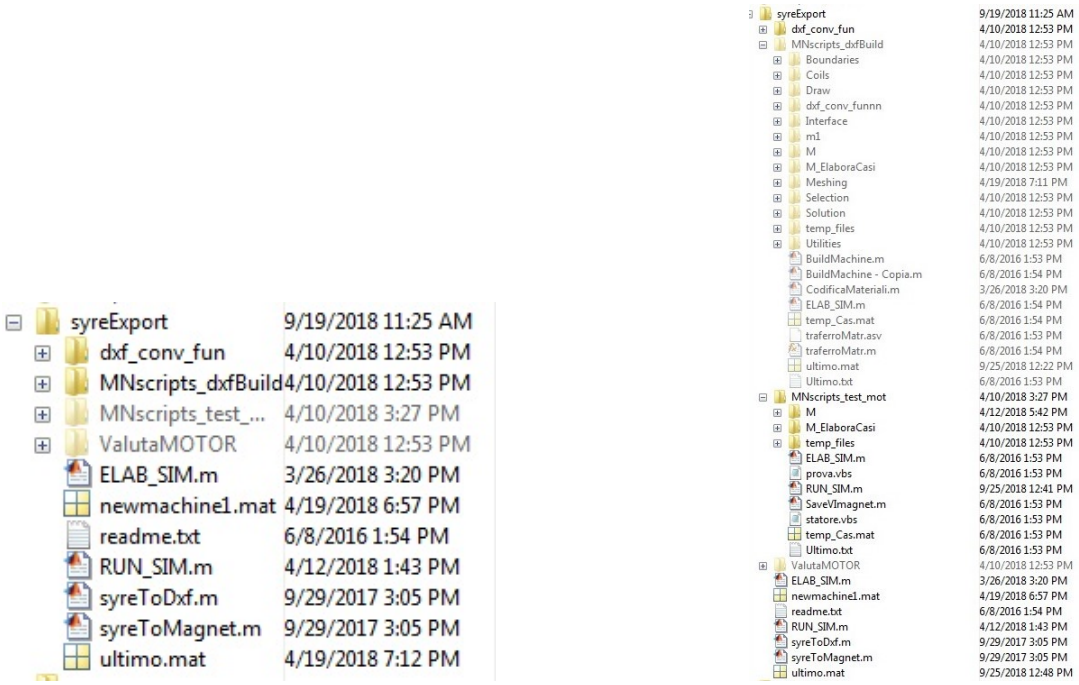
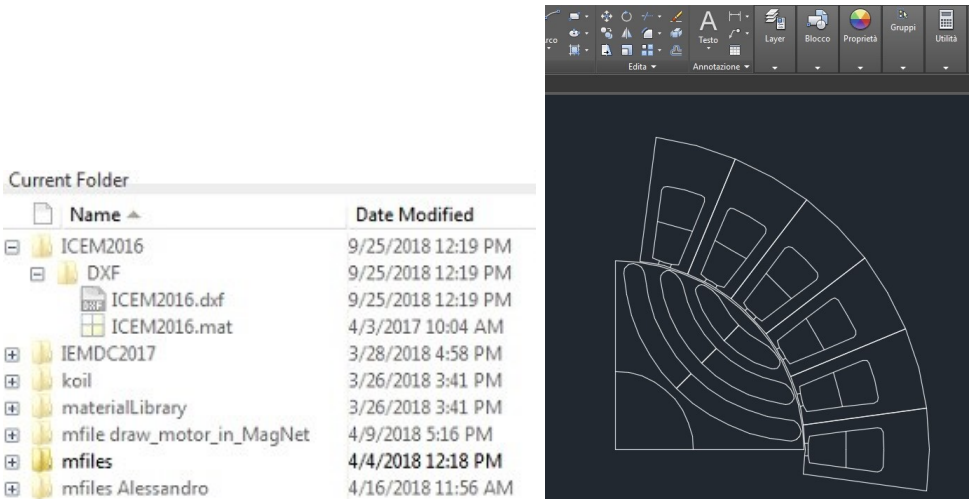


Figura 2.2 Cartella e Sottocartella syreExport [3]

Si ricorda inoltre che il processo di simulazione tramite *MagNet* doveva essere eseguito manualmente, lanciando 4 script separatamente nella giusta sequenza mostrata in seguito. Ricapitolando, dopo aver salvato la macchina desiderata tramite la GUI SyR-e in formato .mat e .fem (ad esempio ICEM2016.mat) si procedere lanciando lo script *syreTodxf* che ci restituisce il disegno della macchine in formato .dxf (nel nostro caso ICEM2016.dxf vedi figura 2.3) nell'apposita cartella nella directory principale (...\ICEM2016\DXF \).

Dopo aver creato il file .dxf della macchina selezionata si procede lanciando il secondo script chiamato *syreToMagnet*: esso costruisce il modello della macchina in *MagNet* accettando in ingresso i file .mat e .dxf appena creati e restituisce in uscita il file .mn creato da *MagNet* della macchina completa. Difatti lo script avvia MagNet, carica il disegno dal file .dxf e leggendo i dati presenti nel file .mat costruisce rotore, statore, traferro e avvolgimenti; ciò che otteniamo è il file .mn (nel nostro esempio ICEM2016.mn) pronto per essere simulato.

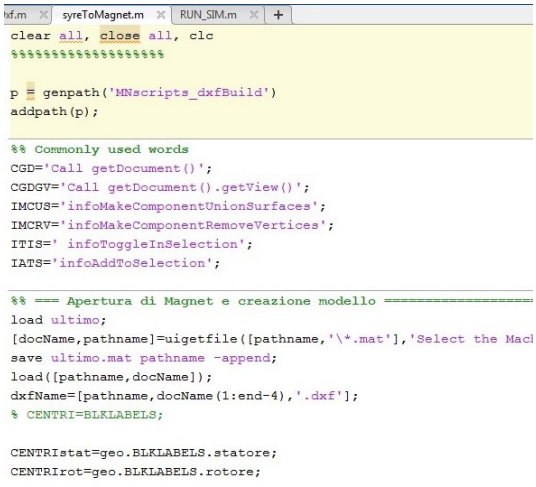


(a) Cartella salvataggio .dxf

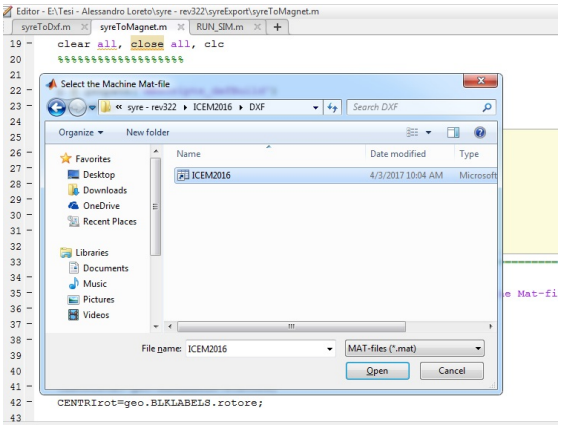
(b) Visualizzazione file .dxf con AutoCAD

Figura 2.3 Funzionamento prima verisone di MagNet [3]

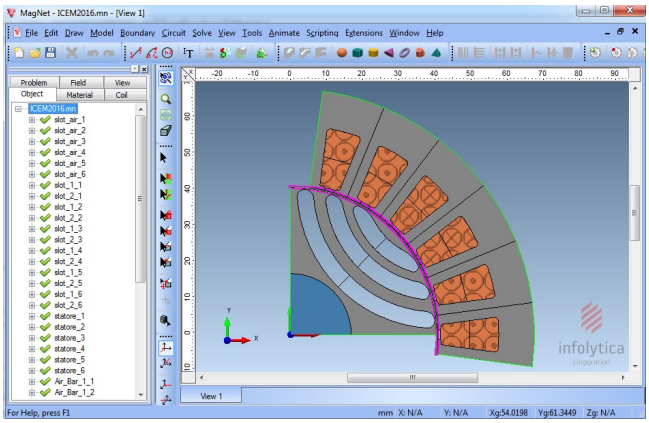
Nella figura 2.4 si mostrano le figure chiavi del processo di creazione della macchina con lo script *syreToMagnet*:



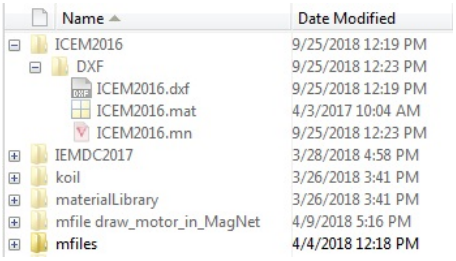
(a) Script syreToMagNet.m



(b) Caricamento file .mat della macchina



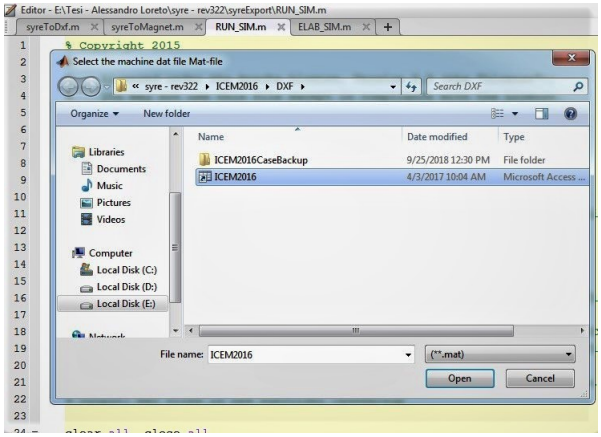
(c) Creazione macchina in MagNet



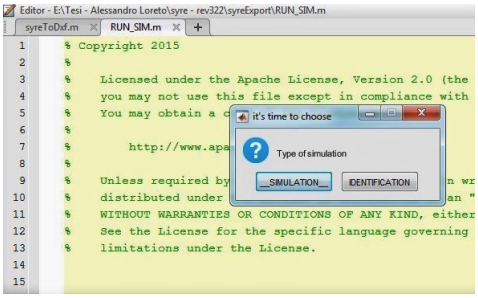
(d) Salvataggio finale dei risultati

Figura 2.4 Creazione macchina in MagNet [3] [4]

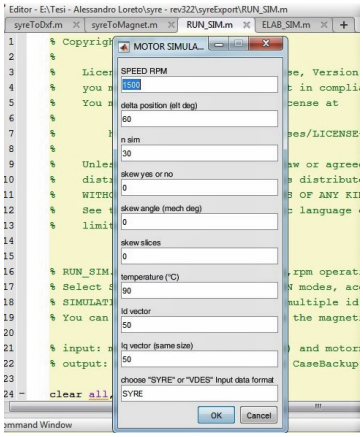
Infine, dopo aver creato il modello della macchina in *MagNet*, si procede con la simulazione: lanciando lo script *RUN_ SIM.m* si avvia il processo di simulazione caricando il file .mat nella stessa cartella in cui è salvato il file .mn della macchina che si vuole simulare (è importante scegliere la cartella per evitare errori durante l'esecuzione).



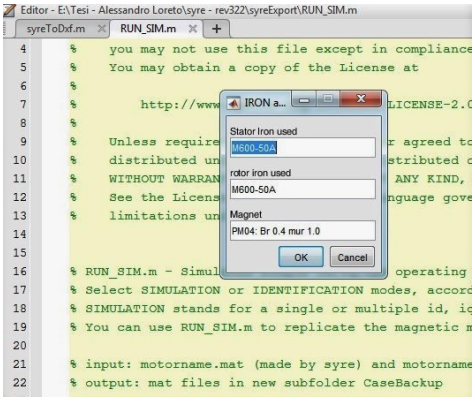
(a) Carciamento macchina



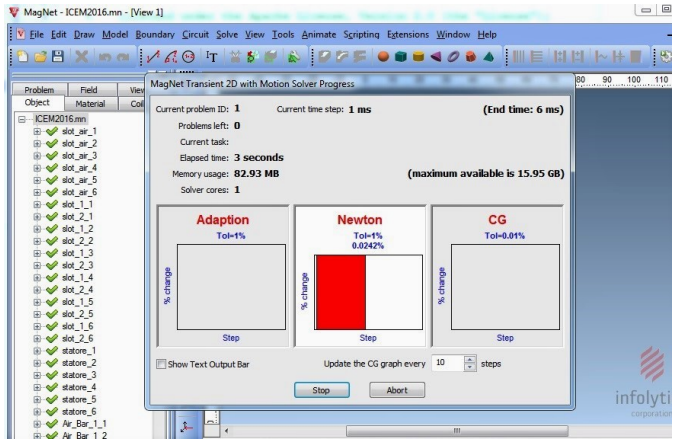
(b) Tipo di simulazione



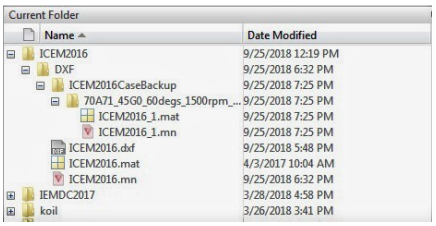
(c) Settaggio parametri della simulazione



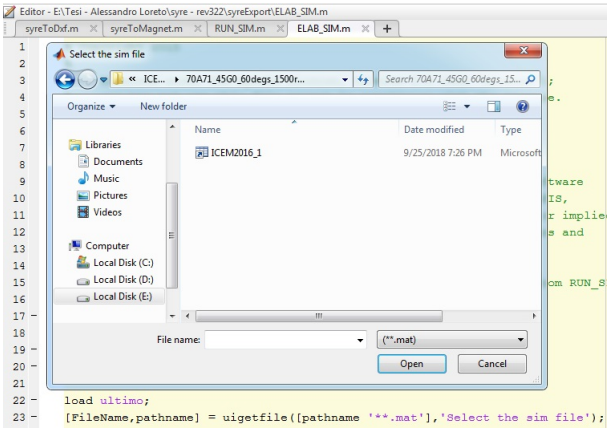
(d) Imposto materiali



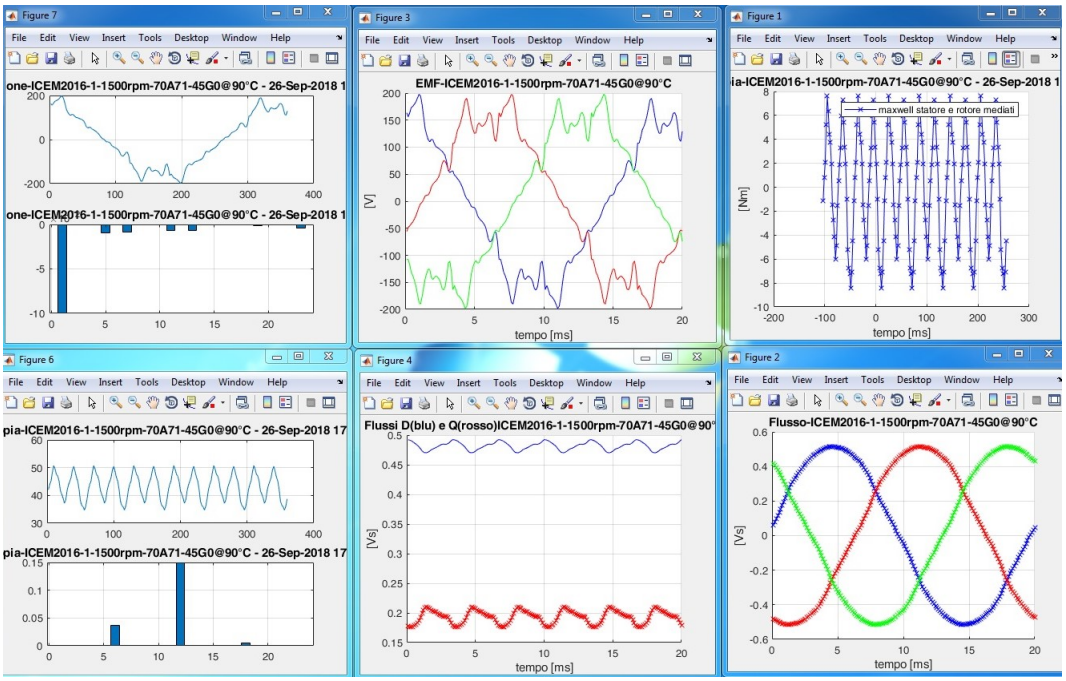
(e) Simulazione in MagNet



(f) Salvataggio finale dei risultati



(g) Caricamento risultati in ELAB_SIM



(h) Plot dei risultati

Figura 2.5 Simulazione in RUN_SIM [3]

In figura 2.5 si mostrano le figure chiavi del processo di simulazione:

a) il caricamneto della macchina avviene automaticamente lanciando lo script: apparirà una finestra in cui dovremmo selezionare la macchina creata.

b) selezionata la macchina bisognerà scegliere il tipo di simualzione :

- *SIMULATION*: si sceglie di simulare la macchina in uno o più punti di lavoro, a seconda che si inserisca un vettore di correnti e angolo di fase della corrente o meno.
- *IDENTIFICATION*: equivalente ad impostare il valore 1000 alla fase della corrente; si accede al calcolo della mappa dei flussi impostando la massima corrente i_{dq} e lo step di variazione delle correnti.

c) se si è scelto *SIMULATION* si accede alla simulazione della macchina; appare un menù dove è possibile settare i parametri per la simulazione come la velocità di rotazione, l'escursione rotorica, il numero delle simulazioni, lo skewing (se esiste), la temperatura della macchina e la corrente i_{dq}.

d) dopo aver impostato i parametri per la simulazione, una nuova finestra chiede di impostare il materiale del ferro di rotore-statore e i magneti. Inoltre si è verifcato che questo modello di simulazione non importa i nuovi materiali, ma avvia la simulazione con i soli materiali di default presenti in MagNet. Si può intuire che questo passaggio risulta essere limitante durante la simulazione di una nuova macchina e quindi dovrà essere migliorato.

e) avvio della simulazione.

f) conclusa la simulazione vi è il salvataggio automatico dei risultati in una nuova cartella di backup appositamente creata, in cui viene salvato sia il file .mn che il file .mat

g) completato il salvataggio, si elaborano i risultati tramite *ELAB_SIM*; lanciato lo script si carica il file .mat precedentemente simulato e salvato. Esso rielabora i risultati simulati e restituisce i plot dei risultati salvandoli nella cartella di backup.

h) plot finale con i risultati (coppia, flusso abc, tensioni di fase, flusso dq, armoniche di coppia, armoniche di tensione e se richiesto le perdite)

2.2 Struttura Finale del Codice

Dopo aver analizzato e compreso il funzionamento degli script e i suoi punti critici, come ad esempio la necessità di avviare manualmente ogni script nella corretta sequenza richiamando i file giusti nelle rispettive cartelle, si è passati all'implementazione automatica della simulazione tramite l'aggiunta di un bottone nella *GUI SyR-e*. Come già detto precedentemente si è voluta mantenere la logica di funzionamento presente in *FEMM* (in figura 2.6) perché risulta molto stabile, performante, lineare e semplice da comprendere qualora la si voglia modificare.

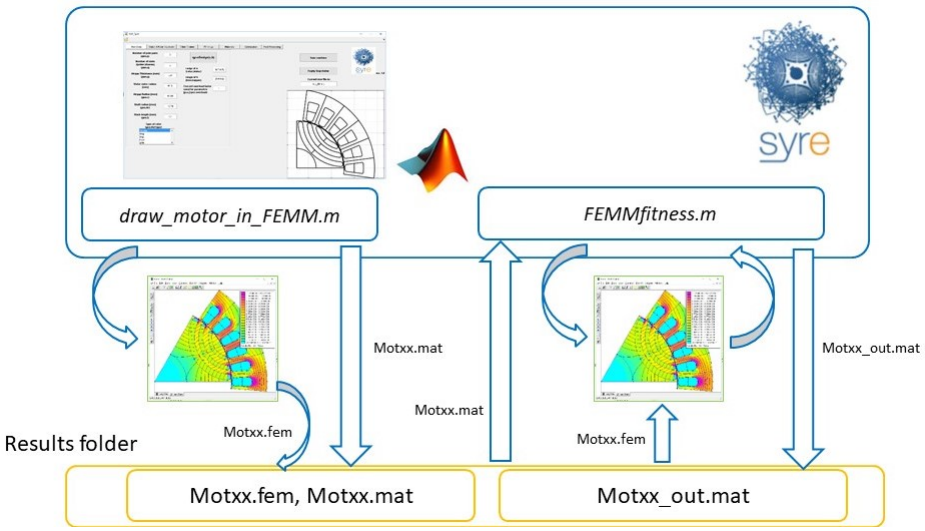


Figura 2.6 Logica già esistente in FEMM

Osservando la figura 2.6 si comprende la logica esistente in FEMM. Si nota che caricando il modello della macchina nella GUI SyR-e, modificato secondo le nostre esigenze, si procede con il salvataggio: la funzione denominata *draw_motor_in_FEMM* lanciata automaticamente dalla GUI attraverso l'appostito bottone *Save Machine*, crea il file .mat e .fem della macchina.

Dopo aver ottenuto il modello della macchina è possibile lanciare la simulazione nella finestra *Post Processing* attraverso il bottone *Start* che richiama automaticamente la funzione *FEMMfitness*: essa è in grado di gestire da sola tutto il processo di simulazione della

macchina prendendo come input i parametri impostati nella GUI SyR-e e ci restituisce il file .mat finale comprensivo dei risultati che poi saranno plottati.

Compreso il funzionamento in FEMM si è implementata la stessa logica in *MagNet*: in figura 2.7 osserviamo la logica di funzionamento finale della simulazione tramite *MagNet*. Come nella figura precedente, caricato il modello della macchina nella GUI SyR-e e opportunamente modificato, si procede al salvataggio tramite la nuova funzione omonima a FEMM chiamata *draw_motor_in_MN*, da me creata, prendendo spunto dagli script già esistenti (*syreTodxf.m e syreToMagnet*). Lanciandola automaticamente dalla GUI SyR-e attraverso un nuovo bottone *MN* (rosso), più piccolo situato alla destra del bottone già esistente, avvia la creazione e il salvataggio della macchina in *MagNet* creando il file .mat e .fem.

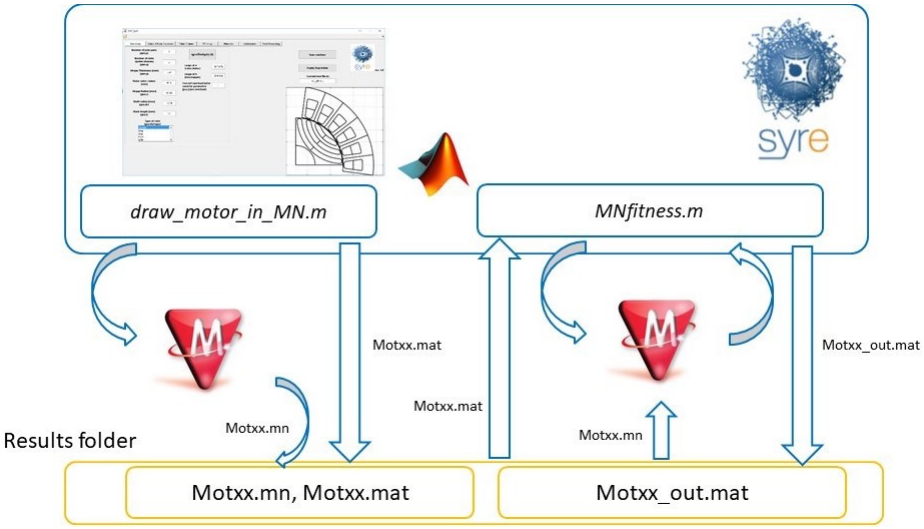


Figura 2.7 Logica finale di MagNet

Dopo aver creato il modello della macchina si procede con la simulazione, anch'essa lanciata nella simulazione nella finestra *Post Processing*, attraverso un nuovo bottone *Start* (rosso), situato a fianco del bottone già esistente. Si avvierà la simulazione con *MagNet* lanciando la nuova funzione *MNfitness.m*: essa riesce a completare l'intera simulazione esattamente come in FEMM e restituendo il file .mat dei risultati che poi verranno salvati e visualizzati.

Capitolo 3

Risultati

3.1 Creazione della Macchina in MagNet

Il processo logico che si è deciso di seguire per la realizzazione del funzionamento del salvataggio della nuova macchina è stato quello di partire dalle ultime funzione annidate e ripercorrere nel senso inverso il processo di salvataggio in modo da a rendere compatibile tutte le funzioni tra di loro.

Il primo passo per la realizzazione del lavoro è stato l'inserimento di un nuovo pulsante chiamato *'MN'* (situato a fianco del pulsante già a esistente *'Save Machine'* nella GUI SyR-e) per il salvataggio della nuova macchina in modo da poter avviare il salvataggio direttamente dalla GUI SyR-e attraverso MagNet.

La scelta di inserire questo bottone con dimensioni più piccole rispetto a quello principale, usato per simulare con FEMM, è data dal fatto che *MagNet* è un programma che richiede la licenza e non un open-source come FEMM, quindi solo una piccola parte di utenti ne potranno usufruire. Si è scelto di inserire questo bottone avente dimensioni più piccole rispetto a quello principale, usato per simulare con FEMM, poichè MagNet a differenza di FEMM è un software protetto da licenza e quindi utilizzabile da pochi utenti.

Le prime operazioni di scrittura e modifica script già esistenti derivano dall'esigenza che *MagNet* per poter funzionare ha bisogno di importare il file *.dxf* in cui è contenuta la

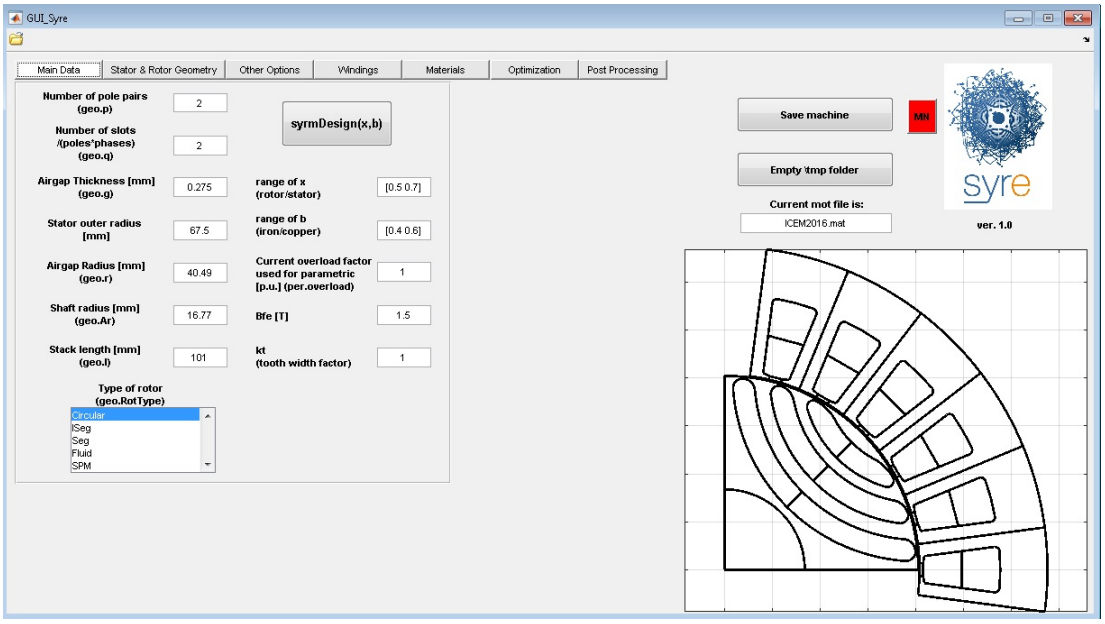


Figura 3.1 Finestra salvataggio macchina

geometria della macchina in 2D: bisogna saper creare il file *.dxf* partendo dal file *.mat* in cui sono riportati tutti i parametri geometrici che permetteranno di disegnare la macchina. Lo script per la creazione del .dxf era già presente, ma è stato convertito in funzione prendendo il nome *SyreToDxf* ed è stato migliorato affinche sia in grado di lavorare anche mancanza di input qualora ne fosse necessario. Il salvataggio del file .dxf avviene in una cartella temporanea appositamente creata in [pathname '\tmp\MagnetExportTmp\mot_0\DXF]

```
1  function syreToDxf(stator,rotor,pathname,filename)
2
3  flag=0;
4  if nargin<4
5      flag=1;
6      close all; clc;
7      load LastPath.mat
8      [filename, pathname] = uigetfile([pathname '\*.mat'], 'Pick a motor');
9      load([pathname filename]);
10     stator = geo.stator;
```

```
11      rotor =  geo.rotor;
12  end
13
14  pathname_DXF=[pathname,filename(1:end-4),'\DXF\'];
15  mkdir(pathname_DXF);
16  nomefile=[pathname_DXF,filename(1:end-4),'.dxf'];
17  raggi=[];
18  avvolgimento=[];
19  magneti=[];
20  DXFconv(raggi,avvolgimento,rotor,stator,magneti,nomefile);
```

Uscendo dalla funzione *SyreToDxf* ci si ritrova nella funzione principale per la creazione e il salvataggio della macchina: *Draw_motor_in_MN*. Gli input di questa funzione sono *geo* (contiene i dati relativi alla geometria), *eval_type* (ci dà indicazioni sul tipo di simulazione scelta), *mat* (ci dà informazioni sui materiali usati per la costruzione della macchina)

Nelle prime righe di *Draw_motor_in_MN* si crea la cartella temporanea che servirà per il salvataggio del file .dxf e del file .mn. Dopo aver definito la mesh, calcolato il fatto di avvolgimento e impostato l'angolo di offset si generano le matrici di rotore e statore grazie alle due funzioni *ROTmatr(geo,fem,mat)* e *STATmatr(geo,fem)*

```
1  %% tmp folder for MagNet export%% tmp folder for MagNet export
2  dname=[cd '\tmp\MagnetExportTmp\']; % creo cartella temporanea
3  mkdir(dname);
4  filename='mot_0.mat';
5  %define mesh size
6  fem = dimMesh(geo,eval_type);
7  % calc winding factor (kavv) and rotor offset (phase1_offset)
8  [kavv, phase1_offset] = calcKwTh0(geo.tempWinTable,geo.ns*geo.p,geo.p);
9
10 % offset angle for coordinate transformations
11 if strcmp(geo.RotType,'SPM')
12     phase1_offset = phase1_offset-90;
13 end
```

```
th_m0 = 0;                                % rotor position [mec deg]
geo.th0 = th_m0*geo.p - phase1_offset;  % d- to alpha-axis offset [elt deg]

% nodes
geo.x0 = geo.r/cos(pi/2/geo.p);
geo.fem=fem;
[rotor,BLKLABELSrot,geo,mat] = ROTmatr(geo,fem,mat); % rotor and ...
    BLKLABELSrot describe the rotor
geo.rotor = rotor;

[geo,statore,BLKLABELSstat] = STATmatr(geo,fem); % statore and ...
    BLKLABELSstat describe the stator
geo.stator=statore;
BLKLABELS.materials=geo.BLKLABELSmaterials;
BLKLABELS.rotore = BLKLABELSrot;
BLKLABELS.statore= BLKLABELSstat;
geo.BLKLABELS=BLKLABELS;
```

Queste due ultime funzioni sono essenziali nel salvataggio della macchina, devono essere eseguite ogni qualvolta la si crea o la si modifica poichè vengono generate matrici e strutture specifiche per ogni macchina che serviranno durante la realizzazione della macchina; è interessante specificare che entrambi i metodi di salvataggio utilizzano le identiche funzione sopra citate per la realizzazione delle matrice di rotore e statore, in modo da garantire l'uguaglianza della geometria per qualunque metodo. In particolare la funzione *ROTmatr* viene creata per la costruzione del rotore. Essa crea 4 nuove variabili:

- rotor: ogni riga identifica una linea o un arco per costruzione attraverso MagNet o FEMM.
- BLKLABELSrot.xy: identifica i punti centrali dei blocchi di MagNet o FEMM.
- BLKLABELSrot.boundary: ogni riga identifica le condizioni a contorno per MagNet o FEMM.
- BLKLABELSrot.BarName: rapresenta il nome del blocco della barriere di flusso.

Proseguendo troviamo la funzione prima descritta *syreToDxf*, che come spiegato crea il file .dxf sfruttando le informazioni presenti nelle matrici *geo.stator* e *geo.rotor* e lo salva nella cartella temporanea che a fine salvataggio verrà eliminata.

```
% creates DXF file
syreToDxf(geo.stator, geo.rotor,dname, filename);
```

Dopo aver creato il file .dxf si avvia *MagNet*, si apre un nuovo documento e si carica il file .dxf della macchina appena creato; queste tre operazioni sono possibili grazie ad *ActiveX* che permette la comunicazione tra *Matlab* e *MagNet*.

```
% OpenMagent
h = OpenMagnet(1);  % 1 significa visibile, 0 invisibile

% New Document
h = NewDocumentMagnet(h);

% Import DXF
ImportDXFMagnet(h,dxfName);
```

Adesso è necessario capire se in *MagNet* sono caricati i materiali selezionati da *SyRe* per la creazione della macchina. Negli script iniziali a me consegnati, *MagNet* non era in grado di importare nuovi materiali da *SyRe*: questo rappresenta una grande limitazione in quanto vi è la possibilità di inserire nuovi materiali in SyR-e, oltre quelli esistenti, qualora ne fosse necessario e importarli nel programma di simulazione proprio come avviene in *FEMM* durante la creazione della macchina. Come sopra spiegato la comunicazione tra *Matlab* e *MagNet* è possibile grazie ad *ActiveX*, eseguendo determinate righe di codici si riesce ad eseguire comandi su MagNet; durante il mio lavoro di tesi si è cercato di creare un'alternativa che fosse in grado di importare nuovi materiali dalla GUI SyR-e a

MagNet. Si sono pensate diverse soluzioni: quella da me applicata prevede la lettura dei soli materiali presenti in MagNet come *UserMaterial* e la stampa nella lista di questi materiali nella variabile *listOfUserMaterials*. Riporto i comandi da eseguire per leggere gli *UserMaterial* da *MagNet*:

```
Command = ['CALL getUserMaterialDatabase().getMaterials(materials)'];
invoke(h.magnetHandler, 'processCommand', Command);
invoke(h.magnetHandler, 'processCommand', 'Call setVariant(0,materials)');
listOfUserMaterials = invoke(h.magnetHandler, 'getVariant', 0);
```

Al suo interno si confronta il nome del materiale per ogni singolo blocco con la lista dei materiali ricavati precedentemente da *MagNet*: se il materiale è presente nella lista, si crea una nuova variabile che identificherà il nome del blocco a cui verrà assegnato il materiale, altrimenti se il materiale non è presente nella lista si procederà con la creazione e importazione del materiale in *MagNet*. Dopo svariate prove tra i comandi riportati nel help di *MagNet* si è riusciti a trovare una soluzione stabile che permettesse di importare qualsiasi tipo di materiale lineare (ferro di rotore, statore o magnete, albero). Di seguito si riporta uno dei comandi da me eseguiti per la creazione, l'importazione e il settaggio del *'Magnete di Rotore'*:

```
%% imposto Rot_Magn tra i materiali presenti in MagNet

if (ismember(mat.LayerMag.MatName,listOfUserMaterials))
      Rot_Magn = mat.LayerMag.MatName;

else
    Command= ['CALL ...
        getUserMaterialDatabase().newMaterial("',mat.LayerMag.MatName,'")'];
    invoke(h.magnetHandler, 'processCommand', Command);
    if (strcmp(mat.LayerMag.MatName, 'Air')) % differente per l'aria ...
        (trasparente)
```

```
10          Command= ['CALL ...
                getUserMaterialDatabase().setMaterialColor("',mat.LayerMag.MatName,'", ...
                255, 255, 255, 255)'];
11          invoke(h.magnetHandler, 'processCommand', Command);
12      else
13          Command= ['CALL ...
                getUserMaterialDatabase().setMaterialColor("',mat.LayerMag.MatName,'", ...
                255, 128, 0, 255)'];
14          invoke(h.magnetHandler, 'processCommand', Command);
15      end
16      %creo ArrayOfValues Permeabilita'
17      Command = 'REDIM ArrayOfValues(0, 2)';
18      invoke(h.magnetHandler, 'processCommand', Command);
19      string ='ArrayOfValues(0, 0)= 20';
20      invoke(h.magnetHandler,'processCommand',string);
21      string=['ArrayOfValues(0, 1)= ' num2str(mat.LayerMag.mu)];
22      invoke(h.magnetHandler,'processCommand',string);
23      string=['ArrayOfValues(0, 2)= '  num2str(mat.LayerMag.Hc)];
24      invoke(h.magnetHandler,'processCommand',string);
25      Command= ['CALL ...
            getUserMaterialDatabase().setMagneticPermeability("',mat.LayerMag.MatName,'", ...
            ArrayOfValues, infoLinearIsotropicReal)'];
26      invoke(h.magnetHandler, 'processCommand', Command);
27      %creo ArrayOfValues Conducibilita'
28      Command = 'REDIM ArrayOfValues1(0, 1)';
29      invoke(h.magnetHandler, 'processCommand', Command);
30      string ='ArrayOfValues1(0, 0)= 20';
31      invoke(h.magnetHandler,'processCommand',string);
32      string=['ArrayOfValues1(0, 1)= '  num2str(mat.LayerMag.sigmaPM)];
33      invoke(h.magnetHandler,'processCommand',string);
34      Command= ['CALL ...
            getUserMaterialDatabase().setElectricConductivity("',mat.LayerMag.MatName,'", ...
            ArrayOfValues1, infoLinearIsotropicReal)'];
35      invoke(h.magnetHandler, 'processCommand', Command);
36      %creo ArrayOfValues Permettivita'
37      Command = 'REDIM ArrayOfValues2(0, 1)';
38      invoke(h.magnetHandler, 'processCommand', Command);
39      string ='ArrayOfValues2(0, 0)= 20';
40      invoke(h.magnetHandler,'processCommand',string);
```

```
41      string='ArrayOfValues2(0, 1)= 1';
42      invoke(h.magnetHandler,'processCommand',string);
43      Command= ['CALL ...
            getUserMaterialDatabase().setElectricPermittivity("',mat.LayerMag.MatName,'", ...
            ArrayOfValues2, infoLinearIsotropicReal)'];
44      invoke(h.magnetHandler, 'processCommand', Command);
45      %creo ArrayOfValues Densita'
46      Command = 'REDIM ArrayOfValues3(0, 1)';
47      invoke(h.magnetHandler, 'processCommand', Command);
48      string ='ArrayOfValues3(0, 0)= 20';
49      invoke(h.magnetHandler,'processCommand',string);
50      string=['ArrayOfValues3(0, 1)= '  num2str(mat.LayerMag.kgm3)];
51      invoke(h.magnetHandler,'processCommand',string);
52      Command= ['CALL ...
            getUserMaterialDatabase().setMassDensity("',mat.LayerMag.MatName,'", ...
            ArrayOfValues3)'];
53      invoke(h.magnetHandler, 'processCommand', Command);
54  %assegno nome nuovo materiale
55    Rot_Magn = mat.LayerMag.MatName;
```

Si nota il confronto della lista degli User Materials con il materiale del Layer preso in esame (nell'esempio il magnete di rotore). Se il materiale è presente nella lista significa che MagNet lo possiede, allora si continua assegnando il nome del materiale alla nuova variabile la quale verrà trasmessa a *MagNet*.
Se il materiale non è presente nella lista allora lo si crea: tramite una serie di comandi si riesce a creare e settare le proprietà del nuovo materiale quali la permeabilità, conducibilità, permettività e la densità in base al tipo di materiale; infine si assegna il nome del materiale alla nuova variabile la quale verrà trasmessa a *MagNet*.

Assegnati i materiali alle variabili si ordinano nel modo desiderato in *BLKLABELS.materials*:

```
1  BLKLABELS.materials=char(Air1, Air2, ...
       Stat_Cu,Stat_Fe,Rot_Fe,Stat_Magn,Rot_Magn,Shaft_Steel,Rot_Cu);  %scelgo ...
       io i nomi dei materiali
```

1. AIR

2. AIR VIRTUAL
3. STATOR COIL
4. STATOR IRON
5. ROTOR IRON
6. STATOR MAGNET
7. ROTOR MAGNET
8. SHAFT STEEL
9. ROTOR COIL

Definiti i materiali e creata la matrice *BLKLABELS* si passa alla realizzazione vera e propria della geometria della macchina in MagNet: si disegna il traferro utilizzando l'algoritmo già presente nella vecchia funzione e si assegnano i compomenti a statore e rotore: la vecchia versione presentava due script che permettevano l'assegnazione dei componenti ai blocchi, ma il procedimento era poco stabile: l'assegnazione avveniva in modo statico, ad ogni riga delle matrice BLKLABELSrot e BLKLABELSstat veniva assegnato il componente, bastava qualsiasi variazione delle righe nelle due matrici che l'algoritmo si bloccava. Si è pensato allora di modificare il procedimento più vulnerabile, ovvero *assign_block_prop_rot_MN*, creando un nuovo algoritmo che fosse in grado di assegnare dinamicamente i materiali/proprietà ai blocchi in base a determinati valori della matrice BLKLABELSrot, nello specifivco in base al valore della terza colonna si assegnano le proprietà del componente. In seguito si riporta l'algoritmo da me utilizzato per l'assegnazione dinamica ai componenti

```
1 for kk=1:length(BLKLABELSrot.xy(:,1))
2         switch BLKLABELSrot.xy(kk,3)
3             case 1 %air
```

```
4                   h = MakeComponentMagnet(h,[BLKLABELSrot.xy(kk,1), ...
                        BLKLABELSrot.xy(kk,2)], ...
                        cell2mat(BLKLABELSrot.BarName{kk,1}) ,l, ...
                        BLKLABELS.materials(2,:),'Uniform', ...
                        [BLKLABELSrot.xy(kk,6),BLKLABELSrot.xy(kk,7), ...
                        BLKLABELSrot.xy(kk,8)],BLKLABELSrot.xy(kk,4));
5
6
7               case 5 %rotor iron
8                   h = MakeComponentMagnet(h,[BLKLABELSrot.xy(kk,1), ...
                        BLKLABELSrot.xy(kk,2)] ...
                        ,'rotor',l,BLKLABELS.materials(5,:), 'None', [0, 0, 1], ...
                        BLKLABELSrot.xy(kk,4));
9
10              case 6 %PM
11                  if Br==0 || strcmp(geo.RotType,'SPM')
12                     h = MakeComponentMagnet(h,[BLKLABELSrot.xy(kk,1), ...
                           BLKLABELSrot.xy(kk,2)] ...
                           ,cell2mat(BLKLABELSrot.BarName{kk,1}),l ...
                           ,BLKLABELS.materials(2,:), 'None',[0, 0, ...
                           1],BLKLABELSrot.xy(kk,4));
13                  else
14                     h = MakeComponentMagnet(h,[BLKLABELSrot.xy(kk,1), ...
                           BLKLABELSrot.xy(kk,2)] ...
                           ,cell2mat(BLKLABELSrot.BarName{kk,1}),l, ...
                           BLKLABELS.materials(7,:),'Uniform',[BLKLABELSrot.xy(kk,6), ...
                           BLKLABELSrot.xy(kk,7), ...
                           BLKLABELSrot.xy(kk,8)],BLKLABELSrot.xy(kk,4));
15                  end
16
17              case 7 %shaft
18                  h = MakeComponentMagnet(h,[BLKLABELSrot.xy(kk,1), ...
                        BLKLABELSrot.xy(kk,2)],'shaft',l,BLKLABELS.materials(8,:), ...
                        'None',[0, 0, 1],-1);
19          end
20      end
```

Continuando si creano le bobine di statore, si imposta la mesh al traferro, si assegnano le condizioni a contorno al rotore e statore, si imposta la parte rotante per la simulazione

Transient with Motion utilizzando le funzioni già presenti nello script *syreToMagnet* effettuando le giuste modifiche delle variabili utilizzate e infine si salva il documento nella cartella temporanea e si chiude *MagNet*. Adesso la macchina è stata creata e salvata: si ritorna nella funzione precedente *DrawPushMachineMN* in cui semplicemente si assegnano le variabili per eseguire la funzione *Draw_motor_in_MN*, si copia il file .mn della macchina appena creata nella current directory e si elimina la cartella temporanea. Infine si aggiorna la GUI SyR-e.
Il salvataggio è stato completato, adesso si è creata la nuova macchina ed è pronta per essere simulati tramite MagNet

```
1
2  function dataSet = DrawPushMachine_MN(handles,filename,dname)
3
4  dataSet = handles.dataSet;
5
6  if nargin < 2
7      [filename,dname] = uiputfile(['newmachine.mat'],'input machine name and ...
           location');
8      dname = dname(1:end-1);
9
10 end
11
12 %% ==== FIRST PART FROM FEMMFitnessX FOR MN
13 [~, ~, geo, per, mat] = data0(dataSet);
14 RQ = dataSet.RQ;
15 currentDir = pwd;
16 eval_type = 'singt';
17
18 [geo,gamma,mat] = interpretRQ(RQ,geo,mat);
19
20 [geo,mat] = draw_motor_in_MN(geo,eval_type,mat);
21 %% SAVE THE FILE
22 cd(dname);
23 geo.RQ = RQ;
```

```
24 copyfile([cd ...
       '\tmp\MagnetExportTmp\mot_0.mn'],[cd,'\',filename(1:end-4),'.mn']);
25 rmdir([cd '\tmp\MagnetExportTmp\'],'s');
26 save(filename,'geo','per','dataSet','mat');
27 set(handles.currentMotFileName,'String',filename);  % update display
28 %% refresh GUI display data
29 load([dname '\' filename]);
30 dataSet.RQ = round(dataSet.RQ,4);
31 dataSet.currentpathname = [dname '\'];
32 dataSet.currentfilename = filename;
33 cd(currentDir);
```

3.2 Simulazione della Macchina in MagNet

Dopo aver avviato la creazione e il salvataggio della macchina si procede in modo analogo con la simulazione. Per eseguire la simulazione bisogna aprire la finestra *Post Processing*, qui troveremo tutti i parametri da settare prima di avviare la simulazione attraverso il pulsante *Start*. Come nel caso del salvataggio macchina anche qui è stato aggiunto un apposito bottone di colore rosso, più piccolo ma ben visibile, a fianco di quello già presente che permette la simulazione attraverso *FEMM* figura 3.2

Successivamente il settaggio dei parametri per la simulazione, quali: l'escursione angolare di rotore, l'angolo di fase della corrente, il carico della corrente in p.u., il numero delle posizioni rotore, il valore di induttanza (qualora ci fosse il magnete) e il numero di punti della griglia se si sceglie di voler ottenere la mappa dei flussi (si ricorda che per attivare il calcolo della mappa dei flussi è necessario inserire il valore pari a '1000' nel campo riguardate la corrente di fase) si potrò procedere con la simulazione cliccando il tasto rosso MN situato a fianco il tasta Start.

Come fatto per la realizzazione delle funzioni per la creazione della macchina, ovvero si è partiti dalle funzioni annidate e è percorso nel senso inverso il processo logico di esecuzione in modo da riuscire a rendere compatibili tutte le funzioni tra loro, è stato fatto anche per la realizzazione della funzione Post Processing.

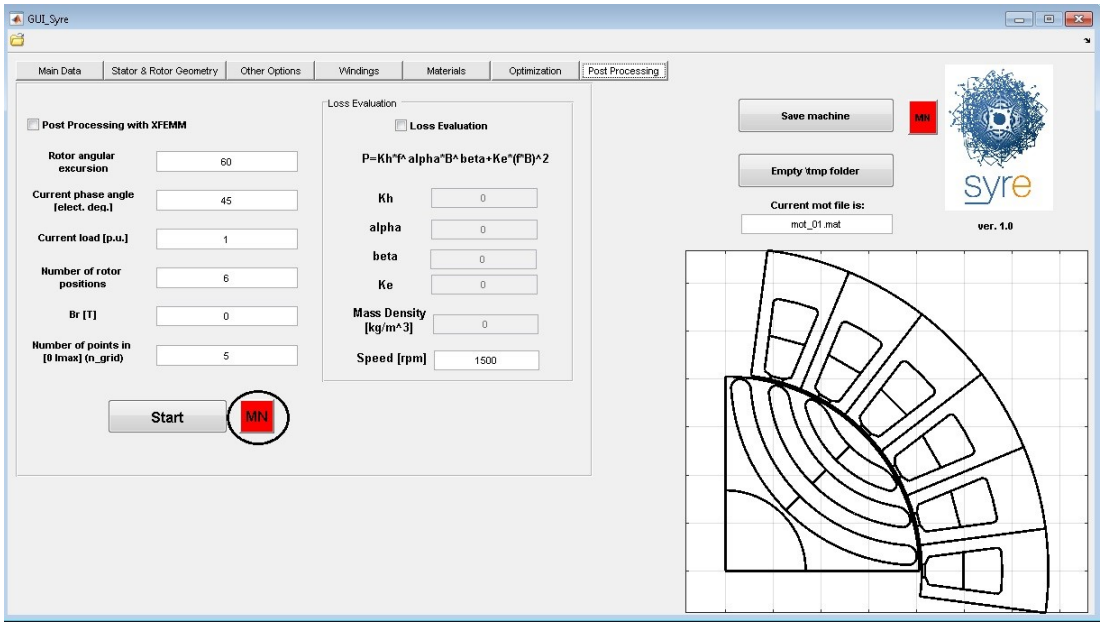


Figura 3.2 Finestra Post Processing

```
477 - if dataIn.LossEvaluationCheck == 0
478 -     set(handles.HysteresisLossFactorEdit,'Enable','off');
479 -     set(handles.HysteresisFrequencyFactorEdit,'Enable','off');
480 -     set(handles.HysteresisFluxDenEdit,'Enable','off');
481 -     set(handles.EddyCurLossFactorEdit,'Enable','off');
482 -     set(handles.MassDensityEdit,'Enable','off');
483 -     set(handles.EvaluatedSpeedEdit,'Enable','on');
```

```
295 -    if dataIn.LossEvaluationCheck == 1
296 -       per.EvalSpeed = dataIn.EvalSpeed;
297 -    end
```

Figura 3.3 Abilitazione lettura variabile 'Speed'

Inoltre è giusto ricordare che per procedere correttamente con la simulazione è necessario abilitare il campo *Speed* nella GUI_SyRe modificando opportunamente la riga numero 483 in GUI_Syre.m, permettendo la lettura valore inserito nella GUI_Syre e passarlo a *MagNet* tramite la variabile per.EvalSpeed questo perché, a differenza di *FEMM*, *MagNet* richiede il settaggio del valore della velocità come dato di input per la simulazione.

Difatti tutte le operazioni di calcolo vengono eseguiti dalla funzione denominata *Simulate_xdeg_MN*, essa rappresenta la funzione più importante in quanto all'interno vengono passate tutte le variabili e i parametri per eseguire la simulazione sia per quanto riguarda il caso 'singt' cioè il calcolo di coppia, IPF e flusso che per il caso 'singm' cioè il calcolo della mappa dei flussi.

Entrati nella funzione *Simulate_xdeg_MN* viene controllata la variabile *eval_type* in

quanto *MagNet* non è possibile eseguire il processo di ottimazione, infatti in caso di ottimizzazione *MO_ OA* o *MO_ GA* SyRe stampa l'errore e interrompe l'esecuzione.
Se *eval_ type* è uguale a 'singt' o 'singm' si caricheranno le variabili sim, xdeg e gamma altrimenti si visualizzerà l'errore, la risoluzione dell'ottimizzazione non è stata implementata come per FEMM: in MagNet non è possibile eseguire il parallel pool e si preferisce non usarla.

```
switch eval_type
    case 'singt'
        nsim = geo.nsim_singt;
        xdeg = geo.delta_sim_singt;
        gamma = per.gamma;
    case 'singm'
        nsim = geo.nsim_singt;
        xdeg = geo.delta_sim_singt;
        gamma = per.gamma;
    otherwise
        error('MagNet Simulations not available during optimization!');
end
```

Continuando si impostano le variabili geometriche utili per il calcolo e si calcola l'ampiezza della corrente *iAmp, id, iq e Imod*

```
th0 = geo.th0;
p   = geo.p;
r  = geo.r;
gap = geo.g;
ns  = geo.ns;
pc  = 360/(ns*p)/2;
ps  = geo.ps;
n3phase = geo.n3phase; %AS number of 3-phase circuits
Q = geo.ns*geo.p;
skew_angle = 0;
N_parallel = 1;
N_cond = geo.Ns/geo.p/geo.q/size(geo.avv,1);
```

```
Br = per.BrPP;
q = geo.q;
gamma_ = gamma*pi/180;
% evaluation of the phase current values for all positions to be simulated
iAmp = per.overload*calc_io(geo,per);

id = iAmp * cos(gamma * pi/180);
iq = iAmp * sin(gamma * pi/180);
Imod = abs(id + 1i* iq);
```

Dopo aver impostato le variabili utili per la simulazione si crea la cartella Backup in base al caso in analisi, 'singt' o 'singm'. Ogni cartella verrà rinominata con il nome del motore preso in analisi, il tipo di simulazione avviata, l'ampiezza della corrente, e l'angolo di fase della corrente come in figura 3.4

```
%% Avvio delle simulazioni e salvataggi
if  (eval_type == 'singt')
       iStr=num2str(Imod,3); iStr = strrep(iStr,'.','A');
       gammaStr=num2str(gamma,4); gammaStr = strrep(gammaStr,'.','d');
       if isempty(strfind(gammaStr, 'd'))
             gammaStr = [gammaStr 'd'];
       end

       CaseDir = [pathname, filename(1:end-4) '_T_eval_',iStr,'_',gammaStr ...
           '_MN'];
       mkdir(CaseDir);
else
      Idstr=num2str(id,3); Idstr = strrep(Idstr,'.','A');
      Iqstr=num2str(iq,3); Iqstr = strrep(Iqstr,'.','A');

         if isempty(strfind(Idstr, 'A'))
             Idstr = [Idstr 'A'];
         end
         if isempty(strfind(Iqstr, 'A'))
             Iqstr = [Iqstr 'A'];
         end
```

```
21
22          CaseDir = [pathname 'CaseBackup\', filename(1:end-4),'_' Idstr 'x' ...
                Iqstr '_MN'];
23          mkdir(CaseDir );
24  end
```

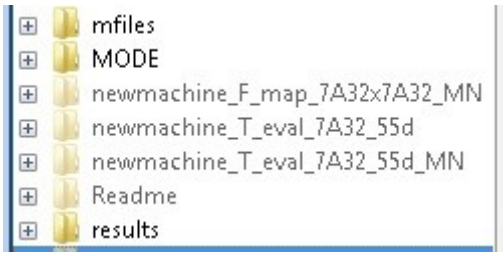


Figura 3.4 Cartella Backup

Si nota che è possibile distinguere la stessa simulazione tra FEMM e MagNet attraverso la sigla finale '_MN nel nome della cartella creata.
Inoltre è anche possibile distingure facilmete il caso riguardante la singola simulazione con il caso in cui si vuole ottenere la mappa dei flussi: rispettivamente *filename_ T_ eval_ iStr_ gammaStr_ MN* per la singola simulazione e *filename_ F_ map_ Idstr'x'Iqstr_ MN* per la mappa dei flussi.
È possibile anche notare attraverso le righe di codice sopra riportati che per il caso 'singt' viene creata la cartella principale in cui si salvano i risultati in file .mn simulato all'interno della funzione *Simulate_ xdeg*, invece per il caso 'singm' si è deciso di creare la cartella principale all'interno della funzione *eval_ FdFq_ tables_ in_ MN* e salvare tutte le macchine simulate per creare la mappa dei flussi dentro la cartella *Backup* in *Simulate_ xdeg*.

Adesso è necessario aprire MagNet, caricare il file .mn creato durante il salvataggio e settare i parametri operazionali: questo richiede l'avvio di ActiveX che permette la comunicazione tra Matlab e MagNet

```
1     MN6 = actxserver('Magnet.application');
2           set(MN6, 'Visible', 1);
3           DocSV = invoke(MN6, 'openDocument',[pathname filename]);
4           Doc = invoke(MN6, 'getDocument');
5           View = invoke(Doc, 'getCurrentView');
```

```
6           Solution = invoke(Doc, 'getSolution');
7           calculator = invoke (MN6, 'getStackCalculator');
```

Si lancia ActiveX, si rende visibile, si apre il file .mn del motore salvato precedentemente. Da questo momento in poi una serie di comandi già esistenti nei vecchi script a me consegnati in *Run_SetParCase.m* impostano i parametri per la risoluzione della simulazione: si parte impostando il transitorio di avviamento a t=0, la mesh, la temperatura d'esercizio, la tipologia di *motion*, gli avvolgimenti, la forma d'onda della corrente e infine si avvia la simulazione tramite il comando *solveTransient2dWithMotion*.
Completata la simulazione è compito dello script chiamato *RunS_PostProcessingMN* leggere i risultati ottenuti da *MagNet*, rielaborarli e caricarli nelle corrette variabili che serviranno alla funzione *simulate_xdeg_MN* la quale le caricherà nelle variabili *SOL*. Durante l'elaborazione dei dati in *RunS_PostProcessingMN* sono sorti dei problemi riguardanti la rappresentazione dei grafici, nello specifico nel confronto tra il grafico ottenuto da FEMM e quello ottenuto da MagNet si notava un'incongruenza.
Difatti sovrapponendoli si notava come il grafico di MagNet (rosso) sia leggermente in anticipo rispetto al grafico di FEMM (blu) (figura 3.6), nonostante l'andamento sia esattamente simmettrico a quello di FEMM preso come metodo di confronto. Questa dissimetria vale sia per la visualizzazione della coppia, IPF medio sia per il flusso in asse d e q.

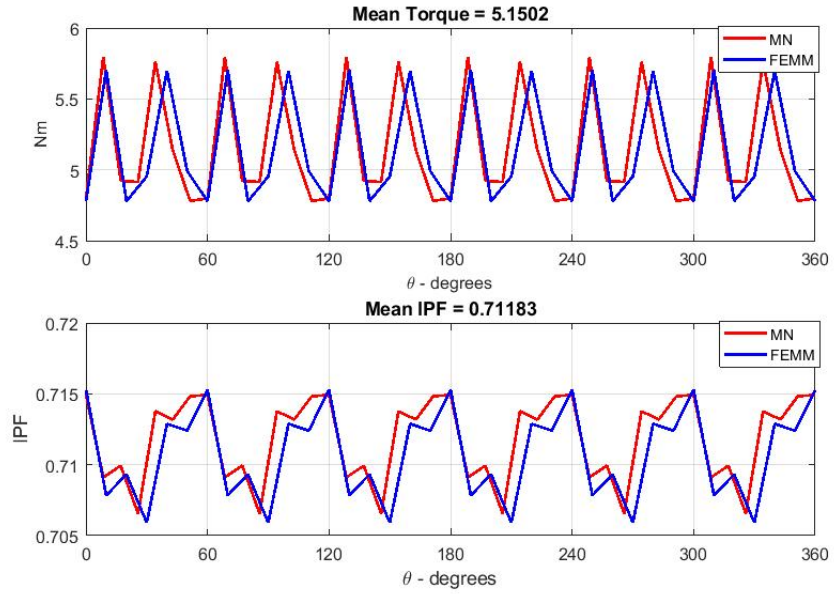


Figura 3.5 [Confronto rosso MN blu FEMM (torque IPF) ERRATO

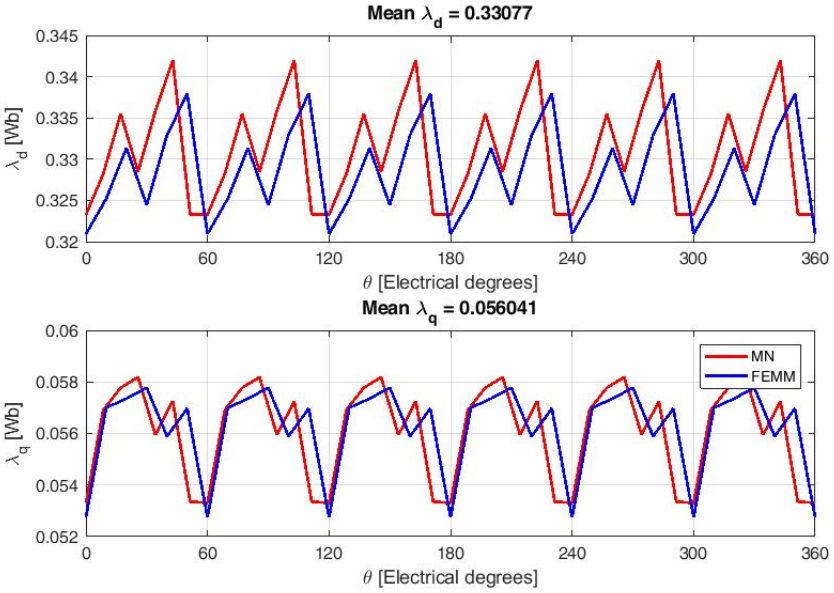


Figura 3.6 Confronto rosso Mn blu FEMM (Fluxd Fluxq) ERRATO

Analizzando il problema si è visto che durante i primi istanti del calcolo della coppia, MagNet stimava un valore diverso rispetto a quello di FEMM, nello specifico si è notato che il vettore *Torque_1sim* risultava sfalsato di una posizione: si presuppone che il motivo

fosse dovuto al tipo dii simulaizone che eseguiva MagNet. Per ovviare a ciò si è pensato di traslare il vettore *Torque_1sim* in modo tale da ottenere la perfetta sovrapposizione dei grafici per dimostrare che la simulazione con MagNet darà gli stessi risultati che si ottengono con FEMM. Di seguito si riportano le righe di codice utilizzato per la traslazione del vettore *Torque_1sim*

```
Torque_1sim = Torque_1sim((2:(end)),:);
Torque_1sim = [Torque_1sim(NTime-1,:);Torque_1sim];
Torque_1sim = Torque_1sim(1:(end-1),:);
```

Naturalmente stesso ragionamento è stato fatto con le variabili *Fluxd, Fluxq*, mentre nulla è stato fatto per la variabile *IPF* in quanto quest'ultima viene ricavata tramite la formula *out.IPF = sin(atan(out.iq./out.id)-atan(out.fq./out.fd))* in cui vi è presente la variabile flusso dq già precedentemente compensata. Si mostra in fig 4.10 i grafici ottenuti dalle modifiche sopra descritte.

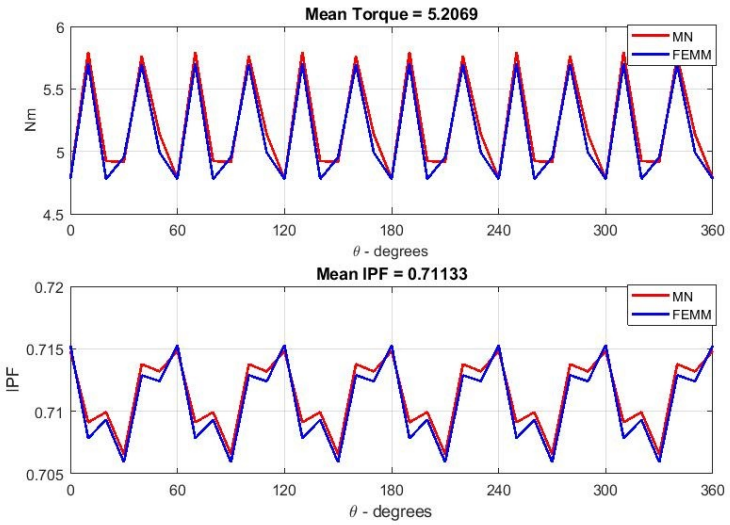


Figura 3.7 [Confronto rosso MN blu FEMM (torque IPF)

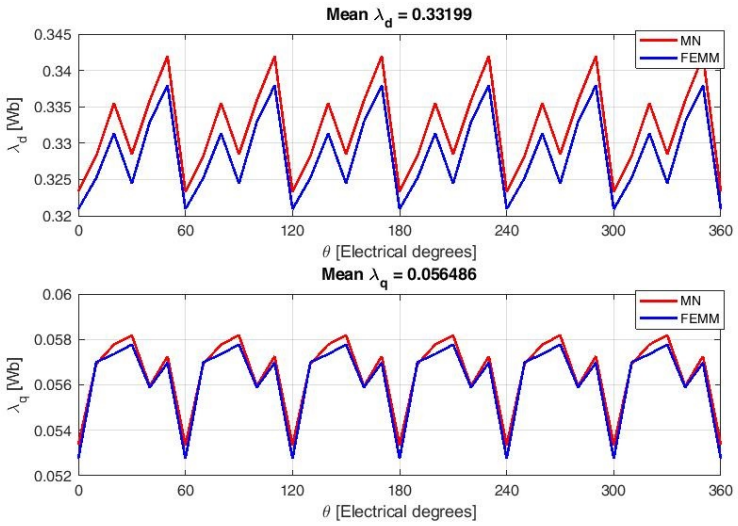


Figura 3.8 Confronto rosso Mn blu FEMM (Fluxd Fluxq)

Capitolo 4

Confronti FEMM vs MN

Dopo aver implementato correttamente il codice e aver testato tutte le funzionalità della simulazione è interessante effettuare dei confronti tra il codice da me creato e quello già esistente in *FEMM*; in linea generale, come è stato detto, la logica di sviluppo è molto simile a quella FEMM, ma in alcune situazioni si sono prese delle decisioni diverse per via di alcune differenze sul funzionamento dei due programmi di simulazione (*MagNet e FEMM*)

4.1 Salvataggio Macchina

Si osservi il *Flow Chart* appositamente creato per evidenziare le differenze sostanziali tra FEMM (a destra) e MagNet (a sinistra) durante la creazione e il salvataggio della macchina:
per capire meglio le operazioni tra i due algoritmi si è scelto di posizionare i blocchetti che svolgono operazioni equivalenti sullo stesso livello, altrimenti su posizioni differenti. Questo aiuta a capir meglio le differenze tra *FEMM e MagNet.*

Dalla GUI SyR-e premendo il tasto Save Machine si chiama la funzione principale *DrawPushMachine* per FEMM e *DrawPushMachine_MN* per MagNet (vedi fig 4.1).

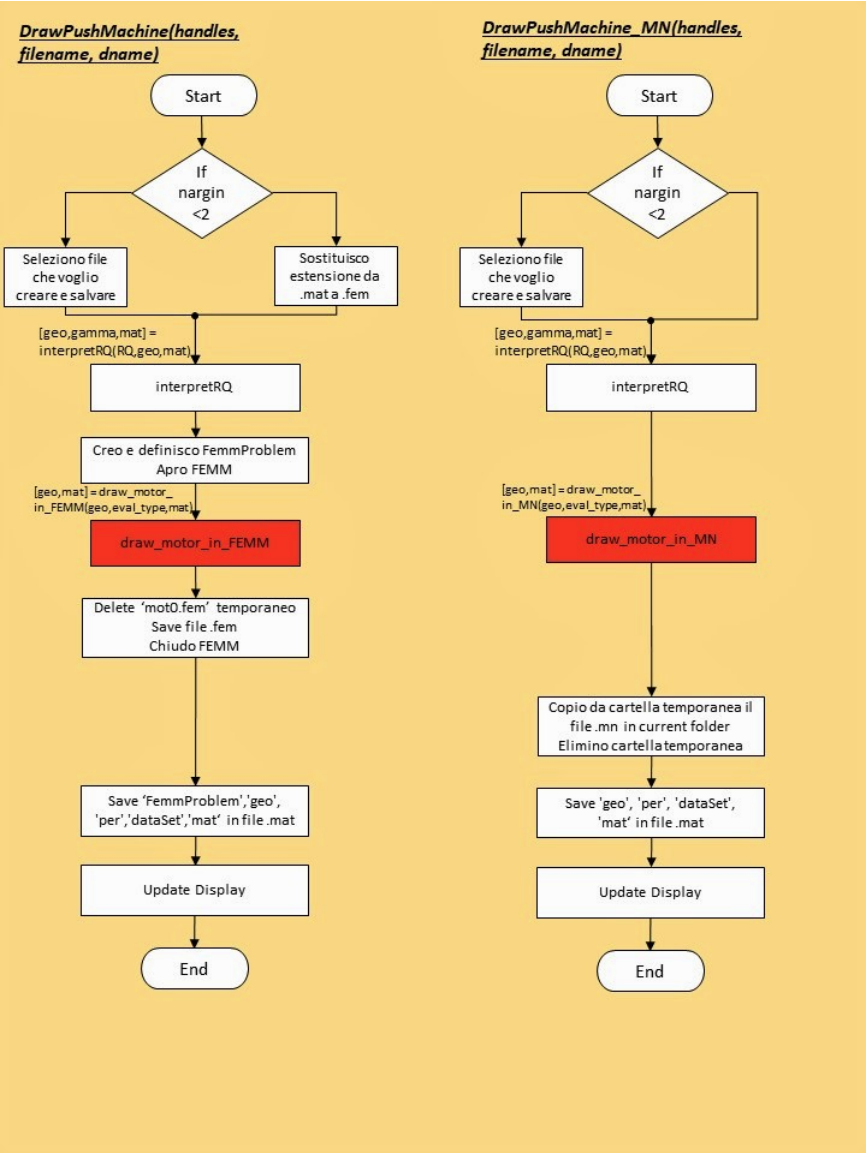


Figura 4.1 Flow Chart DrawPushMachine FEMM vs MN

Si comprende che gran parte delle operazioni sono rimaste invariate: in entrambi i casi le variabili in ingresso sono uguali (handles, filename, dname). Entrati nella funzione si verifica se la condizione (nargin<2) sia vera e si prosegue con la funzione *interpretRQ* che servirà qualora si avviasse un'ottimizzazione, di seguito in FEMM vengono create le variabili *FemmProblem* che serviranno in seguito per settare la risoluzione della simulazione che si andrà ad eseguire (questo in MagNet non è necessario) ed infine si apre

FEMM. Continuando, in entrambi i casi si evoca la funzione *draw_motor_in_FEMM* in FEMM e *draw_motor_in_MN* in MagNet (che analizzeremo in seguito). In MagNet si è costruita quest'ultima funzione in modo da accettare le stesse variabili di FEMM (geo, eval_type, mat). Usciti dalla suddetta funzione si intraprendono due strade differenti per caratteristiche diverse dei programmi: in FEMM si elimina il file temporaneo *mot0.fem* creato dentro la precedente funzione, si salva il file della macchina appena creata e si chiude FEMM, invece su MagNet queste operazione per comodità vengono eseguite dentro la funzione *draw_motor_in_MN*, difatti in MagNet è necessario copiare il file .mn dalla cartella temporanea alla current direct ed eliminare la cartella temporanea (i file .mn creati da MagNet occupano dimensioni maggiori rispetto ai file .femm). Infine per entrambi i casi si salvano le variabili finora usate e si aggiorna la GUI SyR-e.

In figura 4.2 è raffigurato il Flow Chart di confronto tra *draw_motor_in_FEMM e draw_motor_in_MN*: quest'ultima nonostante abbia in ingresso le stesse variabili, presenta differenze significative in quanto per *MagNet* si crea la cartella temporanea per il salvataggio del file temporaneo .dxf durante l'esucuzione della funzione *draw_motor_in_MN*. Analizzando il confronto in entrambi i casi si definisce la dimensione della *mesh*, ma in FEMM è necessario prima creare un file temporaneo *'mot0.fem'*, importare i materiali, procedere nel calcolo del fattore di avvolgimento e offset di rotore. In entrambe le funzioni FEMM e MagNet si eseguono due script *ROTmat e STATmatr* identiche sia per FEMM che per MN: importanti perchè esse creano le matrici di rotore e statore che serviranno per la costruzione della macchina. Da adesso si seguono due strade differenti: in MagNet è necessario creare una cartella temporanea dove salvare il file .dxf. Ciò è effettuato grazie alla funzione *syreToDxf*. Essa crea il file .dxf (Drawing Exchange Format, file di tipo CAD) partendo dal file *mot_0.mat* tramite gli script contenuti al suo interno (raggi, avvolgimenti, magneti) e infine crea la cartella temporanea dove poi si salverà il file appena creato.

Figura 4.2 Flow Chart Draw_motor_in FEMM vs MN

Osservando il flow chart di confronto si può ben notare che ciò non è presente in FEMM, MagNet richiede questo ulteriore passaggio in quanto è necessario caricare il disegno della macchina in versione CAD e dopo associare materiali e variabili utili alla simulazione. Questa è una particolarità del processo di creazione della macchina con MagNet.

Un'altra differenza in MagNet la si riscontra nel processo di caricamento e assegnazione dei

materiali: in FEMM viene tutto gestito dentro la funzione *FEMM_initialize* che in pochi comandi riesce a caricare in FEMM tutti i materiali scelti in *SyRe* con le loro proprietà; invece in MagNet il processo di caricamento e assegnazione avviene in modo più complicato in quanto bisogna prima definire se il materiale scelto in *SyRe* sia già presente in MagNet altrimenti caricarlo come nuovo materiale (script *Codifica Materiali* e poi assegnarlo ai corrispondenti blocchi di rotore, statore e albero (script *assign_block_prop_stat_MN e assign_block_rot_MN*). Un'altra differenza tra FEMM e MagNet la ritroviamo nella creazione delle bobine di statore: in FEMM questa operazione viene svolta all'interno del blocco *assign_block_prop_stat* attraverso un ciclo *for* che utilizzando semplici comandi esistenti in FEMM assegna alle bobine materiale e fase. Mentre in MagNet il tutto viene gestito esternamente grazie allo script *MakeSimpleCoilMagnet_UVW* che al suo interno elabora un serie di passaggi utili per la creazione delle bobine di statore. Infine sia in FEMM che in MagNet si definiscono le boundary conditions al rotore e statore, si costruisce il traferro e si salva il file .fem per FEMM e .mn per MagNet; si chiude l'applicazione MagNet mentre l'operazione di chiusura applicazione in FEMM viene svolta in *draw_motor_in_FEMM*

4.2 Post Processing

Dopo aver illustrato il confronto tra FEMM e MagNet riguardante il salvataggio della macchina è interessante descrivere le differenze sostanziali nel processo di simulazione della macchina; si ricorda che si è cercato di mantenere, dove possibile, la perfetta specularità tra le due funzioni in modo da creare due strade parallele che operano allo stesso modo.

Si parte dalla prima funzione che viene richiamata quando si preme il pulsante *Start* ovvero *post_proc_single_motor* riportata in 4.3. In queste due funzioni non ci sono sostanziali differenze, la logica di procedimento è la stessa eccetto il caso *singt* dove per MagNet non è necessario distinguere se è nececcario calcolare le perdite in quanto la particolarità di MagNet sta proprio nel calcolarle insieme agli altri valori durante la simulazione.

Invece per FEMM è necessario effetuare un check sulla variabile *LossEvaluationCheck* per verificare se è necessario calcolare le perdite entrare: in tal caso si entra nella specifica funzione *FEMMfitnees_IronLoss*, altrimenti si procede avviando la funzione *FEMMfitness*.

Nel caso in analisi ci si è occupati di creare una funzione parallela a *FEMMfitness* che accettasse in ingresso le stesse variabili e mantenesse quanto più possibile le stesse operazioni: la funzione ha preso il nome di *MNfitness*, illustrata nel flow chart in 4.4.

Osservando il flow chart si può affermare che le due funzioni sono identiche, con la differenza che in MagNet si è deciso di non rendere disponibile momentaneamente l'avvio dell'ottimizzione e del calcolo della funzione di costo. Scelta giustificata dal fatto che per poter avviare più simulazioni in parallelo con MagNet è necessario essere in possesso di più licenze ed avere alte prestazioni hardware.

Procedendo nell'analisi sul confronto FEMM e MagNet si analizza l'ultima funzione chiamata *simulate_xdeg*: è la funzione principale dove si svolgono le operazione per l'avvio del calcolo della simulazione. Come tutte le funzioni, le variabili d'ingresso sono identiche; come mostrato dal flow chart in fig 4.5 in entrambi i casi si analizza la variabile *eval_type* con un'unica differenza che in MagNet non è ancora possibile proseguire con simulazioni di tipo *MO_OA e MO_GA*. Compreso che le uniche tipologie di simulazioni disponibili in MagNet sono di tipo *singt e singm* si procede con l'impostazione e la creazione dei paramentri utili per la simualzione come ad esempio Br, Idq, Imod, gamma e altri parametri sulla geometria di rotore e statore. In MagNet in base al tipo di simulazione si crea la cartella identificativa: nello specifico quando si richiede la simulazione per la costruzione della mappa dei flussi si crea una cartella di backup in cui si salvano tutti i punti di lavoro della simulazione per ogni punto della mappa con la determinata Id e Iq. Il motivo principale del salvataggio è dato dal fatto che ogni punto di lavoro della mappa dei flussi ha una dimensione importante e conviene salvarlo in modo da essere disponibile se fosse richiesto. Da qui in avanti troviamo lo stesso procedimento logico tra FEMM e MagNet: in MagNet si eseguono i comandi che richiamano ActiveX, utili

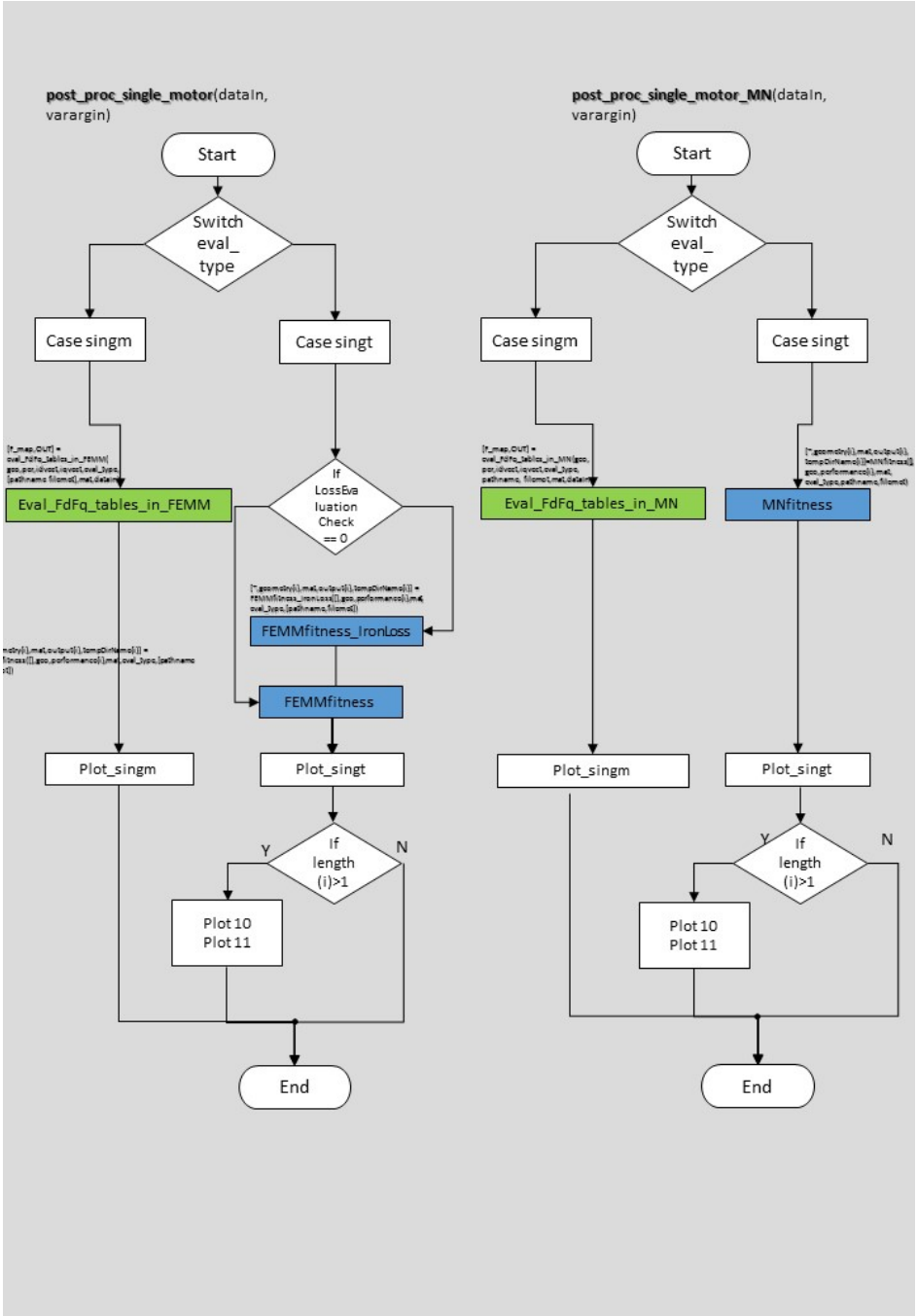


Figura 4.3 Flow Chart post_proc_single_motor FEMM vs MN

per lanciare tutti i comandi da Matlab per MagNet come aprire, caricare il file macchina, impostare i parmametri e avviare la simulazione. In FEMM invece è più semplice: Matlab è compatibile per l'esecuzione dei comadi per operare in FEMM. Avviata e completata la

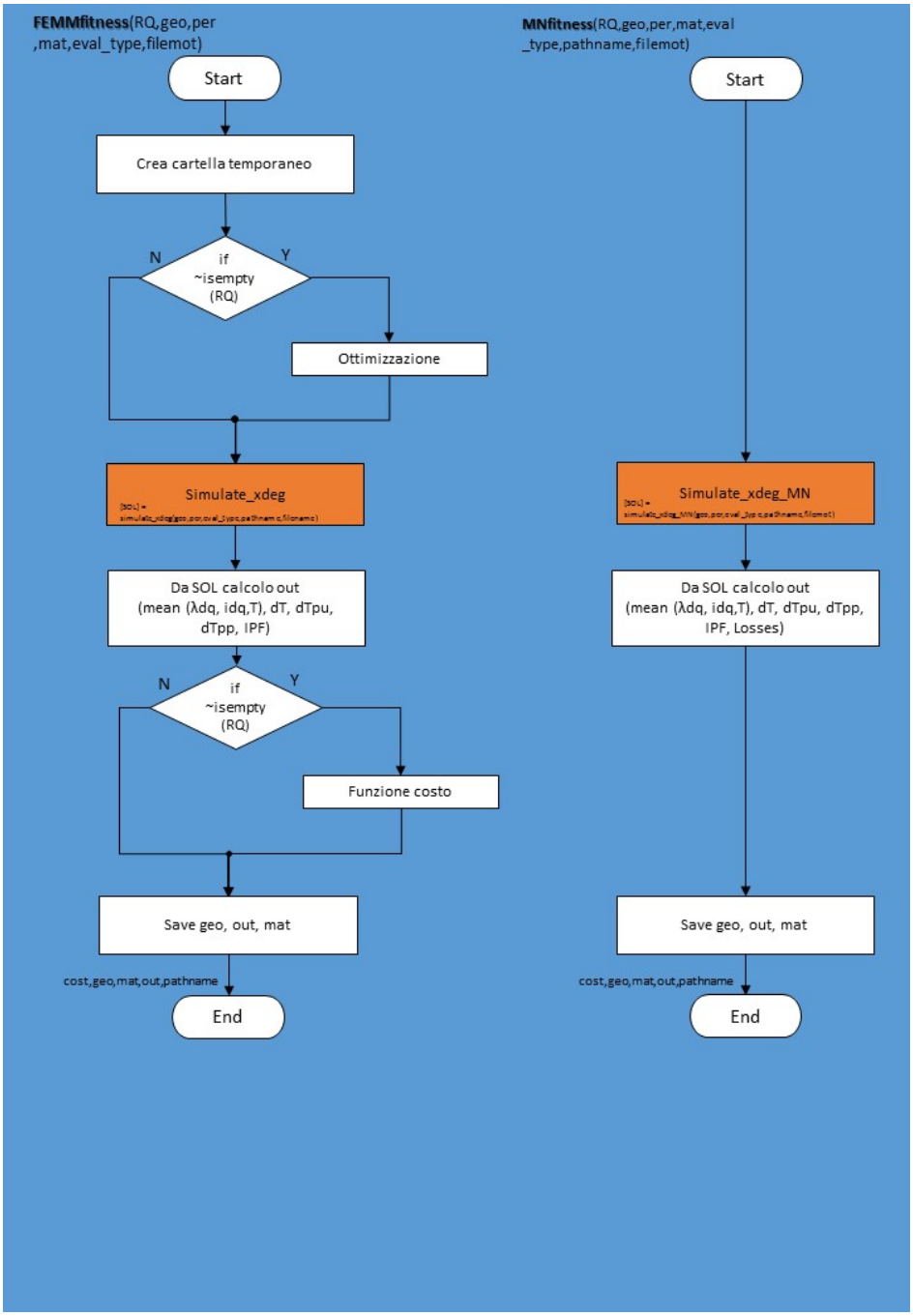


Figura 4.4 Flow Chart FEMMfitness vs MNfitnees

simulazione i valori ottenuti dal programma di simulazione (FEEM e MagNet) vengono letti ed elaborati dalla funzione *Post_proc* per FEMM e *RunS_PostProcessing_MN* per MagNet; esse oltre a leggere i valori calcolati dal programma, li rielaborano e li salvano

nelle apposite variabili. Nello specifico le variabili in uscita sono il flusso d e q, la coppia e le perdite qualora la simulazione lo richieda. Osservando la 4.5 si nota chiaramente che il processo di salvataggio è lo stesso in entrambe le simulazione, lo stesso vale per il caricamento dei risultati nella struttura rinominata SOL. Infine si chiude il programma di simulazione e si continua nelle funzioni superiori.

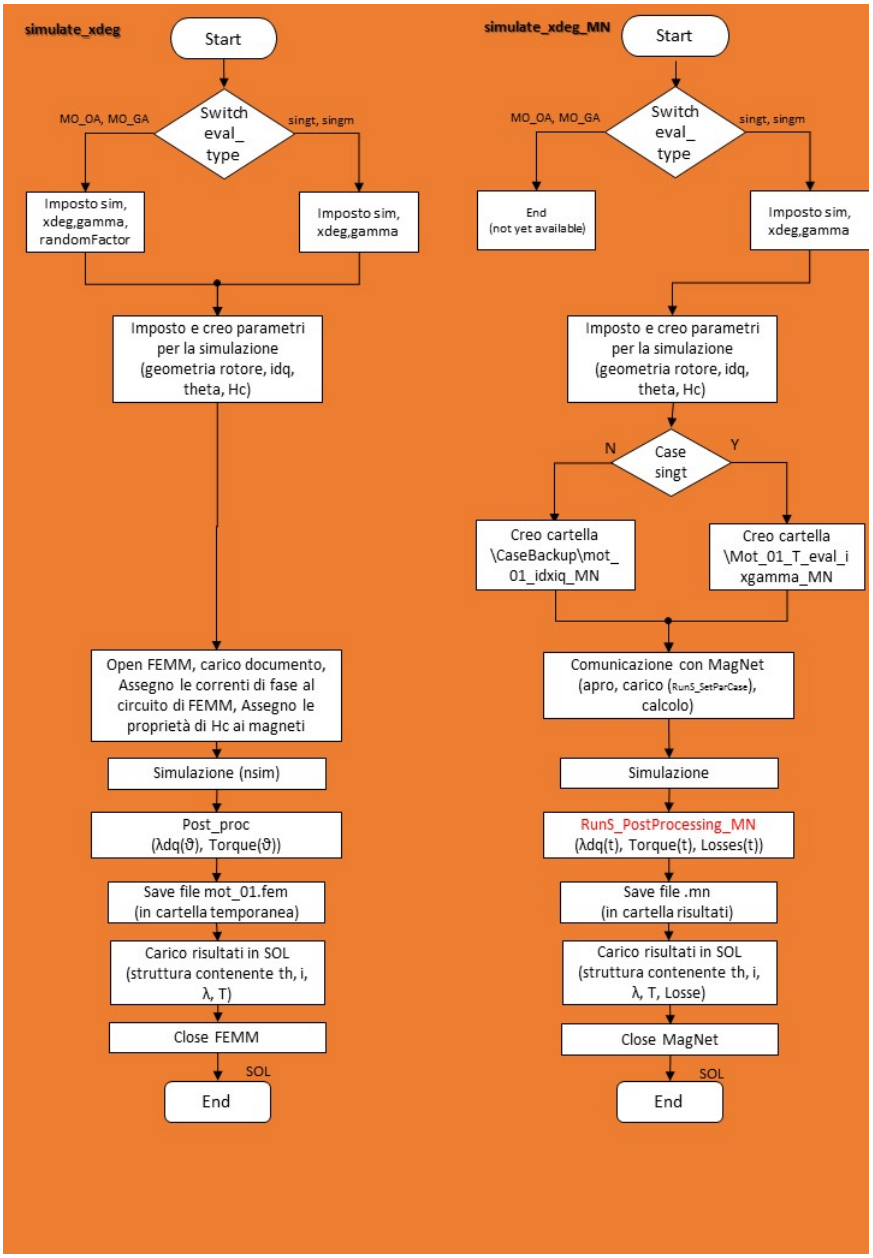


Figura 4.5 Flow Chart simulate_xdeg vs simulate_xdeg_MN

4.3 Avvio Simulazione con MN

Il processo di simulazione con MagNet, come già descritto nei paragrafi precedenti, è molto semplice. Per avviare la simulazione di una macchina è sufficente avviare *GUI_SyRe* da Matlab, selezionare la macchina con cui si vuole avviare la simulazione cioè caricare il file .mat che abbiamo in possesso. La macchina verrà caricata e apparirà un'anteprima della sezione rotore-statore come in figura 4.6

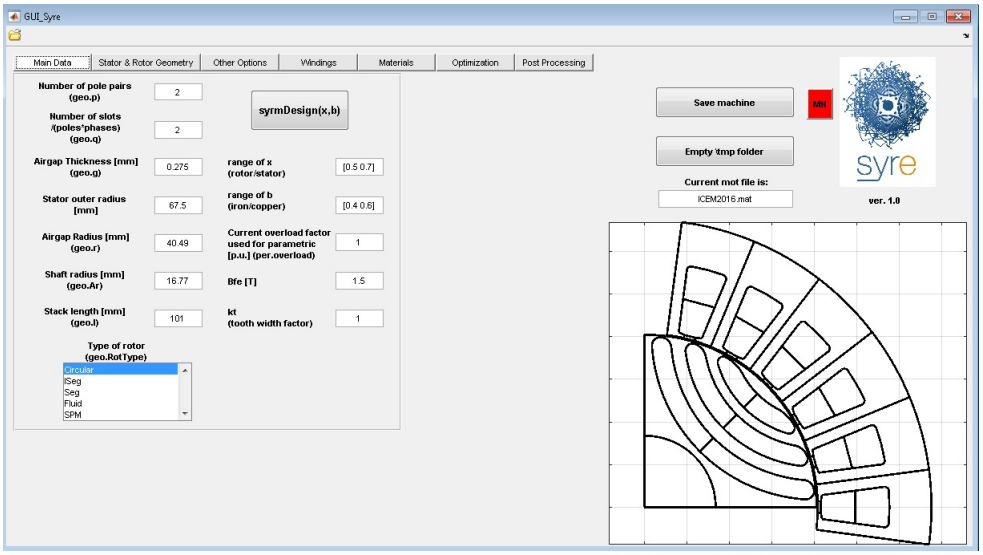


Figura 4.6 Finestra Post Processing

Adesso, se necessario, possiamo modificare i parametri geometrici della macchina entrando nelle altre finestre affianco: qui come già descritto nei capitoli precedenti possiamo modificare oltre alla geometria di rotore e statore anche i materiali e gli avvolgimenti. Dopo aver modificato e salvato la macchina a nostro interesse possiamo entrare nella finestra *Post Processing* mostrata in figura 4.9.

Qui è necessario impostare i parametri necessari per avviare la simulazione: si trovano già impostati alcuni parametri predefiniti salvati durante la creazione della macchina. Fatto ciò si può procedere all'avvio della simulazione con MagNet premendo l'apposito

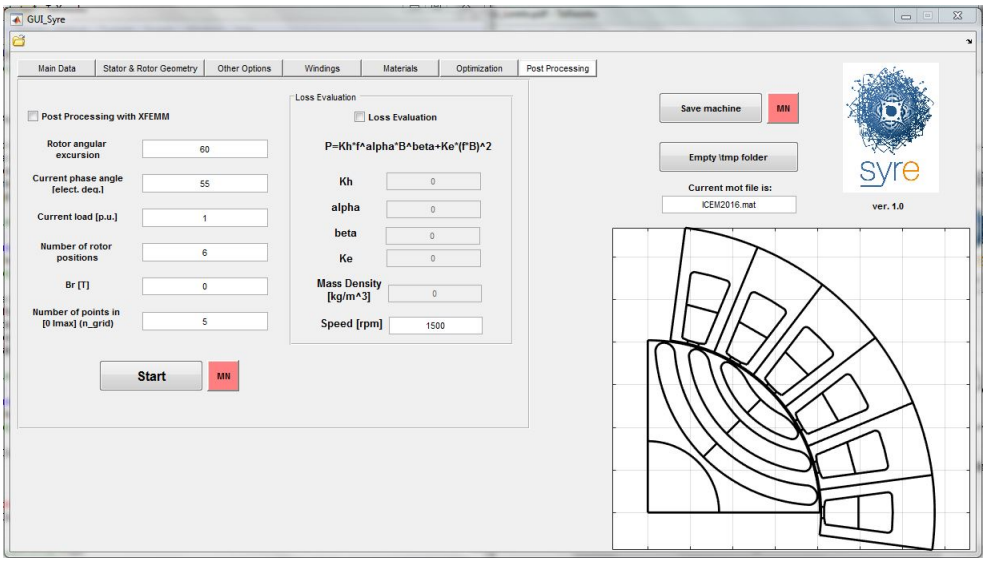


Figura 4.7 Finestra salvataggio macchina

pulsante rosso MN: si richiameranno tutte le funzioni di simulazione già elencate; il risultato grafico sarà l'apertura di MagNet e la visualizzazione dell'avanzamento del processo di simulazione mostrato in 4.8

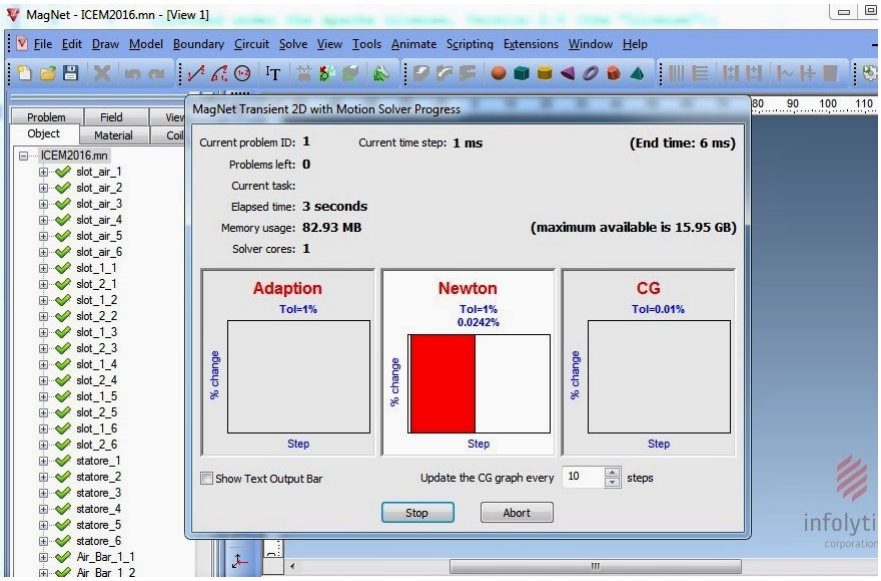


Figura 4.8 Simulazione con MagNet

Completa la simulazione in MagNet si attendono alcuni instanti in modo che SyRe elabori i risultati importati e li stampi negli appositi grafici che si analizzeranno nel

paragrafo successivo.

L'ultima azione della simulaizone in Syre sarà il salvataggio di backup di tutti i dati ottenuti durante la simulazione della macchina scelta, compresi i grafici.

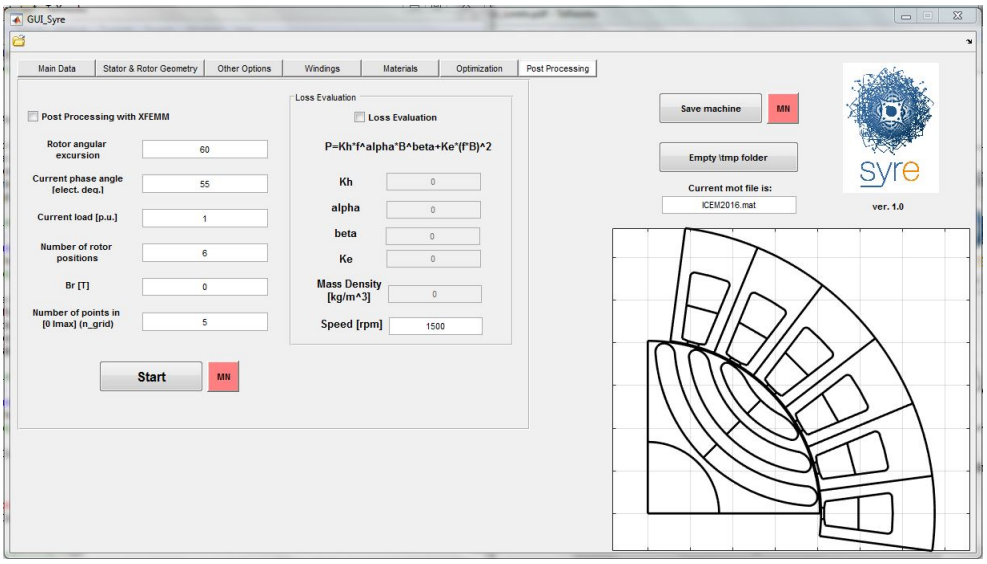
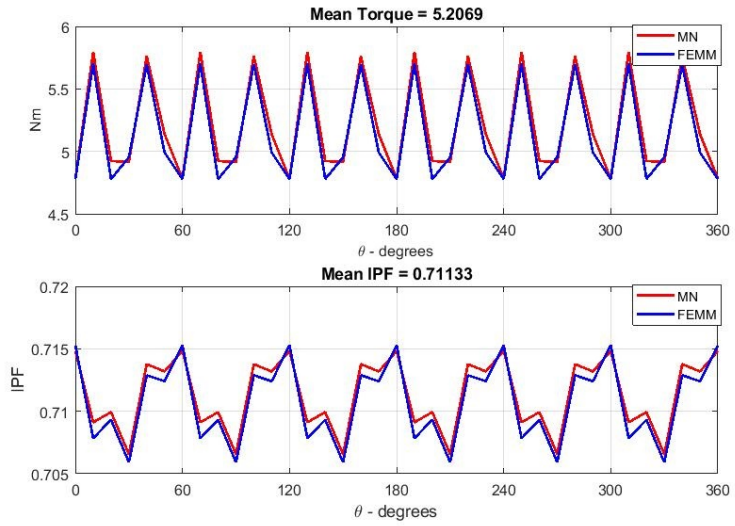


Figura 4.9 Finestra simulazione macchina

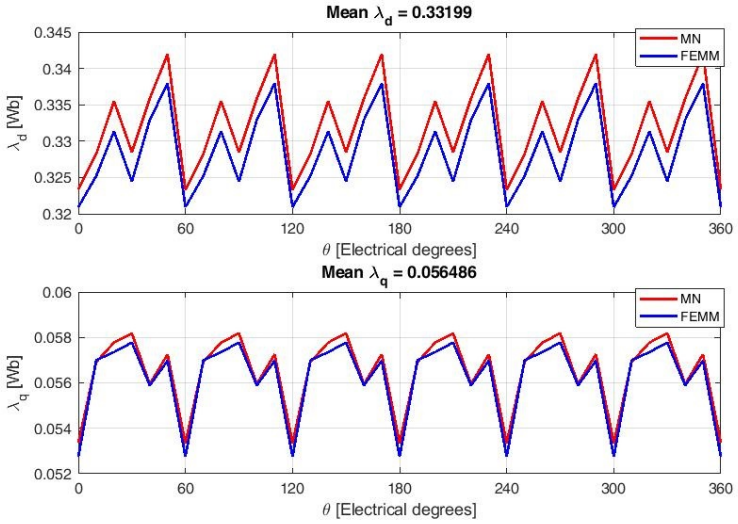
4.4 Confronto Risultati Ottenuti

In figura 4.10 è riportato il confronto tra FEMM e MagNet dei grafici ottenuti a simulazione completata. Quest'ultimi sono stati realizzanti simulando la macchina ICEM2016 e mantenendo costante tutte le variabili geometriche e i parametri di simulazione.

Analizzando i grafici si può ben notare come globalmente sono coincidenti: nello specifico la rappresentazione della coppia in un periodo intero di 360° risulta coincidente a meno di un piccolo tratto in corrispondenza dei 30° e i multipli di 180°. Ciò è dovuto proprio al programma di simulazione, FEMM e MagNet hanno caratteristiche molto diverse nel processo di simulazione della macchina ciò ha portato a lievi differenze nei punti 'critici' (ricordiamo che stiamo parlando di variazione dell'ordine di 2-3 decimi di newton il valore medio è praticamente identico). Con lo stesso ragionamento possiamo attribuire le variazioni del IPF e del flusso nell'asse d e q. Inoltre, come descritto nel capito 4, se



(a)



(b)

Figura 4.10 Confronto grafici (rosso Mn blu FEMM (Fluxd Fluxq))

si vuole calcolare le perdite della macchina è sufficente impostare l'escursione angolare rotorica pari a 360°: così facendo verrà visualizzato il grafico delle perdite dei magneti, delle perdite di isteresi e del rame nel rotore e statore (figura 4.11).

Non è stato possibile effetturare un confronto tra i due programmi di simulazione nel calcolo delle perdite perchè in FEMM l'algoritimo risulta alquanto instabile e poco preciso.

Si ricorda che l'obbiettivo della tesi è stato l'implementazione di MagNet in SyR-e proprio per il calcolo delle perdite in modo dettagliato e stabile.

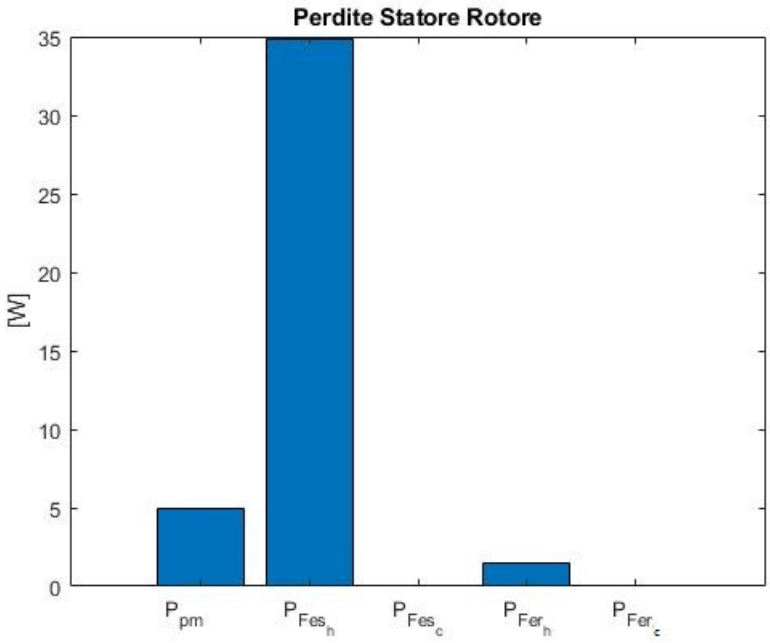


Figura 4.11 Grafico perdite della macchina

Capitolo 5

Conclusioni

Questo lavoro ha aiutato ad apportare miglioramenti in un programma di design già esistente, permettendo la simulazione con un altro programma: *MagNet*.
Gli obiettivi principali raggiunti sono:

- avviare la simulazione con MagNet in SyR-eottenendo gli stessi risultati di FEMM.
- calcolare le perdite della macchina in modo semplice, veloce e preciso.
- aggiornare la grafica GUI SyRe per avviare la simulazione con MagNet.
- importare nuovi materiali da SyRe a MagNet (punto non del tutto compatibile).

I possibili sviluppi futuri possono essere:

- rendere completamente compatibile l'import di nuovi materiali (al momento non è possibile importare i materiali non lineari).
- rendere compatibile la simulazione con tutte le geometrie rotoriche.

Bibliografia

[1] G. Pellegrino, A. Vagati, P. Guglielmi e B. Boazzo, "Performance Comparison Between Surface-Mounted and Interior PM Motor Drives for Electric Vehicle Application", IEEE Transactions on Industrial Electronics, vol. 59, n. 2, pp. 803–811, feb. 2012, issn: 0278-0046. doi: 10 . 1109 / TIE . 2011 . 2151825.

[2] Francesco Cupertino, Gianmario Pellegrino, Politecnico di Torino (Dispense *User's Manual SyR-e* https://sourceforge.net/projects/syr-e/).

[3] Software SyR-e https://sourceforge.net/projects/syr-e/

[4] Software MagNet https://www.mentor.com/products/mechanical/magnet/magnet/ .

[5] Software FEMM https://sourceforge.net/projects/xfemm/

[6] R. Storn e K. Price, "Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces", vol. 11, pp. 341–359, gen. 1997.

[7] Simone Adamo, Tesi di Laurea Magistrale: 'Progettazione di motore elettrci multifase con software open source' (Politecnico di Torino, 2018).