POLITECNICO DI TORINO

Laurea Magistrale in Ingegneria Elettrica
e-Mobility and Smart Grids

Tesi di Laurea Magistrale

Procedura Automatizzata di Calcolo Elettromagnetico e Strutturale di Motori Elettrici da Trazione

Relatore
Prof. Gianmario Pellegrino
Correlatori
Prof. Simone Ferrari
Dott. Cesare Tozzo

Candidato
Nicholas Zanzarelli

Anno Accademico 2023 - 2024

Sommario

La progettazione di motori elettrici rappresenta un campo di studio fondamentale per l'ingegneria elettrica, con applicazioni che spaziano dall'automotive all'industria manifatturiera. La sfida principale in questo ambito è integrare la fase di progettazione preliminare con simulazioni avanzate basate sull'analisi agli elementi finiti (FEA).

SyR-e si distingue come un valido strumento per la progettazione di motori elettrici, mentre COMSOL Multiphysics offre un ambiente di simulazione multiphysics robusto e versatile, capace di modellare una vasta gamma di fenomeni fisici.

La necessità di un'integrazione tra questi strumenti è motivata dall'obiettivo di sfruttare le capacità di ciascun software per ottenere un processo di progettazione e simulazione più completo ed efficiente.

In questo contesto, è essenziale considerare il panorama dei software FEA disponibili sul mercato. Numerose piattaforme, come ANSYS, Abaqus e COMSOL, offrono un'ampia gamma di funzionalità per la simulazione. Tuttavia, spesso sono specializzate in specifici settori industriali o fenomeni fisici, e possono richiedere una grossa mole di lavoro per essere implementate in applicazioni specifiche.

Tra i diversi software presenti sul mercato, COMSOL Multiphysics si distingue come una soluzione versatile e adatta a una vasta gamma di applicazioni. In più, la sua capacità di modellare fenomeni fisici multipli in un'unica piattaforma lo rende particolarmente adatto per l'integrazione con SyR-e.

Questa tesi si propone di colmare il gap tra la progettazione preliminare offerta da SyR-e e le avanzate capacità di simulazione di COMSOL. L'integrazione di COMSOL nella toolchain di SyR-e non solo fornisce un potente strumento di simulazione magnetica, ma soprattutto apre la strada alla simulazione multifisica, a cominciare dall'integrazione della simulazione strutturale con quella elettromagnetica.

Il metodo utilizzato in questa ricerca consiste nello sviluppo di uno script in MATLAB, in linea con lo stile di programmazione di SyR-e. Il processo prevede l'export dei dati del modello progettato in SyR-e verso COMSOL, l'esecuzione di simulazioni elettromagnetiche e strutturali in COMSOL, e infine l'importazione e la visualizzazione dei risultati all'interno di SyR-e. Questo flusso di lavoro integrato è stato implementato sfruttando le capacità dell'ambiente *LiveLink for* MATLAB di COMSOL, garantendo una comunicazione fluida e bidirezionale tra i due software.

Infine, l'interfaccia è stata testata su un caso studio, consentendo di validare l'integrazione e di identificare eventuali miglioramenti necessari. Inoltre, il tool è stato sottoposto a debugging attraverso l'applicazione di numerose modifiche parametriche per garantire

robustezza e flessibilità. I risultati ottenuti hanno dimostrato l'efficacia dell'integrazione, con una significativa riduzione dei tempi di progettazione e una maggiore accuratezza delle simulazioni.

In conclusione, la ricerca ha dimostrato che l'integrazione di COMSOL nella toolchain di SyR-e rappresenta un avanzamento significativo nel campo della progettazione dei motori elettrici, fornendo un approccio più completo e multifisico che migliora la qualità e l'efficienza del processo progettuale.

Indice

Elenco delle figure

Elenco delle tabelle

1 Introduzione

I motori elettrici sono dispositivi fondamentali in diverse applicazioni, capaci di convertire l'energia elettrica in energia meccanica o viceversa. Queste macchine rotanti sono presenti in molteplici contesti, dalle applicazioni domestiche e industriali, quali gli elettrodomestici di casa e i grandi impianti di produzione, ai settori dei veicoli elettrici e ibridi, fino alle grandi navi. I motori elettrici si suddividono principalmente in due categorie: motori a corrente continua (DC) e motori a corrente alternata (AC), con ulteriori suddivisioni come i motori asincroni (o a induzione) e i motori sincroni.

Il funzionamento di queste macchine elettriche rotanti è governato da una serie di fenomeni complessi che devono essere attentamente osservati e ottimizzati. I principali sono:

- **Campi Elettromagnetici**: Alla base del funzionamento delle macchine elettriche c'è la generazione e l'interazione dei campi magnetici e dei flussi di corrente. Questi campi devono essere progettati in modo efficiente per massimizzare la conversione di energia e minimizzare le perdite.

- **Effetti Termici**: Durante il funzionamento, le macchine elettriche generano calore a causa delle perdite elettriche e magnetiche. La gestione del calore è cruciale per evitare il surriscaldamento che può danneggiare i componenti interni e ridurre la durata della macchina.

- **Sollecitazioni Meccaniche**: Le forze generate dalla rotazione e dalle vibrazioni possono influire sulla struttura della macchina. È fondamentale progettare componenti meccanici robusti per garantire un funzionamento sicuro e affidabile nel tempo.

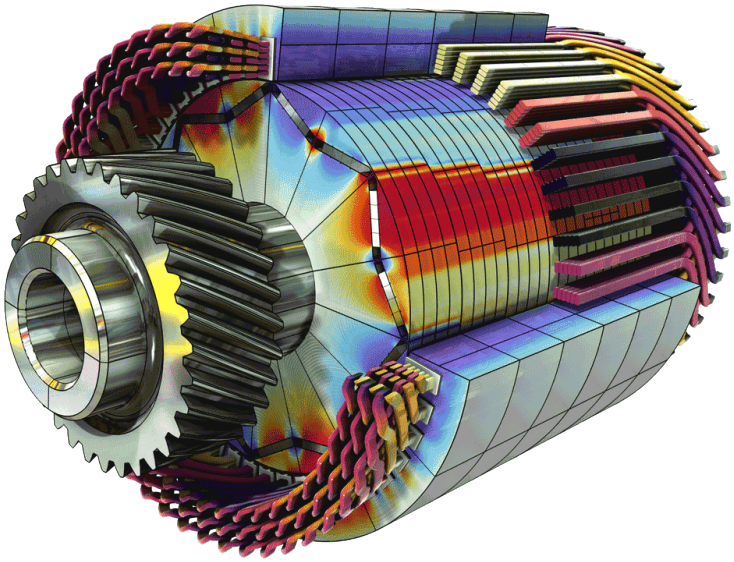

Figura 1.1. Analisi FEM 3D su motore V-shape [4]

Per affrontare la complessità di questi fenomeni, l'analisi agli elementi finiti (Finite Element Analysis, FEA) si rivela uno strumento indispensabile. La FEA consente di simulare in modo dettagliato il comportamento delle macchine elettriche rotanti, permettendo di prevedere e ottimizzare le loro prestazioni prima della costruzione effettiva. I principali benefici di questa analisi sono:

- **Accuratezza e Dettaglio**: La FEA permette di analizzare in dettaglio i campi elettromagnetici, termici e meccanici all'interno della macchina. Questo aiuta a identificare eventuali punti deboli o problematiche di progettazione che potrebbero influire sulle prestazioni.

- **Ottimizzazione delle Prestazioni**: Grazie alle simulazioni precise offerte dalla FEA, è possibile ottimizzare la progettazione della macchina per migliorare l'efficienza e ridurre le perdite. Si possono testare diverse configurazioni e materiali senza la necessità di costruire costosi prototipi fisici.

- **Riduzione dei Costi di Sviluppo**: La capacità di risolvere problemi potenziali in fase di progettazione riduce la necessità di numerosi test fisici, accelerando il processo di sviluppo e riducendo i costi complessivi.

- **Analisi del Comportamento Dinamico**: La FEA consente di studiare come la macchina risponde a diverse condizioni operative, comprese le condizioni di carico e sovraccarico. Questo è essenziale per garantire che la macchina funzioni correttamente in tutte le situazioni previste.
- **Valutazione della Durabilità**: Analizzando le sollecitazioni meccaniche e termiche, la FEA aiuta a prevedere la durata e l'affidabilità della macchina nel tempo. Questo è cruciale per progettare macchine che possano operare per lunghi periodi senza necessità di manutenzione frequente.

In conclusione, le macchine elettriche rotanti sono essenziali per molte applicazioni moderne, e la comprensione dettagliata dei fenomeni che le governano è fondamentale per migliorare le loro prestazioni. L'analisi agli elementi finiti fornisce un approccio potente e dettagliato per la progettazione e l'ottimizzazione di queste macchine, permettendo di sviluppare dispositivi più efficienti, affidabili e duraturi. Grazie alla FEA, si possono creare soluzioni avanzate che rispondono alle crescenti esigenze di efficienza energetica e sostenibilità.

1.1 SyR-e

Figura 1.2. Logo SyR-e [7]

SyR-e, acronimo di Synchronous Reluctance - evolution, è un software open-source sviluppato in collaborazione tra il Politecnico di Torino e il Politecnico di Bari per la progettazione di macchine elettriche rotanti, in particolare macchine sincrone. Utilizza analisi

agli elementi finiti ed algoritmi di ottimizzazione multi-obiettivo [3].

SyR-e è sviluppato in ambiente MATLAB ed è utilizzabile tramite una Graphical User Interface (GUI) che permette di effettuare modifiche a livello geometrico e di intervenire su una vasta gamma di opzioni relative agli aspetti termici, elettrici, magnetici e meccanici del modello di motore in analisi.

Il software utilizza il client gratuito FEMM per eseguire simulazioni magnetiche. I risultati ottenuti da queste simulazioni vengono importati in MATLAB per ulteriori elaborazioni, come illustrato in Figura 1.3.

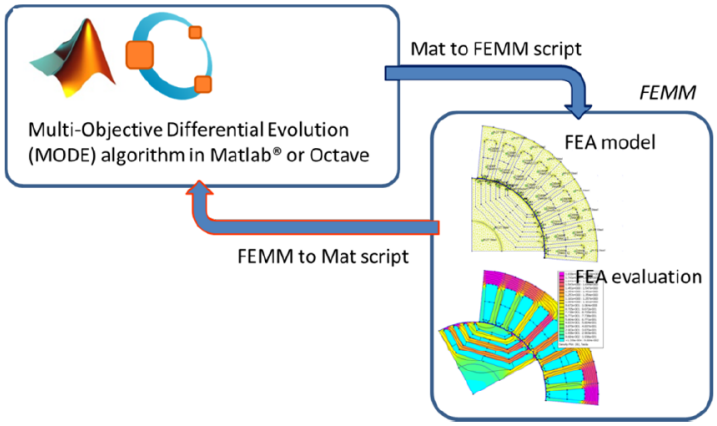

Figura 1.3. Flusso di dati tra SyR-e e FEMM [1]

In MATLAB, è possibile attivare la funzione di Parallel Computing, che consente di eseguire simultaneamente più simulazioni FEA (Finite Element Analysis). Il numero di simulazioni che possono essere eseguite in parallelo dipende dal numero di core del processore del computer in uso [1]. Grazie a questa funzionalità, è possibile ridurre significativamente il tempo necessario per eseguire alcune simulazioni, come quelle relative a ottimizzazioni o identificazioni magnetiche.

1.1.1 Interfaccia Grafica

Per progettare un motore elettrico con SyR-e, è necessario avviare la Graphical User Interface (GUI) utilizzando il comando `GUI_Syre`. Questa GUI (Figura 1.4) è suddivisa in 8 schede, ciascuna dedicata a operazioni specifiche (che verranno descritte in seguito). All'avvio della GUI, viene caricato un motore predefinito denominato `syreDefaultMotor` [1].

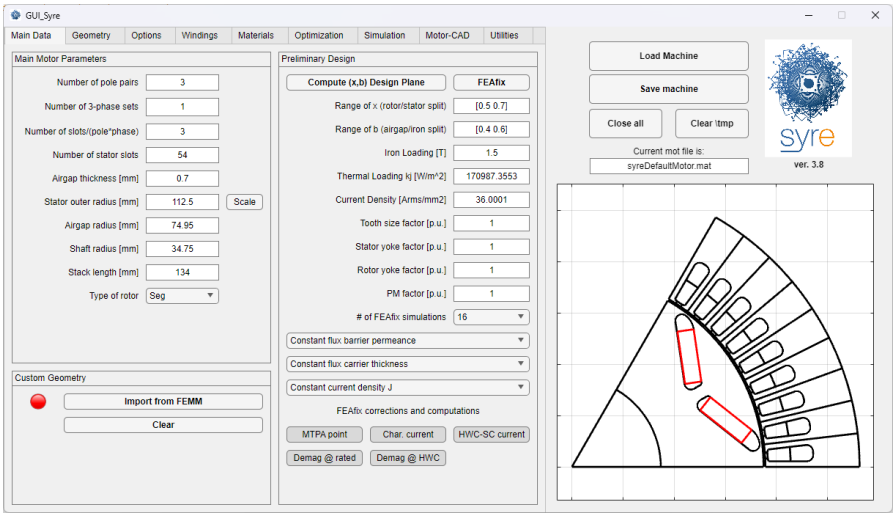

Figura 1.4. Schermata iniziale SyR-e [7]

Di seguito viene riportata una breve descrizione delle finestre presenti nella GUI:

- **Main Data**
  La sezione dedicata all'impostazione dei parametri fondamentali del motore (Figura 1.4).

  Qui, si possono definire input fondamentali per la corretta definizione del progetto, come il tipo di rotore, che definisce, di fatto, la tipologia di macchina che si vuole realizzare. Inoltre, in questa sezione è possibile definire parametri come il numero di coppie polari, il numero di cave, il numero di cave polo fase ma anche la densità di flusso così come la corrente, per cominciare a definire volumi e sfruttamenti della macchina. Infine, sul lato destro della finestra (che rimane visibile in ogni sezione), si può osservare il motore prendere forma attraverso una figura che rappresenta la sezione di un polo della macchina in progetto.

- **Geometry**
  La sezione dedicata alla definizione delle geometrie del motore.

  Attraverso questa finestra è possibile definire i parametri che determinano le geometrie di rotore e di statore. Inoltre, in basso è presente una finestra specifica per inserire ulteriori parametri di rotore nel caso si stia progettando un motore ad induzione.

- **Options**
  La sezione dedicata ai parametri di natura non elettromeccanica del motore.

In questa sezione è possibile definire:

- *Parametri Termici*

  Definendo il fattore di carico termico k_j attraverso dei calcoli interni, il programma è in grado di determinare 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 sulla disposizione degli avvolgimenti, come spiegato in [1]. Il modello termico permette, inoltre, di stimare la temperatura degli avvolgimenti conoscendo la temperatura della carcassa.

- *Parametri Strutturali*

  La velocità massima di rotazione, indicata nel riquadro *Overspeed*, viene utilizzata per dimensionare i ponticelli radiali, tenendo in considerazione esclusivamente gli sforzi di tipo centrifugo. Il parametro *Minimum mech. tolerance* consente di impostare lo spessore dei ponticelli tangenziali. Inoltre, il modello strutturale permette di progettare un rotore con sleeve tramite l'apposito tool *Sleeve Designer* [1].

- *Progettazione dei Ponticelli di Flusso*

  Definendo le dimensioni dei ponticelli è possibile progettare forme e dimensioni delle barriere di flusso. Le impostazioni modificabili di default sono solo quelle tangenziali, mentre quelle radiali vengono calcolate di conseguenza. All'occorrenza è possibile modificare manualmente le impostazioni attraverso i tool *Manual tan ribs* e *Manual rad ribs*.

- **Windings**
  La sezione dedicata all'impostazione dei parametri che definiscono l'avvolgimento del motore.

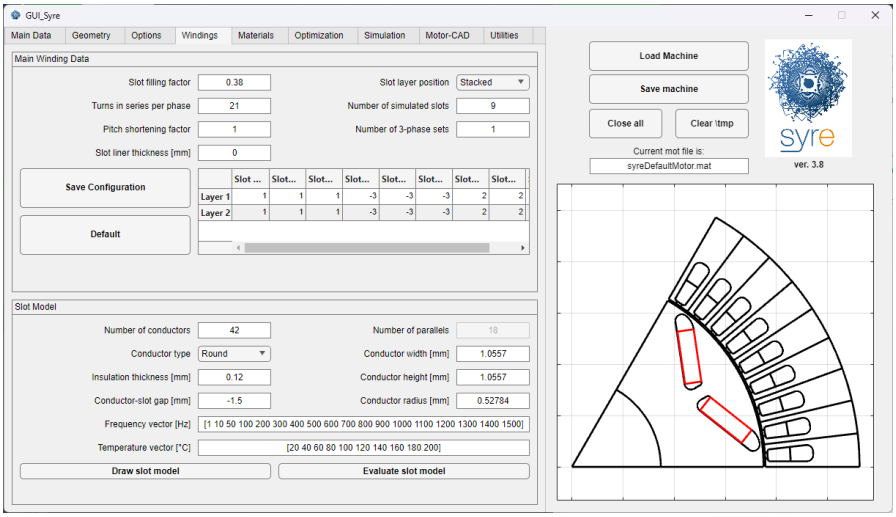

Figura 1.5. Schermata windings SyR-e [7]

Attraverso questa finestra è possibile impostare parametri e geometrie fondamentali. Nella parte superiore, si impostano i dati principali dell'avvolgimento, come: il numero di spire in serie per fase, il numero di cave simulate, la tabella che definisce la distribuzione delle fasi nei diversi layer.
Nella parte inferiore, si definisce la cava della macchina attraverso la definizione delle geometrie dei conduttori e dell'isolante. Inoltre, è possibile impostare due vettori, *Frequency vector* e *Temperature vector*, utili per i calcoli relativi alle perdite [1].

- **Materials**
  La sezione dedicata alle caratteristiche dei materiali della macchina.

  Questa sezione è dedicata all'impostazione dei materiali con cui si vuole realizzare la macchina. La schermata è divisa in tre parti:

  – *Material Data*
    Attraverso cui si assegnano i materiali alle diverse parti del motore: nucleo di statore, cava di statore, nucleo di rotore, barriera di flusso, cava di rotore (nel caso di motore a induzione), albero, sleeve (se si sceglie di realizzare un motore con rotore *wrapped*). Inoltre, in questa finestra è possibile visualizzare la massa e l'inerzia delle parti attive della macchina, una volta scelti tutti i materiali SyR-e ne calcola le masse in base alle dimensioni geometriche scelte in precedenza.

– *Material Library*
Qui si trova il database dei materiali di SyR-e, completamente personalizzabile dall'utente. Infatti, è possibile aggiungere e rimuovere materiali, attraverso gli appositi pulsanti, i materiali nella library vengono divisi per funzione, come possibile notare in Figura 1.7. Infine, nella Figura 1.6 è mostrato un esempio di materiale caricato in libreria. Tale libreria è più limitata rispetto a quelle presenti nei software commerciali come COMSOL, poiché contiene una quantità ridotta di materiali. Inoltre, i dati delle curve B-H forniti dai costruttori spesso riportano valori di campo magnetico limitati, rendendo necessario estendere la curva attraverso approssimazioni numeriche.

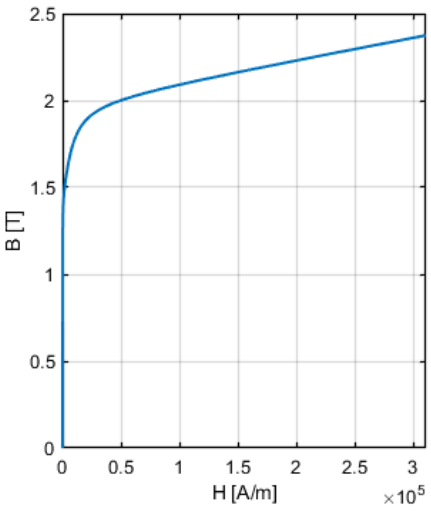

Figura 1.6. Curva B-H del ferro M270-35A [7]

– *Permanent Magnet*
Questa finestra è dedicata alla progettazione dei magneti permanenti dei motori PM-SyR e V-type, con l'obiettivo di ottenere una corrente caratteristica definita. SyR-e utilizza una procedura iterativa FEA [2], una volta inserite manualmente le dimensioni dei PM desiderate o impostate in automatico.

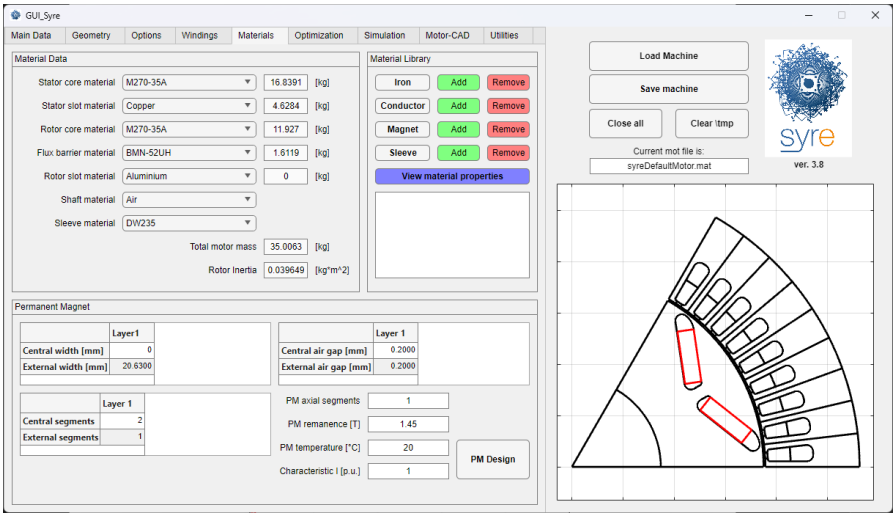

Figura 1.7. Schermata materials SyR-e [7]

- **Optimization**
  Questa finestra consente di effettuare l'ottimizzazione della macchina in esame, permettendo di scegliere se effettuare un'ottimizzazione focalizzata sul design o sul refine della macchina, utilizzando un algoritmo multi-obiettivo basato sull'evoluzione differenziale [1].
  Durante la procedura di ottimizzazione, alcuni parametri del rotore e dello statore vengono modificati automaticamente per raggiungere l'obiettivo prefissato. Per avviare la fase di ottimizzazione, è necessario prima impostare i parametri dell'algoritmo di ottimizzazione, le variabili da ottimizzare e gli obiettivi.

  Lo spazio di ricerca delle variabili da ottimizzare deve essere definito in modo ragionevole per evitare 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;
  – ottimizzazione del flusso a vuoto;
  – ottimizzazione del fattore di potenza.

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

- **Simulation**
  La sezione che permette di effettuare diverse tipologie di simulazioni agli elementi finiti sulla macchina.

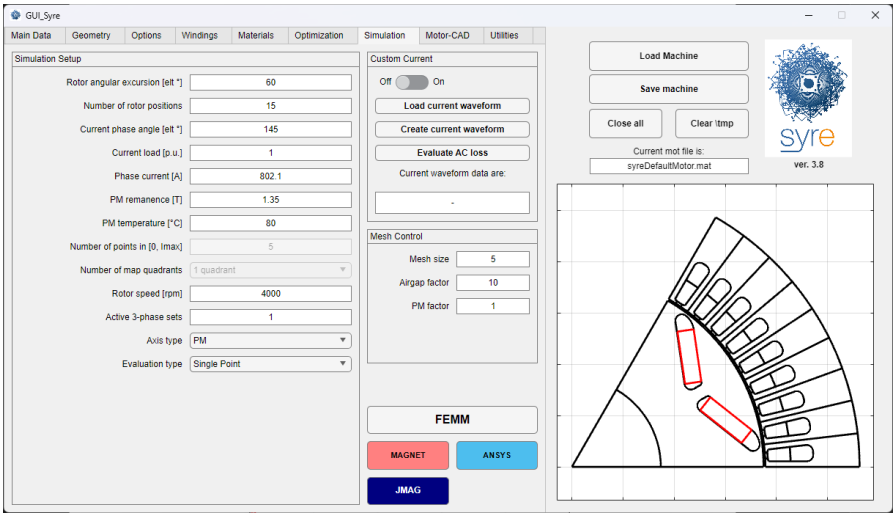

Figura 1.8. Schermata simulation SyR-e [7]

Una volta progettato il motore attraverso gli step precedenti, è possibile effettuare simulazioni di diversa natura su di esso. Una volta impostato il *Simulation Setup* definendo parametri come l'escursione angolare del rotore [1], il numero di posizioni di rotore da simulare, l'angolo della corrente di fase [2], la velocità di rotazione [3] e la convenzione da usare per gli assi [4], si passa a selezionare il tipo di simulazione desiderato.

Attraverso il menù a tendina *Evaluation type* si individua la simulazione da effettuare, distinguendo tra i seguenti tipi di valutazione:

[1] espressa in gradi elettrici [elt deg]

[2] l'angolo tra il fasore corrente e l'asse d

[3] espressa in rpm

[4] Permanent Magnet o Synchronous Reluctance

Tipo di simulazione	**Descrizione**
Single Point	Valutazione di un singolo punto (id, iq) definito dall'ampiezza della corrente e dall'angolo della corrente, calcolato dall'asse d.
Flux Map	Calcolo delle mappe di flusso e coppia su un dominio regolare (id, iq).
Characteristic Current	Calcolo della corrente caratteristica in funzione della temperatura del magnete permanente.
HWC Short-Circuit Current	Stima della corrente massima di corto circuito durante un corto circuito trifase simmetrico.
Demagnetization Curve	Calcolo della curva di smagnetizzazione in funzione della temperatura del magnete permanente.
Demagnetization Analysis	Analisi dell'area demagnetizzata per una corrente di demagnetizzazione e una temperatura del magnete permanente date.
Flux Density Analysis	Analisi della densità di flusso lungo la direzione di magnetizzazione.
Current Offset	Simulazione con un offset di corrente di fase.
Air Gap Force	Calcolo della forza radiale lungo lo spazio d'aria.
Iron Loss - Single Point	Calcolo delle perdite ferromagnetiche per un punto singolo.
Iron Loss - Flux Map	Calcolo delle perdite ferromagnetiche per una mappa di flusso.
Structural Analysis	Analisi strutturale utilizzando la PDE Toolbox in Matlab.

Tabella 1.1. Tipi di simulazione e descrizioni [1]

Una volta impostato l'*evaluation type*, si procede definendo i parametri della mesh e avviando la simulazione facendo clic su uno dei pulsanti disponibili in basso a destra:

- FEMM;
- MAGNET;
- ANSYS;
- JMAG.

I pulsanti corrispondono a software e tipi di FEA specifici. Selezioniamo il software e il tipo di FEA desiderati facendo clic sul pulsante corrispondente.
Tra i risultati delle varie simulazioni, troviamo le prestazioni della macchina in termini di coppia, fattore di potenza e flussi sugli assi dq (Figura 1.9), correnti sugli assi

dq, le mappe di flusso in funzione di i_d e i_q (Figura 1.10) e l'analisi dell'induzione al traferro e nel ferro per i punti di lavoro specifici.

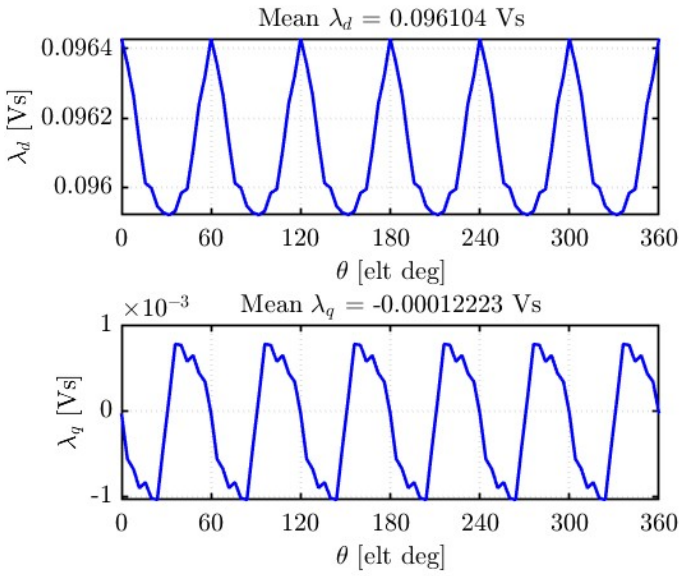

Figura 1.9. Flussi concatenati in assi dq [7]

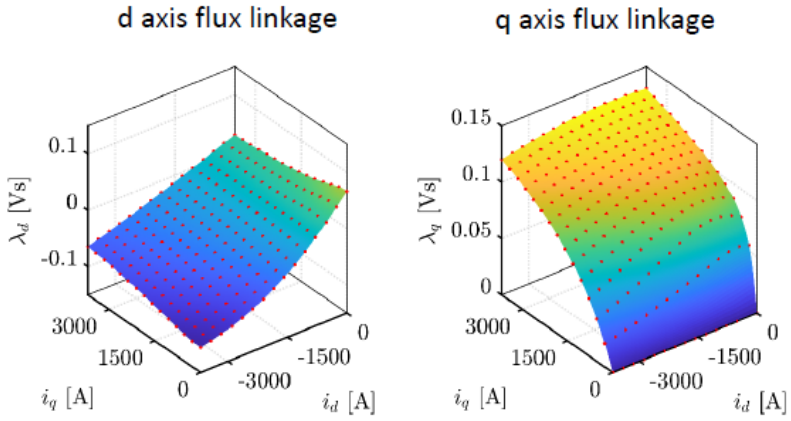

Figura 1.10. Mappe di flusso in funzione di i_d e i_q [7]

- **Motor-CAD**
  La sezione dedicata alla simulazione su Ansys Motor-CAD del motore progettato in

SyR-e.

SyR-e offre diverse opzioni di esportazione verso software FEA commerciali. È possibile esportare la geometria del motore in formato DXF, anche senza dettagli aggiuntivi come bordi o materiali [1]. L'esportazione in Ansys Motor-CAD permette una configurazione completa del modello per simulazioni elettromagnetiche e termiche.

- **Utilities**
  La sezione finale di SyR-e offre una raccolta di strumenti utili per l'utente. Qui, diversi pulsanti sono progettati per migliorare l'esperienza dell'utente, fornendo scorciatoie per l'esportazione dei modelli simulati tramite la sezione *Export*, oltre a documentazione sulla GUI, impostazioni di parallel computing e il pulsante *Launch MMM*.[5].

1.2 COMSOL Multiphysics

Figura 1.11. Logo di COMSOL Multiphysics [6]

COMSOL Multiphysics è un potente ambiente di simulazione interattivo e integrato, utilizzato per modellare e risolvere vari tipi di problemi scientifici e ingegneristici. Il software fornisce un ambiente desktop integrato con un *Model Builder* che offre una panoramica completa del modello e accesso a tutte le funzionalità. COMSOL Multiphysics rende possibile estendere facilmente i modelli convenzionali[6] in modelli multifisici che risolvono

[5] *syreManipulateMM: permette di calcolare e modificare mappe di flusso, correnti, calcolare MTPA e MTPV, valutare curve di coppia e potenza.*

[6] che utilizzano un solo tipo di fisica.

fenomeni fisici accoppiati simultaneamente. L'accesso a questa potenza non richiede una conoscenza approfondita della matematica o dell'analisi numerica.

Utilizzando le interfacce di fisica incorporate e quelle multifisiche, e il supporto avanzato per le proprietà dei materiali, è possibile costruire modelli definendo le quantità fisiche rilevanti, come proprietà dei materiali, carichi, vincoli, sorgenti e flussi, invece di definire le equazioni sottostanti. Queste variabili, espressioni o numeri possono essere applicati direttamente a domini solidi e fluidi, confini, bordi e punti, indipendentemente dalla mesh computazionale. Il software COMSOL Multiphysics compila internamente un insieme di equazioni che rappresentano l'intero modello multifisico (Figura 1.12).

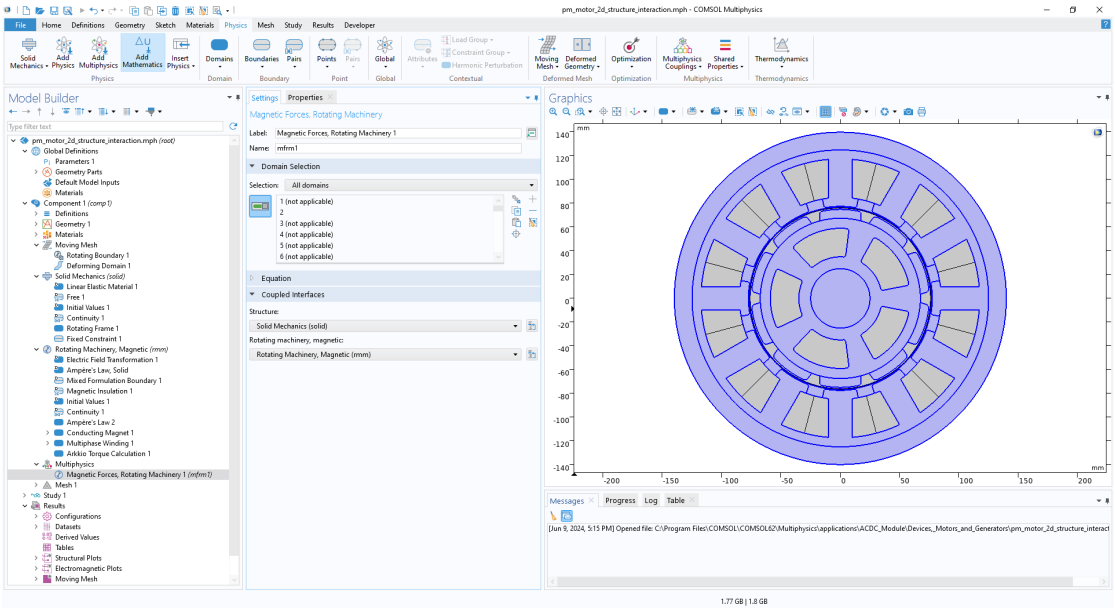

Figura 1.12. Esempio di accoppiamento multifisico in COMSOL [5]

Con queste interfacce di fisica, è possibile eseguire vari tipi di studi, tra cui:

- Studi stazionari e dipendenti dal tempo (transienti)
- Studi lineari e non lineari
- Studi di frequenza propria, modale e di risposta in frequenza

COMSOL Multiphysics può essere utilizzato come prodotto standalone attraverso un'interfaccia utente flessibile (*COMSOL Desktop*) mostrata in Figura 1.13, in app create

utilizzando l'*Application Builder* e distribuite tramite *COMSOL Compiler* o *COMSOL Server*, oppure tramite programmazione in Java o nel linguaggio MATLAB [7] [6].

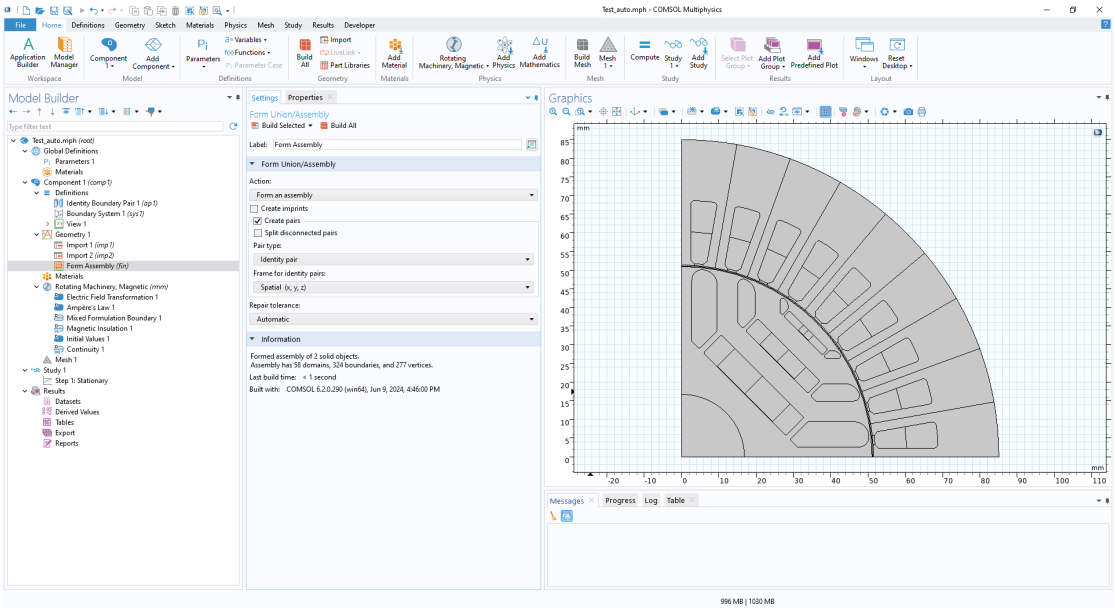

Figura 1.13. L'ambiente desktop di COMSOL Multiphysics [5]

Durante la risoluzione dei modelli, il software COMSOL Multiphysics assembla e risolve il problema utilizzando una serie di strumenti avanzati di analisi numerica. Il software esegue l'analisi insieme al raffinamento adattivo della mesh (se selezionato) e al controllo degli errori utilizzando una varietà di solutori numerici. Gli studi possono sfruttare sistemi multiprocessore e il calcolo a cluster, e si possono eseguire lavori batch e sweep parametrico.

Il software crea sequenze per registrare tutti i passaggi che creano la geometria, la mesh, la fisica, le impostazioni degli studi e del solutore, e la visualizzazione e presentazione dei risultati. Questo rende facile parametrizzare qualsiasi parte del modello; basta cambiare un nodo nell'albero del modello e rieseguire le sequenze. Il programma ricorda e riapplica tutte le altre informazioni e dati nel modello. È possibile eseguire sweep parametrico potenti e flessibili che variano alcune proprietà del materiale, del carico o di altri aspetti del modello.

Vi sono numerosi strumenti di post-elaborazione per tracciare, tabulare e valutare i risultati delle simulazioni COMSOL. L'intero processo di simulazione, dagli input e dalla

[7] quest'ultima opzione richiede una licenza *LiveLink* per MATLAB

geometria ai risultati, viene eseguito in un unico ambiente desktop integrato, il *COMSOL Desktop* (Figura 1.14).

Inoltre, è possibile gestire i progetti di simulazione utilizzando il *Model Manager* e creare report e presentazioni in formato HTML, Word e PowerPoint, includendo grafica e visualizzazioni integrative.

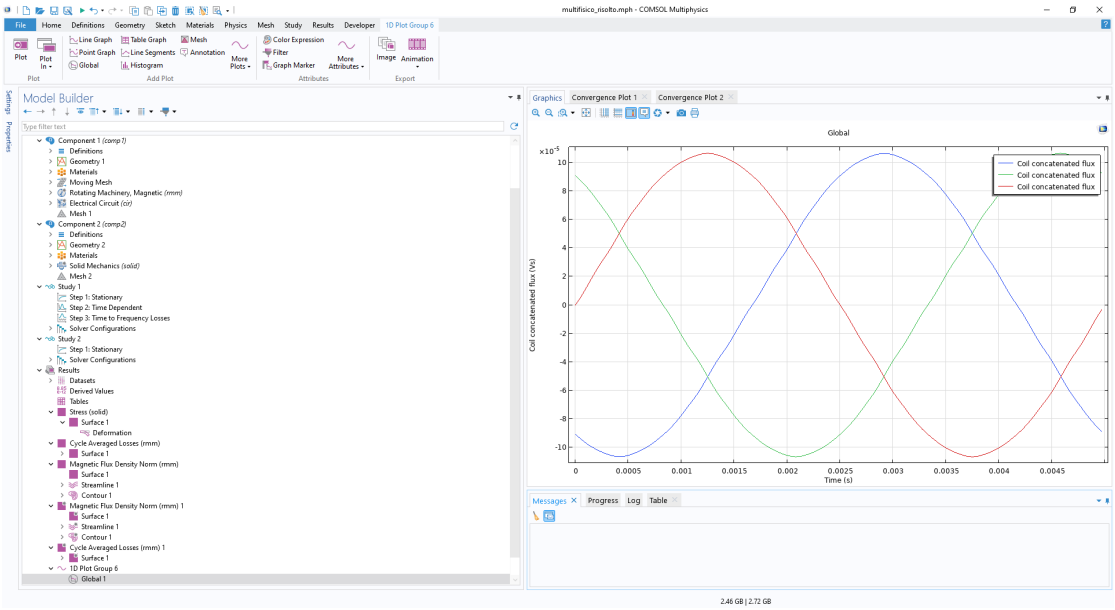

Figura 1.14. Esempio di estrazione di risultati in forma grafica [5]

Le equazioni differenziali parziali (PDE) formano la base delle leggi della scienza e forniscono le fondamenta per la modellazione di una vasta gamma di fenomeni scientifici e ingegneristici. È possibile utilizzare il software COMSOL Multiphysics in molte aree applicative e metodi di analisi, combinando liberamente le fisiche e incorporando PDE, ODE e DAE definiti dall'utente se necessario:

Aree Applicative	
Acustica	Batterie
Bioscienze	Reazioni chimiche
Materiali compositi	Simulazioni di sistemi di controllo
Corrosione e protezione dalla corrosione	Progettazione di esperimenti (DOE)
Diffusione	Elettrochimica
Deposizione elettrolitica	Elettromagnetismo
Analisi della fatica	Fluidodinamica (CFD)
Celle a combustibile ed elettrolizzatori	Geofisica e geomeccanica
Trasferimento di calore	Materiali stratificati e compositi
Liquidi e gas	Lavorazione dei metalli
Microdispositivi elettromeccanici (MEMS)	Microfluidica
Ingegneria delle microonde	Miscelatori e miscelazione di fluidi
Flusso molecolare	Dinamica multibody
Ottica	Ottimizzazione e analisi di sensibilità
Tracciamento delle particelle	Fotonica
Dispositivi piezoelettrici	Flusso nei tubi
Fisica del plasma	Flusso di polimeri
Flusso nei mezzi porosi	Meccanica quantistica
Componenti a radiofrequenza	Tracciamento dei raggi e ottica dei raggi
Dinamica del rotore	Dispositivi a semiconduttore
Meccanica strutturale	Flusso nel sottosuolo
Termodinamica	Fenomeni di trasporto
Quantificazione dell'incertezza (UQ)	Ottica e propagazione delle onde

Tabella 1.2. Aree applicative di COMSOL Multiphysics [6]

I moduli opzionali, incluse le opzioni di interfaccia come il *CAD Import Module* e le interfacce bidirezionali come i prodotti *LiveLink*, sono ottimizzati per specifiche aree applicative e offrono terminologia e interfacce di fisica standardizzate per la disciplina. Per alcuni moduli, sono disponibili anche librerie di materiali aggiuntive, solutori specializzati,

tipi di elementi e strumenti di visualizzazione.

1.3 Confronto tra SyR-e e COMSOL Multiphysics

SyR-e e COMSOL Multiphysics sono entrambi potenti strumenti di simulazione utilizzati per progettare e analizzare una vasta gamma di applicazioni ingegneristiche. Tuttavia, presentano alcune differenze significative nelle loro caratteristiche e funzionalità.
SyR-e è sviluppato in ambiente MATLAB ed è disponibile gratuitamente online, mentre COMSOL Multiphysics è un software commerciale con diverse opzioni di licenza in base ai moduli e alle funzionalità richieste. Questa differenza nell'ambiente di sviluppo può influenzare l'accessibilità e la disponibilità dei due software agli utenti.

In termini di funzionalità di progettazione preliminare, SyR-e offre un modulo di Preliminary Design per eseguire un dimensionamento iniziale della macchina, mentre COMSOL non dispone di una funzionalità equivalente. Tuttavia, COMSOL offre un'ampia gamma di moduli aggiuntivi che consentono di estendere le funzionalità di base del software per adattarsi alle esigenze specifiche dell'utente.

Un'altra differenza significativa è nell'ottimizzazione dei progetti. SyR-e utilizza un algoritmo di ottimizzazione multi-obiettivo per fornire una visione completa della macchina ottimizzata, mentre COMSOL offre analisi di sensitività dei singoli parametri per ottimizzare il progetto. Questo approccio può influenzare la precisione e l'efficienza dell'ottimizzazione dei progetti nei due software.

Il calcolo delle perdite nel ferro è differente tra i due programmi. COMSOL offre un calcolo automatico delle perdite nel ferro, mentre SyR-e richiede l'utilizzo di software aggiuntivi come MagNet. Questa differenza può influenzare la facilità d'uso e l'integrazione delle funzionalità magnetiche nei progetti.

Infine, per quanto riguarda la verifica strutturale e la simulazione termica, COMSOL dispone di strumenti integrati dedicati, mentre SyR-e richiede l'uso di software esterni per queste funzionalità. Questa differenza può influenzare l'efficienza e la coerenza delle analisi strutturali e termiche nei due software.

La Tabella 1.3 riassume le principali caratteristiche del confronto tra i due software:

Caratteristiche	**SyR-e**	**COMSOL Multiphysics**
Ambiente di sviluppo	MATLAB	Software commerciale
Modulo di Preliminary Design	Presente	Assente
Ottimizzazione multi-obiettivo	Presente	Assente
Calcolo delle perdite nel ferro	Presente	Integrato
Verifica strutturale	Richiede software esterni	Integrato
Simulazione termica	Richiede software esterni	Integrato

Tabella 1.3. Confronto tra SyR-e e COMSOL Multiphysics

1.4 Obiettivo della tesi

Il presente lavoro di tesi si propone di implementare un ulteriore strumento di analisi volto al miglioramento di progettazione preliminare e simulazione approfondita dei motori elettrici, attraverso lo sviluppo di uno script di interfacciamento tra i software SyR-e e COMSOL. L'analisi preliminare, svolta nel precedente paragrafo, ha identificato le complementarietà e le sinergie tra questi due ambienti software. SyR-e si distingue come un potente strumento per la progettazione dei motori elettrici, offrendo un'ampia gamma di funzionalità per la modellazione e l'ottimizzazione delle prestazioni. Tuttavia, richiede il supporto di diversi software aggiuntivi per completare il processo di progettazione.

COMSOL è strutturato come un ambiente di simulazione multiphysics altamente avanzato, in grado di modellare una vasta gamma di fenomeni fisici, compresi quelli elettromagnetici, termici e strutturali. Grazie alla sua flessibilità e alle potenti capacità di analisi, COMSOL consente di esplorare in modo dettagliato il comportamento del sistema elettrico in condizioni reali, integrando più fenomeni fisici e consentendo una valutazione completa delle prestazioni della macchina in una singola simulazione.

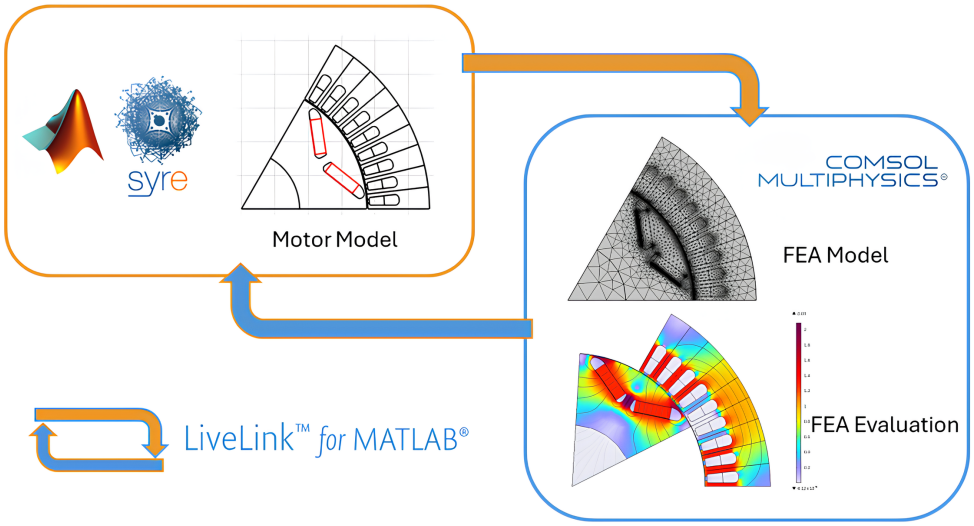

Figura 1.15. Flusso di dati tra SyR-e e COMSOL

L'obiettivo principale è sviluppare un'interfaccia in MATLAB che consenta l'integrazione diretta tra SyR-e e COMSOL, sfruttando l'ambiente LiveLink *for* MATLAB. Lo script avrà lo scopo di trasferire in modo efficiente i dati del modello progettato su SyR-e in COMSOL per l'esecuzione di simulazioni avanzate e dettagliate. Questo approccio integrato consentirà di utilizzare SyR-e come piattaforma per la progettazione preliminare e concettuale del motore, beneficiando delle sue funzionalità avanzate di progettazione della macchina. Successivamente, il modello progettato su SyR-e verrà trasferito a COMSOL per condurre simulazioni agli elementi finiti più approfondite e realistiche, che terranno conto di una serie di fattori, tra cui il comportamento elettromagnetico e strutturale del sistema.

Un aspetto cruciale di questa ricerca è la validazione dei risultati ottenuti tramite lo script di interfacciamento. Ciò sarà realizzato confrontando i risultati delle simulazioni condotte su COMSOL con quelli ottenuti attraverso le simulazioni eseguite singolarmente su SyR-e e gli altri software di cui usufruisce. Questo processo di convalida consentirà di valutare l'accuratezza e l'affidabilità delle simulazioni integrate e di identificare eventuali discrepanze o aree di miglioramento.

In sintesi, lo sviluppo di questo script di interfacciamento rappresenta un passo significativo verso la creazione di un processo di progettazione e simulazione integrato in SyR-e, consentendo di sfruttare al meglio le potenzialità di entrambi gli ambienti software e di

ottenere una valutazione completa delle prestazioni della macchina.

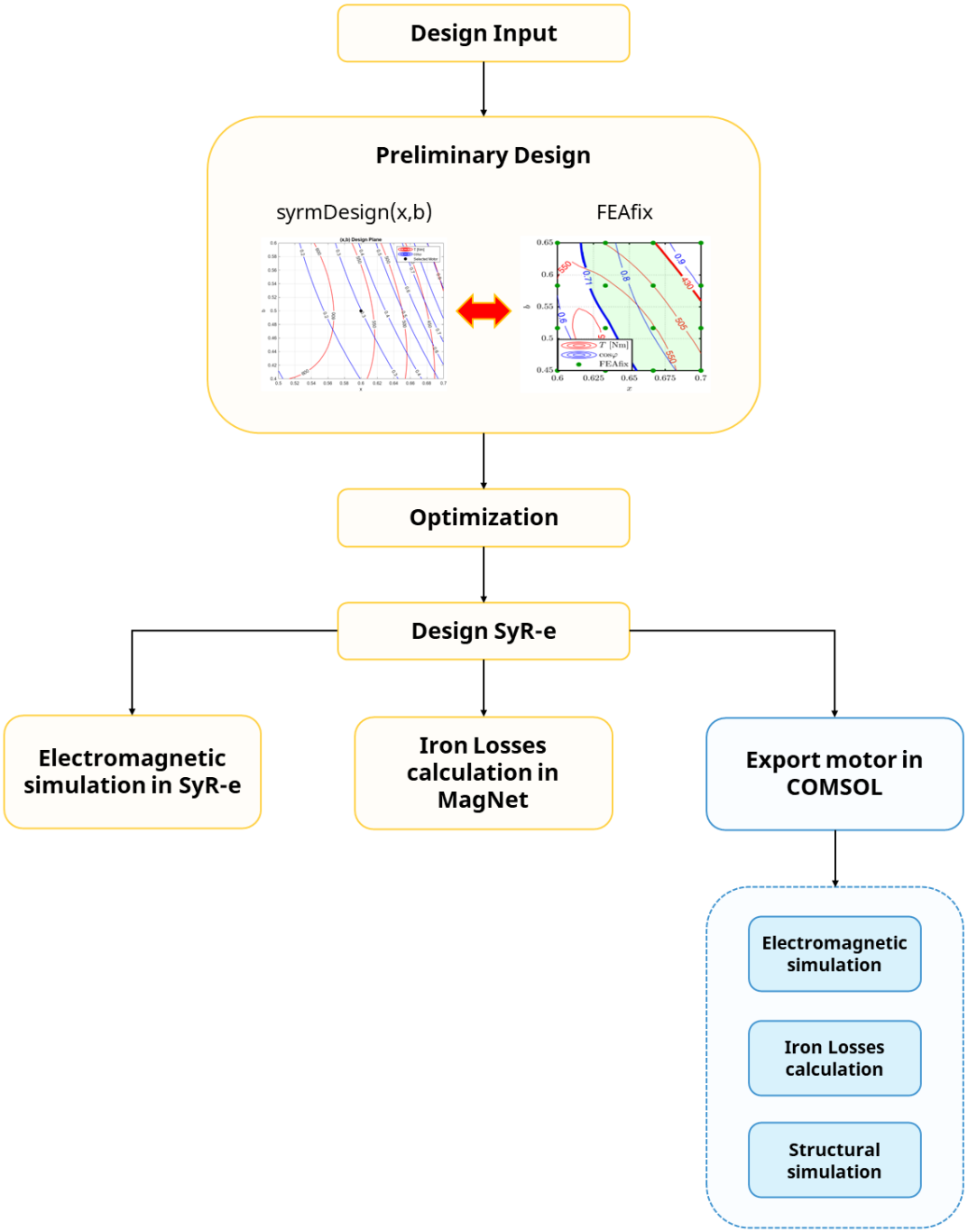

Figura 1.16. Progetto di un motore attraverso l'interfaccia SyR-e – COMSOL

2 Interfaccia SyR-e – COMSOL Multiphysics

Lo sviluppo di uno script di interfaccia tra SyR-e e COMSOL consente di massimizzare le potenzialità di entrambi i software e di valorizzarne le caratteristiche distintive.

L'integrazione di COMSOL nella toolchain di SyR-e rappresenta un avanzamento significativo nella qualità delle simulazioni FEA, grazie alla capacità di COMSOL di simulare, con un unico comando, la macchina progettata in SyR-e. Questo comporta una notevole riduzione dei tempi di progettazione e analisi delle performance del progetto, grazie a due fattori principali:

- l'interfaccia trasferisce e imposta automaticamente l'intero modello in COMSOL;
- la capacità di COMSOL di eseguire simulazioni multifisiche.

L'interfaccia è scritta in ambiente MATLAB utilizzando lo strumento LiveLink *for* MATLAB di COMSOL, che collega COMSOL Multiphysics a MATLAB. Questo consente di integrare SyR-e con COMSOL, permettendo di configurare modelli tramite script con l'API di COMSOL, definire proprietà dei materiali e condizioni al contorno con funzioni MATLAB, controllare il flusso dei programmi e gestire eccezioni. Inoltre, si possono estrarre e analizzare dati direttamente in MATLAB e creare interfacce utente personalizzate, migliorando l'efficienza e la qualità delle simulazioni FEA sviluppate dal modello in SyR-e.

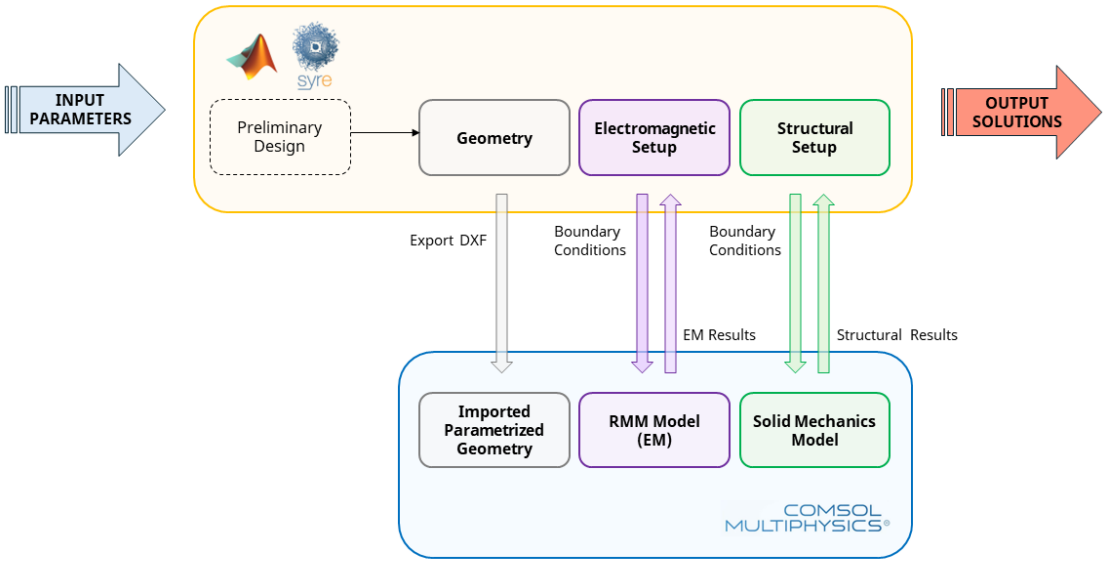

Figura 2.1. Interfaccia SyR-e – COMSOL

Per illustrare il principio di funzionamento dell'interfaccia, si è sviluppato uno schema a blocchi che rappresenta il flusso e la tipologia di dati scambiati tra SyR-e e COMSOL, come mostrato in Figura 2.1. Lo scopo di questo diagramma è offrire una più chiara visualizzazione del processo di trasferimento e utilizzo dei dati, semplificando la comprensione del flusso di informazioni.

Una volta definiti i parametri del progetto preliminare della macchina e inseriti in SyR-e, attraverso le finestre descritte nel capitolo precedente, si passa a definire un design preliminare della macchina, attraverso l'utilizzo dei diversi strumenti presenti nelle finestre della GUI presentate nel capitolo introduttivo.

Il design preliminare permette di ottenere volumi e sfruttamenti che servono a definire la geometria, i materiali e i circuiti della macchina. A questo punto, il modello creato in SyR-e deve essere simulato per verificare che i parametri e i materiali scelti siano ottimali per la realizzazione del motore desiderato.

I dati geometrici della macchina vengono tradotti in un disegno della sezione trasversale, evidenziando statore, rotore, traferro e albero. Questo disegno viene esportato in COMSOL in formato `.dxf` come primo step di costruzione del modello nel software multifisico, come mostrato in Figura 2.2.

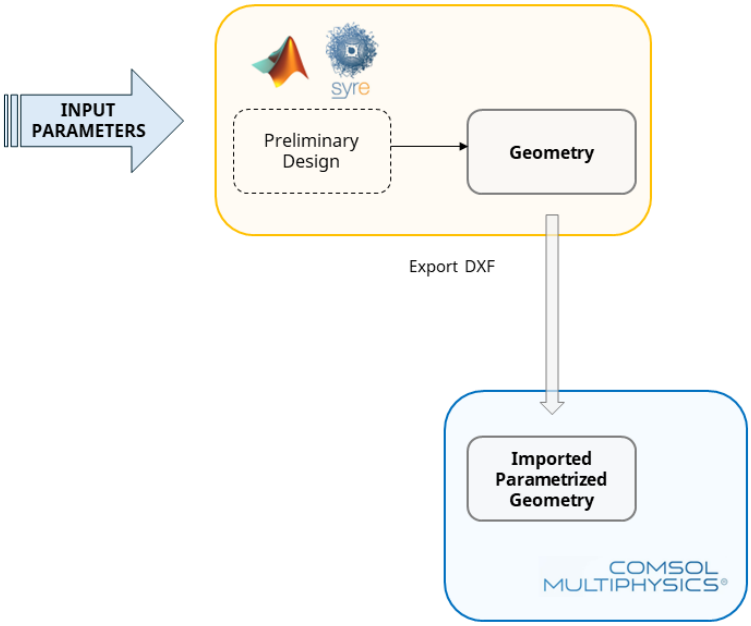

Figura 2.2. Interfaccia SyR-e – COMSOL (Focus Geometria)

Durante questa fase, vengono esportati anche i dati relativi ai materiali necessari per le simulazioni, comprensivi di tutti i parametri elettromagnetici richiesti per l'analisi FEA magnetica, come curve B-H, permeabilità e resistività. Inoltre, vengono definite le boundary conditions necessarie per configurare correttamente la fisica *Rotating Machinery Magnetic (rmm)* al fine di ottenere uno studio quanto più possibile vicino alle condizioni elettromagnetiche reali.

Una volta definiti i parametri geometrici, i materiali e impostate le condizioni a contorno, si esporta il setup dello studio che si vuole realizzare (Figura 2.3). In questo contesto, l'interfaccia non si limita a tradurre il linguaggio tra i due software, ma definisce l'intero studio in COMSOL secondo le impostazioni selezionate in SyR-e, nonostante il primo richieda un'impostazione di studio temporale e l'altro spaziale.

A questo punto, viene avviata la simulazione in COMSOL. Una volta ottenuti i risultati, questi vengono estratti e esportati in SyR-e per le elaborazioni di post-processing, e per essere presentati all'utente sotto forma di immagini e grafici utili.

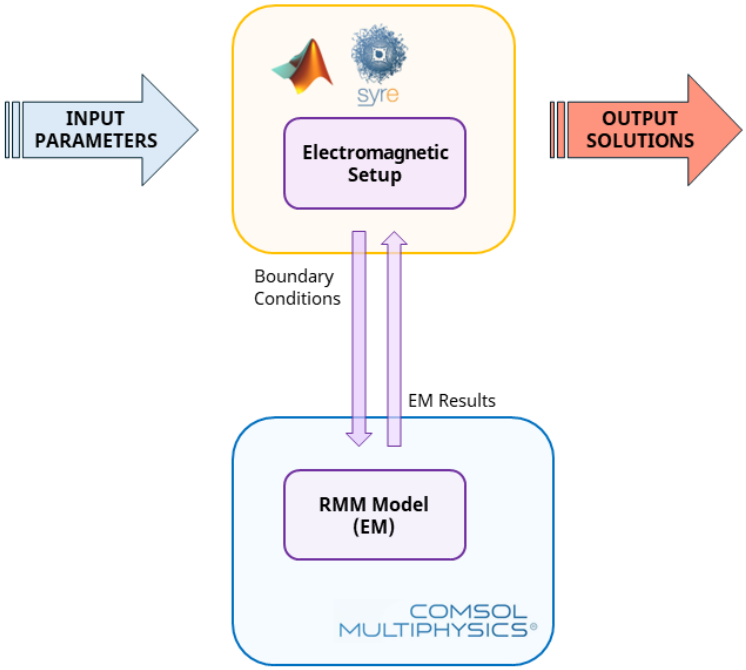

Figura 2.3. Interfaccia SyR-e – COMSOL (Focus Elettromagnetico)

Contemporaneamente, è possibile effettuare uno studio strutturale del lamierino del rotore utilizzando la fisica *Solid Mechanics*. Questo consente di effettuare valutazioni tensoriali per ottenere una panoramica della resistenza meccanica della macchina alle sollecitazioni derivanti dalla velocità di rotazione (Figura 2.4).

Per eseguire questo studio, il setup meccanico viene definito in SyR-e e, attraverso lo script di interfaccia, viene generato un modello in COMSOL. Una volta impostati i parametri fondamentali dello studio, come le proprietà meccaniche dei materiali, la velocità di rotazione da analizzare e le condizioni al contorno necessarie per la simulazione, si lancia la simulazione in COMSOL. I risultati vengono poi importati in SyR-e, presentati sotto forma di figure e grafici utili per valutare la robustezza del progetto dal punto di vista meccanico.

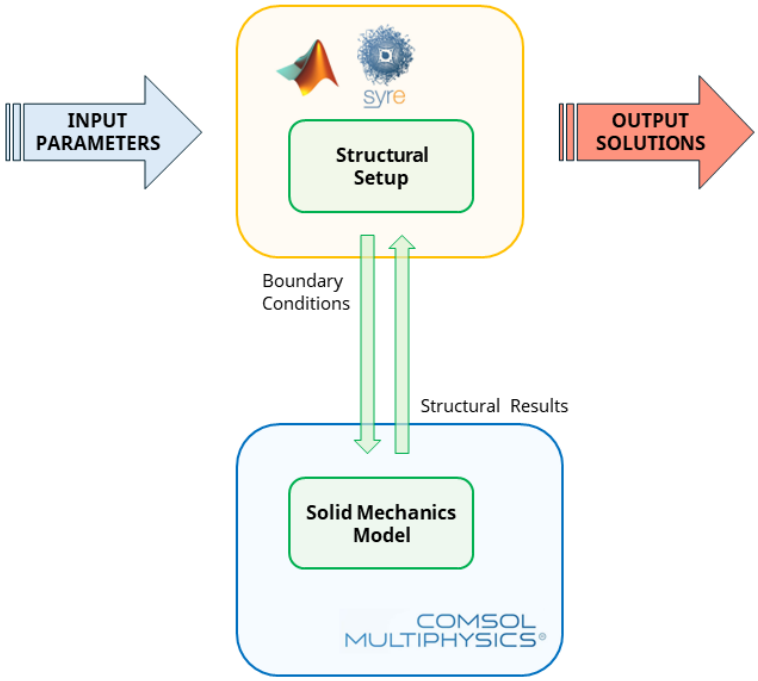

Figura 2.4. Interfaccia SyR-e – COMSOL (Focus Strutturale)

2.1 Struttura dell'interfaccia

In Figura 2.5, è stato sviluppato un flow-chart che include le diverse funzioni componenti dell'interfaccia. Questo diagramma offre una visione d'insieme delle operazioni eseguite e del loro ordine logico, rendendo più chiaro il processo di integrazione tra SyR-e e COMSOL. Il flow-chart illustra schematicamente come ogni funzione si inserisca nel flusso complessivo, evidenziando le interazioni tra i diversi componenti e la sequenza delle operazioni necessarie per completare le simulazioni FEA. Seguendo il diagramma, si identificano:

- `DrawAndSaveMachineCOMSOL`

  La funzione che verifica e imposta i nomi dei file di input e output, determinando il nome del file di input `.mat` e modificandolo in `.mph` per l'uso con COMSOL, e impostando anche il percorso del file. Se la macchina è personalizzata, la funzione chiede all'utente se desidera salvarla. Utilizza la funzione `draw_motor_in_COMSOL` per creare il modello della macchina in COMSOL, basandosi sui dati geometrici e dei materiali forniti. Infine, se necessario, aggiorna il dataset con i nuovi nomi dei

file e percorsi, e salva tutte le informazioni (geometria, prestazioni, materiali) in un file `.mat`.

- `draw_motor_in_COMSOL`

  La funzione che si occupa dell'effettivo disegno della macchina all'interno del modello COMSOL. Questa funzione traduce in geometria i valori dei parametri scelti da SyR-e e li trasferisce a COMSOL. Con essa si costruisce la geometria della macchina, si definiscono i materiali da utilizzare per la simulazione, si impostano la fisica, la mesh e le condizioni al contorno. Inoltre, vengono assegnati i valori nei circuiti e si salva un file `.mph` del modello, pronto per le impostazioni di studio desiderate.

- `eval_operatingPointCOMSOL`

  La funzione progettata per valutare il punto operativo di un motore attraverso simulazioni FEM utilizzando COMSOL Multiphysics. Inizialmente, importa i dati di input da un file `.mat`, successivamente, utilizza questi parametri per preparare l'ambiente di simulazione, inizializzando variabili e strutture dati. Durante l'esecuzione, la funzione chiama `COMSOLfitness` per eseguire la simulazione, ottenendo dati specifici sul comportamento del motore e sulle sue prestazioni nel punto operativo considerato. Infine, la funzione genera diversi grafici per il post-processing dei risultati della simulazione.

- `COMSOLfitness`

  La funzione che funge da collegamento tra `eval_operatingPoint_COMSOL`, che prepara l'ambiente di simulazione, e `simulate_xdeg_COMSOL`, che esegue la simulazione FEA per valutare le prestazioni del motore in esame con parametri definiti in `eval_operatingPoint_COMSOL`. Successivamente, calcola le medie di variabili di output come correnti e flussi dai risultati della simulazione e determina parametri di potenza.

- `simulate_xdeg_COMSOL`

  La funzione che gestisce l'intero ciclo di simulazione e post-processing in COMSOL. Inizialmente, carica il modello inizializzato dalla funzione `draw_motor_in_COMSOL`, configura le impostazioni del solutore, quindi avvia la simulazione FEA. Una volta terminati i calcoli, esegue il post-processing per ricavare e salvare dati fondamentali come coppia motrice, flussi magnetici, correnti e perdite. I risultati vengono esportati in file `.csv` e salvati in una struttura dati `SOL` per l'analisi dei risultati delle prestazioni del sistema.

- `eval_vonMisesstress_COMSOL`

  Questa funzione si occupa di simulare il comportamento strutturale di componenti meccanici utilizzando il software COMSOL. In particolare, carica dati da file, prepara directory e importa file DXF per definire la geometria. Definisce materiali come N52 e NGO 35PN270 con relative proprietà fisiche e li assegna alle parti del modello, specificando le condizioni al contorno. Successivamente, crea uno studio di analisi stazionaria per risolvere il modello, tenendo conto di rigidezza e deformazioni. Infine, esegue post-processing per visualizzare mappe di stress e deformazioni, generando grafici per interpretare i risultati ottenuti.

- `eval_maxstress_COMSOL`

  La funzione progettata per analizzare i dati di stress ottenuti da simulazioni effettuate attraverso `eval_vonMisesstress_COMSOL`. Il suo scopo principale è identificare e registrare le coordinate dei punti nello spazio dove lo stress supera un valore massimo specificato. Inoltre, la funzione identifica il punto in cui si è ottenuto il massimo stress, secondo la simulazione, e ne registra le coordinate. Infine, i risultati vengono restituiti in una struttura `out`, consentendo agli utenti di identificare facilmente i punti critici all'interno del lamierino di rotore.

Nei paragrafi successivi, verranno descritte nel dettaglio le singole funzioni che compongono l'interfaccia, presentate brevemente sopra. Ogni funzione sarà spiegata in termini di operazioni svolte, input e output, e il suo ruolo nel contesto generale dell'interfaccia. Questo approccio sistematico punta a semplificare la comprensione di come ciascuna funzione contribuisca al processo complessivo di simulazione.

Ad esempio, verrà discusso come le funzioni di importazione dei dati geometrici e dei materiali trasferiscono le informazioni da SyR-e a COMSOL, garantendo che tutte le specifiche necessarie siano accuratamente tradotte. Verranno anche esaminate le funzioni che definiscono le condizioni al contorno e configurano i parametri di simulazione in COMSOL, assicurando che le impostazioni selezionate in SyR-e siano applicate correttamente nel modello multifisico.

Infine, verranno esaminate le funzioni di post-processing che estraggono i risultati della simulazione da COMSOL e li importano nuovamente in SyR-e. Queste funzioni non solo facilitano l'analisi dei risultati, ma permettono anche di presentare i dati in formati visivi, come grafici e immagini, con l'obiettivo di mettere a disposizione dell'utente tutte le informazioni necessarie allo sviluppo e alla validazione delle simulazioni, migliorando anche l'efficienza e l'efficacia dell'intero processo di progettazione.

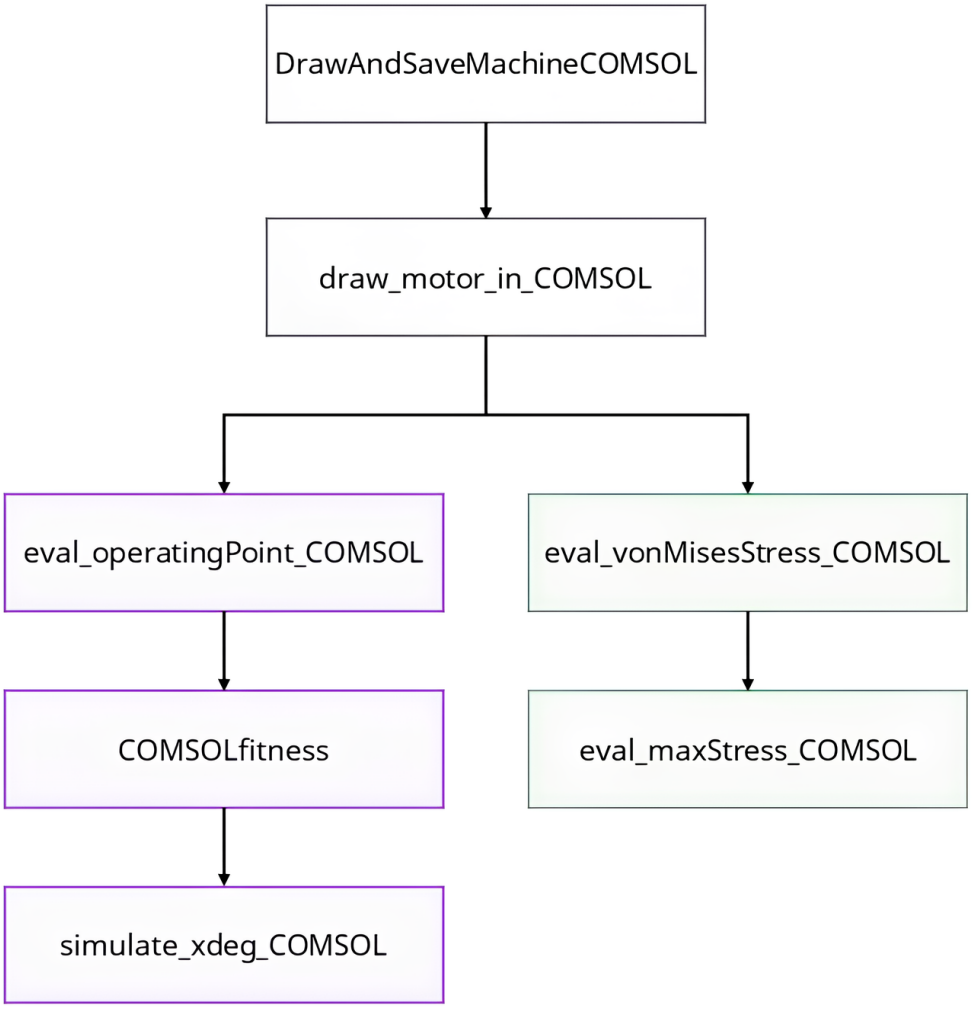

Figura 2.5. Struttura dell'interfaccia SyR-e – COMSOL

3 Creazione del modello in COMSOL

In questo paragrafo viene descritto il processo di esportazione del modello della macchina da SyR-e a COMSOL Multiphysics, fondamentale per avviare simulazioni FEA all'interno dell'ambiente multifisico di COMSOL. Le funzioni centrali in questo contesto sono `DrawAndSaveMachineCOMSOL` e `draw_motor_in_COMSOL`, illustrate nella Figura 3.1, che costituisce la parte iniziale del diagramma a blocchi presentato nel capitolo precedente.

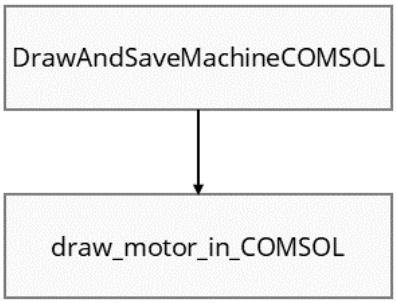

Figura 3.1. Funzioni responsabili della creazione del modello

Le funzioni hanno il compito di trasferire i parametri geometrici, dei materiali e fisici da SyR-e a COMSOL, sfruttando lo strumento LiveLink for MATLAB di COMSOL. Durante la loro implementazione per COMSOL, ci si è basati sull'architettura delle funzioni già presenti in SyR-e, cercando di minimizzare le personalizzazioni necessarie nel codice script, considerando le differenze significative tra SyR-e e COMSOL nella natura delle simulazioni agli elementi finiti.

Questo approccio mira a standardizzare le procedure per agevolare gli utenti che già conoscono la parte di scripting di SyR-e, incoraggiandoli a esplorare personalizzazioni e miglioramenti del software, essendo esso open-source.

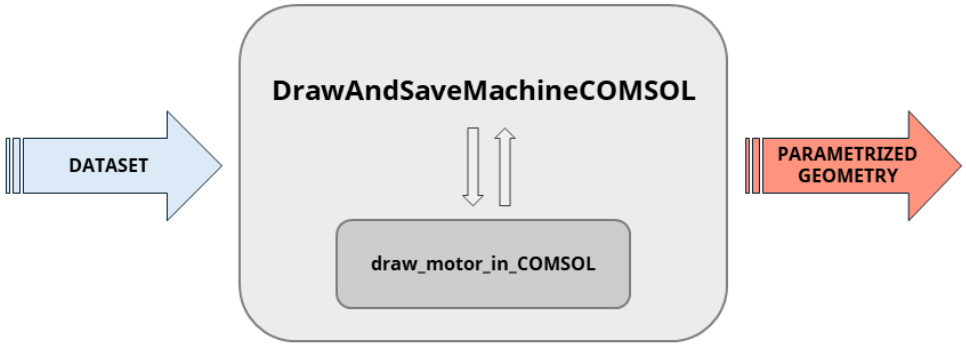

Figura 3.2. Struttura annidata di DrawAndSaveMachineCOMSOL

Nel dettaglio delle funzioni di esportazione del modello, si osserva dalla Figura 3.2 che `DrawandSaveMachineCOMSOL` svolge un ruolo fondamentale come impostazione generale del modello. Questa funzione gestisce la preparazione dei dati e stabilisce accuratamente i percorsi dei file necessari. Utilizza inoltre `draw_motor_in_COMSOL` per la creazione del modello della macchina all'interno di COMSOL, facendo affidamento sui dati geometrici e dei materiali forniti. Successivamente, aggiorna il dataset e memorizza tutte le informazioni rilevanti in un file `.mat`. È importante sottolineare che `draw_motor_in_COMSOL` si occupa direttamente della costruzione dettagliata del modello in COMSOL, partendo dall'implementazione della geometria della macchina, passando per la definizione delle caratteristiche del traferro e arrivando alla configurazione delle fisiche e della mesh necessarie per la simulazione.

L'interfaccia Rotating Machinery, Magnetic (rmm), collocata all'interno delle interfacce elettromagnetiche (AC/DC>Electromagnetics and Mechanics) di COMSOL, è destinata alla progettazione e all'analisi di motori e generatori elettrici. Supporta la modellazione stazionaria e nel dominio del tempo sia in 2D che in 3D. Questa interfaccia risolve le equazioni di Maxwell formulando le variabili dipendenti attraverso una combinazione del potenziale vettore magnetico e del potenziale scalare magnetico [6].

L'aggiunta di questa interfaccia fisica introduce automaticamente i seguenti nodi predefiniti nel Model Builder: Electric Field Transformation, Ampère's Law, Mixed Formulation Boundary, Magnetic Insulation (condizione al contorno predefinita) e Initial Values. Dalla barra degli strumenti *Physics* è possibile aggiungere ulteriori nodi per implementare condizioni al contorno e condizioni puntuali. Inoltre, è possibile selezionare le caratteristiche fisiche facendo clic con il tasto destro del mouse su *Rotating Machinery, Magnetic* e scegliendo le opzioni dal menu contestuale.

L'interfaccia *Electrical Circuit*, situata sotto il ramo AC/DC durante l'aggiunta di un'interfaccia fisica, è impiegata per modellare correnti e tensioni nei circuiti, inclusi sorgenti di tensione e corrente, resistori, condensatori, induttori e dispositivi a semiconduttore. I modelli creati con questa interfaccia possono includere connessioni a modelli di campo distribuito. Supporta la modellazione stazionaria, in frequenza e nel dominio del tempo, risolvendo le leggi di conservazione di Kirchhoff per le tensioni, correnti e cariche associate agli elementi del circuito[6].

L'aggiunta di questa interfaccia fisica introduce un nodo Ground Node predefinito associato al nodo zero nel circuito elettrico.

Questo processo automatizzato garantisce una preparazione accurata dei dati per le analisi e le simulazioni in COMSOL, assicurando un flusso di lavoro efficiente e ben organizzato per gli utenti, così da essere coinvolti maggiormente nello sviluppo e nell'ottimizzazione di SyR-e.

3.1 DrawandSaveMachineCOMSOL

La funzione `DrawAndSaveMachineCOMSOL` è progettata per disegnare e salvare i dati di una macchina in un file da zero oppure aggiornarne una già esistente. Questo processo costituisce il primo tassello dell'interfaccia SyR-e COMSOL per modellare e simulare il comportamento delle macchine elettriche, permettendo agli utenti di visualizzare e inizializzare il design del motore in progetto. La funzione può essere lanciata da direttamente dall'interfaccia GUI di SyR-e attraverso il comando `SaveMachine`, facilitando l'interazione tra l'utente e l'ambiente di lavoro in MATLAB, oppure può essere eseguita in modo autonomo senza il supporto di una GUI direttamente, attraverso la *Command Window*.

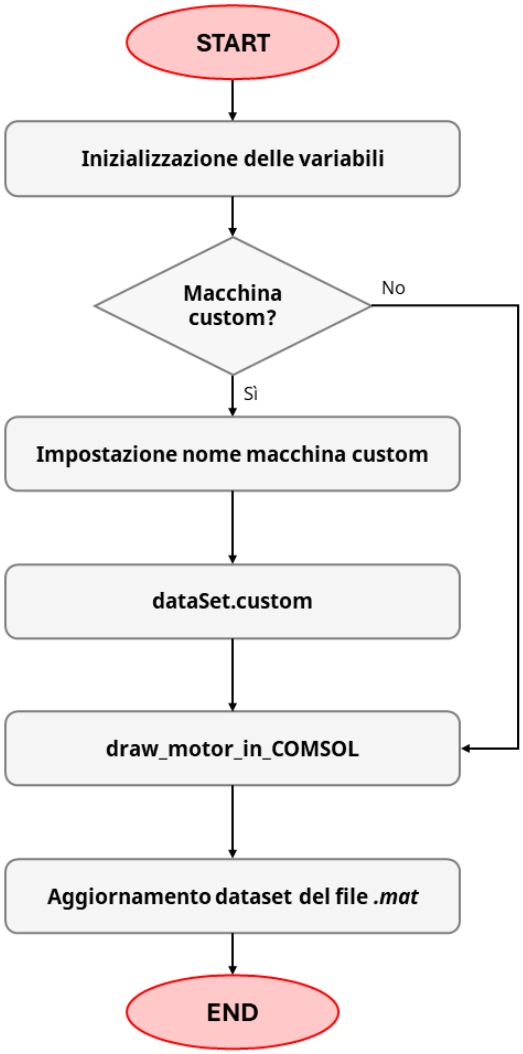

Figura 3.3. Flusso logico di DrawAndSaveMachineCOMSOL

3.1.1 Input della Funzione

- `dataSet`: Struttura contenente una copia di tutti i campi presenti nella GUI, riguardanti la macchina. Infatti, `dataSet` include informazioni fondamentali come il nome del file in uso (`currentfilename`), l'intero percorso del file del motore in progetto (`currentpathname`), il flag che identifica una geometria custom (`custom`), il numero di cave di statore (`NumOfSlots`), il numero di coppie polari (`NumOfPolePairs`) e altri parametri specifici della macchina [1].

- `filename`: Nome del file in cui salvare i dati della macchina. Se non specificato, la funzione richiederà all'utente di fornire un nome attraverso una finestra di dialogo.
- `pathname`: Percorso del file in cui salvare i dati della macchina. Se non specificato, la funzione richiederà all'utente di fornire un percorso attraverso una finestra di dialogo.

3.1.2 Output della Funzione

- `dataSet`: Struttura aggiornata con una copia di tutti dati della macchina. Dopo l'esecuzione della funzione, il `dataSet` include i nuovi valori di `currentfilename` e `currentpathname`, eventuali aggiornamenti ai parametri geometrici e materiali della macchina e conferma che il processo di salvataggio è stato completato correttamente.

3.1.3 Dettagli delle operazioni

- Impostazione iniziale

```
26 nameIn  = dataSet.currentfilename;
27 nameIn  = strrep(nameIn,'.mat','.mph');
28 pathIn  = dataSet.currentpathname;
29 fileIn  = [pathIn nameIn];
```

Nella fase impostazione iniziale, la funzione esegue diverse operazioni. Prima di tutto, ottiene il nome del file `.mat` corrente dalla struttura `dataSet` sopracitata e lo assegna alla variabile `nameIn`. Successivamente, modifica l'estensione del file da `.mat` a `.mph`, adeguandola per l'utilizzo con il software COMSOL Mltiphysics, utilizzando la funzione `strrep`. Dopo di che, recupera il percorso corrente del file da `dataSet` e lo assegna alla variabile `pathIn`. Infine, combina il percorso del file (`pathIn`) con il nome del file modificato (`nameIn`) per ottenere il percorso completo del file di input, assegnandolo alla variabile `fileIn`. Le operazioni appena descritte sono riportate nel codice sopra.

- Gestione di Motori Custom

Se nella struttura `dataSet` appena caricata, è rappresentata una macchina custom, la funzione chiede all'utente se desidera salvare la macchina personalizzata tramite una finestra di dialogo (`questdlg`). Questo processo viene gestito con il seguente codice:

```
31 if dataSet.custom
32    button = questdlg('Save custom machine?','SELECT','Yes','
          Cancel','Yes');
```

```
33 end
```

- Disegno del Motore in COMSOL

  Nella fase di disegno del motore in COMSOL, la funzione esegue diverse operazioni per garantire che il modello del motore venga creato correttamente attraverso il richiamo della funzione `draw_motor_in_COMSOL`. Nel caso in cui in `dataSet` fosse rappresentata una macchina personalizzata, la funzione verifica se l'utente abbia confermato di voler salvare la suddetta macchina. In tal caso, il motore viene disegnato e i file necessari vengono gestiti di conseguenza, attraverso il seguente codice:

```
31 if dataSet.custom
32     if isequal(button, 'Yes')
33         [geo, mat] = draw_motor_in_COMSOL(geo, mat, pathname,
               filename);
34
35         fileTmp = [cd '\tmp\' filename];
36         copyfile(fileIn, fileTmp);
37         copyfile(fileTmp, [pathname filename]);
38         delete(fileTmp);
39
40         if isfile([pathname fileans])
41             delete([pathname fileans]);
42         end
43
44         [geo, mat] = draw_motor_in_COMSOL(geo, mat, pathname,
               filename);
45     else
46         disp('Custom machine not saved');
47     end
48 else
49     [geo, mat] = draw_motor_in_COMSOL(geo, mat, pathname,
           filename);
50 end
```

Le strutture `if` nello script di sopra hanno lo scopo di verificare che se `dataSet` rappresenta una macchina custom e l'utente ha confermato di volerla salvare, si eseguano le seguenti operazioni:

1. Disegno della geometria del motore con `draw_motor_in_COMSOL`, attraverso i dati contenuti nelle strutture `geo`, `mat`, `pathname` e `filename`.

2. Creazione di un file temporaneo (`fileTmp`) nella cartella corrente.
3. Copia del file di input originale (`fileIn`) nel file temporaneo (`fileTmp`).
4. Copia del file temporaneo (`fileTmp`) dove specificato da `pathname` e `filename`.
5. Eliminazione del file temporaneo (`fileTmp`).
6. Se esiste già un file identico nella directory di destinazione, viene eliminato per evitare conflitti.
7. Utilizzando nuovamente la funzione `draw_motor_in_COMSOL` con i parametri aggiornati, il motore viene ridisegnato.

Se l'utente ha deciso di non salvare una macchina custom, la funzione disegna semplicemente il motore attraverso l'utilizzo di `draw_motor_in_COMSOL`, con i parametri specificati in (geo, mat, pathname, filename).

- Aggiornamento `dataSet` e Salvataggio Finale dei Dati

  Una volta definita la geometria del motore, vengono eseguite le operazioni necessarie all'aggiornamento della struttura `dataSet` e salvati i dati della macchina. La parte di codice dedicata a tali scopi è la seguente:

```
56 if dataSet.custom == 0 || (isequal(button, 'Yes') && dataSet.
       custom)
57     geo.RQ = RQ;
58
59     filename = strrep(filename, 'mph', 'mat');
60     dataSet.currentpathname = [pathname '\'];
61     dataSet.currentfilename = filename;
62     dataSet.slidingGap = 1; % R347
63     if dataSet.Qs == 6 * dataSet.Num3PhaseCircuit * dataSet.
           NumOfSlots * dataSet.NumOfPolePairs
64         dataSet.slidingGap = 0;
65     end
66
67     dataSet.RQ = round(dataSet.RQ, 4);
68     dataSet.currentpathname = pathname;
69     dataSet.currentfilename = filename;
70
71     geo = orderfields(geo);
72     per = orderfields(per);
73     dataSet = orderfields(dataSet);
74     mat = orderfields(mat);
```

```
75    save([pathname filename], 'geo', 'per', 'dataSet', 'mat');
76 end
```

Nello script di sopra, sono eseguite le seguenti operazioni:

1. Assegnazione della variabile, che contiene i dati di ottimizzazione, `RQ` alla struttura `geo`, aggiornando i parametri geometrici del motore.

2. Modifica dell'estensione del `filename` da `.mph` a `.mat`, preparando il nome del file per il salvataggio dei dati.

3. Aggiornamento dei campi `currentpathname` e `currentfilename` di `dataSet` con i nuovi valori del percorso e del nome del file.

4. Impostazione del parametro `slidingGap` a 1 per indicare la presenza di una boundary condition scorrevole. Tuttavia, se si decidesse di simulare l'intera macchina viene impostato a 0, indicando che non è necessario una condizione a contorno di questo tipo, poiché non c'è necessità di ricorrere a discorsi di simmetria/antisimmetria che si rendono necessari quando si simula solo una parte della macchina.

5. Arrotondamento del valore delle variabili contenute in `RQ` a quattro cifre decimali e aggiorna `dataSet`.

6. Organizzazione dei campi delle strutture `geo`, `per`, `dataSet` e `mat` in ordine alfabetico utilizzando `orderfields`, garantendo che i dati siano strutturati in modo coerente.

7. Salvataggio dei dati aggiornati in un file con il nome e il percorso specificati, includendo le strutture `geo`, `per`, `dataSet` e `mat`, assicurando che tutte le informazioni rilevanti della macchina siano correttamente archiviate.

3.2 draw_motor_in_COMSOL

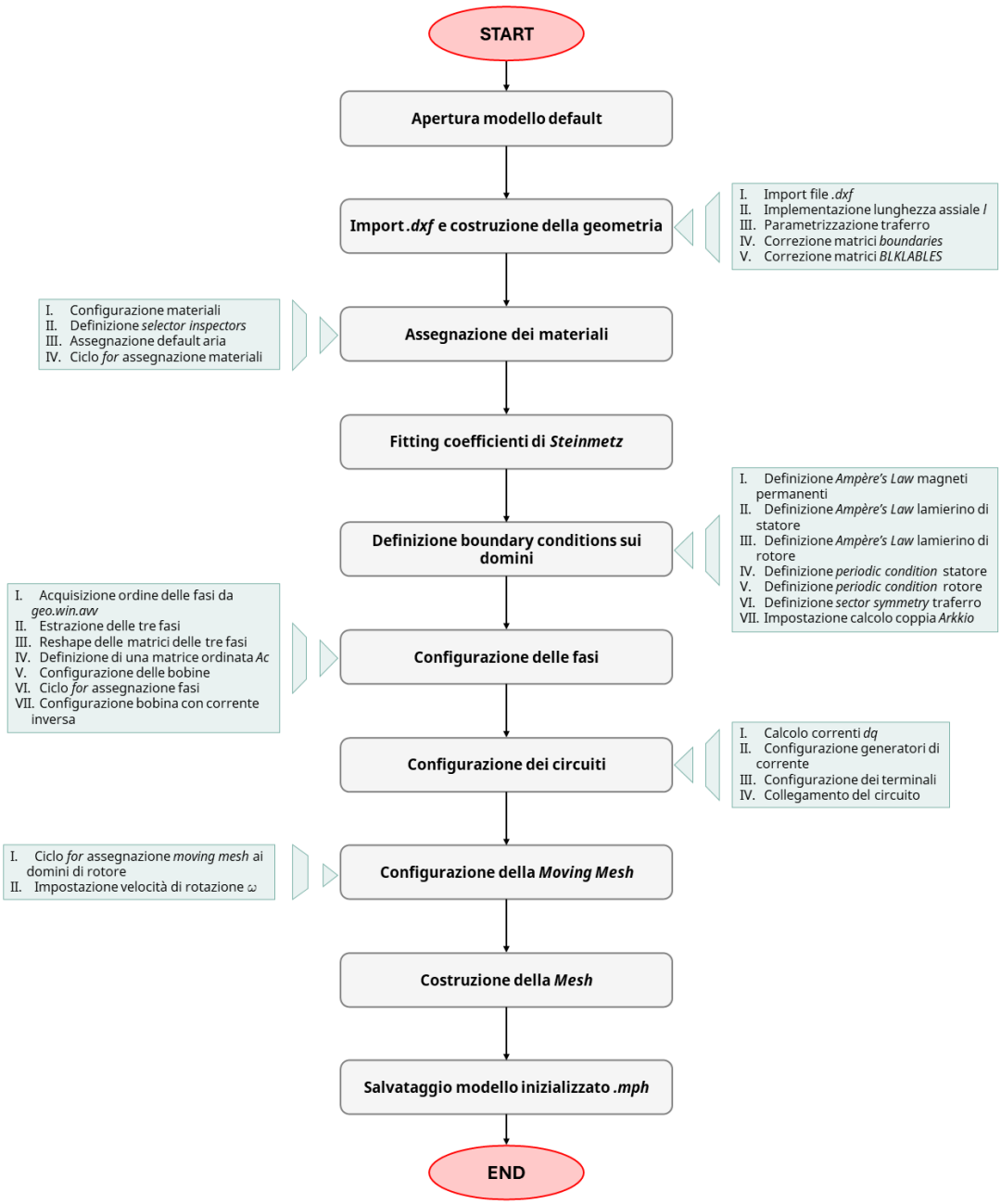

Figura 3.4. Flusso logico di draw_motor_in_COMSOL

La funzione `draw_motor_in_COMSOL`, parte di `DrawandSaveMachine_COMSOL`, è progettata per automatizzare il processo di configurazione del modello di macchina elettrica, sviluppato preliminarmente in SyR-e, utilizzando il tool LiveLink *for* MATLAB per eseguire i comandi di COMSOL Multiphysics. La funzione segue il flusso logico illustrato nella Figura 3.4, che dettaglia le varie fasi necessarie per preparare il modello di simulazione completo del motore.

Di seguito, è presentata una panoramica generale delle operazioni che caratterizzano questa funzione:

1. **Inizializzazione del Modello:**
   - Caricamento del modello COMSOL di default.
   - Estrazione delle informazioni geometriche e dei materiali dalle strutture di input `geo` e `mat`.
2. **Definizione delle Selezioni:**
   - Creazione di selection inspectors con lo scopo di identificare il numero di selezione dei diversi domini/contorni della geometria nel modello COMSOL.
3. **Assegnazione del Materiale di Default (Aria):**
   - Impostazione iniziale dell'aria come materiale predefinito per tutti i domini del modello.
4. **Assegnazione dei Materiali ai Domini:**
   - Identificazione e assegnazione dei materiali appropriati ai vari domini basandosi sulle informazioni fornite dal selection inspector.
5. **Calcolo dei Parametri di Simulazione:**
   - Calcolo della frequenza di alimentazione.
   - Fitting dei parametri di Steinmetz per il calcolo delle perdite nel ferro, attraverso l'utilizzo della funzione `createFit`.
6. **Definizione delle Condizioni al Contorno:**
   - Configurazione delle condizioni al contorno per il calcolo del campo magnetico nelle varie regioni del modello.
7. **Definizione delle Fasi della Macchina:**

- Creazione e impostazione delle bobine per le diverse fasi della macchina.
- Inclusione del calcolo delle perdite per ciascuna fase.

8. **Definizione delle Condizioni di Periodicità:**
   - Configurazione delle condizioni di antiperiodicità e di simmetria per rotore e statore.
9. **Configurazione della Mesh:**
   - Definizione della mesh per il dominio rotante per garantire una discretizzazione adeguata del modello.
10. **Salvataggio del Modello:**
    - Salvataggio del modello configurato in un file `.mph` specificato.

Questa configurazione del modello è essenziale per ottenere una simulazione FEA accurata della macchina. Ogni aspetto, dalla configurazione iniziale alla definizione dettagliata dei materiali e delle condizioni al contorno, fino alla configurazione della mesh e al salvataggio del modello finale, viene impostato per garantire la massima accuratezza nell'analisi.

3.2.1 Input della Funzione

- `geo`: La struttura contenente tutti i parametri geometrici del motore. Comprende dati come: diametri, lunghezze e spessori delle varie parti del motore. Inoltre, definisce anche la configurazione spaziale e le relazioni geometriche tra i vari componenti.
- `mat`: La struttura che contiene le proprietà dei materiali utilizzati nel modello. Include dati come la permeabilità magnetica, la conducibilità elettrica e altre proprietà rilevanti per la simulazione FEA.
- `model`: Un file `.mph` contenente un modello COMSOL pre-configurato con informazioni generali per l'analisi di geometrie 2D, in cui verrà disegnato e configurato il modello del motore impostato in SyR-e.

3.2.2 Output della Funzione

- `model`: Il file `.mph` di default, caricato in precedenza, contenente il modello COMSOL aggiornato che include il disegno e la configurazione del motore secondo le specifiche fornite nelle strutture `geo` e `mat`.

3.2.3 Dettagli delle operazioni

1. Costruzione della Geometria

   All'inizio, si apre il modello COMSOL pre-configurato e si estraggono le informazioni geometriche e materiali dalle strutture `geo` e `mat`. La prima operazione della funzione consiste nel sovrascrivere il modello COMSOL di default e nel definire un componente principale, chiamato `comp1`.

```
% Connessione a COMSOL e Apertura modello default
import com.comsol.model.*
import com.comsol.model.util.*
model = mphopen('Test_auto.mph');
```

   A questo punto, si inizia a configurare la geometria del motore importando i file `stat_dxf` e `rot_dxf`, che rappresentano rispettivamente la geometria dello statore e quella del rotore ottenute dal design preliminare in SyR-e. Successivamente, le due geometrie importate vengono combinate attraverso il comando `Form Assembly`, necessario per indicare a COMSOL la presenza di uno *sliding gap* nella zona del traferro della macchina.

   A seguito della creazione dell'assembly, COMSOL assegna automaticamente una condizione di *Identity Pair* all'arco di circonferenza che separa la geometria del rotore da quella dello statore, rappresentando il centro del traferro nella realtà. Infine, vengono assegnate le unità di misura alla geometria importata, generalmente millimetri, e si definisce la lunghezza assiale della macchina tramite il parametro `l`.

```
% ====== Import e Costruzione della Geometria ====== %

geom = model.component('comp1').geom('geom1');
model.component('comp1').geom('geom1').create('imp1', 'Import'
    );
model.component('comp1').geom('geom1').feature('imp1').set('
    filename', rot_dxf);
model.component('comp1').geom('geom1').feature('imp1').
    importData();
model.component('comp1').geom('geom1').create('imp2', 'Import'
    );
model.component('comp1').geom('geom1').feature('imp2').set('
    filename', stat_dxf);
model.component('comp1').geom('geom1').feature('imp2').
    importData();
```

```
model.component('comp1').geom('geom1').feature('fin').set('
    action', 'assembly');
model.component('comp1').geom('geom1').run('fin');
model.component('comp1').geom('geom1').lengthUnit('mm');

% Implementazione lunghezza assiale
model.component('comp1').physics('rmm').prop('d').set('d', 'l
    [mm]');
```

Queste operazioni di importazione e inizializzazione configurano il modello della macchina in esame nell'interfaccia COMSOL.

2. Parametrizzazione del Traferro

   In questa sezione, si configura l'air gap, ovvero lo spazio tra il rotore e lo statore del motore. Partendo dalle considerazioni fatte in precedenza sulla geometria, si parametrizza il traferro della macchina calcolando le coordinate dei punti che formano le metà del traferro assegnate rispettivamente al rotore e allo statore. Queste coordinate vengono quindi implementate nelle matrici che definiscono le geometrie (`geo.rotor` e `geo.stator`) e nelle matrici che contengono l'assegnazione dei materiali alle coordinate (`geo.BLKLABELS.statore.xy` e `geo.BLKLABELS.rotore.xy`).

   Per il rotore, si calcolano le coordinate del punto medio del traferro al bordo del rotore e le coordinate dell'estremità del rotore. Allo stesso modo, per lo statore, si calcolano le coordinate del punto medio del traferro al bordo dello statore e le coordinate dell'estremità dello statore. Inoltre, vengono definiti e assegnati specifici indici per i domini del rotore e dello statore, insieme a un indice particolare per l'air gap stesso.

```
% Implementazione Air Gap
%rotore
xre2_n = geo.r + geo.g/2;
yre2_n = 0;
xre3_n = (geo.r + geo.g/2)*cos(pi/geo.p*geo.ps);
yre3_n = (geo.r + geo.g/2)*sin(pi/geo.p*geo.ps);

xra2_n = geo.r;
yra2_n = 0;
xra3_n = geo.r*cos(pi/geo.p*geo.ps);
yra3_n = geo.r*sin(pi/geo.p*geo.ps);

index_el_r = max(geo.rotor(:,9)) + 1;
```

```
air_r = 1;

%statore
xse1_n = geo.r + geo.g;
yse1_n = 0;
xse2_n = (geo.r + geo.g)*cos(pi/geo.p*geo.ps);
yse2_n = (geo.r + geo.g)*sin(pi/geo.p*geo.ps);

xsi1_n = geo.r + geo.g/2;
ysi1_n = 0;
xsi2_n = (geo.r + geo.g/2)*cos(pi/geo.p*geo.ps);
ysi2_n = (geo.r + geo.g/2)*sin(pi/geo.p*geo.ps);

index_el_s = max(geo.stator(:,9)) + 1;
air_s = 2;
```

Una volta calcolate le coordinate, queste vengono assegnate alle geometrie di statore e rotore. Le nuove coordinate vengono aggiunte alle matrici esistenti (`geo.rotor` e `geo.stator`). Successivamente, si calcola il numero totale di domini, che rappresenta il totale delle aree definite all'interno del modello del motore. Si rende necessario aggiornare il numero di domini totali, dato che con l'operazione di implementazione delle due metà dell'air gap, si hanno di fatto due domini in più da considerare e a cui assegnare un materiale in seguito. Questo numero è fondamentale per la corretta parametrizzazione dell'interfaccia, in quanto è alla base di molti dei successivi cicli `for` che si occupano di assegnazioni di materiali e condizioni al contorno.

```
%assegnazione coordinate
geo_stator = [];
geo_rotor = [];
geo.stator_n = [xsi1_n, ysi1_n, xse1_n, yse1_n, NaN, NaN, 0,
    air_s, index_el_s;
                0, 0, xse1_n, yse1_n, xse2_n, yse2_n, 1, air_s
                    , index_el_s;
                xse2_n, yse2_n, xsi2_n, ysi2_n, NaN, NaN, 0,
                    air_s, index_el_s
                ];
geo.rotor_n = [xra2_n, yra2_n, xre2_n, yre2_n, NaN, NaN, 0,
    air_r, index_el_r;
              0, 0, xre2_n, yre2_n, xre3_n, yre3_n, 1, air_r,
                  index_el_r;
```

```
94                  xre3_n, yre3_n, xra3_n, yra3_n, NaN, NaN, 0,
                        air_r, index_el_r;];
95
96  geo_stator = [geo.stator; geo.stator_n];
97  geo_rotor = [geo.rotor; geo.rotor_n];
98
99  Ndomains_n = max(geo_stator(:,end))+(geo.Qs)+(geo.Qs-1)+max(
        geo_rotor(:,end));
```

Infatti, per garantire che le condizioni al contorno siano correttamente definite, si calcolano le coordinate dei punti medi per i bordi del rotore e dello statore. Queste coordinate vengono poi assegnate a matrici specifiche che rappresentano i boundaries (`geo.BLKLABELS.statore.boundary` e `geo.BLKLABELS.rotore.boundary`). Tali matrici vengono aggiornate per includere le nuove coordinate, assicurando che le interfacce tra i diversi domini, appena aggiunti alla geometria, siano correttamente definite.

```
102 % Correzione matrici boundaries
103 stat_boundary = [];
104 rot_boundary = [];
105
106 %rotore
107 x_bnd_r_1 = (xra2_n + xre2_n)/2;
108 y_bnd_r_1 = (yra2_n + yre2_n)/2;
109 x_bnd_r_2 = (xre3_n + xra3_n)/2;
110 y_bnd_r_2 = (yre3_n + yra3_n)/2;
111
112 %statore
113 x_bnd_s_1 = (xsi1_n + xse1_n)/2;
114 y_bnd_s_1 = (ysi1_n + yse1_n)/2;
115 x_bnd_s_2 = (xse2_n + xsi2_n)/2;
116 y_bnd_s_2 = (yse2_n + ysi2_n)/2;
117
118 %assegnazione coordinate
119 gm.s.boundary = [x_bnd_s_1, y_bnd_s_1, 10;
120                  x_bnd_s_2, y_bnd_s_2, 10
121                  ];
122 gm.r.boundary = [x_bnd_r_1, y_bnd_r_1, 10;
123                  x_bnd_r_2, y_bnd_r_2, 10
124                  ];
125
```

```
126 stat_boundary = [geo.BLKLABELS.statore.boundary; gm.s.boundary
    ];
127 rot_boundary = [geo.BLKLABELS.rotore.boundary; gm.r.boundary];
```

Infine, si calcolano le coordinate del punto medio dell'arco per il rotore e lo statore. Questi punti medi sono utili per la definizione dei materiali nei passaggi successivi dell'impostazione del modello in COMSOL Multiphysics. Infatti, le matrici dei materiali vengono aggiornate per includere queste nuove coordinate, garantendo che i materiali assegnati alle diverse parti del motore siano correttamente definiti.

```
1  % Correzione matrici materiali (BLKLABELS)
2  stat_mat = [];
3  rot_mat = [];
4
5  %definizione punto medio
6  xm_arc = geo.r*cos(pi/geo.p);
7  ym_arc = geo.r*sin(pi/geo.p);
8
9  %rotore
10 xm_rot = xm_arc + geo.g/4;
11 ym_rot = ym_arc + geo.g/4;
12
13 %statore
14 xm_stat = xm_arc + (3*geo.g/4);
15 ym_stat = ym_arc + (3*geo.g/4);
16
17 stat_mat = [geo.BLKLABELS.statore.xy; xm_stat, ym_stat, 2,
       1.6667, 1];
18 rot_mat = [geo.BLKLABELS.rotore.xy; xm_rot, ym_rot, 1, 1.6667,
        1, 0, 0, 0];
```

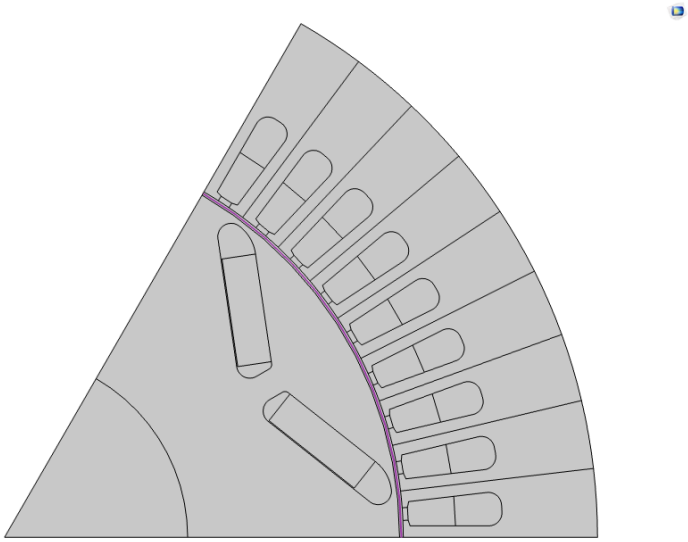

Figura 3.5. Creazione *Identity Pair* al traferro

3. Assegnazione dei Materiali

   I materiali sono assegnati ai domini in base alle categorie specificate all'interno delle matrici `geo.BLKLABELS.rotore.xy` e `geo.BLKLABELS.rotore.xy`, utilizzando un ciclo `for` per iterare su ogni dominio e assegnare il materiale corrispondente.

   Prima di tutto, è necessario definire i materiali, come illustrato nello script seguente. COMSOL offre una vasta selezione di materiali predefiniti per le simulazioni, ma, se necessario, è possibile aggiungere manualmente nuovi materiali alla libreria esistente. Nel codice sottostante viene mostrato un esempio di definizione di un materiale utilizzato nelle simulazioni per caratterizzare i magneti permanenti del motore in progetto.

```
370 % N52 PM
371 model.component('comp1').material().create('mat4', 'Common');
372 model.component('comp1').material('mat4').propertyGroup().
    create('RemanentFluxDensity', 'Remanent flux density');
373 model.component('comp1').material('mat4').label('N52 (Sintered
    NdFeB)');
374 model.component('comp1').material('mat4').set('family', '
    chrome');
375 model.component('comp1').material('mat4').propertyGroup('def')
    .set('electricconductivity', {'1/1.4[uohm*m]', '0', '0', '0
    ', '1/1.4[uohm*m]', '0', '0', '0', '1/1.4[uohm*m]'});
```

```
376 model.component('comp1').material('mat4').propertyGroup('def')
        .set('relpermittivity', {'1', '0', '0', '0', '1', '0', '0',
         '0', '1'});
377 model.component('comp1').material('mat4').propertyGroup('
        RemanentFluxDensity').set('murec', {'1.05', '0', '0', '0',
        '1.05', '0', '0', '0', '1.05'});
378 model.component('comp1').material('mat4').propertyGroup('
        RemanentFluxDensity').set('normBr', '1.44[T]');
379 model.component('comp1').material('mat4').set('family', '
        chrome');
```

Una volta definiti tutti i materiali con le rispettive proprietà necessarie alla tipologia di simulazione che si vuole effettuare, si passa alla definizione dei *selection inspector*. Con questo termine, ci si riferisce a delle entità di tipo *disk*, di dimensioni trascurabili, rispetto a quelle della geometria in esame, che saranno utilizzati per identificare il numero di dominio/boundary che COMSOL assegna automaticamente alle diverse parti che compongono la geometria. Così facendo, è possibile individuare di che tipo di *selection* si tratta, in base alle coordinate e al numero corrispondente al materiale indicati nelle matrici `geo.BLKLABELS.rotore.xy` e `geo.BLKLABELS.statore.xy`.

L'assegnazione dei materiali ai diversi domini della geometria comincia con l'assegnazione dell'aria, come materiale di default, a tutti i domini, per poi essere sovrascritta dagli altri materiali nei rispettivi domini di assegnazione. Nell'ottica di ottimizzazione del codice, si nota che all'assegnazione dell'aria segue anche la definizione della prima boundary condition, ovvero una legge di Ampere e l'assegnazione del calcolo delle perdite nei domini che contengono aria.

Successivamente, vengono inizializzati i vettori temporanei (caratterizzati dalla radice `tmp`) che saranno utilizzati all'interno della struttura iterativa per l'assegnazione effettiva dei materiali ai domini, insieme con i vettori che ospiteranno i numeri di selection dei diversi domini.

```
381 % Definizione selection inspector
382 model.component('comp1').selection().create('disk1', 'Disk');
383 model.component('comp1').selection().create('disk2', 'Disk');
384
385 % Assegnazione default Aria
386 sel = 1:Ndomains_n;
387 model.component('comp1').material('mat1').selection().set(sel)
        ;
```

```
388 model.component('comp1').physics('rmm').feature('al1').label("
        Ampere's Law - Air");
389 model.component('comp1').physics('rmm').feature('al1').create(
        'loss1', 'LossCalculation', 2);
390
391 % Vettori per assegnazione materiali
392 tmp = [];
393 tmp_rot = rot_mat(:, 1:3);
394 tmp_stat = stat_mat(:, 1:3);
395 tmp = [tmp_rot; tmp_stat];
396 disk1 = model.component('comp1').selection('disk1');
397
398 % Vettori per settare i domini
399 A = [];
400 B = [];
401 C = [];
```

Infine, l'assegnazione dei materiali viene effettuata tramite un ciclo `for`, in cui le matrici `rot_mat` e `stat_mat` vengono esaminate per estrarre le coordinate dei punti da assegnare al selector inspector. Utilizzando il comando `disk1.entities()`, il numero del dominio identificato dalle coordinate x e y correnti viene restituito. Successivamente, si analizza il numero contenuto nella terza colonna della matrice `tmp`, che contiene informazioni sul tipo di materiale da assegnare, come specificato nella progettazione preliminare della macchina in SyR-e. Il numero del dominio in esame viene quindi assegnato a uno dei vettori A, B o C. Questo processo si ripete per tutti i domini della geometria. Alla fine del ciclo, i diversi materiali vengono assegnati ai rispettivi domini.

```
403 % Assegnazione materiali
404 for kk = 1:Ndomains_n
405     x = tmp(kk,1);
406     y = tmp(kk,2);
407     disk1.set('posx', x);
408     disk1.set('posy', y);
409     selNumber = disk1.entities();
410     switch tmp(kk, 3)
411         case 3
412             A = horzcat(A, selNumber);
413         case {4, 5}
414             B = horzcat(B, selNumber);
```

```
415         case 6
416             C = horzcat(C, selNumber);
417     end
418 end
419
420 model.component('comp1').material('mat3').selection().set(A);
421 model.component('comp1').material('mat2').selection().set(B);
422 model.component('comp1').material('mat4').selection().set(C);
```

Questo processo non solo stabilisce le proprietà elettriche e magnetiche di ogni regione del motore, ma prepara anche il terreno per la modellazione fisica dettagliata, che verrà eseguita più avanti dall'interfaccia.

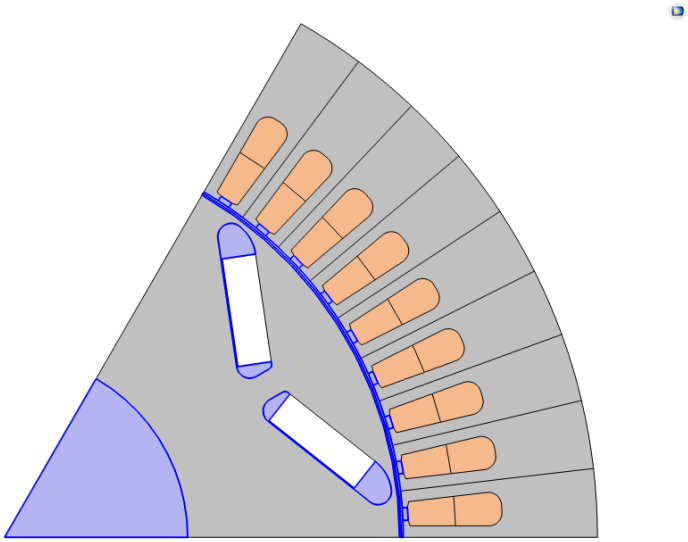

Figura 3.6. Assegnazione materiali al modello

4. Fitting dei Coefficienti di Steinmetz

   Prima di definire le condizioni al contorno del modello, viene eseguita una procedura per adattare i parametri del modello Steinmetz per il calcolo delle perdite nel ferro, calcolando i parametri `KH`, `ALPHA` e `BETA` tramite un fitting dei dati. La calibrazione di questi parametri è fondamentale per garantire che la simulazione rifletta accuratamente il comportamento reale del motore elettrico. In questo caso, i parametri `KH`, `ALPHA` e `BETA` sono determinati attraverso l'utilizzo della funzione `createFit`, che realizza il fitting partendo dai dati contenuti nella matrice `mat` fornita da SyR-e.

Questo passaggio assicura che il modello COMSOL sia calibrato al meglio per fornire risultati in linea con le condizioni operative reali del motore.

```
% Fitting Steinmetz per calcolare KH, ALPHA, BETA
ff = linspace(50, freq, 51);
Bf = linspace(0, Bf_max, 51);
kh = mat.Rotor.kh;
ke = mat.Rotor.ke;
alpha = mat.Rotor.alpha;
beta = mat.Rotor.beta;
pfe_f = kh .*(ff .^alpha) .*(Bf .^beta) +  ke .*(ff .^2) .*(Bf
    .^2);

[fitresult, gof] = createFit(ff, Bf, pfe_f);
KH = fitresult.KH;
ALPHA = fitresult.ALPHA;
BETA = fitresult.BETA;
```

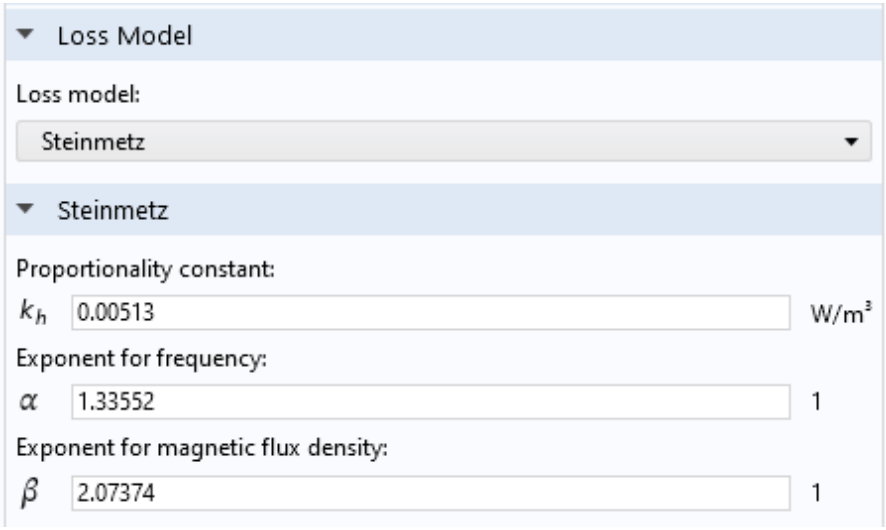

Figura 3.7. Coefficienti di Steinmetz ottenuti dal Fitting

5. Definizione delle Condizioni al Contorno sui Domini

   Prima di tutto, vengono definite le condizioni al contorno per i magneti permanenti presenti nella geometria del rotore. Attraverso un ciclo `for`, illustrato di seguito, si iterano le righe della matrice `rot_mat`. Per ogni riga, viene verificato il tipo di materiale corrispondente alle coordinate x, y correnti. Se il valore nella terza colonna

è 6 (il codice materiale per il magnete permanente), vengono estratte le coordinate xm, ym e zm che indicano le direzioni di magnetizzazione del magnete permanente in questione. Nella stessa struttura `if`, viene anche incrementato progressivamente il numero delle condizioni al contorno *Ampère's Law* per assegnarne una a ciascun magnete, permettendo così la parametrizzazione di questa parte dello script. Infine, vengono impostate le relazioni costitutive della boundary condition, specificamente la *Remanent Flux Density*, e si richiede il calcolo delle perdite anche in questa fase.

```
447 % Definizione delle condizioni al contorno PM
448 tmp_rot_righe = size(rot_mat, 1);
449 tmp_stat_righe = size(stat_mat, 1);
450 BC_pm = [];
451 AM = [];
452
453 for kk = 1:tmp_rot_righe
454     if rot_mat(kk, 3)==6
455         BC_pm = horzcat(BC_pm, disk1.entities());
456         AM = [rot_mat(kk, 6), rot_mat(kk, 7), rot_mat(kk, 8)];
457         alnumber_m = ['al_m' num2str(kk+1)];
458         model.component('comp1').physics('rmm').create(
                alnumber_m, 'AmperesLaw', 2);
459         model.component('comp1').physics('rmm').feature(
                alnumber_m).selection().set(selNumber);
460         model.component('comp1').physics('rmm').feature(
                alnumber_m).set('ConstitutiveRelationBH', '
                RemanentFluxDensity');
461         model.component('comp1').physics('rmm').feature(
                alnumber_m).set('e_crel_BH_RemanentFluxDensity', AM
                );
462         model.component('comp1').physics('rmm').feature(
                alnumber_m).create('loss1', 'LossCalculation', 2);
463     end
464 end
```

Successivamente, vengono definite le condizioni al contorno per i lamierini del statore e del rotore, seguendo una procedura simile a quella descritta per i magneti permanenti.

Nello script riportato di seguito, le boundary conditions *Ampère's Law* vengono applicate ai lamierini del statore. Attraverso l'iterazione delle righe della matrice `stat_mat` e l'uso dello selection inspector, vengono identificati i numeri dei domini

come mostrato nella Figura 3.8. Anche in questo caso, il criterio distintivo utilizzato è il codice del materiale contenuto nella matrice geometrica.

Terminata la fase di identificazione delle selezioni, si procede alla configurazione delle relazioni costitutive della fisica specifica, utilizzando *BH Curve* per il ferro, trattandosi di un materiale ferromagnetico non lineare. Inoltre, si attiva il calcolo delle perdite nel ferro utilizzando i coefficienti ottenuti dal fitting descritto precedentemente.

```
% Definizione boundary condition su ferro di statore
BC_fe_s = [];

for kk = 1:size(tmp_stat, 1)
    x = tmp_stat(kk,1);
    y = tmp_stat(kk,2);
    disk1.set('posx', x);
    disk1.set('posy', y);
    selNumber = disk1.entities();
    if tmp_stat(kk, 3)==4
       BC_fe_s = horzcat(BC_fe_s, selNumber);
    end
end

model.component('comp1').physics('rmm').create('al_fs', '
    AmperesLaw', 2);
model.component('comp1').physics('rmm').feature('al_fs').
    selection().set(BC_fe_s);
model.component('comp1').physics('rmm').feature('al_fs').set('
    ConstitutiveRelationBH', 'BHCurve');
model.component('comp1').physics('rmm').feature('al_fs').
    create('loss_f', 'LossCalculation', 2);
model.component('comp1').physics('rmm').feature('al_fs').
    feature('loss_f').set('LossModel', 'Steinmetz');
model.component('comp1').physics('rmm').feature('al_fs').label
    ("Ampere's Law - Stator");
model.component('comp1').physics('rmm').feature('al_fs').
    feature('loss_f').set('kh_steinmetz', KH);
model.component('comp1').physics('rmm').feature('al_fs').
    feature('loss_f').set('alpha', ALPHA);
model.component('comp1').physics('rmm').feature('al_fs').
    feature('loss_f').set('beta_steinmetz', BETA);
```

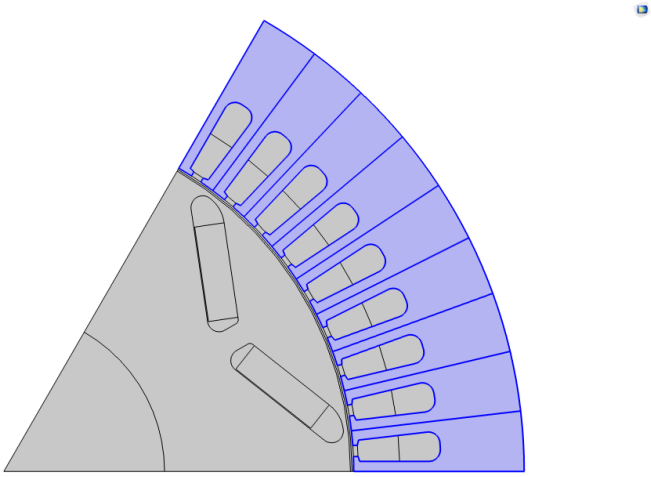

Figura 3.8. Assegnazione *Ampère's Law* al lamierino di statore

6. Definizione delle Condizioni di Periodicità e Simmetria

   Le condizioni periodiche sono impostate su alcuni bordi per garantire la continuità o l'antiperiodicità tra le aree specificate. Questa fase di configurazione del modello richiede tali condizioni per ottimizzare i tempi di simulazione, consentendo la modellazione di parti isolate della macchina mentre si preserva l'integrità dei risultati attraverso la simmetria geometrica e magnetica.

   Nello script seguente, viene mostrata l'assegnazione di queste condizioni ai bordi laterali dello statore, utilizzando un procedimento simile a quelli precedentemente descritti. Vengono identificati i numeri delle selezioni dei bordi e, in base al numero di poli simulati `geo.ps`, viene applicata la condizione di continuità o antiperiodicità utilizzando la boundary condition *Periodic Condition* fornita da COMSOL.

   Questo approccio garantisce una modellazione accurata delle interazioni tra rotore e statore, tenendo conto dell'influenza delle correnti indotte e delle forze magnetomotrici.

```
612 % Vettori assegnazione periodic condition sui bordi
613 tmp = [];
614 tmp_rot = rot_boundary(:, 1:3);
615 tmp_stat = stat_boundary(:, 1:3);
616 tmp_righe_s = size(tmp_stat, 1);
617 tmp_righe_r = size(tmp_rot, 1);
618 AP_s = [];
619 AP_r = [];
```

```
620
621 disk2 = model.component('comp1').selection('disk2').set('
    entitydim', 1);
622 disk2 = model.component('comp1').selection('disk2').set('r',
    0.1);
623
624 % Definizione periodic condition statore
625 for kk = 1:tmp_righe_s
626     x = tmp_stat(kk,1);
627     y = tmp_stat(kk,2);
628     disk2.set('posx', x);
629     disk2.set('posy', y);
630     selNumber = disk2.entities();
631     if tmp_stat(kk,3)==10
632         AP_s = horzcat(AP_s, selNumber);
633     end
634 end
635
636 model.component('comp1').physics('rmm').create('pc1', '
    PeriodicCondition', 1);
637 model.component('comp1').physics('rmm').feature('pc1').
    selection().set(AP_s);
638 if mod(geo.ps, 2)==0
639 model.component('comp1').physics('rmm').feature('pc1').set('
    PeriodicType', 'Continuity');
640 else
641 model.component('comp1').physics('rmm').feature('pc1').set('
    PeriodicType', 'AntiPeriodicity');
642 end
643 model.component('comp1').physics('rmm').feature('pc1').label("
    Periodic Condition - Stator");
```

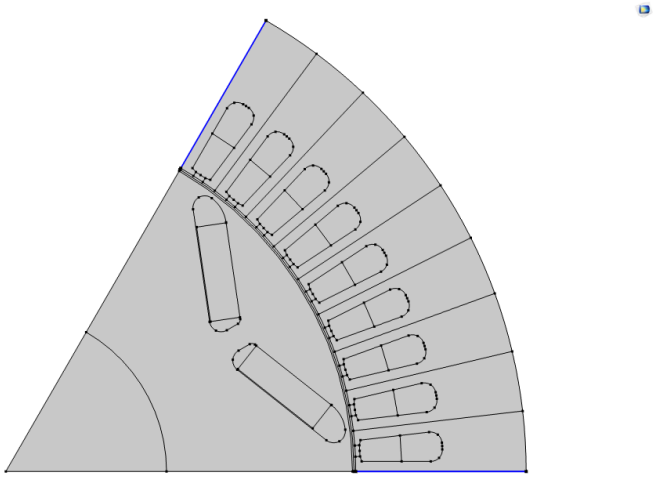

Figura 3.9. Assegnazione condizione di simmetria/antiperiodicità allo statore

Successivamente, viene assegnata la condizione di simmetria sul traferro utilizzando l'*Identity Pair* creato automaticamente da COMSOL, motivo per cui le geometrie di statore e rotore vengono importate separatamente. La condizione *Sector Symmetry* consente di interpretare la geometria totale come composta da due componenti, una rotante e una fissa, specificando contemporaneamente il numero di settori che costituiscono la macchina reale. Questo viene realizzato attraverso i comandi seguenti, rendendo l'interfaccia parametrica e quindi adattabile a tutti i tipi di geometria e modellizzazione.

```
% Definizione di Sector Symmetry su Air Gap
model.component('comp1').physics('rmm').create('ssc1', '
    SectorSymmetry', 1);
model.component('comp1').physics('rmm').feature('ssc1').set('
    pairs', 'ap1');
if mod(geo.ps, 2)==0
model.component('comp1').physics('rmm').feature('ssc1').set('
    nsector', geo.p);
model.component('comp1').physics('rmm').feature('ssc1').set('
    PeriodicType', 'Continuity');
else
model.component('comp1').physics('rmm').feature('ssc1').set('
    nsector', geo.p*2);
model.component('comp1').physics('rmm').feature('ssc1').set('
    PeriodicType', 'AntiPeriodicity');
```

```
675 end
676 model.component('comp1').physics('rmm').feature('ssc1').set('
    constraintOptions', 'weakConstraints');
```

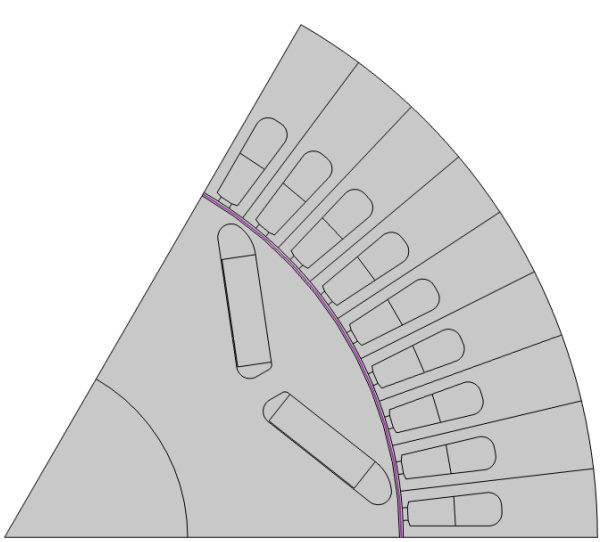

Figura 3.10. Assegnazione condizione di simmetria/antiperiodicità al traferro

7. Definizione della Moving Mesh e Calcolo della Coppia

   Come ultima condizione al contorno da impostare sulla fisica *rmm* di COMSOL, si configura il calcolo della coppia con il metodo di Arkkio. Si è scelto questo metodo di calcolo perché più accurato rispetto ad altri, specialmente nell'analisi di motori sincroni a magneti permanenti, che sono i più comuni in progetto all'interno di SyR-e.

   In ultimo, prima della configurazione dei circuiti, si definisce una condizione che permette alla simulazione di essere effettuata considerando la mesh mobile sulle parti rotanti della geometria, come mostrato in Figura 3.11. La mesh mobile permette a COMSOL di effettuare le simulazioni dinamiche sulla macchina, senza dover necessariamente ricostruire la mesh per ogni posizione di rotore, ma operando un aggiustamento dei triangoli durante il movimento delle geometrie.

```
678 % Definizione Arkkio Torque Calculation
679 model.component('comp1').physics('rmm').create('ark1', '
    ArkkioTorqueCalculation', 2);
680
681 % ============== Definizione Moving Mesh ============== %
```

```
682 T = [];
683 tmp = rot_mat;
684
685 for kk = 1:tmp_rot_righe
686     x = tmp(kk,1);
687     y = tmp(kk,2);
688     disk1.set('posx', x);
689     disk1.set('posy', y);
690     selNumber = disk1.entities();
691     T = horzcat(T, selNumber);
692 end
693
694 model.component('comp1').common().create('rot1', '
    RotatingDomain');
695 model.component('comp1').common('rot1').selection().set(T);
696 model.component('comp1').common('rot1').set('rotationType', '
    rotationalVelocity');
697 model.component('comp1').common('rot1').set('
    rotationalVelocityExpression', 'constantAngularVelocity');
698 model.component('comp1').common('rot1').set('angularVelocity',
     w);
```

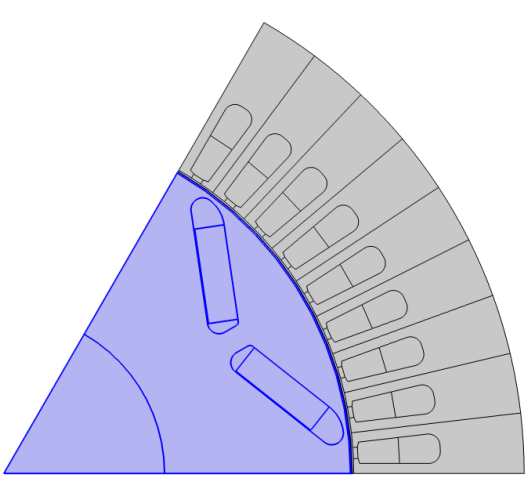

Figura 3.11. Assegnazione *Moving Mesh* al rotore

8. Configurazione delle Fasi di Statore

   Le fasi della macchina sono create utilizzando le *coils*, che la fisica *rmm* incorpora al proprio interno, e i domini sono assegnati alle diverse bobine in base alle specifiche della matrice `geo.win.avv` che rappresenta la distribuzione di fasi e strati impostata dall'utente nella finestra *Windings* di SyR-e (Figura 1.5). COMSOL permette di importare da SyR-e tutte le caratteristiche fondamentali dell'avvolgimento della macchina e di poterle implementare nel modello. Infatti, come indicato nello script di sotto, la configurazione di una fase della macchina prevede la specifica della tipologia di spira (bobina multispire o hairpin), del numero di spire che costituiscono l'avvolgimento del singolo dominio a cui è assegnata tale boundary condition, del fattore di riempimento della cava e, ancora, del tipo di eccitazione a cui è sottoposta la bobina. In quest'ultima sezione è possibile specificare se la macchina è semplicemente alimentata in corrente continua o alternata, ma nel caso di valutazioni più avanzate, relative all'analisi delle performance, è possibile associare ogni bobina a un circuito elettrico, che verrà configurato in seguito nella funzione.

```
model.component('comp1').physics('rmm').create('coil1', 'Coil', 2);
model.component('comp1').physics('rmm').feature('coil1').label('Phase 1');
model.component('comp1').physics('rmm').feature('coil1').set('ConductorModel', 'Multi');
model.component('comp1').physics('rmm').feature('coil1').set('coilGroup', true);
model.component('comp1').physics('rmm').feature('coil1').set('CoilExcitation', 'CircuitCurrent');
model.component('comp1').physics('rmm').feature('coil1').set('N', {num2str(geo.win.Nbob*2*geo.p)});
model.component('comp1').physics('rmm').feature('coil1').set('AreaFrom', 'FillingFactor');
model.component('comp1').physics('rmm').feature('coil1').set('FillingFactor', {num2str(geo.win.kcu)});
```

   Le fasi vengono configurate utilizzando cicli `for` per iterare attraverso le righe e le colonne della matrice `geo.win.avv` e assegnare i domini corrispondenti a ciascuna bobina. La discriminante di questo tipo di distribuzione di domini è il numero assegnato alle fasi nella finestra *Windings* di SyR-e, che viene memorizzato in `geo.win.avv` esattamente nell'ordine in cui deve essere assrgnato, quindi il ciclo assegna ad ogni bobina il corrispondente dominio dividendo le fasi in fase 1, fase 2 e fase 3. Inoltre,

nel caso il numero di fase fosse negativo, assegna la condizione di corrente inversa all'avvolgimento interessato. Infine, viene assegnato il calcolo delle perdite per ogni fase. Di seguito, la parte di codice incaricata di assegnare le bobine ai rispettivi domini.

```
% Creazione delle bobine e assegnazione dei domini
coil1 = [];
coil2 = [];
coil3 = [];

for i = 1:num_righe_avv
    for c = 1:num_colonne_avv
        switch avv(i, c)
            case 1
                coil1 = horzcat(coil1, Ac(i, c));
            case -3
                coil3 = horzcat(coil3, Ac(i, c));
            case 2
                coil2 = horzcat(coil2, Ac(i, c));
        end
    end
end

model.component('comp1').physics('rmm').feature('coil1').
    selection().set(coil1);
model.component('comp1').physics('rmm').feature('coil2').
    selection().set(coil2);
model.component('comp1').physics('rmm').feature('coil3').
    selection().set(coil3);

% Assegna i domini alla bobina con corrente inversa
model.component('comp1').physics('rmm').feature('coil3').
    create('rcd1', 'ReverseCoilGroupDomain', 2);
model.component('comp1').physics('rmm').feature('coil3').
    feature('rcd1').selection().set(coil3);
model.component('comp1').physics('rmm').feature('coil3').
    feature('rcd1').label('Reverse Current Phase 3');

%Assegna calcolo perdite alle bobine
model.component('comp1').physics('rmm').feature('coil1').
    create('loss1', 'LossCalculation', 2);
```

```
30 model.component('comp1').physics('rmm').feature('coil2').
       create('loss1', 'LossCalculation', 2);
31 model.component('comp1').physics('rmm').feature('coil3').
       create('loss1', 'LossCalculation', 2);
```

Questo passaggio non solo facilita la simulazione delle correnti elettriche indotte, ma permette anche di valutare l'efficienza e le prestazioni del motore in base alla distribuzione delle bobine e alla loro interazione con il campo magnetico.

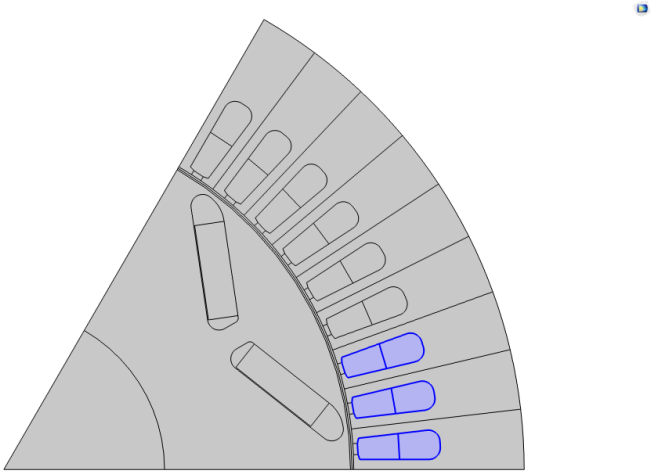

Figura 3.12. Assegnazione della condizione *Coil* alla fase 1

9. Configurazione dei Circuiti

Il completamento del modello di simulazione avviene con l'integrazione del circuito elettrico che interfaccia gli avvolgimenti della macchina con la simulazione all'interno del modello COMSOL. La configurazione del circuito elettrico è essenziale per rappresentare accuratamente le connessioni elettriche tra le bobine e le fonti di alimentazione del motore. La funzione definisce e configura tre circuiti distinti: *I1*, *I2*, e *I3*, che rappresentano le sorgenti di corrente necessarie per il funzionamento delle bobine e per generare il campo magnetico rotante all'interno del motore.

Per impostare correttamente il circuito, la funzione importa da SyR-e l'ampiezza e la fase della corrente (in assi *dq*) specificate nella finestra *Simulation* e ne estrae le componenti, utilizzandole per il calcolo del modulo. A questo punto, definisce i tre circuiti e le connessioni tra di essi. Infine, imposta i tre generatori di corrente come

fonti di corrente sinusoidale, specificandone la frequenza di alimentazione, il modulo della corrente e la fase, sfasando opportunamente le tre fasi di 120° elettrici tra loro.

Questo processo è fondamentale per modellare accuratamente il comportamento dinamico del motore, garantendo che le equazioni del circuito siano risolte in modo efficiente durante la simulazione. Inoltre, permette di simulare e valutare il comportamento del sistema in diverse condizioni di carico e operazione.

```
% ============== Definizione Circuito ============== %

% Calcolo correnti dq
iAmp = per.overload*per.i0;
gamma = per.GammaPP;
theta_i = (geo.th0 + gamma)*pi/180;

id = iAmp*cos(theta_i);
iq = iAmp*sin(theta_i);
Imod = abs(id + 1i*iq);
Iarg = angle(Imod(end));

model.component('comp1').physics().create('cir', 'Circuit', '
    geom1');
model.component('comp1').physics('cir').create('I1', '
    CurrentSourceCircuit', -1);
model.component('comp1').physics('cir').create('I2', '
    CurrentSourceCircuit', -1);
model.component('comp1').physics('cir').create('I3', '
    CurrentSourceCircuit', -1);
model.component('comp1').physics('cir').create('termI1', '
    ModelTerminalIV', -1);
model.component('comp1').physics('cir').create('termI2', '
    ModelTerminalIV', -1);
model.component('comp1').physics('cir').create('termI3', '
    ModelTerminalIV', -1);
model.component('comp1').physics('cir').feature('I2').setIndex
    ('Connections', 1, 0, 0);
model.component('comp1').physics('cir').feature('I2').setIndex
    ('Connections', 3, 1, 0);
model.component('comp1').physics('cir').feature('I3').setIndex
    ('Connections', 1, 0, 0);
```

```
23 model.component('comp1').physics('cir').feature('I3').setIndex
       ('Connections', 4, 1, 0);
24 model.component('comp1').physics('cir').feature('termI1').set(
       'Connections', 2);
25 model.component('comp1').physics('cir').feature('termI2').set(
       'Connections', 3);
26 model.component('comp1').physics('cir').feature('termI3').set(
       'Connections', 4);
27 model.component('comp1').physics('cir').create('R1', 'Resistor
       ', -1);
28 model.component('comp1').physics('cir').feature('R1').set('R',
        '10000 [ohm]');
29 model.component('comp1').physics('cir').feature('R1').setIndex
       ('Connections', 1, 0, 0);
30 model.component('comp1').physics('cir').feature('R1').setIndex
       ('Connections', 0, 1, 0);
31 model.component('comp1').physics('cir').feature('I1').set('
       sourceType', 'SineSource');
32 model.component('comp1').physics('cir').feature('I2').set('
       sourceType', 'SineSource');
33 model.component('comp1').physics('cir').feature('I3').set('
       sourceType', 'SineSource');
34 model.component('comp1').physics('cir').feature('I1').set('
       value', [num2str(Imod) ' [A]']);
35 model.component('comp1').physics('cir').feature('I1').set('
       freq', [num2str(freq) ' [Hz]']);
36 model.component('comp1').physics('cir').feature('I1').set('
       phase', [num2str(theta_i)]);
37 model.component('comp1').physics('cir').feature('I2').set('
       value', [num2str(Imod) ' [A]']);
38 model.component('comp1').physics('cir').feature('I2').set('
       freq', [num2str(freq) ' [Hz]']);
39 model.component('comp1').physics('cir').feature('I2').set('
       phase', [num2str(theta_i) ' - 120*pi/180']);
40 model.component('comp1').physics('cir').feature('I3').set('
       value', [num2str(Imod) ' [A]']);
41 model.component('comp1').physics('cir').feature('I3').set('
       freq', [num2str(freq) ' [Hz]']);
42 model.component('comp1').physics('cir').feature('I3').set('
       phase', [num2str(theta_i) ' + 120*pi/180']);
```

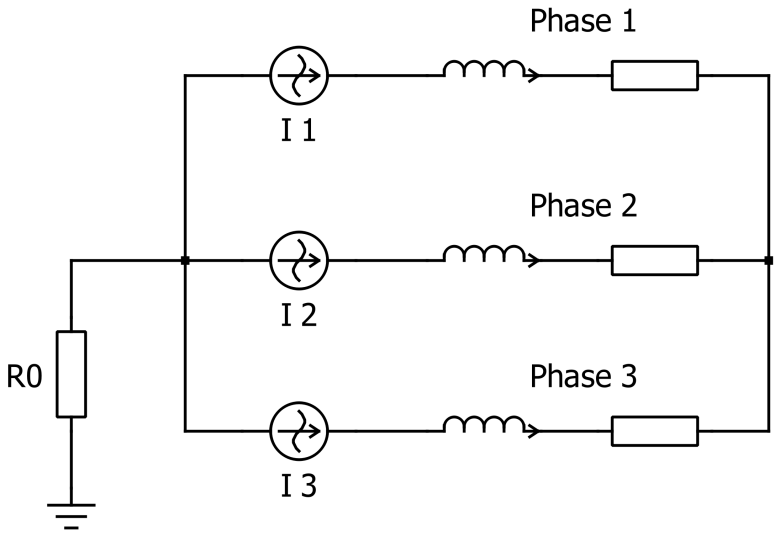

Figura 3.13. Schema del circuito realizzato in COMSOL

10. Configurazione della Mesh

L'ultima operazione di configurazione del modello consiste nella definizione della mesh per garantire la precisione e l'efficienza della simulazione. La funzione prepara una mesh dettagliata per il modello COMSOL, ottimizzando la risoluzione spaziale per catturare accuratamente le variazioni nel campo magnetico all'interno della geometria del motore. COMSOL Multiphysics permette di impostare automaticamente la mesh, come mostrato nello script seguente. Tuttavia, se l'utente desidera impostare manualmente una mesh dettagliata, può scegliere tra 8 livelli di fittezza o configurare tutto manualmente. In quest'ultimo caso, l'interfaccia realizzata nel presente lavoro non offre questa possibilità; quindi, è necessario aprire il file `.mph` del modello della macchina e apportare le modifiche desiderate. Una volta impostate le dimensioni della mesh, il codice la applica alla geometria (Figura 3.14) per concludere l'impostazione del modello e passare alla fase successiva della simulazione.

```
% Configurazione della mesh
model.component('comp1').mesh('mesh1').autoMeshSize(6);
model.component('comp1').mesh('mesh1').run();
```

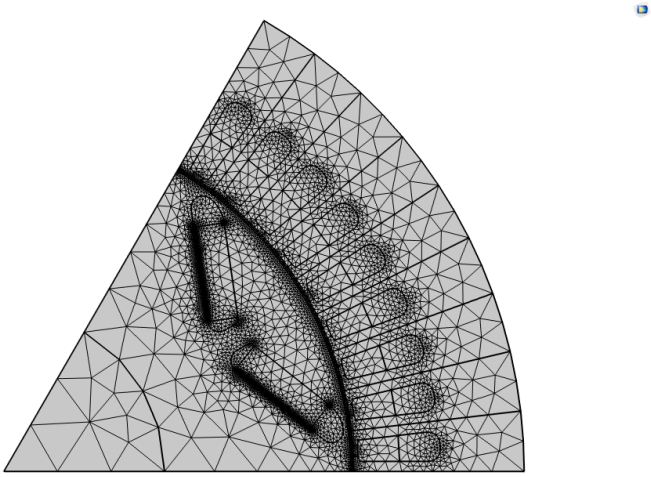

Figura 3.14. Meshing della geometria

Questi punti rappresentano il flusso di lavoro completo e dettagliato per la simulazione di motori elettrici utilizzando COMSOL Multiphysics, integrando aspetti fondamentali della modellazione geometrica, fisica e numerica per fornire una rappresentazione accurata delle prestazioni del motore sotto diverse condizioni di funzionamento.

4 Simulazione magnetica

In questo capitolo viene descritta l'essenza dell'interfaccia che è stata realizzata, ovvero il processo di analisi elettromagnetica del modello della macchina esportata da SyR-e a COMSOL Multiphysics, fondamentale per configurare e ottimizzare le simulazioni FEA all'interno dell'ambiente multifisico di COMSOL ma anche per estrarre i risultati e mostrarli all'utente in ambiente MATLAB. Le funzioni centrali in questo contesto sono `eval_operatingPointCOMSOL`, `COMSOLfitness` e `simulate_xdeg_COMSOL`, illustrate nella Figura 4.1, che costituisce la parte evidenziata in basso a sinistra del diagramma a blocchi presentato nel capitolo precedente.

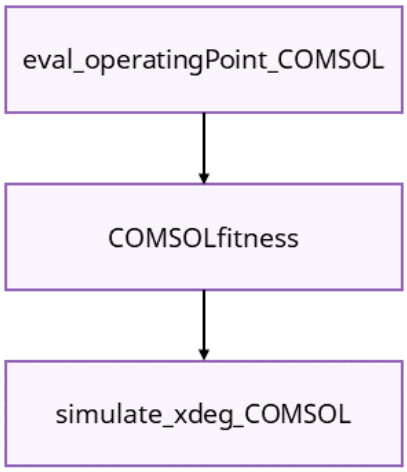

Figura 4.1. Funzioni responsabili della simulazione elettromagnetica del modello

Le funzioni hanno il compito di calcolare e/o estrarre tutti i parametri geometrici, dei materiali e fisici da SyR-e, sfruttando la definizione del *preliminary design*, le impostazioni di simulazione inserite dall'utente nella finestra *Simulation* di SyR-e e le operazioni svolte dalla funzione `draw_motor_in_COMSOL`. Anche in questo caso, come avvenuto per le funzioni presentate nel capitolo precedente, per l'implementazione in COMSOL, ci si

è basati sull'architettura delle funzioni già presenti in SyR-e, cercando di minimizzare le personalizzazioni necessarie nel codice script, considerando le differenze significative tra SyR-e e COMSOL nell'implementazione delle simulazioni agli elementi finiti.

Questo approccio mira a standardizzare le procedure per agevolare gli utenti che già conoscono lo scripting di SyR-e, incoraggiandoli a esplorare personalizzazioni e miglioramenti del software, essendo esso open-source.

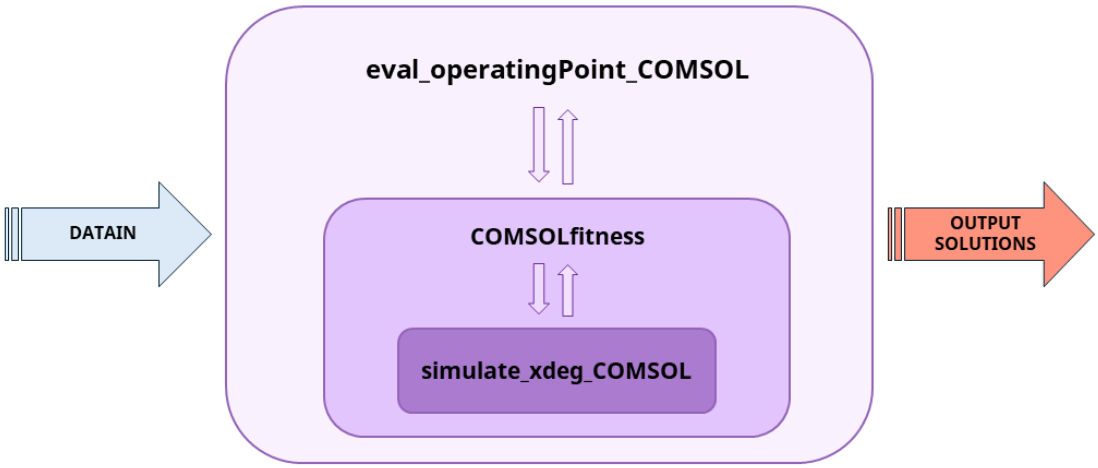

Figura 4.2. Struttura annidata di eval_operatingPoint_COMSOL

Nel dettaglio delle funzioni di simulazione del modello, si osserva dalla Figura 4.2 che `eval_operatingPointCOMSOL` svolge un ruolo fondamentale sia nell'impostazione generale dei parametri utili all'analisi sia nella presentazione dei risultati ottenuti dalla simulazione del modello. Questa funzione gestisce la preparazione dei dati e stabilisce accuratamente i percorsi dei file necessari, utilizza `COMSOLfitness` per gestire i dati in uscita dalla simulazione, mentre la simulazione vera e propria viene lanciata da `simulate_xdegCOMSOL`, con `COMSOLfitness` che funge da interfaccia per avviare la simulazione in COMSOL e ottenere i dati grezzi.

Il post-processing dei dati, come verrà illustrato nei paragrafi seguenti, è eseguito in parte da `COMSOLfitness` e in parte da `eval_operatingPointCOMSOL`. Una volta completato il processo dei dati in background, i risultati vengono presentati all'utente tramite figure, il cui numero e contenuto possono variare in base alla tipologia di analisi svolta.

L'interfaccia SyR-e - COMSOL assicura una preparazione accurata dei dati per le analisi e le simulazioni FEA, garantendo un flusso di lavoro efficiente e ben organizzato per gli utenti, migliorando così la qualità e l'efficienza nell'utilizzo di SyR-e.

4.1 eval_operatingPoint_COMSOL

La funzione `eval_operatingPoint_COMSOL` è progettata con lo scopo di automatizzare il processo di valutazione e visualizzazione delle prestazioni operative del motore, progettato in SyR-e, attraverso una simulazione FEA in COMSOL Multiphysics. Carica i dati di simulazione dal file `.mat` creato da SyR-e e inizializzato tramite `DrawandSaveMachineCOMSOL`, esegue una simulazione FEA utilizzando COMSOL attraverso la funzione `COMSOLfitness` ed elabora e visualizza i risultati ottenuti dalla struttura `SOL`.

La funzione segue il flusso logico definito in Figura 4.3 che definisce le diverse fasi necessarie per lanciare la simulazione del modello e presentare i dati processati all'utente. In più, può essere lanciata direttamente dall'interfaccia GUI di SyR-e attraverso il pulsante *COMSOL*, nella finestra *Simulation*, oppure può essere eseguita in modo autonomo attraverso la *Command Window*, a patto di stare usando LiveLink for MATLAB di COMSOL.

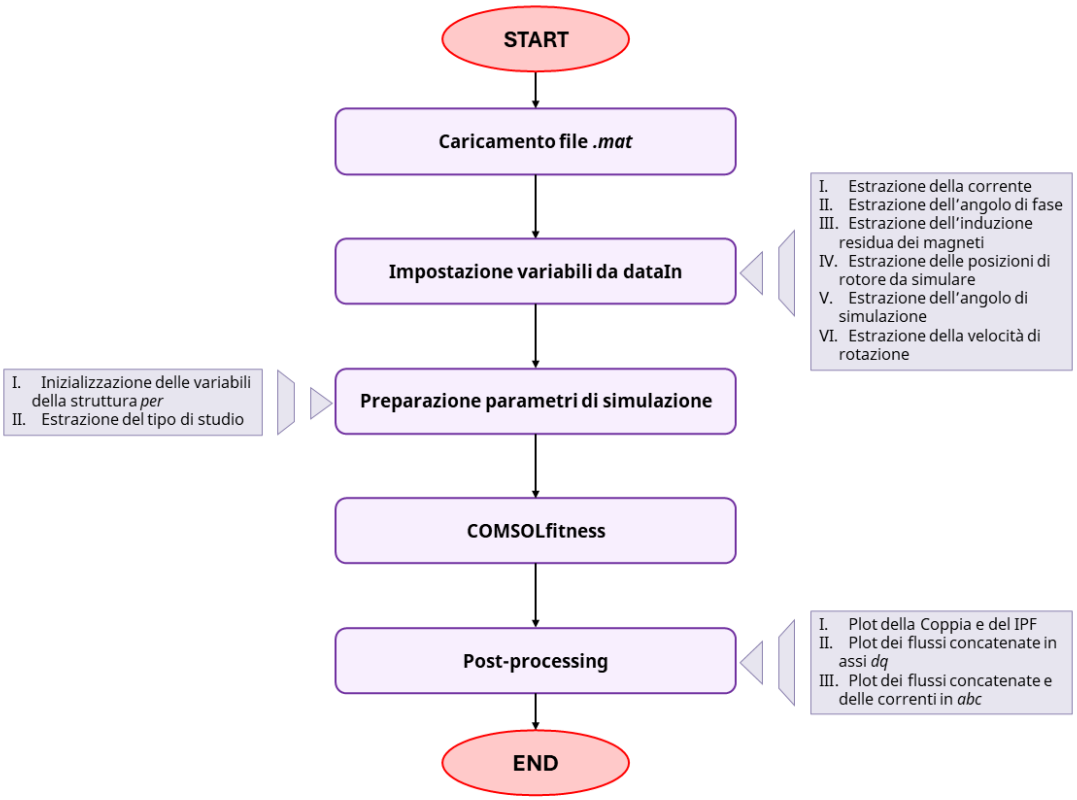

Figura 4.3. Flusso logico di eval_operatingPoint_COMSOL

4.1.1 Input della funzione

L'input della funzione è una struttura `dataIn` che contiene i seguenti campi:

- `currentpathname`: Percorso del file `.mat`.
- `currentfilename`: Nome del file `.mat`.
- `RatedCurrent`: Corrente nominale.
- `CurrLoPP`: Fattore di sovraccarico della corrente.
- `SimulatedCurrent`: Corrente simulata.
- `GammaPP`: Angolo di fase della corrente in assi dq in $elt\ deg$.
- `BrPP`: Induzione residua dei magneti permanenti in T.
- `NumOfRotPosPP`: Numero di posizioni di rotore simulate.
- `AngularSpanPP`: Ampiezza del passo angolare della simulazione in $mec\ deg$.
- `NumGrid`: Numero di livelli di corrente nella griglia delle mappe di flusso.
- `EvalSpeed`: Velocità di rotazione a cui effettuare la simulazione in rpm.
- `EvalType`: Tipo di studio da effettuare.

4.1.2 Output della funzione

La funzione non restituisce direttamente dei valori in output, ma genera delle figure che mostrano i risultati della simulazione. Queste includono:

1. Grafici di coppia e del fattore di potenza intrinseco (IPF).
2. Grafici dei flussi concatenati nel sistema di riferimento dq.
3. Grafici dei flussi concatenati e delle correnti nel sistema di riferimento trifase.

4.1.3 Dettagli delle operazioni

In prima istanza la funzione raccoglie nelle variabili elencate nel seguente script i valori delle grandezze elencate nel sottoparagrafo "Input della Funzione", caricando il file `.mat` distintivo della macchina da simulare, per assicurare alla funzione `COMSOLfitness`

di avere tutte le informazione necessarie da passare a `simulate_xdeg_COMSOL` e lanciare effettivamente la simulazione.

Nello specifico, vengono caricati i dati dalla struttura `dataIn` e preparati per la simulazione FEA.

```
pathname = dataIn.currentpathname;
filemot = strrep(dataIn.currentfilename, '.mat');
load([dataIn.currentpathname dataIn.currentfilename]);

RatedCurrent = dataIn.RatedCurrent;
CurrLoPP = dataIn.CurrLoPP;
SimulatedCurrent = dataIn.SimulatedCurrent;
GammaPP = dataIn.GammaPP;
BrPP = dataIn.BrPP;
NumOfRotPosPP = dataIn.NumOfRotPosPP;
AngularSpanPP = dataIn.AngularSpanPP;
NumGrid = dataIn.NumGrid;

per.EvalSpeed = dataIn.EvalSpeed;
overload_temp = CurrLoPP;
gamma_temp = GammaPP;
Br = BrPP;
eval_type = dataIn.EvalType;

per.overload = CurrLoPP;
per.i0 = RatedCurrent;
per.BrPP = BrPP;
per.nsim_singt = NumOfRotPosPP;
per.delta_sim_singt = AngularSpanPP;
```

Una volta raccolti tutti i dati necessari, la funzione chiama `COMSOLfitness` per tradurli e renderli digeribili per `simulate_xdeg_COMSOL` che si occuppa della configurazione del solver e della gestione della simulazione in ambiente COMSOL Multiphysics.

A questo punto, `COMSOLfitness` riceve i risultati dell'analisi FEM processati all'interno di `simulate_xdeg_COMSOL` e li rende, attraverso la struttura `out`, adatti a essere trattati in `evaluate_operatingPointCOMSOL`.

```
[~, geometry, ~, output, tempDirName] = COMSOLfitness([], geoTmp,
    perTmp, matTmp, eval_type, fileMotWithPath);
```

Dopo aver concluso la simulazione, lo script riceve da `COMSOLfitness` i vettori contenenti i valori delle grandezze che caratterizzeranno gli assi x e y dei vari grafici da mostrare

all'utente come risultati. Nel codice sottostante viene mostrato come viene effettuato il plot in figura 1 dell'andamento della coppia motrice e del fattore di potenza intrinseco ottenuti dalla simulazione. Entrambi vengono rappresentati in funzione della pulsazione elettrica espressa in gradi. Inoltre, la figura mostra il valore medio delle suddette grandezze durante l'analisi effettuata, facilitando la fruizione del software da parte dell'utente che riceve una chiara indicazione su un dato di fondamentale importanza.

```
figure();
set(gcf, 'color', 'w');
subplot(2, 1, 1);
hold on;
plot((theta_elt_deg - theta_elt_deg(1)), SOL.T, 'b', 'LineWidth',
    1.5);
mean_Tor = out.T;
grid on;
set(gca, 'GridLineStyle', ':');
xlabel('$\theta$ [elt deg]', 'Interpreter', 'latex', 'FontName', '
    Times', 'FontSize', 12);
ylabel('[Nm]', 'FontName', 'Times', 'FontSize', 12, 'Interpreter',
     'latex');
set(gca, 'XLim', [0 360], 'FontName', 'Times', 'FontSize', 12);
set(gca, 'XTick', 0:60:360, 'TickLabelInterpreter', 'latex', '
    FontName', 'Times', 'FontSize', 12);
title(['Mean Torque = ', num2str(mean_Tor), ' Nm'], 'FontWeight',
    'normal', 'FontName', 'Times', 'FontSize', 12, 'Interpreter',
    'latex');
box on;

subplot(2, 1, 2);
hold on;
plot((theta_elt_deg - theta_elt_deg(1)), out.IPF, 'b', 'LineWidth'
    , 1.5);
mean_IPF = mean(IPF);
grid on;
set(gca, 'GridLineStyle', ':');
xlabel('$\theta$ [elt deg]', 'Interpreter', 'latex', 'FontName', '
    Times', 'FontSize', 12);
ylabel('IPF', 'FontName', 'Times', 'FontSize', 12, 'Interpreter',
    'latex');
set(gca, 'XLim', [0 360], 'FontName', 'Times', 'FontSize', 12);
```

```
set(gca, 'XTick', 0:60:360, 'TickLabelInterpreter', 'latex', '
    FontName', 'Times', 'FontSize', 12);
title(['Mean IPF = ', num2str(mean_IPF)], 'FontWeight', 'normal',
    'FontName', 'Times', 'FontSize', 12, 'Interpreter', 'latex');
box on;
```

Successivamente, con un procedimento del tutto analogo a quello mostrato prima, nella seguente parte di codice vengono mostrati all'utente gli andamenti dei flussi concatenati, rispettivamente in asse d e in asse q, sempre in funzione della pulsazione elettrica espressa in gradi.

```
figure();
set(gcf, 'color', 'w');
subplot(2, 1, 1);
hold on;
plot((theta_elt_deg - theta_elt_deg(1)), SOL.fd, 'b', 'LineWidth',
     1.5);
mean_lambda_d = out.fd;
grid on;
set(gca, 'GridLineStyle', ':');
xlabel('$\theta$ [elt deg]', 'Interpreter', 'latex', 'FontName', '
    Times', 'FontSize', 12);
ylabel('$\lambda_d$ [Vs]', 'Interpreter', 'latex', 'FontName', '
    Times', 'FontSize', 12);
set(gca, 'XLim', [0 360], 'FontName', 'Times', 'FontSize', 12);
set(gca, 'XTick', 0:60:360, 'TickLabelInterpreter', 'latex', '
    FontName', 'Times', 'FontSize', 12);
title(['Mean $\lambda_d$ = ', num2str(mean_lambda_d), ' Vs'], '
    Interpreter', 'latex', 'FontName', 'Times', 'FontSize', 12);
box on;

subplot(2, 1, 2);
hold on;
plot((theta_elt_deg - theta_elt_deg(1)), SOL.fq, 'b', 'LineWidth',
     1.5);
mean_lambda_q = out.fq;
grid on;
set(gca, 'GridLineStyle', ':');
xlabel('$\theta$ [elt deg]', 'Interpreter', 'latex', 'FontName', '
    Times', 'FontSize', 12);
```

```
ylabel('$\lambda_q$ [Vs]', 'Interpreter', 'latex', 'FontName', '
    Times', 'FontSize', 12);
set(gca, 'XLim', [0 360], 'FontName', 'Times', 'FontSize', 12);
set(gca, 'XTick', 0:60:360, 'TickLabelInterpreter', 'latex', '
    FontName', 'Times', 'FontSize', 12);
title(['Mean $\lambda_q$ = ', num2str(mean_lambda_q), 'Vs'], '
    Interpreter', 'latex', 'FontName', 'Times', 'FontSize', 12);
box on;
```

Infine, con un procedimento ancora analogo a quello mostrato in precedenza, nella seguente parte di codice vengono mostrati all'utente gli andamenti dei flussi e delle correnti di fase, entrambi in sistema di riferimento trifase, sempre in funzione della pulsazione elettrica espressa in gradi.

```
figure();
set(gcf, 'color', 'w');

subplot(2, 1, 1);
plot((theta_elt_deg - theta_elt_deg(1)), SOL.fa, 'b', 'LineWidth',
    1.5);
hold on;
plot((theta_elt_deg - theta_elt_deg(1)), SOL.fb, 'r', 'LineWidth',
    1.5);
hold on;
plot((theta_elt_deg - theta_elt_deg(1)), SOL.fc, 'g', 'LineWidth',
    1.5);
grid on;
set(gca, 'GridLineStyle', ':');
xlabel('$\theta$ [elt deg]', 'Interpreter', 'latex', 'FontName', '
    Times', 'FontSize', 12);
ylabel('$\lambda_{abc}$ [Vs]', 'Interpreter', 'latex', 'FontName',
    'Times', 'FontSize', 12);
set(gca, 'XLim', [0 360], 'FontName', 'Times', 'FontSize', 12);
set(gca, 'XTick', 0:60:360, 'TickLabelInterpreter', 'latex', '
    FontName', 'Times', 'FontSize', 12);
title('Phase flux linkages', 'Interpreter', 'latex', 'FontWeight',
    'normal', 'FontName', 'Times', 'FontSize', 12);
box on;

subplot(2, 1, 2);
```

```
145 plot((theta_i_deg - theta_i_deg(1)), SOL.ia, 'b', 'LineWidth',
    1.5);
146 hold on;
147 plot((theta_i_deg - theta_i_deg(1)), SOL.ib, 'r', 'LineWidth',
    1.5);
148 hold on;
149 plot((theta_i_deg - theta_i_deg(1)), SOL.ic, 'g', 'LineWidth',
    1.5);
150 grid on;
151 set(gca, 'GridLineStyle', ':');
152 xlabel('$\theta$ [elt deg]', 'Interpreter', 'latex', 'FontName', '
    Times', 'FontSize', 12);
153 ylabel('$i_{abc}$ [A]', 'Interpreter', 'latex', 'FontName', 'Times
    ', 'FontSize', 12);
154 set(gca, 'XLim', [0 360], 'FontName', 'Times', 'FontSize', 12);
155 set(gca, 'XTick', 0:60:360, 'TickLabelInterpreter', 'latex', '
    FontName', 'Times', 'FontSize', 12);
156 title('Phase currents', 'Interpreter', 'latex', 'FontWeight', '
    normal', 'FontName', 'Times', 'FontSize', 12);
157 box on;
158 legend({'$i_a$', '$i_b$', '$i_c$'}, 'Interpreter', 'latex', '
    FontName', 'Times', 'FontSize', 12);
```

4.2 COMSOL_fitness

La funzione `COMSOLfitness` è progettata per fungere da interfaccia tra la funzione più esterna, `eval_operatingPointCOMSOL` e quella più interna, `simulate_xdeg_COMSOL`. Il suo scopo principale è convertire i dati ottenuti dalla simulazione FEA dalla struttura `SOL` alla struttura `out`, in modo che possano essere presentati correttamente all'utente attraverso l'ambiente di SyR-e. In dettaglio, la funzione:

1. **Prepara l'ambiente per l'analisi agli elementi finiti**: La funzione crea una directory temporanea e copia i file del modello necessari per la simulazione.
2. **Esegue la simulazione FEA**: Utilizzando `simulate_xdeg_COMSOL`, la funzione configura il solver e gestisce l'esecuzione della simulazione in COMSOL Multiphysics.
3. **Elabora i risultati**: La funzione raccoglie i risultati della simulazione e li organizza nella struttura `out`, calcolando le medie delle grandezze di interesse e altre metriche rilevanti come il fattore di potenza intrinseco (IPF) e le perdite di potenza.

4. **Restituisce i risultati**: I dati elaborati vengono restituiti in una struttura di output, pronta per essere utilizzata e visualizzata nell'ambiente SyR-e.

Nel paragrafo successivo, verrà illustrato in dettaglio come la funzione gestisce la configurazione del solver e l'esecuzione della simulazione tramite `simulate_xdeg_COMSOL`, nonché l'elaborazione finale dei risultati.

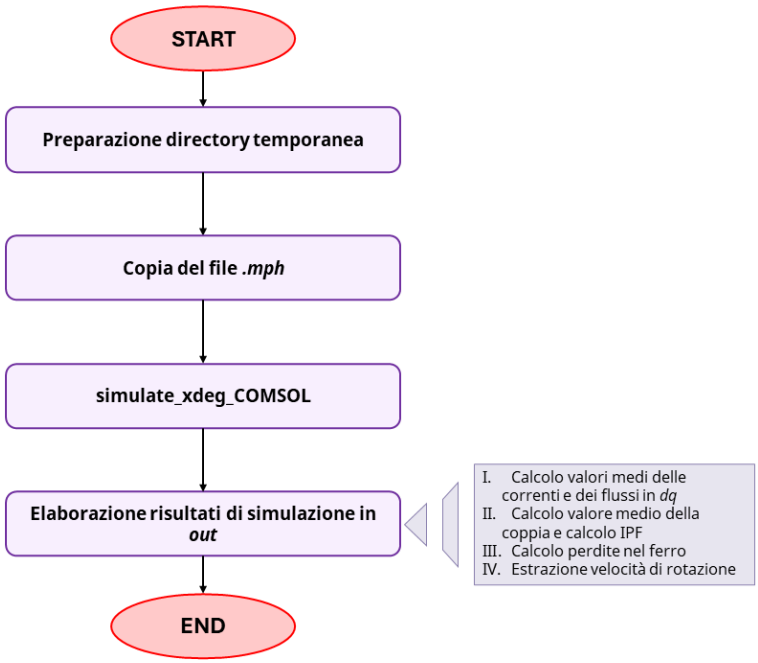

Figura 4.4. Flusso logico di COMSOLfitness

La funzione segue il flusso logico definito in Figura 4.4 che illustra le diverse fasi necessarie per avviare la simulazione del modello e presentare i dati processati all'utente. Inoltre, può essere eseguita direttamente dall'interfaccia GUI di SyR-e tramite il pulsante *COMSOL* nella finestra *Simulation*, oppure può essere eseguita autonomamente tramite la *Command Window*, a condizione che si stia utilizzando LiveLink for MATLAB di COMSOL.

4.2.1 Input della funzione

La funzione richiede i seguenti input:

- `RQ`: Un parametro opzionale (non utilizzato direttamente).

- `geo`: La struttura contenente i dati geometrici del modello da simulare.
- `per`: La struttura contenente i parametri di performance e di simulazione.
- `mat`: La struttura contenente le proprietà dei materiali utilizzati nel modello.
- `eval_type`: Il parametro che specifica il tipo di studio da eseguire.
- `pathname`: Il percorso della directory contenente i file necessari per la simulazione.
- `filename`: Il nome del file del modello COMSOL `.mph` da utilizzare per la simulazione.

4.2.2 Output della funzione

La funzione restituisce i seguenti output:

- `geo`: La struttura dei dati geometrici, eventualmente modificata durante la simulazione.
- `mat`: La struttura dei dati dei materiali, eventualmente modificata durante la simulazione.
- `out`: La struttura contenente i risultati elaborati della simulazione.
- `pathname`: Il percorso della directory temporanea creata per la simulazione.

4.2.3 Dettagli delle operazioni

Di seguito sono riportate le operazioni dettagliate eseguite dalla funzione:

1. Creazione Directory Temporanea

```
17 pathnameIn = pathname;
18 [~, pathname] = createTempDir();
19 copyfile([pathnameIn strrep(filename,'.mat','.mph')], [
       pathname strrep(filename,'.mat','.mph')]);
```

   La funzione crea una directory temporanea per eseguire la simulazione e copia il file del modello COMSOL `.mph` nella directory temporanea.

2. Esecuzione della Simulazione

```
21 [SOL] = simulate_xdeg_COMSOL(geo, per, eval_type, pathname,
       filename);
```

La funzione chiama `simulate_xdeg_COMSOL` per eseguire la simulazione in ambiente COMSOL Multiphysics, passando i dati raccolti nelle strutture `geo` e `per`, la tipologia di studio indicata dall'utente e i percorsi dei file. Alla fine della simulazione, i risultati ottenuti e processati da `simulate_xdeg_COMSOL` vengono restituiti a `COMSOLfitess` attravero la struttura `SOL`.

3. Elaborazione dei Risultati della Simulazione

```
23 out.id = mean(SOL.id);
24 out.iq = mean(SOL.iq);
25 out.fd = mean(SOL.fd);
26 out.fq = mean(SOL.fq);
27 out.T  = abs(mean(SOL.T));
28 out.IPF = sin(atan2(out.iq,out.id)-atan2(out.fq,out.fd));
29 out.SOL = SOL;
30
31 out.Ppm    = SOL.Ppm;
32 out.Pfes   = SOL.Pfes;
33 out.Pfer   = SOL.Pfer;
34
35 out.Pfe    = out.Pfes + out.Pfer;
36 out.velDim = per.EvalSpeed;
```

Infine, la funzione si occupa dell'elaborazione ulteriore dei risultati ottenuti dalla simulazione, salvandoli nella struttura `out`. Nello specifico, calcola le medie di alcune grandezze di particolare interesse, come le correnti *id* e *iq*, i flussi *fd* e *fq*, la coppia *T* e il fattore di potenza intrinseco *IPF*. Inoltre, trasferisce i dati relativi alle quote di perdita di potenza *Ppm*, *Pfes* e *Pfer*, e calcola le perdite nel ferro complessive della macchina (*Pfe*). Infine, salva la velocità di rotazione utilizzata per effettuare le simulazioni (*velDim*) dalla struttura `per`.

4.3 simulate_xdeg_COMSOL

La funzione è progettata per eseguire la simulazione agli elementi finiti (FEA) impostata attraverso le funzioni presentate in precedenza, utilizzando l'ambiente di COMSOL Multiphysics attraverso LiveLink *for* MATLAB.

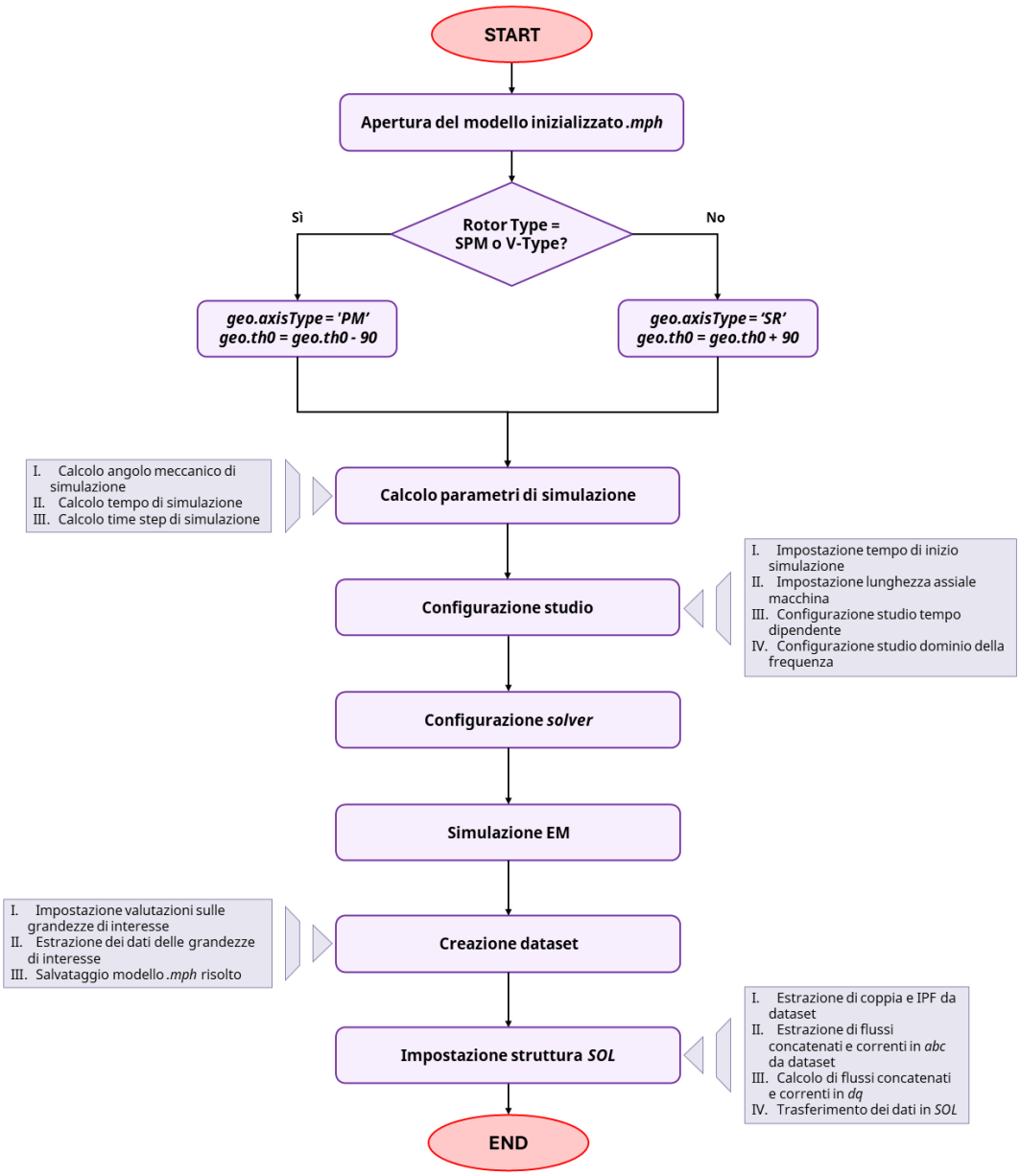

Figura 4.5. Flusso logico di simulate_xdeg_COMSOL

Di seguito viene presentata una panoramica delle principali operazioni che svolge:

1. Connessione al Server e Configurazione del Modello COMSOL
2. Impostazione della Simulazione

3. Definizione dello Studio e del Solver
4. Esecuzione della Simulazione
5. Post-Processing e Creazione dei Dataset
6. Salvataggio del Modello Risolto
7. Impostazione delle Variabili di Output

Questa funzione permette di automatizzare l'intero processo di simulazione, dalla configurazione iniziale del modello alla presentazione dei risultati, completando di fatto il processo automatizzato di analisi FEM da SyR-e attraverso l'utilizzo di COMSOL Multiphysics in background.

La funzione segue il flusso logico definito in Figura 4.5 che illustra le diverse fasi necessarie per avviare la simulazione del modello e presentare i dati processati all'utente. In più, può essere eseguita direttamente dall'interfaccia GUI di SyR-e tramite il pulsante *COMSOL* nella finestra *Simulation*, oppure può essere eseguita autonomamente tramite la *Command Window*, a condizione che si stia utilizzando LiveLink for MATLAB di COMSOL.

4.3.1 Input della Funzione

- `geo`: struttura contenente i parametri geometrici della macchina.
- `per`: struttura contenente i parametri operativi e di simulazione.
- `eval_type`: tipo di valutazione.
- `pathname`: percorso del file COMSOL.
- `filename`: nome del file COMSOL.

4.3.2 Output della Funzione

`SOL`: struttura contenente i risultati della simulazione, inclusi:

- `Pfes`: perdite nel ferro dello statore.
- `Pfer`: perdite nel ferro del rotore.
- `PjrBar`: perdite nelle barriere di flusso del rotore.
- `Ppm`: perdite nei magneti permanenti.

- `th`: angolo elettrico.
- `id, iq`: componenti della corrente in dq.
- `fd, fq`: componenti del flusso concatenato in dq.
- `T`: coppia motrice.
- `ia, ib, ic`: correnti nelle fasi abc.
- `fa, fb, fc`: flussi concatenati nelle fasi abc.

4.3.3 Dettagli delle Operazioni

1. **Connessione e Configurazione del Modello COMSOL**

   - Come prima operazione, la funzione importa le librerie COMSOL e apre il modello specificato. Nonostante le operazioni si svolgano in LiveLink *for* MATLAB, è necessario `import com.comsol.model.util.*` che permette di utilizzare i comandi di COMSOL in ambiente MATLAB, tale operazione viene svolta solo in `simulate_xdeg_COMSOL` poiché è in questo contesto che avviene la configurazione della simulazione del modello in COMSOL.

```
18 import com.comsol.model.*
19 import com.comsol.model.util.*
20 model = mphopen([pathname filename(1:end-4), '.mph']);
```

   - Insieme all'import del modello, avviene la verifica del tipo di geometria di rotore selezionata dall'utente nella fase di *preliminary design*. Infatti, da questa informazione, il codice seleziona quale convenzioni applicare in termini di orientamento del sistema di riferimento in assi *dq* con conseguente correzione dell'offset sulla posizione di rotore rispetto agli 0° elettrici.

```
26 if strcmp(geo.RotType, 'SPM') || strcmp(geo.RotType, '
   Vtype')
27     geo.axisType = 'PM';
28     geo.th0 = geo.th0 - 90;
29 else
30     geo.axisType = 'SR';
31     geo.th0 = geo.th0 + 90;
32 end
```

- La configurazione iniziale termina con l'impostazione del grado di discretizzazione nel calcolo del vettore potenziale magnetico con cui svolgere i calcoli. In questo caso, è stato scelto di impostare di default il valore 1, che corrisponde ad una discretizzazione di tipo lineare. Se si desiderasse aumentare il livello di precisione, con conseguente dilatazione dei tempi di simulazione, è possibile scegliere i livelli 2 e 3, corrispondenti rispettivamente a discretizzazioni quadratiche e cubiche.

```
35 model.component('comp1').physics('rmm').prop('
       ShapeProperty').set('order_magneticvectorpotential', 1)
       ;
```

2. **Impostazione della Simulazione**

- L'impostazione della simulazione parte dall'assegnazione dei dati presenti nelle strutture `geo` e `per`, che arrivano da `eval_operatingPointCOMSOL` passando attraverso `COMSOLfitness`, a delle variabili per semplificare la scrittura delle operazioni di calcolo da effettuare per definire un tempo di simulazione e un time step conseguente. Infatti, sebbene in SyR-e, attraverso la finestra *Simulation*, si specifichino un numero di posizioni di rotore da simulare e un angolo elettrico di simulazione, COMSOL Multiphysics necessita di una impostazione con intervalli di tempo. Quindi, la funzione si occupa della traduzione in tempo delle grandezze spaziali definite dall'utente, grazie al valore specificato di velocità di rotazione meccanica. Di seguito, viene mostrato la parte di codice dedita alla conversione appena spiegata.

```
37 nsim = per.nsim_singt;
38 xdeg = per.delta_sim_singt;
39 gamma = per.gammaPP;
40 th0 = geo.th0;
41 p = geo.p;
42 w = per.EvalSpeed * 30 / pi;
43 theta_m_tot = xdeg / p * pi / 180;
44 t = theta_m_tot / w;
45 ts = t / nsim;
```

- Successivamente, si passa al calcolo delle componenti della corrente (`id` e `iq`) dalle quali si ottiene il modulo del vettore corrente in assi *dq*.

```
47 iAmp = per.overload * per.i0;
48 theta_i = (th0 + gamma) * pi / 180;
```

```
49
50 id = iAmp * cos(theta_i);
51 iq = iAmp * sin(theta_i);
52 Imod = abs(id + 1i * iq);
53 Iarg = angle(Imod(end));
```

3. **Definizione dello Studio e del Solver**

- La configurazione dello studio viene svolta inizializzando: il parametro `t`, che COMSOL identifica intrinsecamente come parametro temporale, uguale a 0 secondi e il parametro `l`, espresso in mm, che rappresenta la lunghezza assiale del motore in analisi. Successivamente, si imposta il range temporale di simulazione per la parte tempo dipendente dello *study1* (elettromagnetico), utilizzando i valori di `t` e `ts`, ottenuti dai calcoli precedenti.

```
57 model.param().set('t', '0 [s]');
58 model.param().set('l', [num2str(geo.l) ' [mm]']);
59 model.study('std1').setGenIntermediatePlots(true);
60 model.study('std1').create('time', 'Transient');
61 model.study('std1').feature('time').set('tlist', ['range
    (0,' num2str(ts) ',' num2str(t) ')']);
62 model.study('std1').feature('time').set('plot', true);
```

- In seguito, si passa alla configurazione del calcolo delle perdite. Questo processo viene effettuato in COMSOL, specificando l'istante di tempo iniziale e finale di calcolo delle perdite nel ferro per il modello della macchina, attraverso l'impostazione di *TimeToFrequencyLosses*. Inoltre, viene anche specificato il vettore di valori di frequenza ai quali effettuare le simulazioni per il calcolo delle perdite.

```
65 model.study('std1').create('emloss', '
    TimeToFrequencyLosses');
66 model.study('std1').feature('emloss').set('endtmethod', '
    userdef');
67 model.study('std1').feature('emloss').set('lossstarttime',
     '0');
68 model.study('std1').feature('emloss').set('lossendtime',
    num2str(t));
69 model.study('std1').feature('emloss').set('fftmaxfreq', ['
    6/(' num2str(t) '-0)']);
```

- L'ultima configurazione riguarda l'assegnazione dei circuiti elettrici di fase agli avvolgimenti della macchina. Così facendo, si interfaccia il motore a un circuito elettrico esterno, permettendo di iniettare correnti sinusoidali nelle bobine ed estrarre dati come flussi concatenati e tensioni dalla simulazione.

```
71 model.component('comp1').physics('cir').feature('termI1').
       set('V_src', 'root.comp1.rmm.VCoil_1');
72 model.component('comp1').physics('cir').feature('termI2').
       set('V_src', 'root.comp1.rmm.VCoil_2');
73 model.component('comp1').physics('cir').feature('termI3').
       set('V_src', 'root.comp1.rmm.VCoil_3');
```

4. **Esecuzione della Simulazione**

- Il comando mostrato sotto esegue la simulazione del modello COMSOL con l'aggiornamento della barra di progresso.

```
167 model.study('std1').run();
```

5. **Post-Processing e Creazione dei Dataset**

- Dopo aver effettuato la simulazione, la prima operazione di post-processing si focalizza sulla creazione di dataset, sotto forma di tabelle in cui vengono immagazzinati i risultati ottenuti in termini di valori assunti dalle grandezze d'interesse nell'intervallo di tempo di simulazione. Per estrarre tutti i dati utili all'utente e processarli, è necessario svolegere delle *evaluations* globali e/o locali sul modello risolto.

  Nel codice mostrato in basso, vi è l'esempio di creazione del dataset riguardante l'andamento della coppia motrice della macchina, durante la simulazione. Nella tabella `tbl1` verranno inseriti i dati da estrarre dal modello risolto ed esportati in un file `.csv`.

```
175 % Coppia Motrice (Arkkio)
176 model.result().numerical().create('gev1', 'EvalGlobal');
177 model.result().table().create('tbl1', 'Table');
178 model.result().export().create('tbl1', 'Table');
```

  Restando sempre sul processo dei dati relativi alla coppia (le operazioni svolte per le altre grandezze sono del tutto analoghe), una volta creati gli strumenti di valutazione ed esportazione, questi vengono configurati e lanciati. Infatti, per l'estrazione dei dati, la funzione seleziona la grandezza da estrarre

(`rmm.Tark_1`), attraverso i successivi comandi li estrae e li importa nella tabella `tbl1`. Infine, trasforma la tabella in un file (`torque_csv`) e lancia il comando di export, salvando i dati in una cartella ad hoc.

```
% Coppia Motrice (Arkkio)
model.result().numerical('gev1').set('data', 'dset3');
model.result().numerical('gev1').set('expr', {'rmm.Tark_1'
    });
model.result().numerical('gev1').set('descr', {'Axial
    Torque'});
model.result().table('tbl1').comments('Global Evaluation 1
    ');
model.result().numerical('gev1').set('table', 'tbl1');
model.result().numerical('gev1').setResult();
model.result().export('tbl1').set('filename', torque_csv);
model.result().export('tbl1').set('table', 'tbl1');
model.result().export('tbl1').run();
```

6. **Salvataggio del Modello Risolto**

- Dopo aver estratto i dati, la funzione salva il modello COMSOL risolto con lo stesso nome, aggiungendo `_solved.mph`, attraverso il comando mostrato sotto.

```
mphsave(model, [pathname filename(1:end-4), '_solved.mph'
    ]);
```

7. **Impostazione delle Variabili di Output**

- L'ultima parte del post-processing riguarda l'elaborazione dei dati estratti dal modello, per renderli informazioni fruibili e utili all'utente in SyR-e. Continuando con l'analisi del processo dei dati relativi alla coppia motrice della macchina, si apre il file `.csv` appena estratto da COMSOL e si estraggono i valori di coppia e quelli dei time step di simulazione. Successivamente si trasforma il tempo in un angolo elettrico `theta_elt_deg`, grazie al parametro di frequenza di alimentazione della macchina, già noto.

```
% Coppia Motrice (Arkkio)
axial_torque = readmatrix(fullfile(torque_csv));
tempo = [axial_torque(:, 1)]';
Tor = [axial_torque(:, 2)]';
theta_elt = tempo*2*pi*freq+geo.th0*pi/180-pi/2;
theta_elt_deg = theta_elt*180/pi;
```

Per quanto riguarda grandezze come i flussi e le correnti, che hanno necessità di essere mostrate all'utente nel sistema di riferimento in assi *dq*, il post-processing dei dati è costituito da un ulteriore passaggio dopo l'estrazione dal file `.csv`, che consiste nella trasformazione da trifase a *dq*, attraverso l'utilizzo della funzione `abc2dq`. Infine, la funzione calcola anche il modulo e la fase del vettore flusso concatenato in assi *dq*.

```
% Flussi concatenati
concatenated_flux = readmatrix(flux_csv);
lambda_1 = [concatenated_flux(:, 2)]';
lambda_2 = [concatenated_flux(:, 3)]';
lambda_3 = [concatenated_flux(:, 4)]';

lambda_dq = abc2dq(lambda_1, lambda_2, lambda_3, theta_elt
    );
lambda_d = lambda_dq(1, :);
lambda_q = lambda_dq(2, :);

lambda_mod = abs(lambda_d + 1i*lambda_q);
lambda_arg = angle(lambda_mod(end));
```

- Come ultima operazione, la funzione prepara i dati per l'esportazione, verso la funzione più esterna `COMSOLfitness`, assegnando i valori alla struttura di output `SOL`, che contiene perdite, correnti e flussi pronti per essere presentati all'utente.

```
SOL.Pfes    = Pfes;
SOL.Pfer    = Pfer;
SOL.PjrBar  = PjrBar;
SOL.Ppm     = Ppm;

SOL.th = theta_elt_deg;
SOL.id = id_pp;
SOL.iq = iq_pp;
SOL.fd = lambda_d;
SOL.fq = lambda_q;

SOL.T = Tor;

SOL.ia = i_1;
SOL.ib = i_2;
```

```
361 SOL.ic = i_3;
362
363 SOL.fa = lambda_1;
364 SOL.fb = lambda_2;
365 SOL.fc = lambda_3;
```

5 Simulazione strutturale

In questo capitolo viene descritta un'altra sezione fondamentale dell'interfaccia che è stata realizzata, ovvero il processo di simulazione strutturale del modello della macchina esportata da SyR-e a COMSOL Multiphysics. Le funzioni centrali in questo contesto sono `eval_vonMisesStress_COMSOL` e `eval_maxStress_COMSOL`, illustrate nella Figura 5.1, che costituisce la parte evidenziata in basso a destra del diagramma a blocchi presentato nel capitolo di presentazione dell'interfaccia.

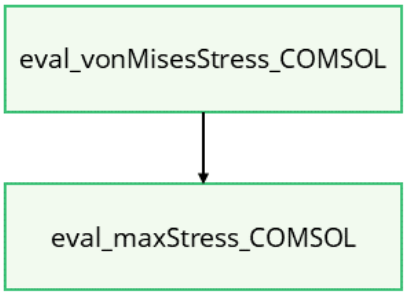

Figura 5.1. Funzioni responsabili della simulazione strutturale del modello

Le funzioni hanno il compito di calcolare e/o estrarre tutti i parametri geometrici, dei materiali e fisici da SyR-e, sfruttando la definizione del *preliminary design* e le impostazioni di simulazione inserite dall'utente nella finestra *Simulation* di SyR-e. Anche in questo caso, come avvenuto per le funzioni presentate in precedenza, per l'implementazione in COMSOL, ci si è basati sull'architettura delle funzioni già presenti in SyR-e, cercando di minimizzare le personalizzazioni necessarie nel codice script, considerando le differenze significative tra SyR-e e COMSOL nell'implementazione delle simulazioni agli elementi finiti.

Questo approccio mira a standardizzare le procedure per agevolare gli utenti che già conoscono lo scripting di SyR-e, incoraggiandoli a esplorare personalizzazioni e miglioramenti del software, essendo esso open-source.

Figura 5.2. Struttura annidata di eval_vonMisesStress_COMSOL

Nel dettaglio delle funzioni di simulazione del modello, si osserva dalla Figura 5.2 che `eval_vonMisesStress_COMSOL` svolge un ruolo fondamentale sia nell'impostazione generale dei parametri utili all'analisi sia nella presentazione dei risultati ottenuti dalla simulazione del modello. Questa funzione gestisce la preparazione dei dati e stabilisce accuratamente i percorsi dei file necessari. Utilizza inoltre `eval_maxStress_COMSOL` per una parte delle operazioni di post-processing relative alla valutazione del superamento dello stress massimo consentito nei diversi punti della geometria in analisi.

A differenza di quanto avviene con la simulazione elettromagnetica, per la simulazione strutturale `eval_vonMisesStress_COMSOL` si occupa di creare il modello, definire materiali e boundary conditions, effettuare la simulazione, in ambiente COMSOL, ed elaborare una parte dei risultati finali, come verrà illustrato nei paragrafi successivi. Una volta completato il processo dei dati in background, i risultati vengono presentati all'utente tramite figure, il cui numero e contenuto possono variare in base alla tipologia di analisi svolta.

Inoltre, va sottolineato che per condurre studi di meccanica strutturale in COMSOL, è essenziale utilizzare il modulo *Solid Mechanics*, dedicato alla risoluzione di problemi nei settori della meccanica strutturale e dei solidi. Le interfacce fisiche contenute in questo modulo possono essere accoppiate con qualsiasi altra interfaccia fisica in COMSOL Multiphysics [6].

In più, questo modulo offre capacità di studio che comprendono analisi statiche, di frequenza propria, transitorie, di risposta in frequenza, buckling e studi parametrici. Una vasta gamma di modelli materiali è disponibile, tra cui materiali elastici lineari, viscoelastici, piezoelettrici, e magnetostrittivi (con il modulo AC/DC), oltre a modelli per iperelasticità, plasticità dei metalli, creep, viscoplasticità, elasticità non lineare, plasticità dei

suoli, calcestruzzo, rocce e argilla tramite moduli opzionali [6].

Infine, *Solid Mechanics* consente la modellazione di grandi deformazioni, contatti e attrito, e facilita l'accoppiamento con l'analisi termica e altre forme di multiphysics. Supporta completamente materiali piezoelettrici, permettendo accoppiamenti speciali che risolvono sia il campo elettrico che la deformazione in entrambe le direzioni. Inoltre, le simulazioni in COMSOL Multiphysics frequentemente integrano la meccanica strutturale con discipline come il flusso fluido (FSI), le reazioni chimiche, l'acustica, i campi elettrici, i campi magnetici e la propagazione delle onde ottiche [6].

5.1 eval_vonMisesStress_COMSOL

La funzione `eval_vonMisesStress_COMSOL` è progettata con lo scopo di effettuare una simulazione strutturale sulla macchina in progetto utilizzando COMSOL Multiphysics.

La funzione opera sullo stesso file `.mph` utilizzato per svolgere le analisi magnetiche, questo passaggio permette di effettuare studi di diversa tipologia attraverso l'utilizzo del minor numero possibile di configurazioni, consentendo all'utente di velocizzare il processo di progettazione e analisi.

Nonostante si utilizzi lo stesso file in COMSOL, si deve riconfigurare il modello, `eval_vonMisesStress_COMSOL` si occupa dell'impostazione, della simulazione e di una parte del post-processing, mentre l'individuazione dei punti di stress massimo e l'elaborazione dei dati relativi viene effettuata da `eval_maxStress_COMSOL`.

Nello specifico, la funzione estrae i componenti del tensore di stress attraverso l'analisi agli elementi finiti del lamierino di rotore, comprensivo di magneti (se presenti nel progetto) e restituisce i risultati per ciascun punto di interesse, sotto forma di grafici.

Il processo logico, secondo cui la funzione è stata scritta, è mostrato in Figura 5.3 attraverso un flow-chart. In più, può essere eseguita attraverso l'apposito pulsante `COMSOL`, presente nella finestra *Simulation* di SyR-e (avendo selezionato *Structural Analysis*) oppure in modo autonomo attraverso la *Command Window*, a patto di stare usando LiveLink *for* MATLAB di COMSOL.

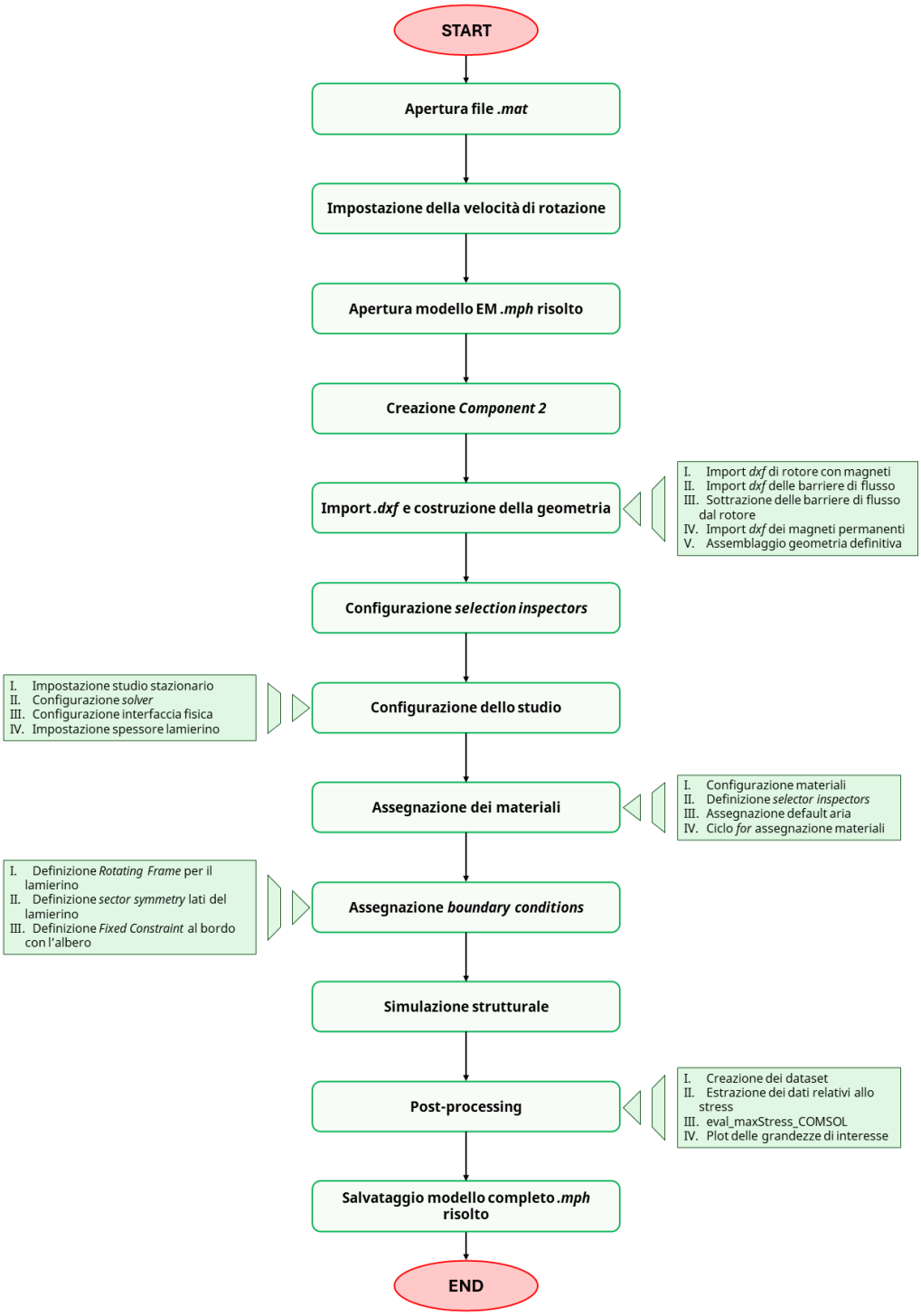

Figura 5.3. Flusso logico di eval_vonMisesStress_COMSOL

5.1.1 Input della Funzione

La funzione riceve in input la struttura `dataIn` che contiene i seguenti campi impostati dell'utente nell'interfaccia di SyR-e:

- `dataIn.currentpathname`: La stringa che rappresenta il percorso della cartella corrente.
- `dataIn.currentfilename`: La stringa che rappresenta il nome del file corrente.
- `dataIn.EvalSpeed`: Il valore numerico che rappresenta la velocità di rotazione alla quale effettuare la simulazione in rpm.

5.1.2 Output della Funzione

La funzione restituisce:

- `newDir`: Una stringa che rappresenta il percorso della nuova cartella creata, che è una sottocartella all'interno della directory dei risultati del file `.mph` specificato in `dataIn`.

5.1.3 Dettagli delle Operazioni

Di seguito sono riportati i dettagli delle operazioni eseguite dalla funzione:

1. **Apertura del modello COMSOL**

   Innanzitutto, si caricano le librerie di COMSOL e si apre il modello già utilizzato per le analisi elettromagnetiche.

```
47 import com.comsol.model.*
48 import com.comsol.model.util.*
49 model = mphopen(fullfile(dataIn.currentpathname, [dataIn.
       currentfilename(1:end-4), '.mph']));
```

2. **Configurazione della geometria**

   La simulazione strutturale, viene gestita attraverso l'aggiunta di un componente (`comp2`) al modello esistente in COMSOL Multiphysics. Di conseguenza, bisogna caricare nuovamente i disegni della geometria, attraverso i file `.dxf` e si rende necessaria la creazione di una mesh specifica per le analisi FEM da svolgere sul modello strutturale.

```
model.component().create('comp2', true);
model.component('comp2').geom().create('geom2', 2);
model.component('comp2').mesh().create('mesh2');
model.component('comp2').geom('geom2').lengthUnit('mm');
```

Vengono svolte diverse operazioni per montare la geometria utile alle analisi strutturali, all'interno del modello. Infatti, vengono utilizzati 3 file, uno contenente la geometria di rotore, alla quale viene sottratta l'area relativa alle barriere di flusso e successivamente aggiunti i magneti, come mostrato nello script di sotto. Tutto questo si rende necessario per ottenere la formazione di un *Identity Pair* tra i magneti e il lamierino, che permette, di fatto, alla simulazione di funzionare correttamente.

```
model.component('comp2').geom('geom2').create('imp1', 'Import'
    );
model.component('comp2').geom('geom2').feature('imp1').set('
    filename', rot_mecc_dxf);
model.component('comp2').geom('geom2').run('imp1');
model.component('comp2').geom('geom2').create('imp2', 'Import'
    );
model.component('comp2').geom('geom2').feature('imp2').set('
    filename', flux_barr_dxf);
model.component('comp2').geom('geom2').run('imp2');
model.component('comp2').geom('geom2').create('dif1', '
    Difference');
model.component('comp2').geom('geom2').feature('dif1').
    selection('input').set('imp1');
model.component('comp2').geom('geom2').feature('dif1').
    selection('input2').set('imp2');
model.component('comp2').geom('geom2').run('dif1');
model.component('comp2').geom('geom2').create('imp3', 'Import'
    );
model.component('comp2').geom('geom2').feature('imp3').set('
    filename', magn_dxf);
model.component('comp2').geom('geom2').run('imp3');
model.component('comp2').geom('geom2').feature('fin').set('
    action', 'assembly');
model.component('comp2').geom('geom2').run('fin');
```

3. **Configurazione dello Studio**

   La configurazione dello studio avviene in maniera analoga a quanto mostrato per le analisi elettromagnetiche. Il tipo di analisi da effettuare è stazionario, in più viene aggiunta la fisica *Solid Mechanics* che definisce le condizioni al contorno per gli studi di carattere strutturale. Infine, vengono impostate le diverse condizioni sui solver e viene definita la lunghezza assiale della geometria che, per questo tipo di analisi, è lo spessore del singolo lamierino di rotore.

```
93  model.study().create('std2');
94  model.study('std2').create('stat', 'Stationary');
95  model.study('std2').feature('stat').setSolveFor('/physics/rmm'
        , true);
96  model.study('std2').feature('stat').setSolveFor('/physics/cir'
        , true);
97  model.study('std2').feature('stat').setEntry('activate', 'cir'
        , false);
98  model.study('std2').feature('stat').setEntry('activate', 'rmm'
        , false);
99  model.component('comp2').physics().create('solid', '
        SolidMechanics', 'geom2');
100 model.component('comp2').physics('solid').prop('d').set('d',
        0.001);
101 model.study('std1').feature('stat').setSolveFor('/physics/
        solid', true);
102 model.study('std1').feature('time').setSolveFor('/physics/
        solid', true);
103 model.study('std1').feature('emloss').setSolveFor('/physics/
        solid', true);
104 model.study('std2').feature('stat').setSolveFor('/physics/
        solid', true);
105 model.study('std1').feature('stat').setEntry('activate', '
        solid', false);
106 model.study('std1').feature('time').setEntry('activate', '
        solid', false);
```

4. **Impostazione dei materiali**

   Trattandosi di un componente diverso, vengono definiti i materiali utili alla simulazione, con la specifica dei valori delle proprietà meccaniche come: modulo di Young, densità e coefficiente di Poisson.

```
model.component('comp2').material().create('mat5', 'Common');
model.component('comp2').material('mat5').propertyGroup().
    create('RemanentFluxDensity', 'Remanent flux density');
model.component('comp2').material('mat5').label('N52 (Sintered
    NdFeB)');
model.component('comp2').material('mat5').set('family', '
    chrome');
model.component('comp2').material('mat5').propertyGroup('def')
    .set('electricconductivity', {'1/1.4[uohm*m]', '0', '0', '0
    ', '1/1.4[uohm*m]', '0', '0', '0', '1/1.4[uohm*m]'});
model.component('comp2').material('mat5').propertyGroup('def')
    .set('relpermittivity', {'1', '0', '0', '0', '1', '0', '0',
    '0', '1'});
model.component('comp2').material('mat5').propertyGroup('
    RemanentFluxDensity').set('murec', {'1.05', '0', '0', '0',
    '1.05', '0', '0', '0', '1.05'});
model.component('comp2').material('mat5').propertyGroup('
    RemanentFluxDensity').set('normBr', '1.44[T]');
model.component('comp2').material('mat5').propertyGroup().
    create('Enu', 'Youngs_modulus_and_Poissons_ratio');
model.component('comp2').material('mat5').propertyGroup('Enu')
    .set('E', '175e9');
model.component('comp2').material('mat5').propertyGroup('Enu')
    .set('nu', '0.24');
model.component('comp2').material('mat5').propertyGroup('def')
    .set('density', '7550');
```

5. **Assegnazione dei Materiali**

   L'assegnazione dei materiali viene gestita in modo del tutto analogo a come mostrato in precedenza nella funzione `draw_motor_in_COMSOL`. Infatti, si sfrutta un *selection inspector* e, attraverso il codice materiale indicato in `geo.BLKLABELS.rotore.xy`, si popola il vettore `fer` con i numeri dei domini a cui assegnare il materiale.

```
selNumber_fer = [];
fer = [];

for kk = 1:size(geo.BLKLABELS.rotore.xy, 1)
    x = geo.BLKLABELS.rotore.xy(kk, 1);
    y = geo.BLKLABELS.rotore.xy(kk, 2);
    disk3.set('posx', x);
```

```
250     disk3.set('posx', y);
251     selNumber_fer = disk3.entities();
252     if geo.BLKLABELS.rotore.xy(kk, 3) == 5
253        fer = horzcat(fer, selNumber_fer);
254     end
255 end
256
257 model.component('comp2').material('mat6').selection().set(fer)
    ;
```

6. **Impostazione delle condizioni al contorno**

   Anche per l'assegnazione delle condizioni al contorno, il processo è del tutto analogo a quello visto in `draw_motor_in_COMSOL`. Nel codice mostrato in basso, si effettua l'assegnazione del *Fixed Constraint* sul bordo di contatto tra albero motore e lamierino di rotore.

```
281 selNumber_shaft = [];
282
283 raggio = geo.Ar;
284 alpha_mecc = pi/geo.p/2;
285 x_mecc_m = raggio*cos(alpha_mecc);
286 y_mecc_m = raggio*sin(alpha_mecc);
287 disk4.set('posx', x_mecc_m);
288 disk4.set('posy', y_mecc_m);
289 selNumber_shaft = disk4.entities();
290
291 model.component('comp2').physics('solid').create('fix1', '
    Fixed', 1);
292 model.component('comp2').physics('solid').feature('fix1').
    selection().set(selNumber_shaft);
```

7. **Esecuzione della simulazione**

```
294 model.study('std2').run();
```

8. **Post-processing**

   Infine, si effettua il post-processing, lanciando la funzione `eval_maxStress_COMSOL` per l'individuazione dei punti che superano lo stress massimo consentito e, successivamente, presentando i risultati ottenuti dall'analisi FEM strutturale all'utente attraverso dei grafici.

```
% von Mises Stress
model.result().create('pg7', 'PlotGroup2D');
model.result('pg7').set('data', 'dset6');
model.result('pg7').create('surf1', 'Surface');
model.result('pg7').set('edges', false);
model.result('pg7').feature('surf1').set('expr', 'solid.misesGp');
model.result('pg7').feature('surf1').set("descr", 'von Mises stress');
model.result('pg7').feature('surf1').set('unit', 'MPa');
model.result('pg7').feature('surf1').set('colortable', 'RainbowLightClassic');

figure()

mphplot(model, 'pg7', 'rangenum', 1);

set(gcf,'defaultTextInterpreter','Latex');
set(gcf,'defaultLegendInterpreter','Latex');
set(gcf,'defaultAxesTickLabelInterpreter','Latex');
set(gcf,'defaultAxesLineWidth',1);
set(gcf,'defaultLineLineWidth',1.5);
set(gcf,'defaultAxesGridLineStyle',':');
set(gcf,'defaultAxesXGrid','on');
set(gcf,'defaultAxesYGrid','on');
set(gcf,'defaultAxesZGrid','on');
set(gcf,'defaultAxesXColor',0*[1 1 1]);
set(gcf,'defaultAxesYColor',0*[1 1 1]);
set(gcf,'defaultAxesZColor',0*[1 1 1]);
set(gcf,'defaultAxesBox','on');
set(gcf,'defaultAxesNextPlot','add');

% Define text size
textSize = 12;  % Or set it to your desired value

set(gcf,'defaultAxesFontSize',textSize);
set(gcf,'defaultTextFontSize',textSize);
set(gcf,'defaultLegendFontSize',textSize);
set(gcf,'defaultAxesFontSizeMode','manual');
set(gcf,'defaultTextFontSizeMode','manual');
```

```
374 set(gcf,'defaultLegendFontSizeMode','manual');
375 set(gcf,'defaultAxesLabelFontSizeMultiplier',1);
376 set(gcf,'defaultAxesTitleFontSizeMultiplier',1);
377 set(gcf,'defaultAxesFontName','Times');
378 set(gcf,'defaultTextFontName','Times');
379 screenPos=get(groot,'ScreenSize')/get(groot,'
    ScreenPixelsPerInch')*2.54;
380 figPos(1)=screenPos(3)/2-width/2;
381 figPos(2)=screenPos(4)/2-height/2;
382 figPos(3)=width;
383 figPos(4)=height;
384 colors{1} = [0.0 0.0 1.0];
385 colors{2} = [1.0 0.0 0.0];
386 colors{3} = [0.0 0.8 0.0];
387 colors{4} = [1.0 0.5 0.0];
388 colors{5} = [0.0 0.8 0.8];
389 colors{6} = [0.8 0.0 0.8];
390 colors{7} = [1.0 0.8 0.0];
391 set(gcf,'defaultAxesColorOrder',[colors{1};colors{2};colors
    {3};colors{4};colors{5};colors{6};colors{7}]);
392 set(gcf,'Units','centimeters');
393 set(gcf,'Position',figPos);
394 set(gcf,'Color',[1 1 1]);
395 colormap('turbo');
396 title('von Mises Stress [MPa]', 'FontWeight', 'normal', '
    FontName', 'Times', 'FontSize', 12, 'Interpreter', 'latex')
    ;
```

5.2 eval_maxStress_COMSOL

La funzione `eval_maxStress_COMSOL` è progettata per analizzare i risultati di stress ottenuti dalla simulazione COMSOL configurata e lanciata da `eval_vonMisesStress_COMSOL`. Infatti, riceve in input la matrice dei valori di stress, calcolati dalla simulazione, e la struttura contenente il limite massimo di stress ammissibile per il lamierino della macchina in esame.

La funzione verifica ogni punto di stress nella matrice e, se il valore di stress supera il limite, registra le coordinate di quel punto. Inoltre, individua il punto con il massimo stress e ne salva le coordinate. In ultimo, restituisce a `eval_vonMisesStress_COMSOL` le

coordinate dei punti critici e del punto con il massimo stress attraverso la struttura `out`. In questo modo, si facilita l'identificazione delle aree potenzialmente problematiche e si offre la possibilità di correggere il progetto in relazione al funzionamento ad alte velocità.

Anche per questa funzione, viene presentato un flow-chart, in Figura 5.4, in cui è mostrato il flusso logico seguito dalla funzione, una volta lanciata. Inoltre, la funzione può essere eseguita in modo autonomo attraverso la *Command Window*, a patto di stare usando LiveLink *for* MATLAB di COMSOL.

Figura 5.4. Flusso logico di eval_maxStress_COMSOL

5.2.1 Input della Funzione

- **Stress**: La matrice $Nx3$ dove ogni riga rappresenta un punto con le coordinate x, y e il valore di stress in quel punto.

- **mat**: La struttura contenente i dati relativi ai materiali, incluso il valore massimo di stress ammissibile.

5.2.2 Output della Funzione

- **out**: La struttura contenente i risultati dell'analisi sullo stress massimo.

5.2.3 Dettagli delle Operazioni

1. **Inizializzazione delle Variabili:**

```
17 sigma_max = mat.Rotor.sigma_max; %[MPa]
```

   In questo passaggio si estrae il valore massimo di stress ammissibile in MPa, `sigma_max`, dalla struttura `mat`.

2. **Inizializzazione dei Vettori di Accumulo:**

```
19 stress_rot = [];
20 x_stress = [];
21 y_stress = [];
```

   Successivamente, vengono inizializzati tre vettori vuoti in cui memorizzare le informazioni sui punti critici di stress e sul punto di massimo stress.

3. **Ciclo di Analisi dei Punti di Stress:**

```
23 for kk = 1:size(Stress, 1)
24     if Stress(kk, 3) > sigma_max
25         x_stress = Stress(kk, 1);
26         y_stress = Stress(kk, 2);
27     end
28     stress_rot = [stress_rot; x_stress, y_stress];
29 end
```

   Il nucleo della funzione, dove si esplora la matrice degli stress e, se il valore di tensione meccanica, nella riga corrente, supera il massimo ammissibile, si memorizzano le coordinate x e y in `x_stress` e `y_stress`, e si aggiungono al vettore `stress_rot`.

4. **Individuazione del Punto con Stress Massimo:**

```
34 [~, idx] = max(Stress(:, 3));
35 x_max_stress = Stress(idx, 1);
36 y_max_stress = Stress(idx, 2);
```

Come ultima analisi, si utilizza la funzione `max` per trovare l'indice del massimo valore di stress, nella terza colonna della matrice `Stress`, e si estraggono le coordinate x, y del punto corrispondente.

5. **Output dei Risultati:**

```
40 out.stressrot = stress_rot;
41 out.x_over = x_stress;
42 out.y_over = y_stress;
43 out.x_max = x_max_stress;
44 out.y_max = y_max_stress;
```

Infine, si raggruppano i risultati in una struttura `out`, contenente le coordinate dei punti che superano i valori massimi di stress consentiti e del punto di stress massimo.

6 Validazione delle simulazioni

In questo capitolo viene descritta la fase di validazione delle simulazioni elettromagnetiche e strutturali effettuate utilizzando COMSOL Multiphysics. Questa fase è essenziale per garantire l'accuratezza e l'affidabilità dei risultati ottenuti, assicurando che l'interfaccia sia stata implementata correttamente. Il processo di validazione delle simulazioni rappresenta un passaggio fondamentale per lavori di questo tipo, poiché permette di confrontare i risultati numerici con dati sperimentali o con soluzioni analitiche note, assicurando che i modelli implementati siano corretti e che le ipotesi adottate siano valide.

L'interfaccia è stata progettata per consentire simulazioni elettromagnetiche e strutturali sul modello della macchina sviluppato in SyR-e. Di conseguenza, per ottenere un riscontro sulla bontà dell'implementazione, è necessario effettuare delle verifiche. La validazione delle simulazioni presentata nei paragrafi successivi viene effettuata attraverso un confronto tra i risultati ottenuti tramite le simulazioni effettuate da SyR-e, con gli strumenti attualmente disponibili, e le simulazioni effettuate da COMSOL Multiphysics, utilizzando lo stesso modello di partenza e configurato con gli stessi parametri.

Nel caso delle simulazioni magnetiche, la validazione viene effettuata utilizzando il *DefaultMotor* di SyR-e, lanciando due simulazioni: una a vuoto e una a carico con punto di lavoro sul MTPA, entrambe alla velocità base, sia in SyR-e che in COMSOL. Questo approccio permette di osservare, a parità di condizioni iniziali, i risultati ottenuti dai due software ed eventualmente apportare le modifiche necessarie allo script dell'interfaccia. Le funzioni principali per la validazione delle simulazioni elettromagnetiche sono `eval_operatingPointCOMSOL`, `COMSOLfitness` e `simulate_xdeg_COMSOL`. Queste funzioni, descritte nei capitoli precedenti, sono fondamentali per effettuare le simulazioni FEA all'interno dell'ambiente multifisico di COMSOL e per estrarre i risultati da presentare all'utente in ambiente MATLAB.

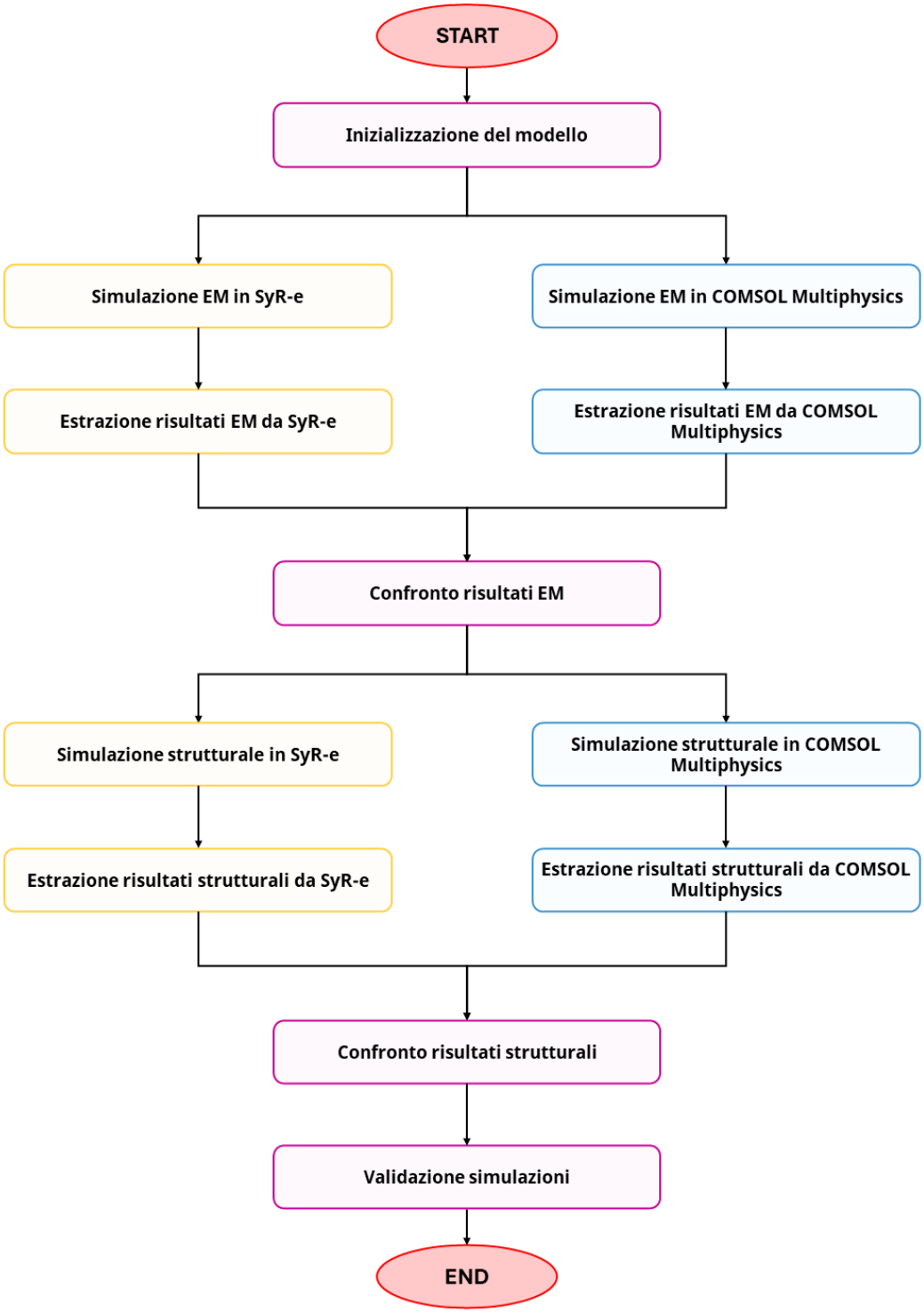

Figura 6.1. Flusso logico di validazione delle simulazioni

Nel caso delle simulazioni strutturali, la validazione viene effettuata, sempre utilizzando il *DefaultMotor* di SyR-e, lanciando due simulazioni: una alla velocità massima e una a velocità superiore, per evidenziare la capacità dell'interfaccia di segnalare superamenti dei limiti strutturali della macchina, sia in SyR-e che in COMSOL. Anche in questo caso, è possibile osservare, a parità di condizioni iniziali, i risultati ottenuti dai due software ed eventualmente apportare le modifiche necessarie allo script dell'interfaccia. Le funzioni `eval_vonMisesStress_COMSOL` e `eval_maxStress_COMSOL` sono impiegate per valutare le tensioni strutturali nei materiali, permettendo di determinare i punti critici e di verificare la resistenza del lamierino di rotore sottoposto a carico. Queste funzioni calcolano rispettivamente le tensioni di von Mises e le massime tensioni nel modello, offrendo un'analisi dettagliata delle sollecitazioni strutturali.

L'accurata preparazione dei dati, la corretta esecuzione delle simulazioni e il rigoroso post-processing dei risultati sono essenziali per una valida analisi delle prestazioni elettromagnetiche e strutturali. L'interfaccia tra SyR-e e COMSOL garantisce un approccio sistematico alla validazione, migliorando la qualità delle simulazioni e facilitando il processo di ottimizzazione e verifica per gli utenti. Questo approccio non solo aumenta la fiducia nei risultati delle simulazioni, ma promuove anche un utilizzo efficiente e accurato degli strumenti disponibili in COMSOL Multiphysics. La Figura 6.1 illustra il processo di validazione in dettaglio, fornendo una chiara visione d'insieme delle diverse fasi coinvolte.

6.1 Simulazioni Magnetiche

In questo paragrafo vengono presentati i risultati della validazione delle simulazioni elettromagnetiche, ottenuti confrontando i dati generati da SyR-e e COMSOL Multiphysics. La validazione delle simulazioni magnetiche è fondamentale per garantire l'accuratezza e l'affidabilità del modello creato attraverso l'interfaccia in COMSOL. A tal fine, sono state condotte due simulazioni su due diversi punti di lavoro del motore, valutando la coerenza dei risultati tra i due software.

Prima di entrare nel merito della validazione magnetica, è necessario effettuare un richiamo delle interfacce fisiche utilizzate in COMSOL, per realizzare l'analisi FEM:

- Rotating Machinery, Magnetic (rmm), collocata all'interno delle interfacce elettromagnetiche (AC/DC>Electromagnetics and Mechanics) di COMSOL, è destinata alla progettazione e all'analisi di motori e generatori elettrici. Supporta la modellazione stazionaria e nel dominio del tempo sia in 2D che in 3D. Questa interfaccia risolve le equazioni di Maxwell formulando le variabili dipendenti attraverso una

combinazione del potenziale vettore magnetico e del potenziale scalare magnetico [6].

- Electrical Circuit (cir), collocata all'interno delle interfacce elettromagnetiche (AC/-DC>Electromagnetics and Mechanics) di COMSOL, è impiegata per modellare correnti e tensioni nei circuiti, inclusi sorgenti di tensione e corrente, resistori, condensatori, induttori e dispositivi a semiconduttore. I modelli creati con questa interfaccia possono includere connessioni a modelli di campo distribuito. Inoltre, supporta la modellazione stazionaria, in frequenza e nel dominio del tempo, risolvendo le leggi di conservazione di Kirchhoff per le tensioni, correnti e cariche associate agli elementi del circuito[6].

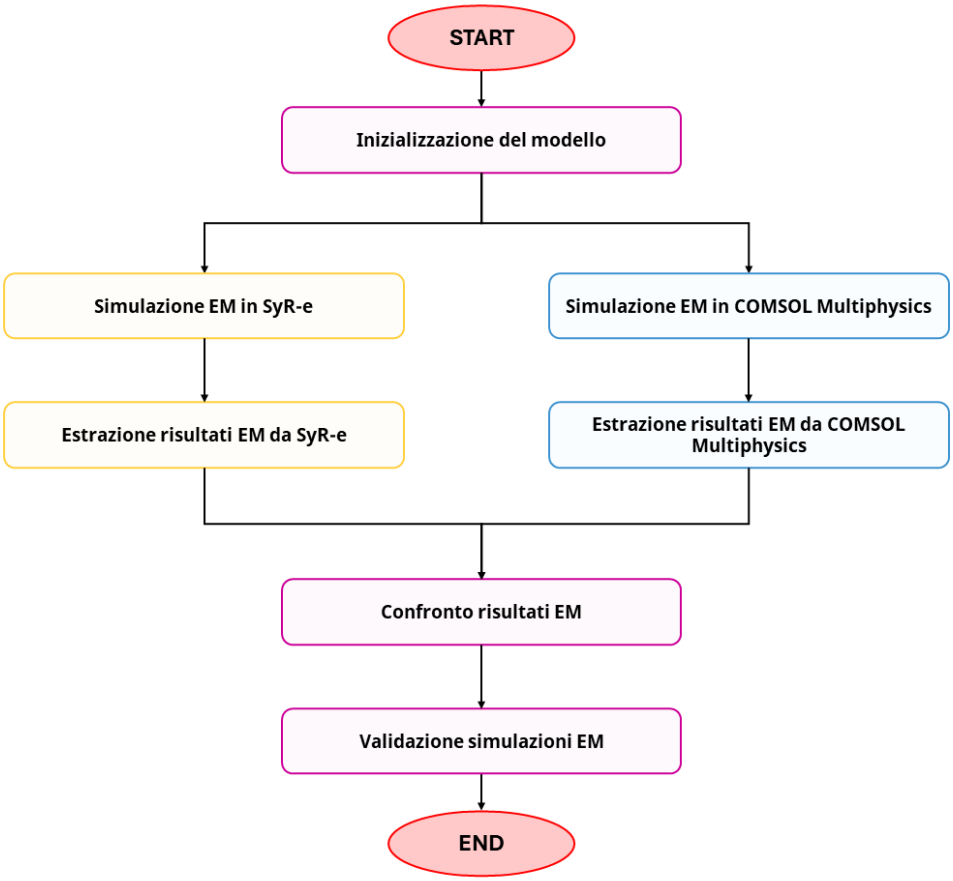

Figura 6.2. Flusso logico di validazione delle simulazioni elettromagnetiche

Dopo aver introdotto le fisiche che stanno alla base delle simulazioni in COMSOL, si passa a presentare il modus operandi utilizzato per effettuare le validazioni elettromagnetiche, presentato concettualmente nel flow-chart di Figura 6.2.

Nel corso di questo studio sono state eseguite due simulazioni utilizzando sia COMSOL Multiphysics che SyR-e, mirando a ottenere risultati il più possibile coerenti tra i due software:

1. La prima simulazione ha impiegato la geometria e la configurazione dati del *syreDefaultMotor* dalla cartella *motorExamples* di SyR-e. Entrambi i software sono stati configurati per simulare il motore con una corrente di fase di 0 A, emulando un funzionamento a vuoto. La velocità di rotazione del motore è stata mantenuta a 4000 rpm, mentre gli altri parametri di simulazione come l'escursione angolare del rotore, il numero di posizioni di rotore simulate e l'induzione residua dei magneti sono stati mantenuti invariati rispetto alle impostazioni predefinite.

2. Nella seconda simulazione, è stata utilizzata la medesima geometria e configurazione dati del *syreDefaultMotor*. In questo caso, entrambi i software sono stati configurati per simulare il motore con una corrente di fase di 802.1 A e un angolo di fase di 148° elettrici, emulando un funzionamento a carico con la corrente nominale sul MTPA. Anche in questa simulazione, la velocità di rotazione del motore è stata mantenuta a 4000 rpm, e gli altri parametri di simulazione sono stati mantenuti invariati rispetto alle impostazioni predefinite.

I risultati delle simulazioni sono stati raccolti e saranno esaminati dettagliatamente nel paragrafo successivo al fine di valutare la robustezza dell'interfaccia tra SyR-e e COMSOL Multiphysics.

6.1.1 Confronto risultati SyR-e – COMSOL

In questo paragrafo vengono presentati i risultati della validazione delle simulazioni magnetiche, effettuate confrontando i dati ottenuti da SyR-e e COMSOL Multiphysics. La validazione delle simulazioni magnetiche è fondamentale per garantire che i modelli utilizzati siano accurati e che le previsioni fatte dai software siano affidabili. A tale scopo, sono state eseguite due simulazioni in due diversi punti di lavoro del motore, al fine di valutare la coerenza dei risultati tra SyR-e e COMSOL.

La prima simulazione è stata effettuata impostando una corrente di 0 A di ampiezza e una velocità di rotazione della macchina pari circa alla velocità base, ovvero 4000 rpm. L'obiettivo di questa simulazione è assicurarsi che il modello sia stato configurato correttamente, in particolare le condizioni al contorno riguardanti il traferro della macchina e

i lati della geometria, che nella realtà non esistono in quanto la sezione trasversale è una circonferenza (con tutti i poli presenti).

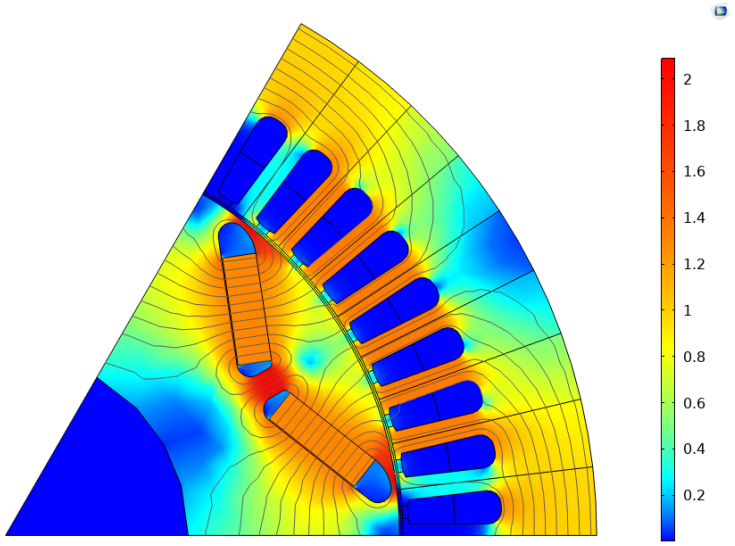

Figura 6.3. Analisi induzione a vuoto COMSOL

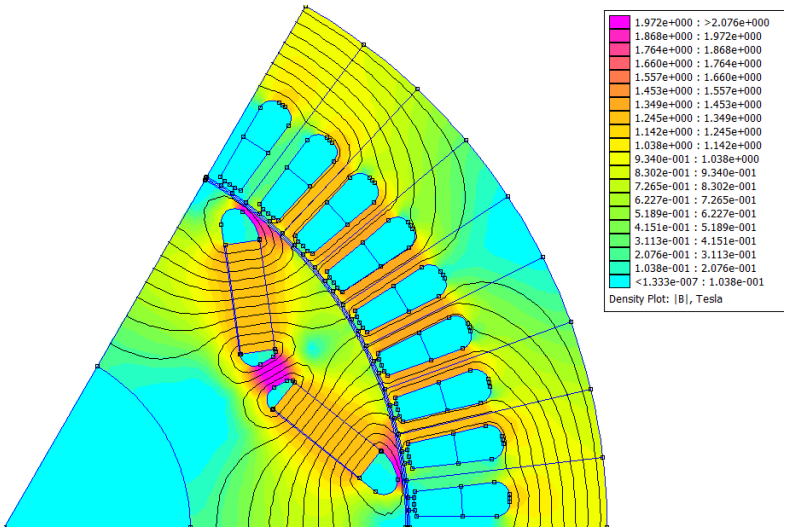

Figura 6.4. Analisi induzione a vuoto FEMM

Inizialmente, è stata verificata la distribuzione dell'induzione magnetica nel polo simulato della macchina in COMSOL Multiphysics rispetto ai risultati della simulazione svolta

nelle stesse condizioni in FEMM da SyR-e. I risultati delle analisi rispettivamente ottenuti da COMSOL e SyR-e, sono mostrati in Figura 6.3 e Figura 6.4. Si può osservare che la distribuzione di induzione calcolata da COMSOL è del tutto analoga a quella calcolata in FEMM (ad eccezione della scala cromatica usata per i due grafici). Inoltre, è possibile osservare che la distribuzione di linee di flusso magnetico è praticamente identica evidenziando in entrambi i casi una distribuzione in asse d del flusso prodotto dai soli magneti permanenti, come ci si aspetterebbe da una situazione di questo tipo.

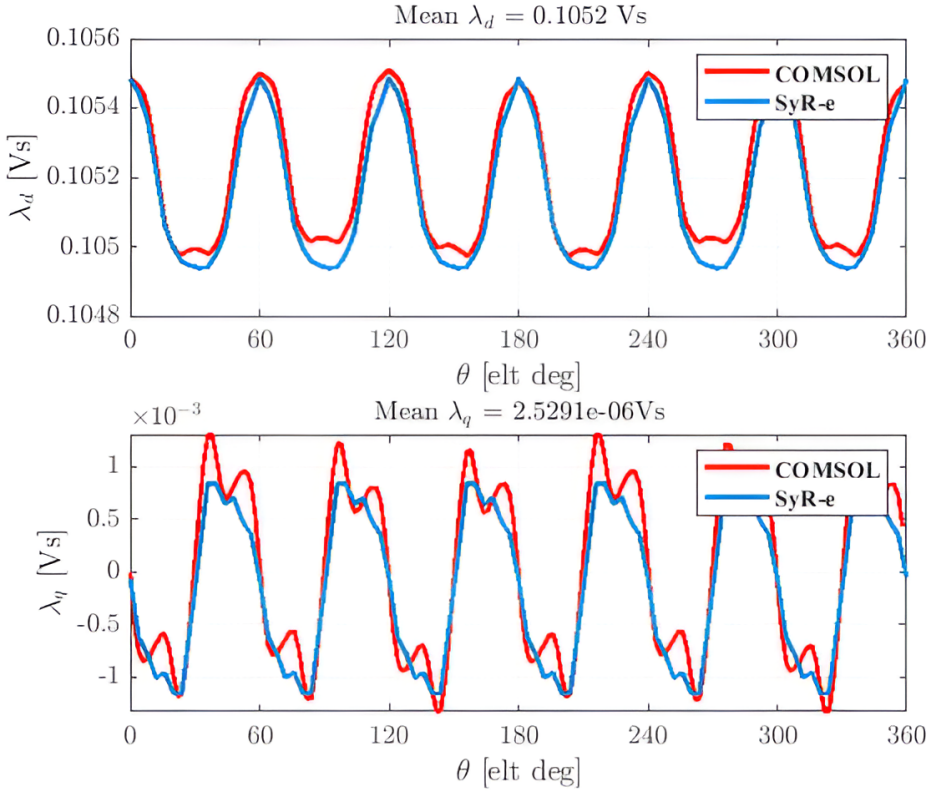

Figura 6.5. Confronto flussi in dq a vuoto SyR-e – COMSOL

Successivamente, si sono effettuate le misure del flusso della macchina, in termini di andamenti di flusso nel sistema di riferimento dq e si sono confrontati i risultati ottenuti da i due software. In Figura 6.5 sono mostrati i grafici del flusso in asse d (in alto) e in asse q (in basso) ottenuti dalla simulazione a vuoto del motore. La curva in rosso rappresenta il flusso calcolato da COMSOL, mentre quella in blu il flusso calcolato da SyR-e. Il punto di interesse maggiore è rappresentato dalla semifigura superiore, poiché essendo a vuoto

e trattandosi di una macchina *V-type*, in questa condizione di funzionamento il flusso si concentra totalmente in asse d il cui totale contributo ricade sui magneti permanenti.

Dall'analisi del flusso in asse q emerge un valore medio della grandezza del tutto trascurabile rispetto a quello in asse d, esattamente come ci si aspetterebbe da un funzionamento a vuoto reale, indice del fatto che l'interfaccia sia calibrata bene. Per quanto riguarda la leggera differenza di forme d'onda tra COMSOL e SyR-e, non è molto utile commentarla in quanto si tratta praticamente solo di rumore della simulazione.

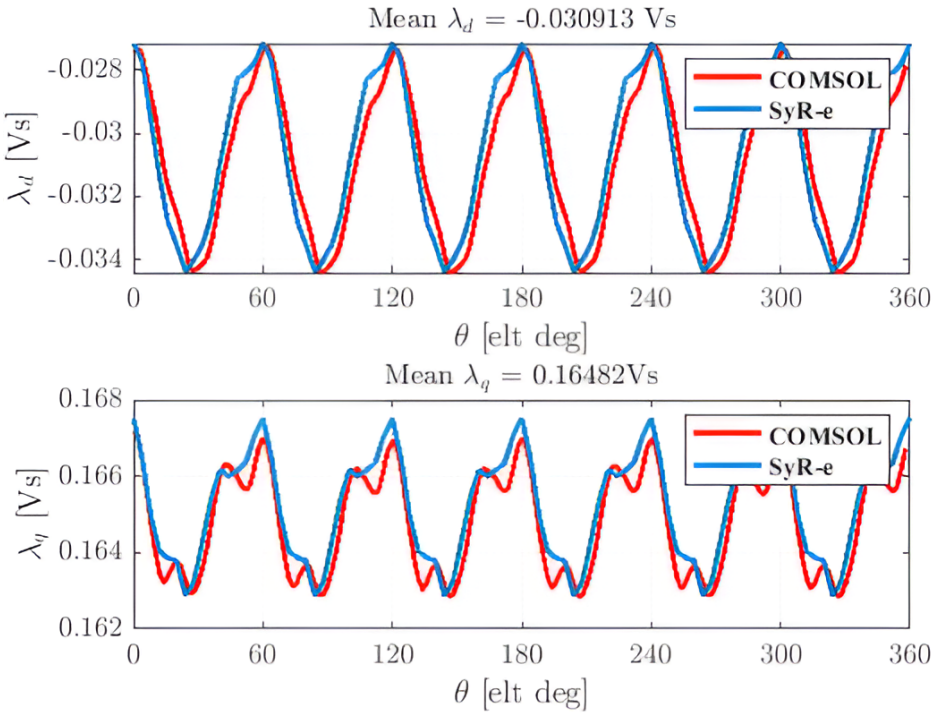

Figura 6.6. Confronto flussi in dq su MTPA SyR-e – COMSOL

In conclusione della prima analisi a vuoto del modello, si può affermare che l'interfaccia sia stata in grado di svolgere un'analisi sufficientemente accurata sui flussi e sull'induzione. Non è stata analizzata in questo caso la coppia a vuoto perché non si è ritenuto di interesse analizzare un segnale di di coppia praticamente trascurabile, lo stesso dicasi per il fattore di potenza intrinseco della macchina.

La seconda simulazione è stata effettuata alimentando la macchina con la corrente nominale, di ampiezza pari a 802.1 A e con un angolo di fase di 148° elettrici, corrispondente al MTPA della macchina e imponendo una velocità di rotazione di 4000 rpm che, in altri termini, equivale a una frequenza di alimentazione di 200 Hz. In Figura 6.6 sono mostrati

i risultati ottenuti dalle simulazioni in SyR-e e COMSOL, riguardanti i flussi dq mentre in Figura 6.7 sono mostrati i grafici della coppia e del fattore di potenza intrinseco (IPF) del motore.

Durante la calibrazione dell'interfaccia, si è notato che impostando in COMSOL, una corrente di 802.1 A, i valori di flusso e coppia calcolati dalla conseguente simulazione erano eccessivamente più elevati di quelli ottenuti da SyR-e. Così tanto maggiori da non poter essere giustificati da una diversa modalità risolutiva della simulazione o da impostazioni diverse dei solver dei due software.

Continuando la calibrazione si è osservato che, per questa macchina, è necessario dividere il valore della corrente inserita nel modello COMSOL per 6, al fine di ottenere dei risultati analoghi a quelli di SyR-e. Il tutto si è tradotto, attraverso altre analisi, alla parametrizzazione di un fattore correttivo, infatti l'interfaccia divide il valore della corrente per il rapporto tra il numero di poli della macchina e il numero di poli simulati, che per il *syreDefaultMotor* si traduce proprio nel numero 6. Questa incongruenza di valori di corrente è molto probabilmente dovuta a un'inconsistenza della boundary condition che definisce la continuità/antiperiodicità in COMSOL.

Fatta la necessaria premessa riguardante la calibrazione della simulazione COMSOL a carico, si può procedere con l'analisi dei risultati ottenuti, tenendola comunque in considerazione. Infatti, analizzando la Figura 6.6, il flusso in asse d calcolato da COMSOL, non si sovrappone perfettamente a quello calcolato da SyR-e, in tutti i punti, nonostante le due forme d'onda presentino andamenti del tutto analoghi. L'unica differenza degna di nota è rappresentata dai valori medi dei due flussi che differiscono 0.5 mVs. Passando ad analizzare l'asse q, si osserva come, anche in questo caso, le due forme d'onda ottenute dai due software non si sovrappongano perfettamente, seppur mantenendo gli stessi andamenti, per via di quanto spiegato in precedenza.

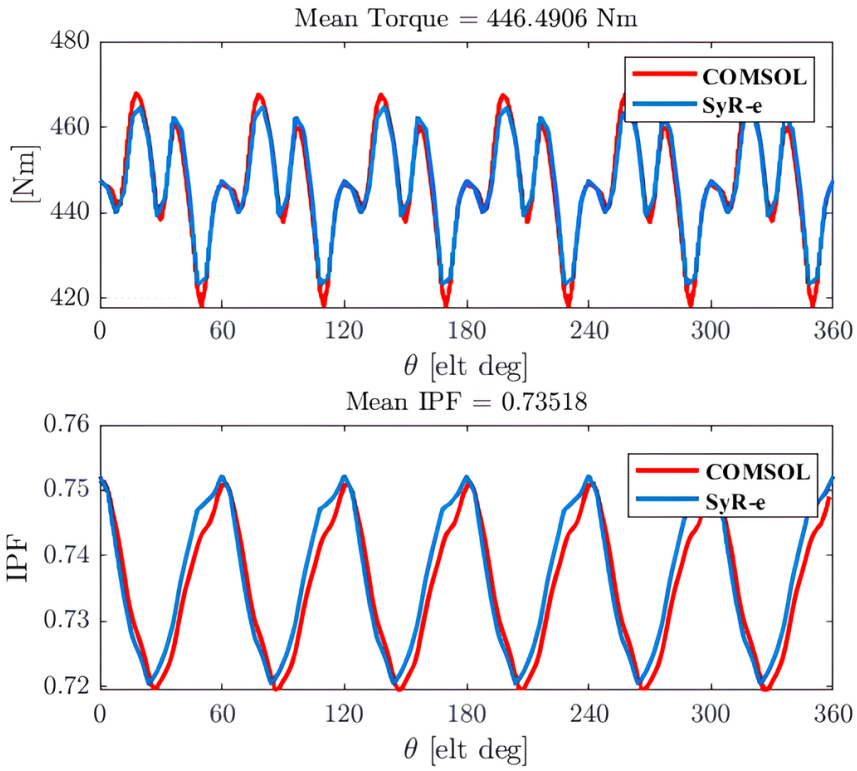

Figura 6.7. Confronto coppia e IPF su MTPA SyR-e – COMSOL

In questa seconda simulazione, si sono confrontati i grafici di coppia e IPF ottenuti dalla simulazione in SyR-e (blu) e COMSOL (rosso), evidenziati in Figura 6.7. Dall'analisi della coppia si può osservare come le due curve presentino andamenti analoghi e siano quasi totalmente sovrapposte. I valori medi delle due coppie differiscono di qualche Nm e la coppia calcolata da COMSOL presenta un andamento caratterizzato da picchi più marcati, probabilmente dovuti al problema riguardante la corrente spiegato in precedenza. Per quanto concerne il fattore di potenza intrinseco della macchina, si evidenziano due forme d'onda del tutto analoghe e con una discrepanza che, di nuovo si può ricondurre al fattore correttivo usato per le correnti, dato che, sia questo fattore, che la coppia, ma anche i flussi, sono fortemente influenzati dalla corrente, tenendo costanti gli altri parametri.

In conclusione, la validazione delle simulazioni elettromagnetiche tramite il confronto tra SyR-e e COMSOL ha confermato la consistenza dell'interfaccia realizzata, evidenziando anche un punto di fragilità del modello in COMSOL, su cui si può agire in futuro per ottimizzare il processo di simulazione che, comunque resta valido, come mostrato dai risultati ottenuti da questo confronto.

6.2 Simulazioni Strutturali

In questo paragrafo viene descritta la fase di validazione delle simulazioni strutturali effettuate utilizzando COMSOL Multiphysics, essenziale per garantire l'accuratezza e l'affidabilità dei risultati ottenuti. Anche in questo caso, come per le simulazioni elettromagnetiche, è opportuno riprendere in considerazione l'interfaccia fisica che COMSOL Multiphysics utilizza per le simulazioni di tipo strutturale, già presentata nel dettaglio, nel capitolo relativo alla simulazione strutturale del modello.

Per realizzare uno studio meccanico strutturale in COMSOL, è necessario utilizzare la fisica *Solid Mechanics*, destinata all'analisi strutturale generale di corpi 3D, 2D o assialsimmetrici. L'interfaccia Solid Mechanics si basa sulla risoluzione delle equazioni del moto insieme a un modello costitutivo per un materiale solido. Vengono calcolati risultati come spostamenti, sforzi e deformazioni [6].

Dopo aver introdotto la fisica che sta alla base delle simulazioni in COMSOL, si passa a presentare il modus operandi utilizzato per effettuare le validazioni strutturali, presentato concettualmente nel flow-chart di Figura 6.8.

Nel corso di questo studio sono state eseguite due simulazioni utilizzando sia COMSOL Multiphysics che SyR-e, mirando a ottenere risultati il più possibile coerenti tra i due software:

1. La prima simulazione ha impiegato la geometria e la configurazione dati del *syreDefaultMotor* dalla cartella *motorExamples* di SyR-e. Entrambi i software sono stati configurati per simulare il solo lamierino di rotore del motore predefinito. Su entrambi gli ambienti di calcolo è stata effettuata una valutazione esclusivamente meccanica del lamierino in accoppiamento meccanico con i magneti permanenti. Infine, il sistema è stato simulato alla velocità massima di rotazione della macchina, ovvero 18100 rpm.

2. Nella seconda simulazione, è stata utilizzata la medesima geometria e configurazione dati del *syreDefaultMotor*. Anche in questo caso, entrambi i software sono stati configurati per simulare il lamierino di rotore con i magneti all'interno delle barriere di flusso ed effettuando una valutazione esclusivamente meccanica. Per questa simulazione il sistema è stato fatto ruotare a 32000 rpm, in modo tale da ottenere una simulazione delle condizioni critiche del materiale che costituisce il rotore e poter testare effettivamente la bontà delle analisi strutturali via COMSOL.

I risultati delle simulazioni sono stati raccolti e saranno esaminati dettagliatamente nel paragrafo successivo al fine di valutare la robustezza dell'interfaccia tra SyR-e e COMSOL Multiphysics.

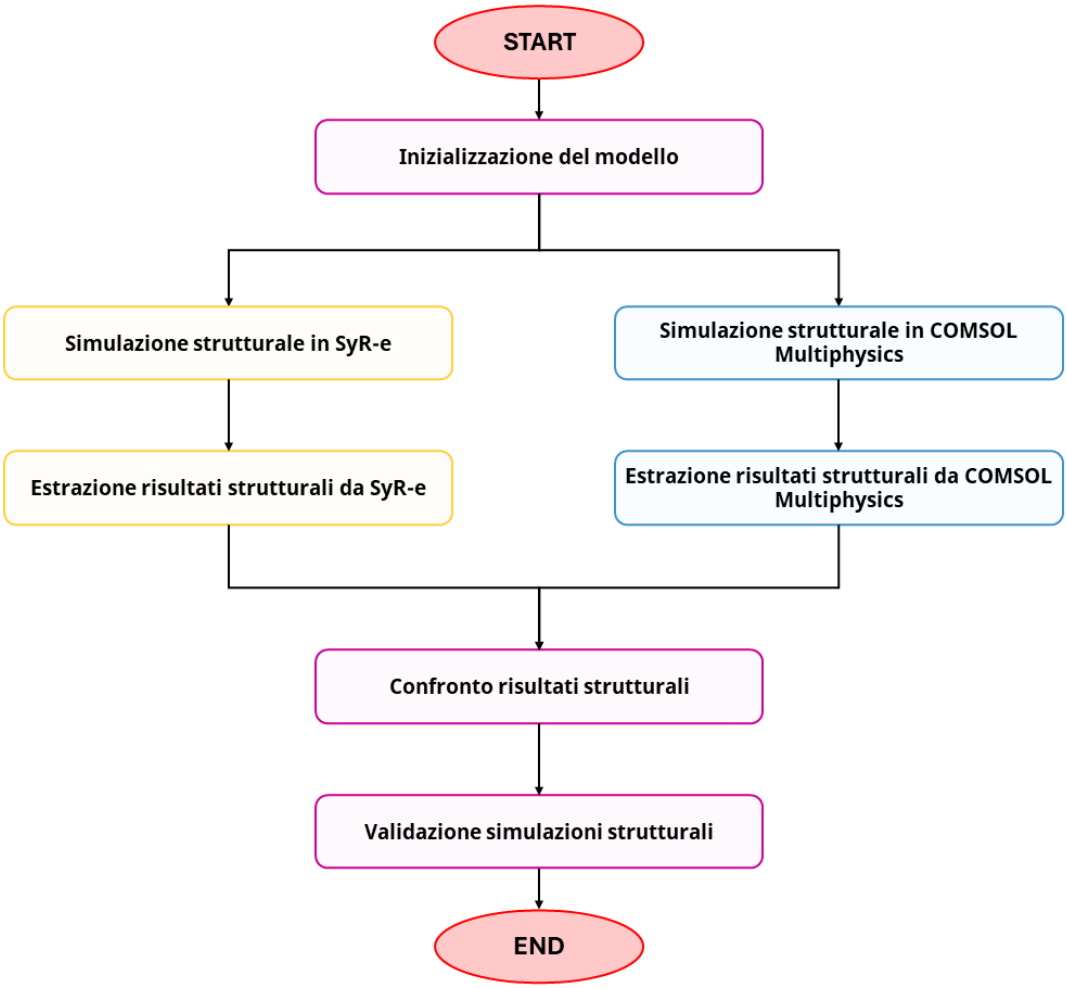

Figura 6.8. Flusso logico di validazione delle simulazioni strutturali

6.2.1 Confronto risultati SyR-e – COMSOL

In questo paragrafo vengono presentati i risultati della validazione delle simulazioni strutturali, ottenuti confrontando i dati generati da SyR-e e COMSOL Multiphysics. La validazione delle simulazioni meccaniche è fondamentale per garantire l'accuratezza e l'affidabilità del modello creato attraverso l'interfaccia in COMSOL. A tal fine, sono state condotte due simulazioni su due diversi punti di lavoro del motore, valutando la coerenza dei risultati tra i due software.

In particolare, è stato analizzato il *syreDefaultMotor*, come fatto in precedenza per

le simulazioni elettromagnetiche, eseguendo una simulazione strutturale sul lamierino di rotore e i magneti permanenti, impostando una velocità di rotazione del sistema di 18100 rpm, ovvero la velocità massima prevista per questo motore, per la prima simulazione e una velocità di rotazione di 32000 rpm, per la seconda simulazione.

I risultati ottenuti dalle simulazioni sono illustrati nelle figure seguenti, all'interno del paragrafo.

Come mostrato in Figura 6.9 e Figura 6.10, si osserva una leggera differenza nei risultati tra le analisi dei due software. Nella simulazione condotta con COMSOL (Fig. 6.9), lo stress massimo rilevato nel lamierino è di circa 464 MPa, mentre nella simulazione condotta con FEMM (Fig. 6.10), lo stress massimo risulta essere di circa 490 MPa. Questa discrepanza può essere attribuita all'utilizzo di una mesh impostata in modo automatico, in COMSOL. Sebbene, sia stata impostata la mesh più fine possibile, è presente una differenza del 5.4% circa tra i due risultati, che tuttavia risulta accettabile considerato la mesh impostata automaticamente in COMSOL Multiphysics.

L'obiettivo di questo confronto rimane l'uniformità nella distribuzione degli sforzi sul lamierino tra le analisi di SyR-e e COMSOL e la possibilità di effettuare simulazioni strutturali attendibili in COMSOL. In questo senso, si può affermare che l'obiettivo sia stato raggiunto, poiché le due figure mostrano una distribuzione degli sforzi molto simile, con concentrazione degli sforzi negli stessi punti critici.

La validazione incrociata dei risultati tra SyR-e e COMSOL Multiphysics dimostra l'accuratezza del modello di simulazione meccanica e supporta ulteriori studi e ottimizzazioni del motore basati su questi strumenti.

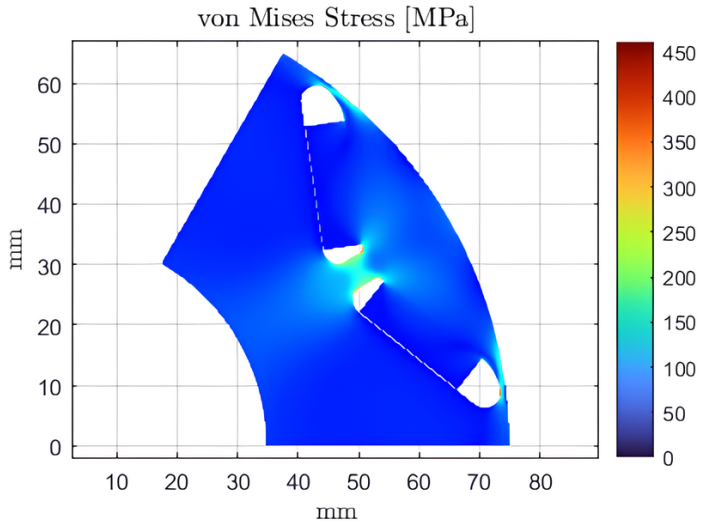

Figura 6.9. Analisi von Mises Stress COMSOL

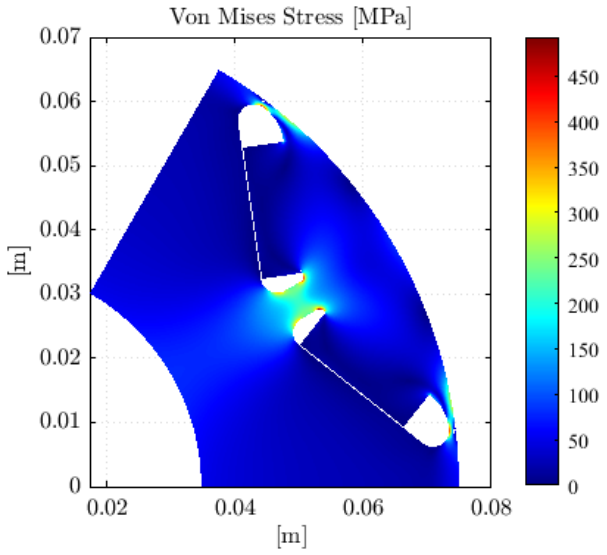

Figura 6.10. Analisi von Mises Stress SyR-e (FEMM)

Proseguendo l'analisi della simulazione effettuata alla velocità massima di rotazione, nelle figure seguenti sono illustrati i diagrammi di *displacement* della macchina. I grafici

mostrano gli spostamenti delle diverse parti del rotore durante la rotazione alla velocità precedentemente specificata, causati dagli sforzi centrifughi dovuti all'alta velocità.

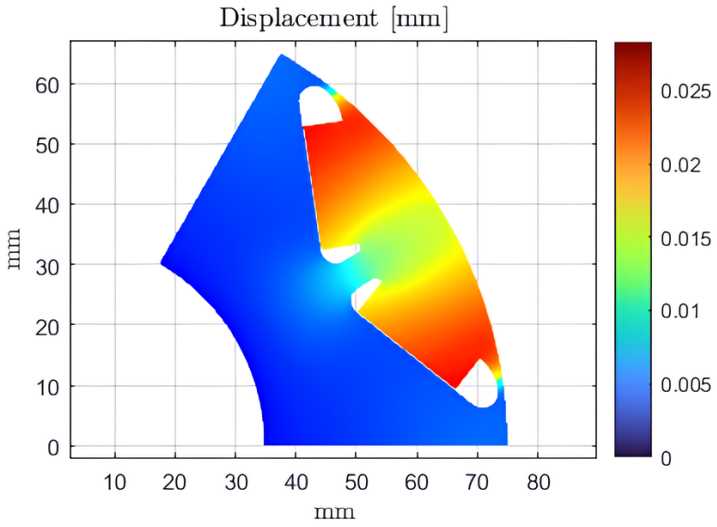

Figura 6.11. Analisi displacement COMSOL

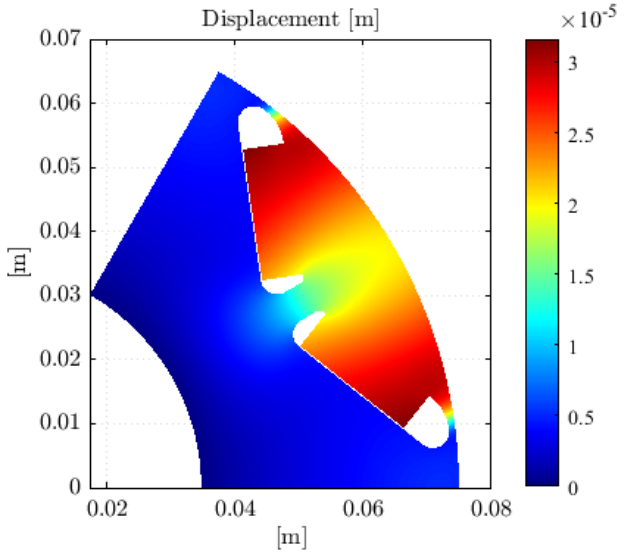

Figura 6.12. Analisi displacement SyR-e (FEMM)

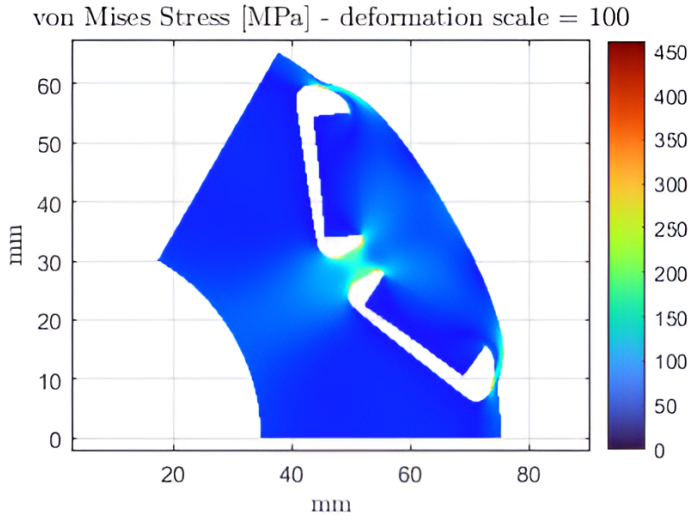

Figura 6.13. Analisi entità del displacement COMSOL

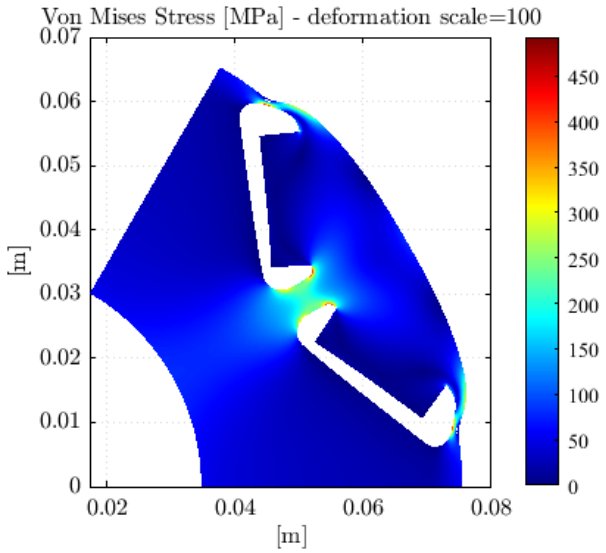

Figura 6.14. Analisi entità del displacement SyR-e (FEMM)

In particolare, le Figure 6.13 e 6.14 presentano rispettivamente i risultati ottenuti da COMSOL e SyR-e. Anche in questo caso, si nota una lieve differenza nei valori massimi

di spostamento registrati dai due software. La simulazione condotta con SyR-e indica un displacement massimo di circa 31.6 μm, mentre quella con COMSOL riporta uno spostamento massimo di 28.3 μm.

Infine, la simulazione a velocità massima si conclude con due figure che mostrano l'entità degli spostamenti subiti dal rotore, utilizzando una *deformation scale* impostata a 100. Queste figure hanno lo scopo di evidenziare l'effetto degli sforzi sul rotore, fornendo all'utente un'idea della deformazione del lamierino all'aumentare della velocità. Inoltre, questi grafici aiutano a identificare quale parte del rotore è maggiormente affetta dalle tensioni generate dalla velocità elevata. Accanto alle figure, sono riportate le scale dei valori di sforzo raggiunti, espressi in MPa.

La seconda simulazione strutturale è stata eseguita impostando una velocità di 32000 rpm con l'obiettivo di evidenziare la capacità dell'interfaccia di individuare e mostrare i punti del lamierino in cui il limite di snervamento viene superato. Come ci si aspetterebbe, questo limite è stato superato in molti punti, come mostrato nella Figura 6.15, che riportano rispettivamente i risultati ottenuti da COMSOL e SyR-e.

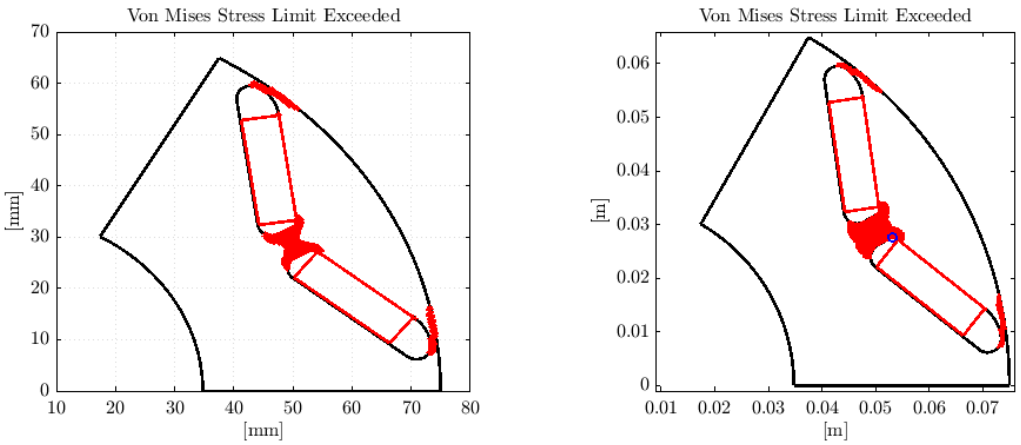

Figura 6.15. A sinistra: Analisi superamento limite di snervamento COMSOL. A destra: Analisi superamento limite di snervamento SyR-e (FEMM).

Sebbene questa analisi abbia un significato fisico marginale, poiché è stata condotta a una velocità doppia rispetto alla velocità massima prevista, è stata eseguita per mostrare il comportamento dello script in caso di superamento del limite di snervamento dell'acciaio del lamierino. I punti rossi nelle figure indicano le aree della mesh dove i software hanno calcolato una tensione superiore ai limiti intrinsechi dei materiali.

In conclusione, la validazione delle simulazioni strutturali tramite il confronto tra SyR-e e COMSOL ha confermato la robustezza dello script di interfaccia tra i due software,

confermando come sia possibile lanciare simulazioni strutturali in COMSOL Multiphysics semplicemente utilizzando il pulsante `COMSOL` presente nella schermata *Simulation* di SyR-e.

7 Conclusioni

Il lavoro di tesi ha avuto come obiettivo principale lo sviluppo di un'interfaccia in MATLAB tra SyR-e e COMSOL Multiphysics, consentendo un'integrazione efficace e automatizzata tra i due software. L'interfaccia sviluppata permette di trasferire il modello del motore elettrico progettato in SyR-e a COMSOL, eseguire simulazioni agli elementi finiti (FEA) in COMSOL ed esportare i risultati nuovamente in SyR-e. Tutto ciò per permettere all'utente di gestire l'intero processo di progettazione, simulazione e analisi direttamente da SyR-e, senza la necessità di dover ricostruire il modello da zero utilizzando manualmente COMSOL.

Il processo di sviluppo dell'interfaccia ha comportato diverse fasi. Inizialmente, è stata definita l'architettura generale del modello della macchina in COMSOL, attraverso l'implementazione di due funzioni. Successivamente, si è proceduto con la configurazione del modello e l'impostazione della simulazione. In questa fase, è stata posta particolare attenzione alla gestione dei file di input e output e alla impostazione delle funzioni nello stesse modalità con cui sono scritte per altri software che si interfacciano con SyR-e, per garantire un flusso di lavoro ottimale e opportunità di personalizzazione dell'interfaccia da parte dell'utente.

Infine, per validare l'efficacia dell'interfaccia, sono state effettuate simulazioni elettromagnetiche e strutturali sul modello del motore elettrico in COMSOL, confrontando i risultati ottenuti con quelli delle simulazioni eseguite direttamente in SyR-e. I confronti hanno mostrato un'elevata concordanza delle grandezze analizzate, sia in termini di forme d'onda che di valori assoluti, confermando la corretta funzionalità e l'affidabilità dell'interfaccia automatizzata. In particolare, le curve dei flussi magnetici, delle sollecitazioni strutturali e delle coppie di trazione ottenute dalle simulazioni in COMSOL hanno mostrato una sovrapposizione del tutto coerente con quelle calcolate in SyR-e, dimostrando la validità del trasferimento di modello e della successiva analisi.

In sintesi, questo lavoro ha dimostrato che l'integrazione tra SyR-e e COMSOL attraverso un'interfaccia MATLAB è non solo fattibile, ma anche efficiente. L'automazione del

processo di simulazione permette di risparmiare tempo e ridurre la possibilità di errori manuali, migliorando la produttività e l'accuratezza delle analisi sul progetto. Le similitudini riscontrate nei risultati delle simulazioni svolte rappresentano un'ulteriore conferma della validità dell'interfaccia sviluppata, che offre un notevole vantaggio in termini di velocità e semplicità d'uso, in quanto consente all'utente di eseguire complesse analisi FEA senza dover necessariamente essere esperti di COMSOL Multiphysics.

Considerando i risultati ottenuti, si possono delineare alcune possibili direzioni per futuri sviluppi:

1. Ottimizzazione dell'Interfaccia:

   Migliorare ulteriormente l'interfaccia per eliminare bug e ridurre i tempi di trasferimento dei dati per aumentare l'efficienza computazionale. Questo potrebbe includere l'ottimizzazione dei codici di scripting MATLAB che costituiscono l'interfaccia. Per esempio si potrebbe investigare la presenza di un disturbo armonico del secondo ordine presente nel flusso in asse d a vuoto, rilevato nella forma d'onda estratta da COMSOL (Figura 6.5).

2. Estensione delle Funzionalità:

   L'integrazione delle analisi termiche e fluidodinamiche è fondamentale per affrontare le problematiche legate al raffreddamento della macchina. L'analisi termica permetterebbe di identificare i punti caldi e ottimizzare la dissipazione del calore, mentre l'analisi fluidodinamica permetterebbe di migliorare il design dei sistemi di raffreddamento. Implementando questi moduli, l'interfaccia renderebbe SyR-e uno strumento multifisico completo ed efficace, in grado di soddisfare una gamma ancora più ampia di esigenze progettuali.

3. Migliorare l'usabilità:

   Continuare a sviluppare la GUI (interfaccia grafica utente) di SyR-e, in modo tale da renderla più *user-friendly* per facilitare l'utilizzo anche da parte di utenti non esperti. Si potrebbero includere elementi di visualizzazione dei risultati in tempo reale, strumenti di diagnostica e opzioni di personalizzazione in base ai software installati dall'utente che si possono interfacciare con SyR-e.

La buona riuscita di questo progetto rappresenta un passo significativo verso l'automazione e l'integrazione dei processi di progettazione e simulazione, aprendo la strada a nuove opportunità di sviluppo multifisico delle analisi FEM in SyR-e. L'interfaccia sviluppata ha il potenziale di diventare uno strumento valido per l'utente, offrendo un modo automatico ed efficiente per condurre analisi complesse e ottimizzare i progetti di motori elettrici in SyR-e.

Bibliografia

[1] S. Ferrari. *SyR-e User Manual*, 2023.

[2] S. Ferrari, E. Armando, and G. Pellegrino. Torque ripple minimization of pm-assisted synchronous reluctance machines via asymmetric rotor poles. In *2019 IEEE Energy Conversion Congress and Exposition (ECCE)*, pages 4895–4902, 2019. doi: 10.1109/ECCE.2019.8912470.

[3] A. Loreto. Progettazione e simulazione di motori elettrici con software magnet in ambiente open-source syr-e. 2019.

[4] C. Multiphysics. "synchronous electric drive 3d". 2020. Available at: https://www.comsol.it/model/motor-tutorial-series-110261.

[5] C. Multiphysics. *COMSOL Multiphysics - Reference Manual*. COMSOL Multiphysics, 2022.

[6] C. Multiphysics. *COMSOL Multiphysics - Reference Manual*, 2022. Available at: https://doc.comsol.com/6.2/doc/com.comsol.help.comsol/COMSOL_ReferenceManual.pdf.

[7] G. Pellegrino and F. Cupertino. Syr-e. Available online at: https://github.com/SyR-e.