

Modélisation des machines électriques dans Flux 3D Overlay 3D

Ghania Bara

▶ **To cite this version:**

Ghania Bara. Modélisation des machines électriques dans Flux 3D Overlay 3D. Sciences de l'ingénieur [physics]. 2015. hal-01827424

HAL Id: hal-01827424
https://hal.univ-lorraine.fr/hal-01827424

Submitted on 2 Jul 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

AVERTISSEMENT

Ce document est le fruit d'un long travail approuvé par le jury de soutenance et mis à disposition de l'ensemble de la communauté universitaire élargie.

Il est soumis à la propriété intellectuelle de l'auteur. Ceci implique une obligation de citation et de référencement lors de l'utilisation de ce document.

D'autre part, toute contrefaçon, plagiat, reproduction illicite encourt une poursuite pénale.

Contact : ddoc-memoires-contact@univ-lorraine.fr

LIENS

Code de la Propriété Intellectuelle. articles L 122. 4
Code de la Propriété Intellectuelle. articles L 335.2- L 335.10
http://www.cfcopies.com/V2/leg/leg_droi.php
http://www.culture.gouv.fr/culture/infos-pratiques/droits/protection.htm

Master I2E2I

Faculté des sciences et technologies
BP70239

54506 VANDOEUVRE LES NANCY

Université de Lorraine
Faculté des Sciences et Technologies

Master Ingénierie Electrique Electronique et Informatique Industrielle

Spécialité « Energie Electrique »

Année universitaire 2014/2015

Modélisation des machines électriques dans Flux 3D Overlay 3D
Electricals machines modeling in Flux 3D Overlay 3D

Mémoire présenté par « BARA Ghania »

Soutenu le 18 septembre 2015

Stage effectué à CEDRAT SA
15 Chemin de Malacher – Inovallée – 38240 MEYLAN
Cedex - France

Tuteur industriel : SOUALMI Abdessamed

Tuteur universitaire : DOUINE Bruno

Table des matières

Remerciements **3**

Introduction **4**

Partie I : Présentation de l'entreprise

I.1. Introduction : **5**

I.2. Présentation des différents logiciels **5**

I.3. Evolution de Flux **7**

I.4. Différents services et équipes de CEDRAT **8**

I.4.1. Equipe R&D 8

I.4.2. Equipe commerciale et Marketing 8

I.4.3. Equipe application 8

I.5. Projet de fin d'étude : Overlay 3D **9**

I.5.1. Problématique et objectif 9

I.5.2. Organisation du travail 9

I.6. Conclusion **9**

Partie II : Modélisation d'une machine synchrone à aimants permanents enterrés

II.1. Introduction **10**

II.2. Modélisation de la machine synchrone à aimants permanents **10**

enterrés dans Flux 3D **10**

II.2.1. Construction de la géométrie 11

II.2.2. Génération du maillage 13

II.2.3. Propriétés physiques 14

II2.4. Résolution 16

II.2.5. Exploitation des résultats 17

II.3. Conclusion **21**

Partie III : Overlay 3D

III.1. Introduction **21**

III.2. Overlay 2D **22**

III.3. Overlay 3D **23**

III.3.1. Mise en place du diagramme UML 23

III.3.2. Génération de l'interface homme machine (IHM) 24

III.3.3. Implémentation dans le langage de commande Flux 24
III.3.4. Construction de la machine en 3D 26
III.3.5. Boîte infinie 26
III.3.6. Boîte air 28
III.3.7. La symétrie et périodicité 32
III.3.8. Le maillage 33
III.3.9. Le bobinage 35
III.4. Overlay 3D : tests 38
III.5. Conclusion 38
Conclusion et perspectives 39
Bibliographie 40
Annexes 41

Remerciements

Le présent rapport de stage a été effectué au sein de l'entreprise CEDRAT au service application dans le cadre de mon projet de fin d'étude de Master en Énergie Electrique à l'Université de Lorraine.

Je tiens tout d'abord à exprimer ma sincère gratitude envers mon Responsable de stage, Dr. Abdessamed SOUALMI, pour sa disponibilité pour ce projet, la confiance qu'il m'a accordé, ainsi que ses idées et conseils judicieux apportés tout au long de ce stage. Sa grande humanité, sa bienveillance à mon égard, ainsi que ses hautes compétences scientifiques font de lui un Ingénieur exceptionnel et un modèle à suivre, je lui en suis infiniment reconnaissante.

Mes remerciements chaleureux vont aussi à l'égard du Dr. Patrick LAMBART, Responsable du service application, pour son support, ses conseils constructifs et son aide inestimable pour la réalisation de ce projet et aussi à Guy JEROME et Vincent LECONTE pour l'intérêt qu'ils ont porté à ce travail.

Je suis profondément reconnaissante envers Katalin TAMAS pour sa disponibilité ainsi que les conseils pertinents et avisés qu'elle m'a prodiguée pour la réussite de ce travail.

Je tiens à remercier également Dr. Farid ZIDAT pour sa rigueur scientifique, ses conseils judicieux et éclairés et ses qualités humaines, qu'il trouve ici l'expression de ma profonde gratitude. J'aimerai remercier maintenant Diana MAVRUDIEVA, Yassine SALHI et Cyril FAVE avec qui nos discussions ont enrichi mes connaissances, ainsi que toute l'équipe support et l'ensemble du personnel de CEDRAT pour leur accueil chaleureux et leur bonne humeur.

J'adresse aussi ma reconnaissance à tous les enseignants du département Electronique, Electrotechnique et Informatique industrielle qui ont contribué à ma formation, qu'ils trouvent ici mon grand respect.

L'expression de mes remerciements va aussi vers mes collègues d'études avec lesquels j'ai eu plaisir à partager des moments conviviaux et des échanges fructueux. Je leurs exprime ma profonde sympathie et leurs souhaite beaucoup de bonheur et de réussite dans leurs vies personnelles et professionnelles.

J'adresse également mes remerciements à mes amis pour le soutien et les encouragements qu'ils m'ont fournis durant mes années d'études.

J'exprime enfin ma profonde gratitude envers les membres de ma famille, en particulier à la mémoire de mes parents. Ce projet n'aurait pu être possible sans leur amour, leur encouragement et leur soutien inconditionnels.

Introduction

Le dimensionnement des machines électriques reste une étape essentielle lors de la réalisation d'un dispositif électromagnétique (machines électriques). Parmi les méthodes utilisées, on trouve :

- Méthode analytique à base des équations de Maxwell
- Méthode semi-analytique à base de réseaux de reluctances
- Méthode numérique

Dans ce mémoire, nous nous intéressons à la modélisation des machines électriques avec la méthode des éléments finis en utilisant le logiciel Flux de CEDRAT. Le projet a pour but de développer une plateforme à base de python qui permet de créer facilement les géométries pour modéliser les dispositifs électromagnétiques (machines électriques, transformateurs, câbles électriques, torons,…). L'interface doit permettre à partir de peu de paramètres de créer l'ensemble de la géométrie.

Le travail réalisé est divisé en trois parties dans ce rapport :

- Partie I

Cette partie consiste à faire une présentation générale de l'entreprise CEDRAT, ainsi que les différents produits à savoir : Flux 2D/3D, Got It, Speed, Inca3D et Portunus et l'évolution du logiciel Flux depuis 1981 à ce jour.

- Partie II

Dans la deuxième partie, une machine synchrone à aimants enterrés est modélisée dans Flux 3D. Les différentes étapes de modélisation (la construction de la géométrie, la définition de la physique, le maillage, la résolution et le post-traitement) sont détaillées. Nous essayons ainsi de dégager les démarches liées à la construction d'une machine dans Flux 3D.

- Partie III

La troisième partie concerne la réalisation de l'Overlay 3D. Par définition, un Overlay consiste à automatiser les différentes étapes de modélisation d'un projet dans Flux 2D/3D. L'Overlay 3D est réalisé en partant des démarches dégagées de la partie II.

Dans notre cas, l'Overlay 3D réalisé consiste à automatiser :

- La création de la géométrie (données géométriques, conditions aux limites, plan de périodicité, plan de symmetrie…)
- Le maillage (automatique, extrusif et mixte)
- Le bobinage : en utilisant des bobines non maillées

Partie I

Présentation de l’entreprise

I.1. Introduction :

Cedrat, éditeur de solutions logicielles dans le domaine du génie électrique, est une entreprise créée en 1971 et implantée à Meylan. Elle réalise un chiffre d'affaires d'environ 5.8 millions d'euros par an. Cette entreprise, composée de 60 employés, vend ses produits à travers le monde (vente d'un peu plus de 50% à l'étranger). Elle distribue ses logiciels grâce à un large réseau de distribution, en Europe et au Moyen-Orient et la société Magsoft, entreprise associée à Cedrat, en Amérique et en Asie.

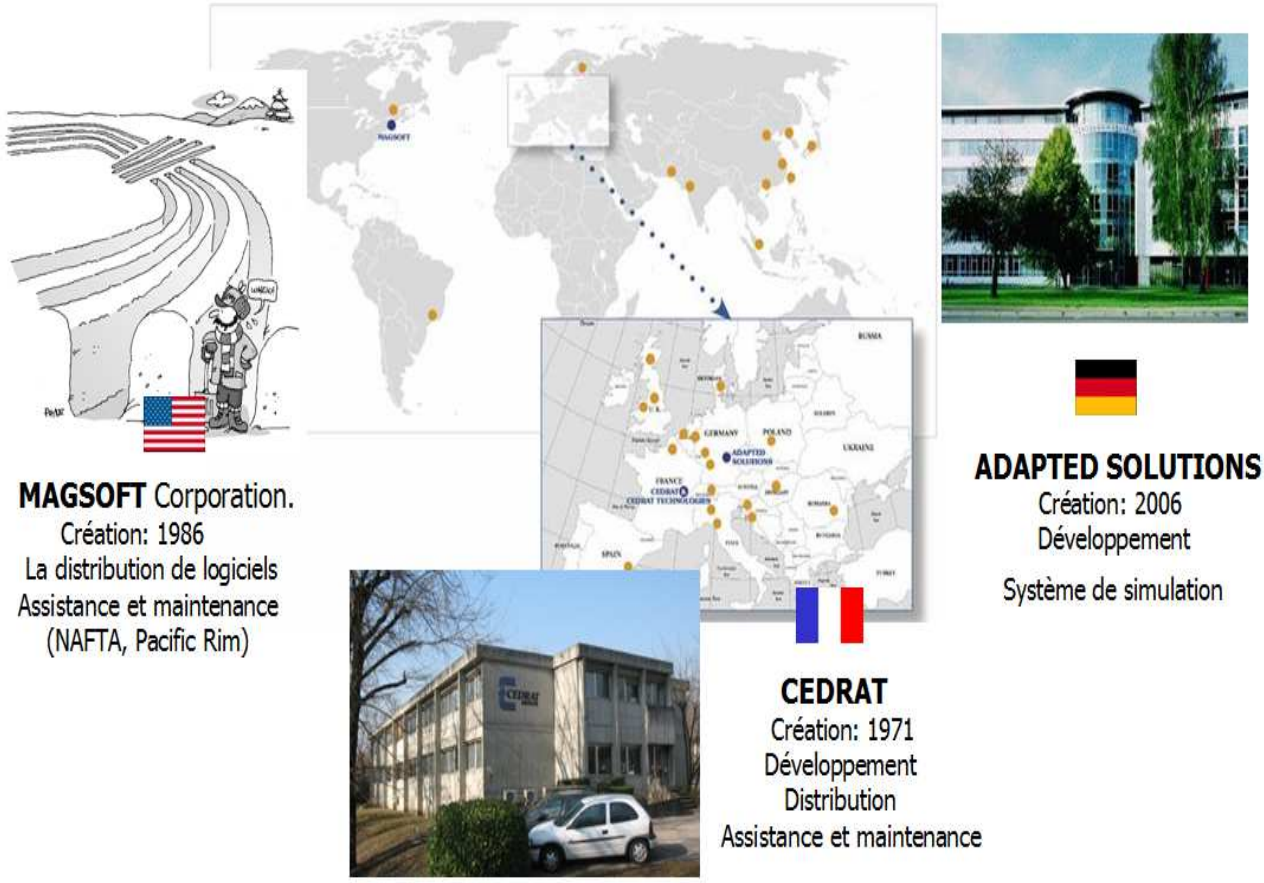


Figure 1 : Cedrat et ses filiales

Cedrat développe une large gamme de logiciels permettant la conception, l'étude et la validation de systèmes dans de nombreux domaines d'application comme l'électronique, l'électromagnétisme ou encore la thermique. Cette entreprise vend ses différents produits aux entreprises ainsi qu'aux universités.

I.2. Présentation des différents logiciels

La renommée de la société a notamment été bâtie autour du logiciel Flux, développé en partenariat avec le G2Elab, celui-ci est spécialisé dans la modélisation et la simulation de systèmes électromagnétiques et thermiques 2D et 3D. Ce logiciel permet l'étude complète de dispositifs complexes grâce aux méthodes des éléments finis et offre aux utilisateurs un large panel de solutions pour leurs travaux. On retrouve plus succinctement les logiciels suivant dans le catalogue de l'entreprise :

GOT-It : permet d'optimiser les configurations existantes de systèmes en fonction de différentes contraintes utilisateurs.

InCa3D : dédié à la modélisation des connexions électriques dans les domaines de l'électronique de puissance et l'électrotechnique

Portunus : dédié pour des simulations systèmes multi-domaines

Speed : dimensionnement de moteurs électriques.

Flux : permet la conception, l'analyse et l'optimisation de système Electromagnétique.

Il s'agit du logiciel que nous allons utiliser dans la suite du projet. Flux 2D/3D permet de calculer et de visualiser les grandeurs utiles à l'ingénieur, pour des dispositifs bidimensionnels, tridimensionnels ou à symétrie de révolution comportant des matériaux à caractéristiques linéaires ou non, isotropes ou non. C'est un logiciel complet ayant l'avantage de permettre le couplage avec les équations de circuits ainsi que l'ajout d'une région surfacique particulière dite « bande de roulement » pour l'étude des machines tournantes avec différentes positions du rotor, sans avoir à modifier la géométrie et le maillage.
La résolution d'un problème fait appel aux modules spécialisés suivants:

- **Module préprocesseur**

Il permet, de définir la géométrie du dispositif à étudier, de choisir et/ou de construire une banque de matériaux, d'affecter les propriétés physiques aux différentes régions géométriques prédéfinies et de définir le schéma et/ou les données du circuit électrique. Il permet également un maillage automatique d'une géométrie 2D/3D prédéfinie.

- **Module processeur**

Il est principalement constitué d'un module de résolution 2D/3D des différents modèles usuels de l'électromagnétisme et des problèmes thermiques.

- **Module postprocesseur de Flux 2D/3D**

Il permet, entre autres, de tracer les équipotentielles ou les lignes de flux, le maillage, la géométrie et les courbes 2D/3D ou selon un chemin prédéfini. Il permet aussi de calculer des grandeurs globales telles que le couple ou la force appliqués à un contour fermé, les inductions, les flux, les inductances, etc.

I.3. Evolution de Flux

La figure 2 montre l'évolution de Flux 2D/3D depuis 1981 à ce jour, plusieurs améliorations ont été réalisés telles que : la prise en compte de la rotation, la translation, le module thermique, couplage circuit, le vrillage, la co-simulation, la vibro acoustique, le maillage adaptatif et le modeleur.

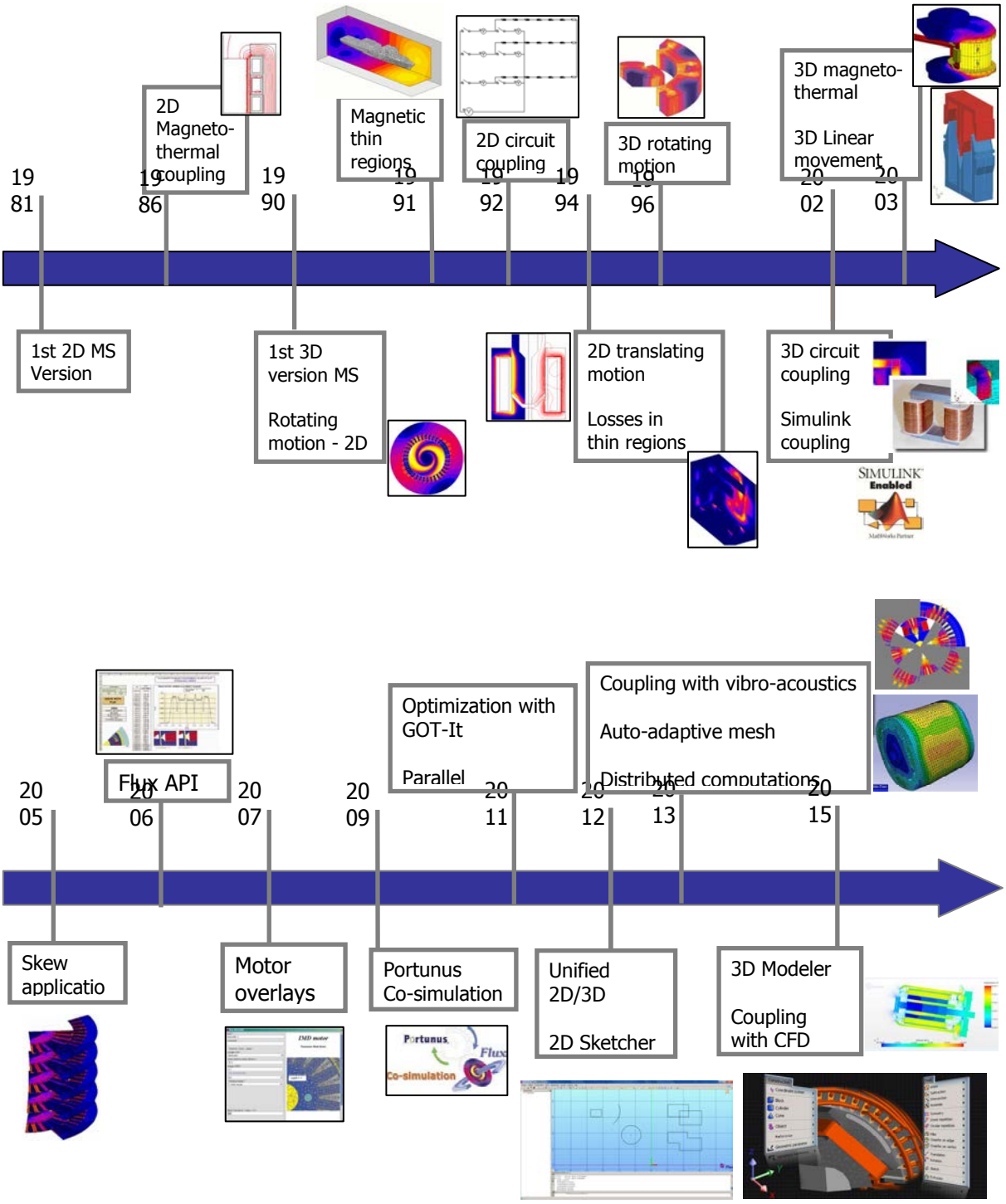


Figure 2 : Evolutions du logiciel Flux 2D/3D

I.4. Différents services et équipes de CEDRAT

CEDRAT est composée de trois équipes : R&D, commerciale et application. Comme le montre la figure 3, les trois équipes sont en contact avec le client d'une façon directe (équipe application et commerciale) ou indirecte (équipe R&D).

I.4.1. Equipe R&D

La principale tâche de cette équipe est le développement des logiciels : Flux 2D/3D, Inca3D et Got-It.

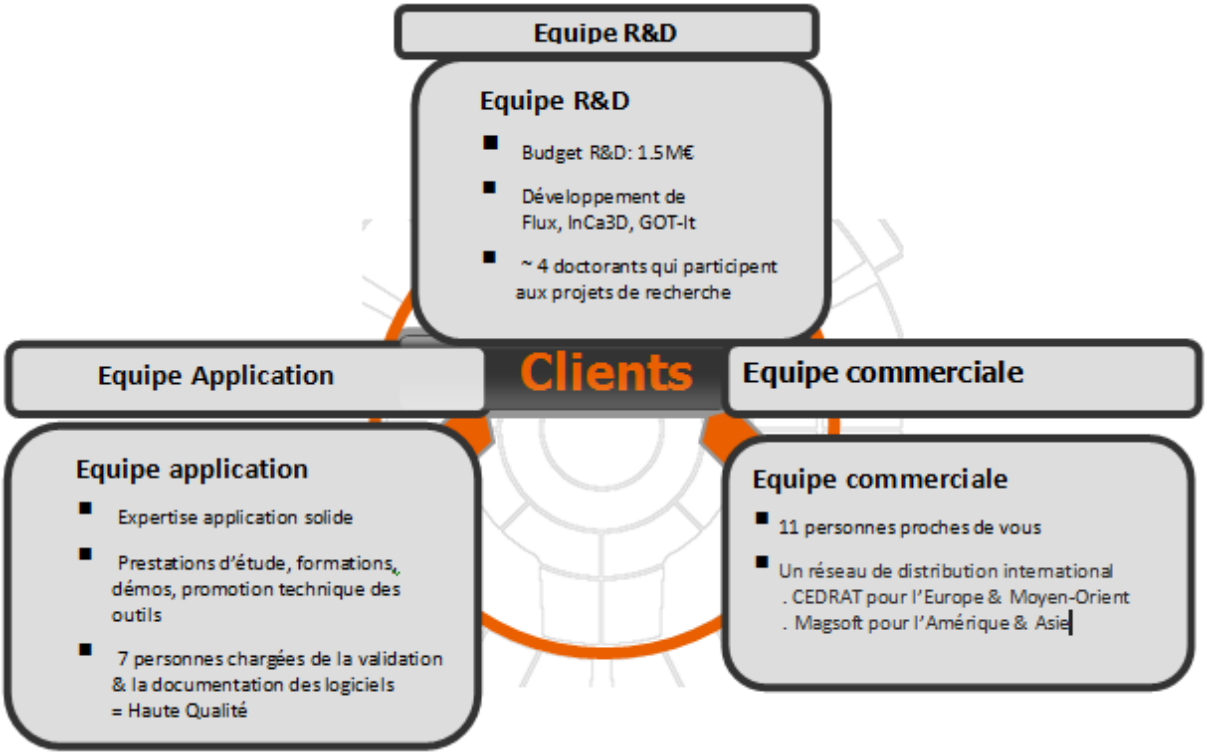


Figure 3 : Différents services de CEDRAT

I.4.2. Equipe commerciale et Marketing

La tâche principale de l'équipe commerciale est la vente des produits de CEDRAT : Flux 2D/3D, Inca3D et Got-It.

I.4.3. Equipe application

Dans l'équipe application, on trouve deux groupes :

Groupe N°1 documentation et validation :

Il s'occupe de la documentation des produits CEDRAT, la mise à jour de la documentation, la création des exemples dans le superviseur de Flux et la validation des versions Flux (le bon fonctionnement des produits CEDRAT).

Groupe N°2 : On trouve deux équipes :

Equipe métier 1 : Moteur, Capteurs, Actionneurs et Chauffage par induction (Flux2D/3D, Got-It)

Equipe métier 2 : Connections électriques & Isolation, Réseaux Electriques, EP-CEM (InCa 3D)

Trois tâches principales sont confiées à l'équipe :

- **Le support technique :** Consiste à apporter de l'aide aux clients sur l'utilisation des différents logiciels. Ces contacts peuvent s'établir par mails ou par téléphone.

- **Le consulting :** Consiste à la réalisation d'études. Des clients, ne souhaitant pas forcément acheter une licence d'un logiciel par exemple, peuvent demander la réalisation de leur étude à l'entreprise Cedrat.

- **Les formations :** Consistent à former les clients sur l'utilisation des logiciels distribués par Cedrat. Elles peuvent être dispensées dans les locaux de l'entreprise cliente ou de ceux de Cedrat.

I.5. Projet de fin d'étude : Overlay 3D

Le projet Overlay 3D est réalisé au sein de l'équipe application avec l'aide de l'équipe R&D.

I.5.1. Problématique et objectif

Le temps de mise en œuvre pour créer des dispositifs éléctromagnetiques peut être considérable. Face à cette problématique, il apparait nécessaire pour Cedrat de proposer une solution qui est l'Overlay 3D. L'objectif du projet de fin d'étude est donc de développer une plateforme à partir de python afin de créer facilement l'ensemble de la géométrie à partir de peu de paramètres pour modéliser les dispositifs électromagnétiques (machines électriques, transformateurs, câbles électriques, torons…).

I.5.2. Organisation du travail

Pour mener à bien ce projet, mon travail a été réalisé selon le planning de la figure 4. Les tâches ont été réalisées les unes après les autres.

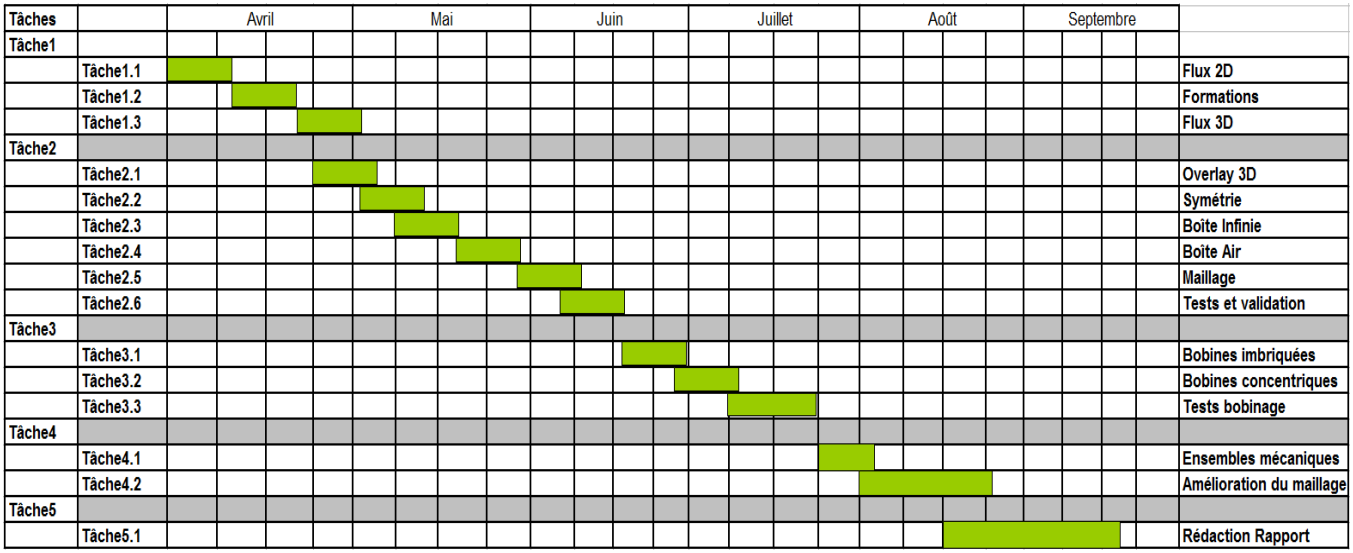


Figure 4 : Planning du PFE

I.6. Conclusion

Une présentation de la société CEDRAT est abordée à savoir la création, l'évolution des produits de CEDRAT (Flux 2D/3D, Inca3D et Got-It), les différents services et équipes (R&D, application et commerciale).

Partie II
Modélisation d’une machine synchrone à aimants permanents enterrés

II.1. Introduction

Après des années de domination (presque exclusive) de la machine asynchrone et les progrès considérables au niveau des matériaux (aimants permanents plus performants), la machine synchrone à aimants permanents s'impose de plus en plus dans divers domaines, et particulièrement dans la traction ferroviaire. Ce choix de motorisation est devenu intéressant, notamment compte tenu des contraintes de réduction de la masse et du volume des équipements embarqués. L'optimisation de la masse est obtenue grâce aux bonnes performances que les aimants de type Néodyme-Fer- Bore ou Samarium-Cobalt confèrent à ce type de machines.

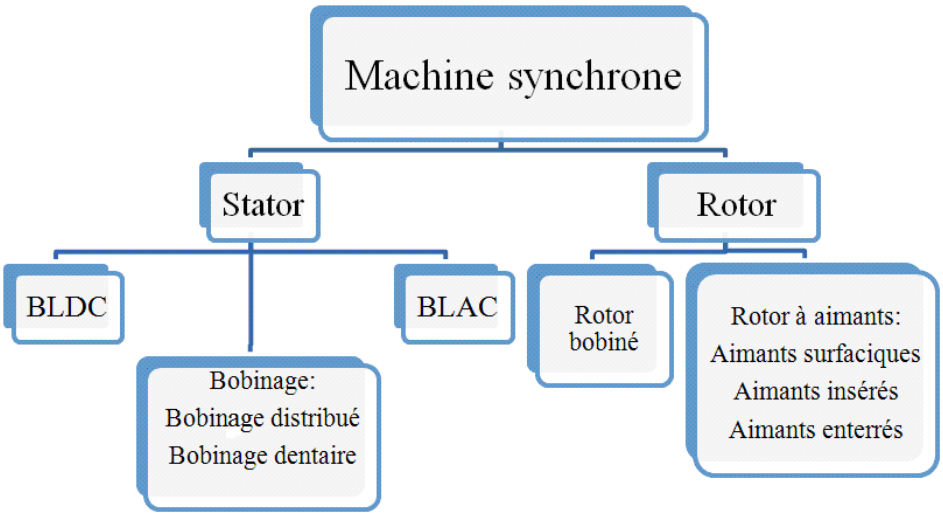


Figure 5 : Classification des machines synchrones à aimants permanents

La figure 5 récapitule la classification des machines synchrones à aimants selon la nature de la force électromotrice, le type de bobinage et la position des aimants au rotor. Plus particulièrement, selon la position des aimants au rotor, on trouve trois types de machines synchrones à aimants : montés en surface, insérés dans la culasse rotorique et enterrés dans la culasse rotorique. Dans ce dernier cas, on peut imaginer que les aimants enterrés sont répartis sur plusieurs couches, mais ce nombre de couches ne peut généralement pas dépasser trois pour des raisons de faisabilité mécanique. Le but de cette partie du stage est d'apprendre et de comprendre le principe de modélisation de machines électriques en 3D. (ex : voir les différentes difficultés rencontrées et le temps de mise en œuvre d'un modèle 3D).

II.2. Modélisation de la machine synchrone à aimants permanents enterrés dans Flux 3D

En général la modélisation d'un dispositif dans Flux 2D/3D en générale passe par 5 étapes essentielles (voir Figure 6) :

- Construction géométrique
- Génération du maillage
- Définition des propriétés physiques
- Résolution
- Exploitation des résultats

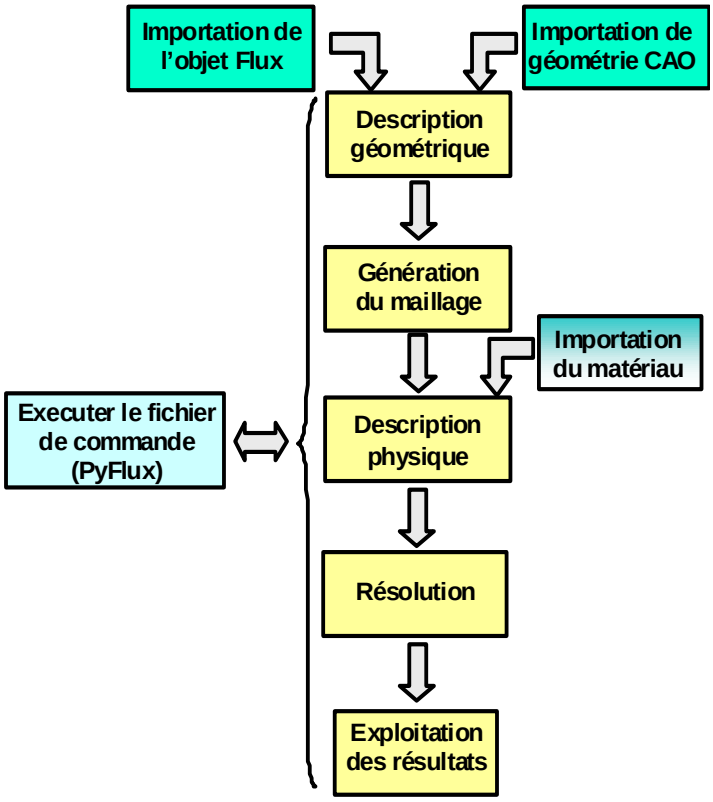


Figure 6 : Phases de construction d'un projet Flux

II.2.1. Construction de la géométrie

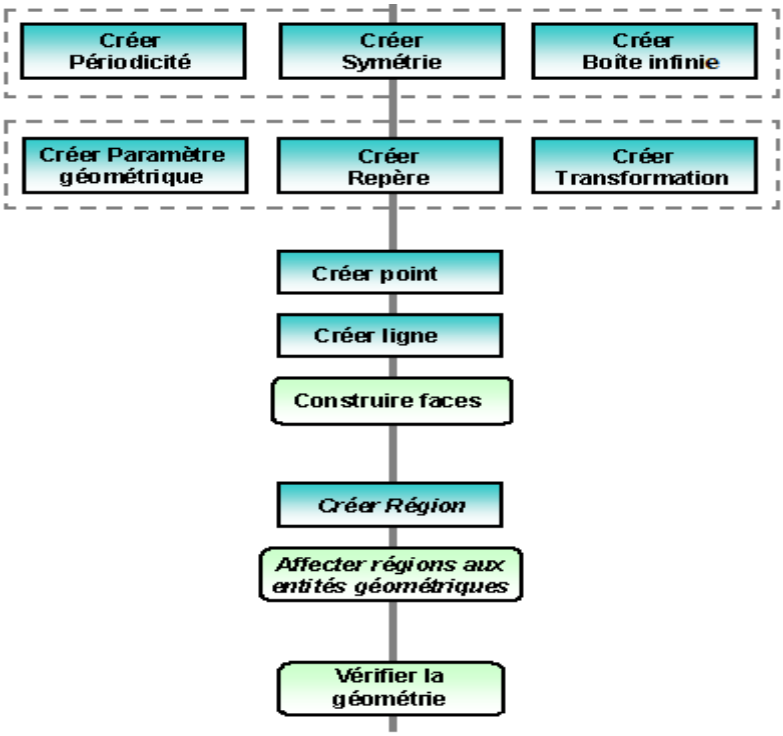
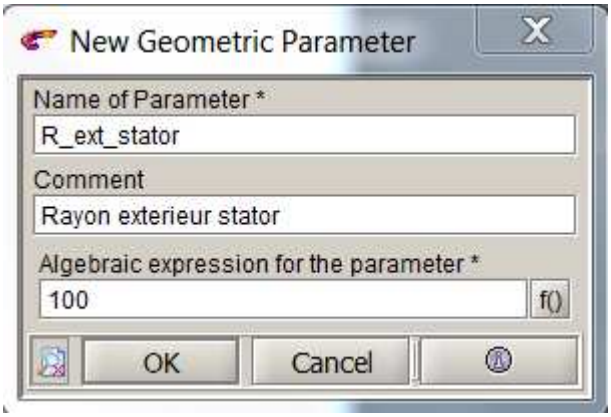
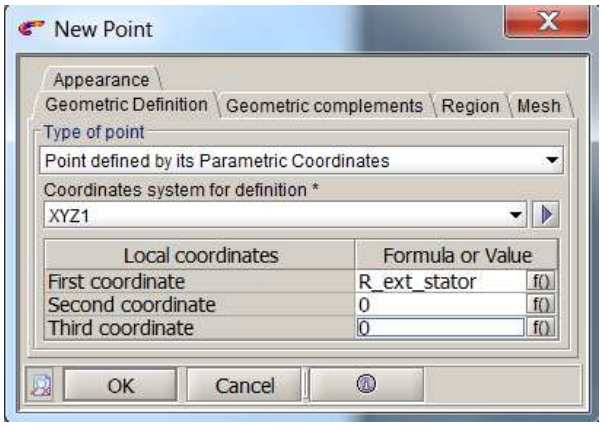


Figure 7 : Démarche générale de construction de la géométrie

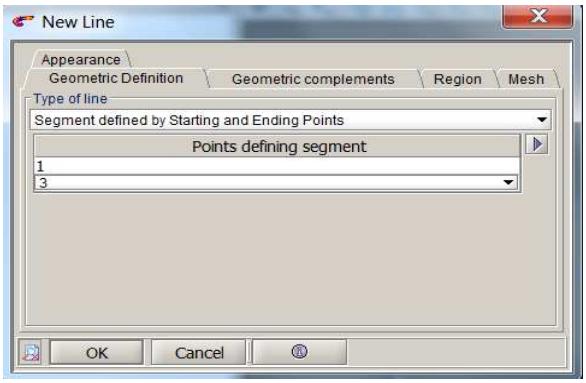
La première étape (voir figure 7) consiste à créer le dispositif dans Flux 2D/3D. Pour cela nous créons les paramètres géométriques qui seront utilisés pour la création des points, lignes, faces et volumes. Pour un gain de temps de calcul, on utilise également les périodicités et les symétries, ce qui permet de ne représenter qu'une portion du dispositif (1/2, 1/4…).



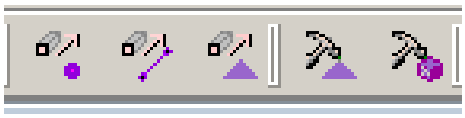
Paramètre géométrique



Création point



Création ligne



Création des faces et des volumes avec des commandes intégrées dans Flux

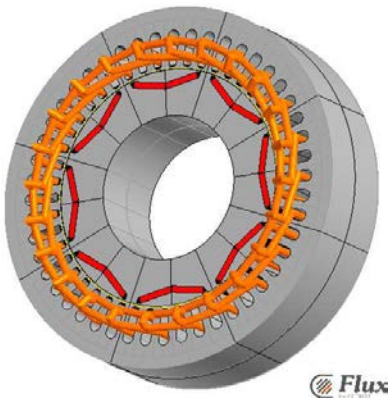


Figure 8 : Machine synchrone à aimants enterrés dans Flux 3D

Dans les nouvelles versions de Flux (Flux 12), la construction d'un dispositif dans Flux 3D est simplifiée avec l'intégration d'un modeleur 3D. De même, dans Flux 2D un Sketcher est intégré, ce qui permet un gain de temps pour la construction géométrique.

II.2.2. Génération du maillage

En général, la résolution par la méthode des éléments finis comprend plusieurs étapes. Une de ces étapes est le maillage qui correspond à la discrétisation du domaine d'étude en éléments (triangles, tétraèdres, hexaèdres…) et nœuds sur lesquels nous calculons:

- Le potentiel magnétique dans le cas d'une application magnétique
- Le potentiel électrique dans le cas d'une application électrique
- La température dans le cas d'une application thermique

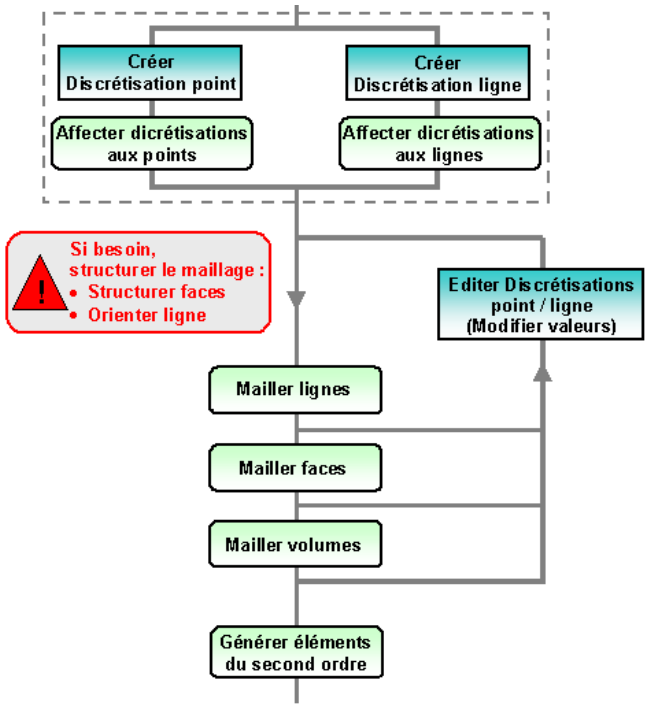


Figure 9 : Démarche générale pour le maillage

La figure 9 montre les différentes étapes du maillage à savoir: la création de la discrétisation (ponctuelle ou linéique), l'affectation des discrétisations et le maillage du dispositif. Dans Flux, il y a trois outils permettant de caractérisé le maillage:

- Le type de mailleur pour définir la forme

Quatre types de mailleurs sont disponibles:

  - Automatique: utilise des éléments triangulaires et tétraédriques. Son avantage est d'être simple et robuste.
  - Réglé: éléments quadrangulaires et parallélépipédiques. L'avantage de ce mailleur permet un contrôle facile de la qualité, adapté aux anisotropies physiques (courants de Foucault...).
  - Relié: ce mailleur permet de copier le maillage d'une face à une autre.
  - Extrusif : il permet de créer le maillage identique sur les couches extrudées et il utilise des éléments quadrangulaires sur les cotés.

- Les discrétisations ponctuelles et linéiques pour définir la densité de mailles

- L'ordre du maillage:
  - 1 er ordre
  - 2 ème ordre: on créé un nœud supplémentaire entre chaque deux nœuds, ce qui permet d'augmenter la précision de calcul

La figure 10 montre le maillage de la machine synchrone à aimants enterrés. Le maillage utilisé est un maillage mixte: automatique et extrusif.

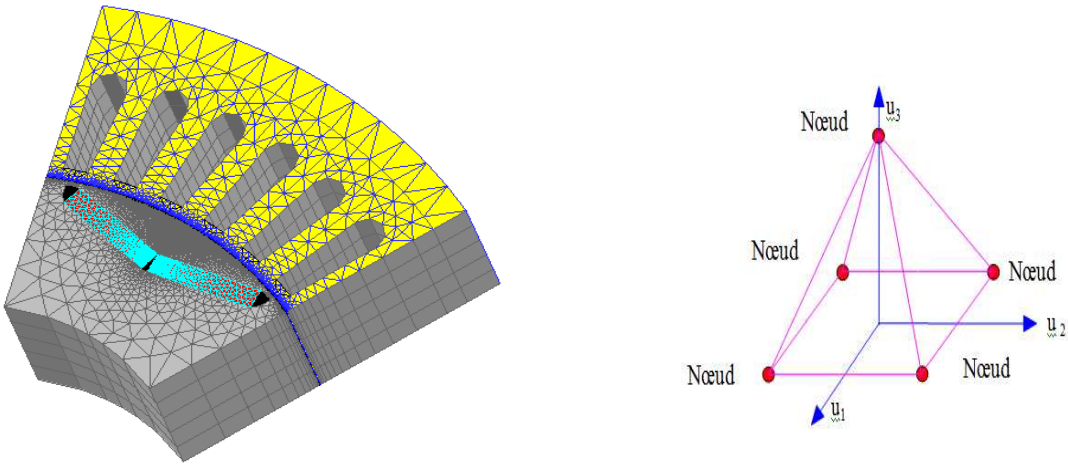


Figure 10 : Maillage de la machine synchrone à aimants enterrés

II.2.3. Propriétés physiques

Dans cette étape on définit:

- Le type d'application

Applications Flux	Le champ magnétique est créé par …		L'application permet de prendre en compte …
Magnéto Statique	Courants stationnaires (régime permanent)	aimants	
Magnétique Transitoire	Courants variables (régime variable / régime permanent)	aimants	Courants induits (courants de Foucault)
Magnéto Harmonique	Courant sinusoïdaux (régime permanent)	! Aimants interdits !	Effet de peau / effet de proximité

- Les conditions aux limites
  - Boîte infinie : Automatique
  - Plans de symétrie :
    - Champ magnétique tangent / Champ électrique Normal
    - Champ magnétique normal / Champ électrique Tangent
  - Plans de périodicité
    - Pair = Cyclique: V(T+u) = V(u)
    - Impair = Anticyclique: V(T+u) = -V(u)

- Les propriétés magnétiques des matériaux

Un matériau est caractérisé par un ensemble de propriétés physiques :

- Caractéristique B(H) : perméabilité (μ)
- Caractéristique J(E) : résistivité (ρ)
- Caractéristique D(E) : permittivité (ε)
- Caractéristique Thermique : conductivité k(T)
  capacité calorifique ρCp(T)
- Masse volumique : masse volumique (ρV)

Différents modèles sont disponibles

- Constant
- Linéaire
- Exponentiel

- Les régions : une région se définit par un groupe d'entités géométriques de même type (volumes, faces, lignes et points) qui ont les mêmes caractéristiques physiques.
- Les ensembles mécaniques permettant de définir la partie mobile et la partie fixe

La figure 11 montre la démarche générale à suivre pour définir les propriétés physiques.

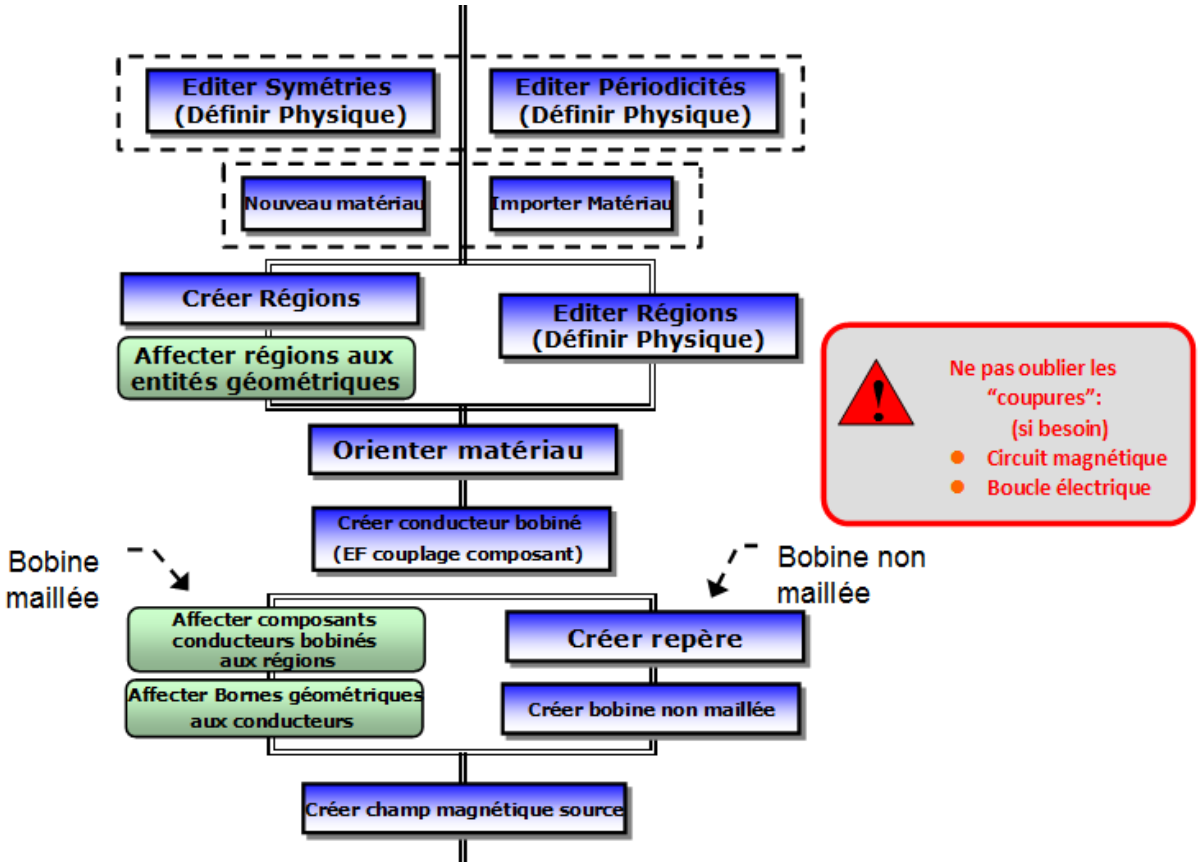


Figure 11 : Démarche générale pour la définition de la physique

II2.4. Résolution

De manière simplifiée, la résolution suit le processus suivant :

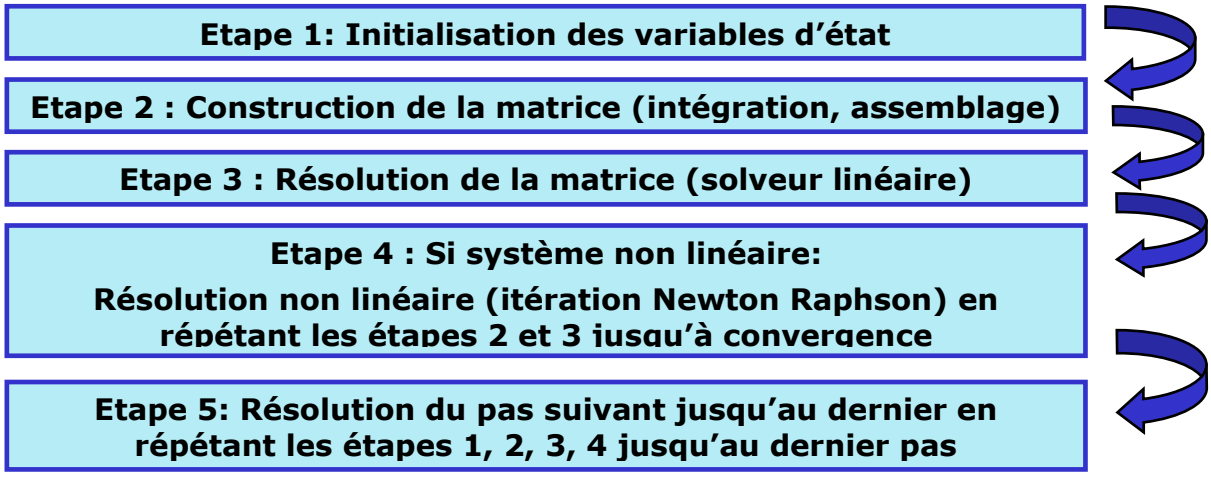


Figure 12 : Etapes de résolution d'un projet dans Flux

Le processus de résolution est réalisé automatiquement dans Flux. Pour les applications statiques (Magnéto Statique, Electro Statique,…), la résolution est complètement automatique. Par contre, dans des applications transitoires (Magnétique transitoire,…), il faut obligatoirement définir la loi de variation du temps (créer un scénario de résolution).

Il ya deux types d'études: temporelle et paramétrique.

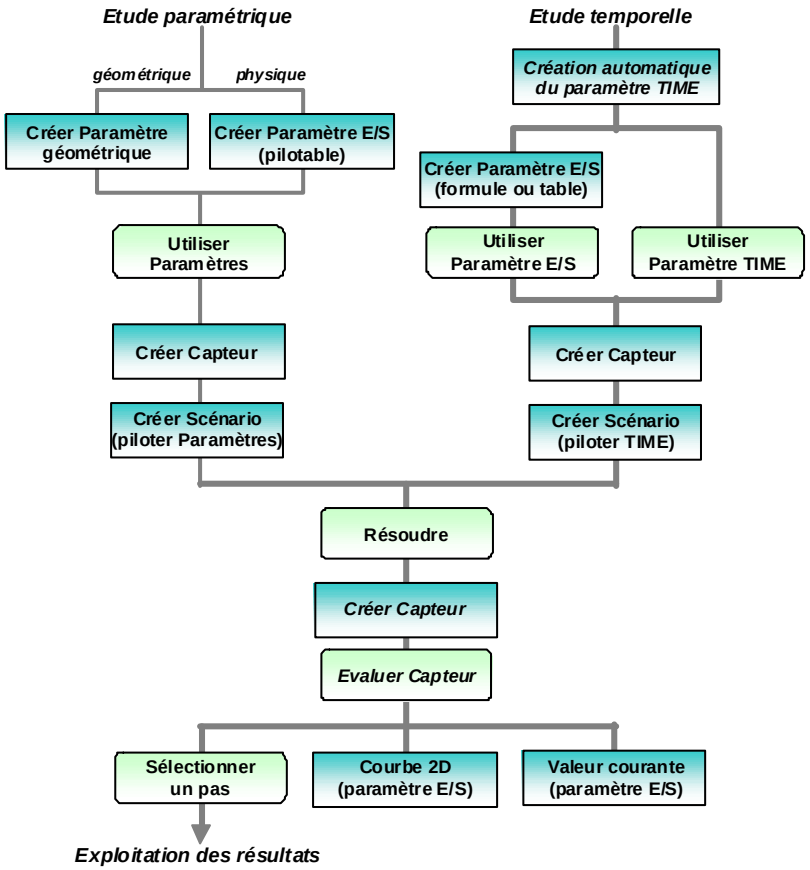


Figure 13 : Démarche générale de la résolution

II.2.5. Exploitation des résultats

Une fois le projet résolu, on passe à l'exploitation des résultats qui dépend du type de simulation réalisée:

- Calcul à vide :
  - Détermination du couple de détente
  - Détermination de l'induction dans l'entrefer et analyse FFT
  - Détermination des formes de l'induction dans les différentes parties du stator
  - Détermination du flux dans les différentes phases et analyse FFT
  - Détermination de la FEM

- Calcul en charge :
  - Calcul du couple à fonctionnement nominal
  - Calcul des pertes :
    - Pertes fer
    - Pertes dans les aimants permanents
  - Calcul du rendement

- Calcul en court circuit

- Calcul des inductances Ld et Lq

La figure 14 montre la démarche générale à suivre dans le post traitement pour tous les cas de simulations (à vide, en charge…).

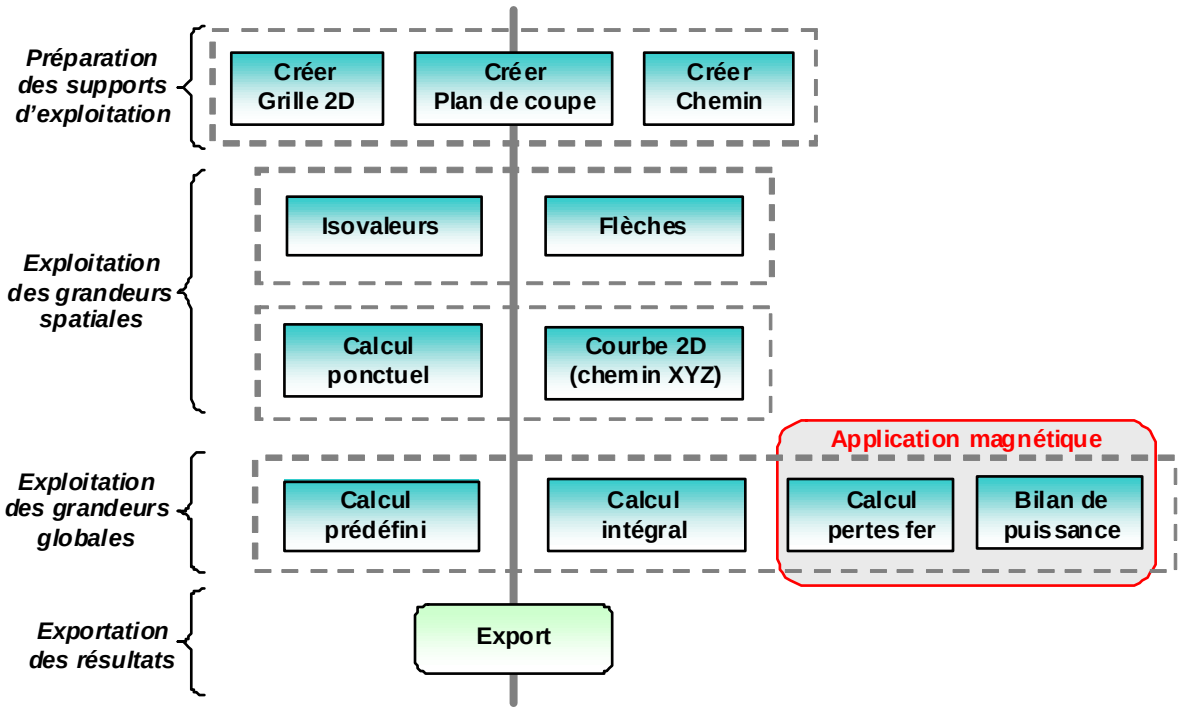


Figure 14 : Démarche générale de l'exploitation

- **Essai à vide**

Dans le cas de l'essai à vide, les grandeurs calculées sont: le flux, la force électromotrice (FEM), la FFT de la FEM, les pertes fer dans le stator, le couple de détente et les inductions magnétiques dans différentes parties de la machine.

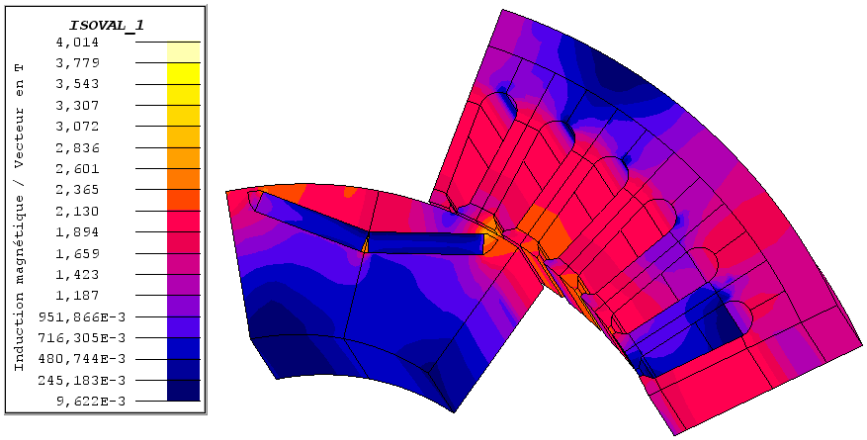


Figure 15: Dégradés de l'induction magnétique pour une position donnée

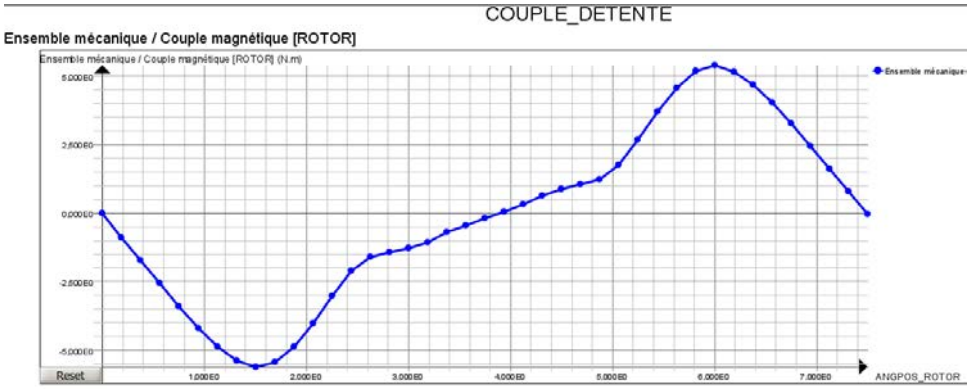


Figure 16: Evolution du couple de détente en fonction de la position

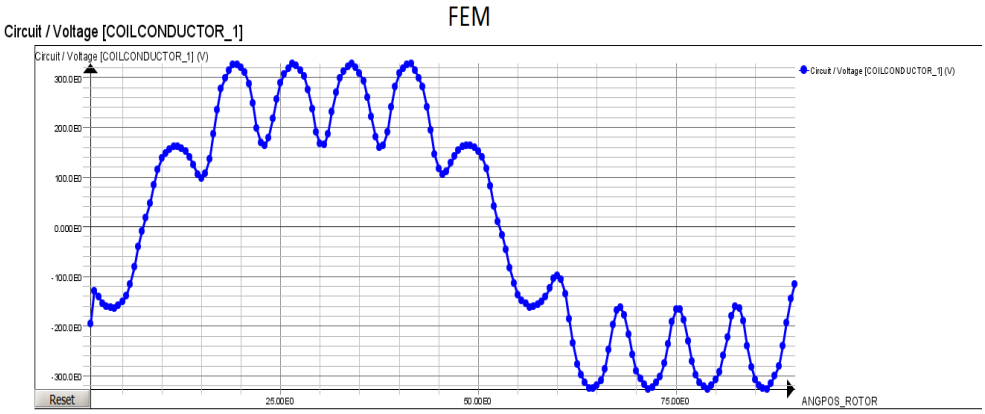


Figure 17: Evolution de la FEM en fonction de la position

La première chose à faire après la résolution est de tracer les dégradés d'induction. Cela nous donne une idée sur les valeurs et la répartition de l'induction magnétique dans la machine et des niveaux de saturation..

- **Essai en charge**

Dans le cas de l'essai en charge, on alimente la machine par une source de courant sinusoïdale (on prend le 1er harmonique). On calcule l'évolution du couple électromagnétique, les inductions magnétiques, les pertes fer (modèle de Bertotti), les pertes dans les aimants permanents, les pertes Joule et on peut alors faire le bilan de puissance pour calculer le rendement de la machine.

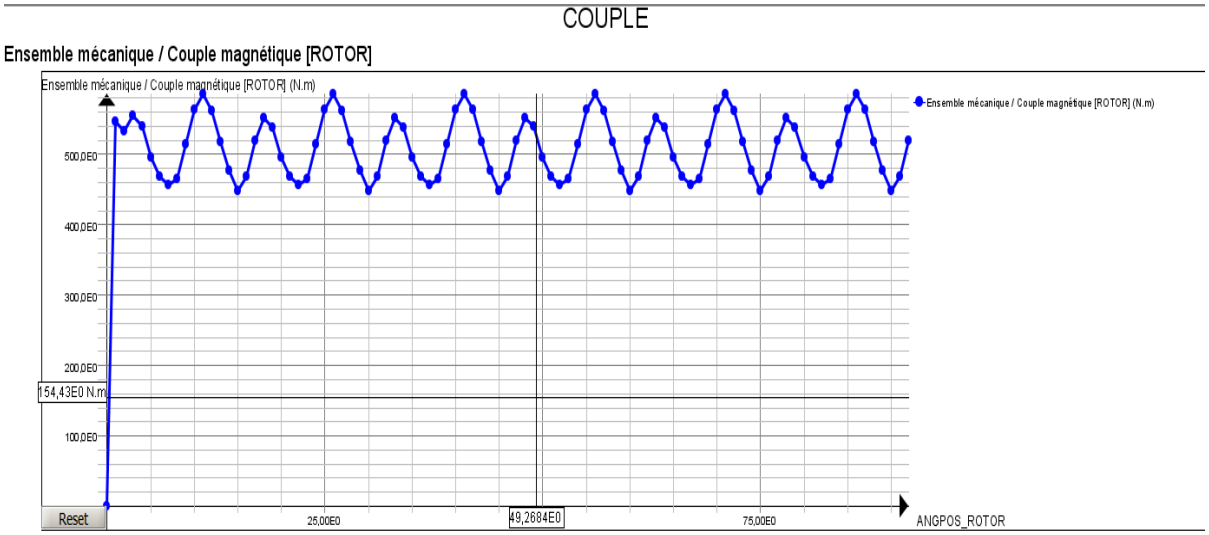


Figure 18: Evolution du couple électromagnétique en fonction de la position

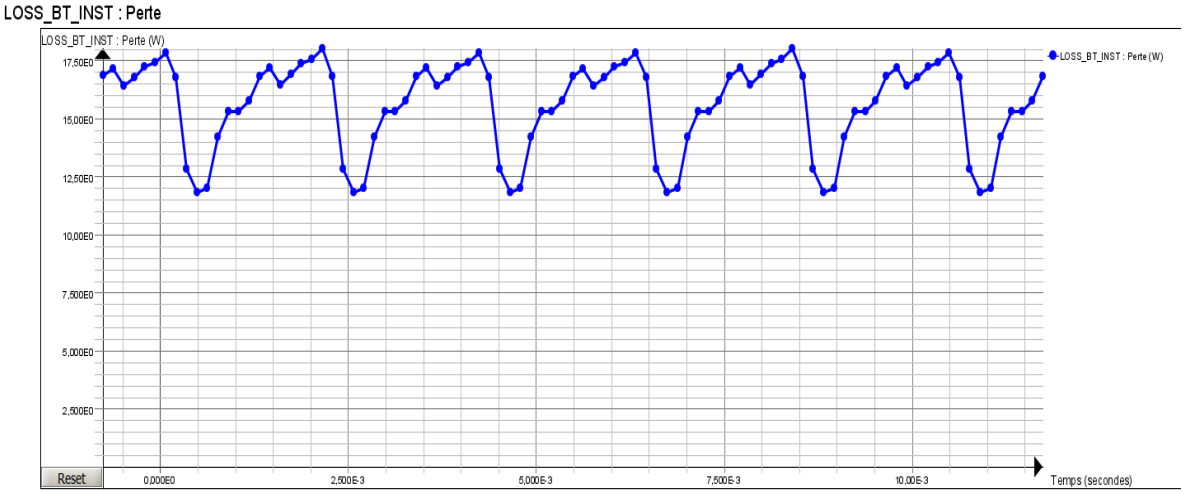


Figure 19: Pertes fer dans le stator en fonction de la position

Tous calculs faits (les pertes, la puissance absorbée, la puissance mécanique…), le rendement final de la machine pour ce point de fonctionnement est de :

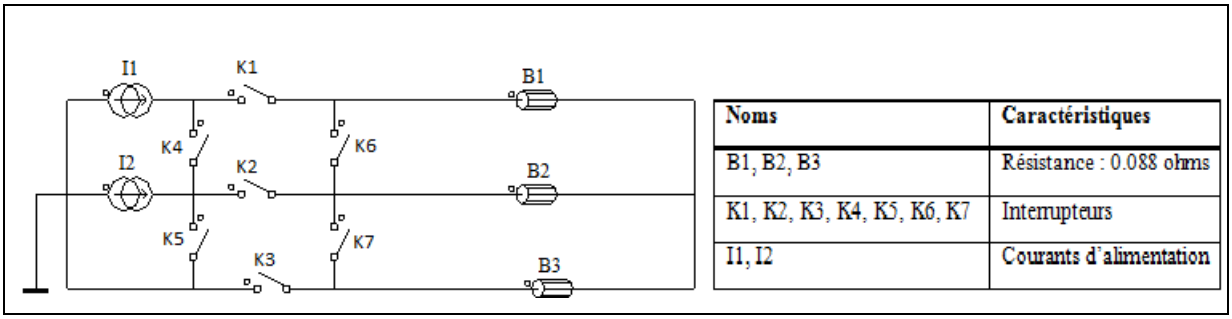
Quantités	Valeurs
Puissance absorbée	56247 w
Puissance mécanique	48915 w
Perte fer	134 w
Rendement	86.72%

- **Essai en court circuit**

L'essai en court circuit permet de voir le comportement de la machine dans le cas d'un appel de courant fort. Pour réaliser cet essai dans Flux, nous avons utilisé des interrupteurs qui permettent de contrôler le passage de courant dans chaque phase. Au début, on alimente la machine avec des courants sinusoïdaux pendant une période mécanique. Suite à cela, on ferme les interrupteurs de manière à créer un court circuit triphasé. Cela va générer un appel de courant très fort au moment de l'application du court circuit. On trace ensuite l'évolution du courant et le couple électromagnétique en fonction du temps (cf. figure 20).

Au moment de l'application du court circuit, un régime transitoire apparait (un fort appel du courant) et au bout d'un certain nombre de périodes, le courant tend à se stabiliser.

- **Le circuit :**



Noms	Caractéristiques
B1, B2, B3	Résistance : 0.088 ohms
K1, K2, K3, K4, K5, K6, K7	Interrupteurs
I1, I2	Courants d'alimentation

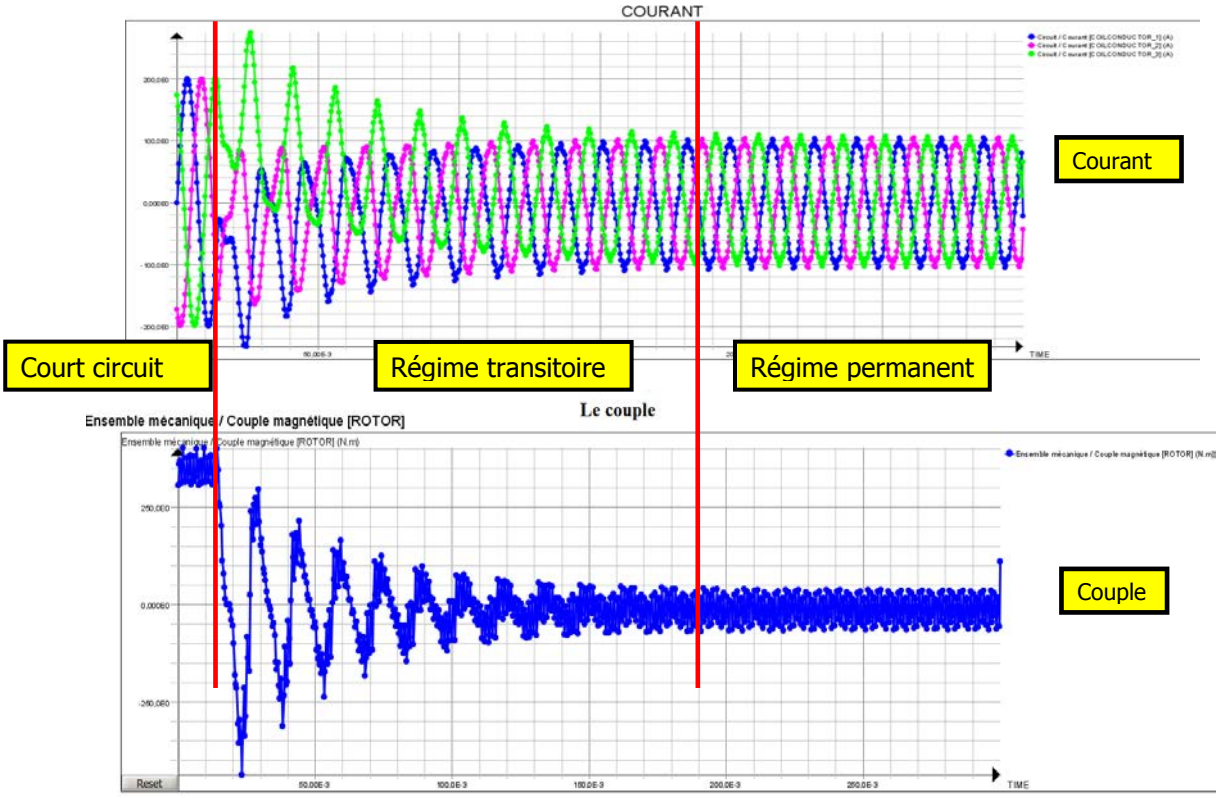


Figure 20 : Evolution du courant et du couple électromagnétique en fonction du temps

- **Calcul des inductances**

Cette dernière simulation consiste à calculer les inductances dans les axes d et q. Ces inductances permettent de calculer le couple électromagnétique et la commande de la machine.

Les phases sont alimentées par :

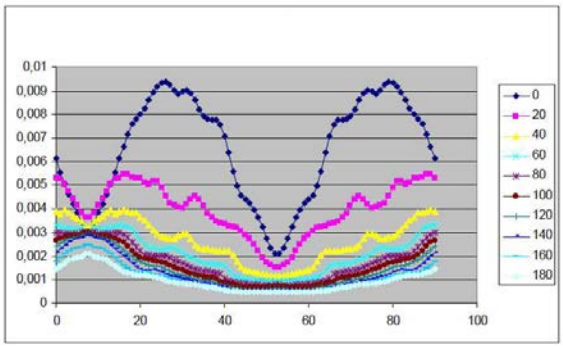
$$I_1 = 10 * \cos(\omega t)$$

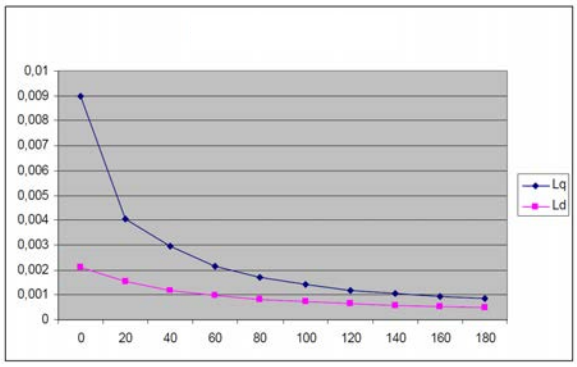
$$I_2 = 10 * \cos(\omega t - \frac{2\pi}{3})$$

A l'instant t=0:

$$I_1 = 10A$$
$$I_2 = -5A$$

$$L = \frac{(\Phi_2 - \Phi_1)}{(I_2 - I_1)}$$





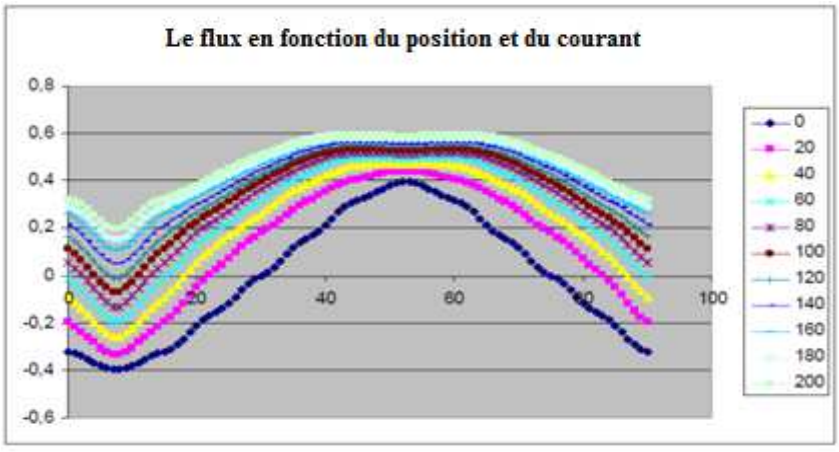


Figure 21: Calcul des inductances Ld et Lq

II.3. Conclusion

Dans cette partie, nous nous sommes intéressés à la modélisation d'une machine synchrone à aimants enterrés dans Flux 3D, plus particulièrement aux différentes étapes pour la construction de la machine. Une fois la machine construite, on a réalisé des essais à savoir : essai à vide, en charge, en court circuit et calcul des inductances dans les axes d et q. Le but de cette partie était d'apprendre les différentes étapes pour la construction d'un projet dans Flux 3D. Cela m'a permis de mieux appréhender la troisième partie qui consiste à automatiser toutes ces étapes.

Partie III
Overlay 3D

III.1. Introduction

Cette partie commence par des généralités sur l'Overlay 2D afin d'introduire le concept de l'Overlay 3D dont nous détaillerons ensuite les différentes modifications effectuées sur l'Overlay 2D à savoir : la boite infinie, la boite air, la symétrie, le maillage et le bobinage pour construire l'Overlay 3D. Il est à noter que nous entendons par un Overlay 2D/3D une interface qui permet de réaliser et de construire une machine électrique à partir de peu de paramètres géometriques.

III.2. Overlay 2D

L'Overlay 2D est un outil dédié pour la création des machines électriques (asynchrone, synchrone, à reluctance variable, à courant continu…).

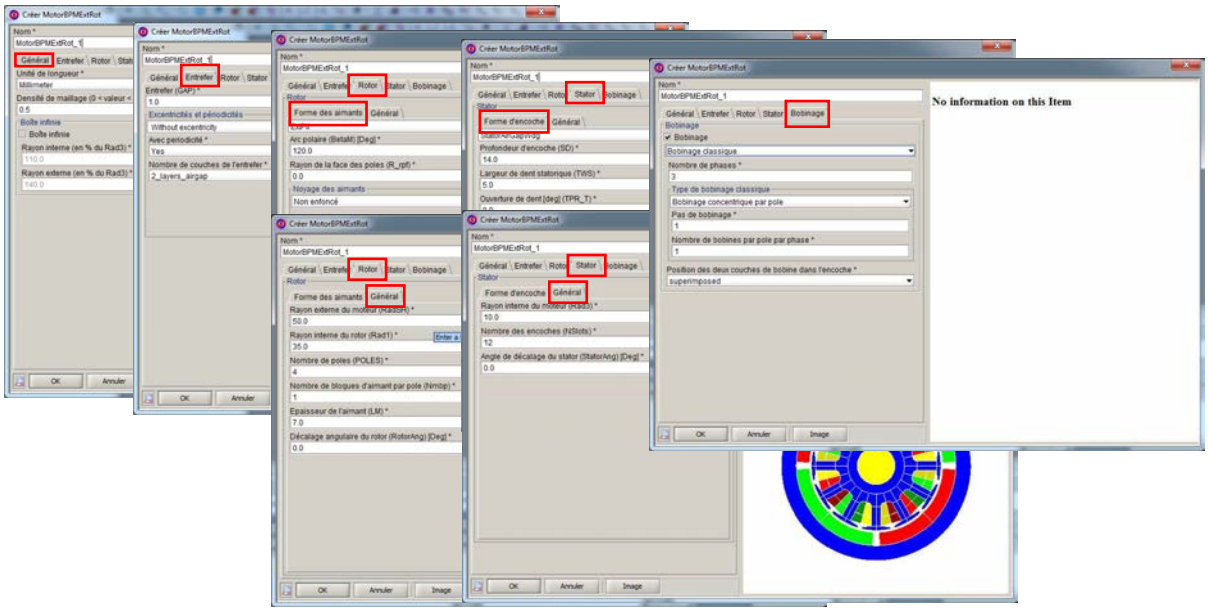


Figure 22 : Structure d'un Overlay 2D

L'interface de l'Overlay 2D est constituée de cinq onglets essentiels :

- Général : on renseigne l'unité de longueur de la boite infinie.
- Entrefer : l'utilisateur indique l'épaisseur de l'entrefer, la périodicité, l'excentricité et le nombre de couches dans l'entrefer.
- Rotor : deux parties sont à renseigner
    - La forme des aimants : dimensions des aimants, la position des aimants au niveau du rotor…
    - Des informations générales comme le nombre de Pôle, le rayon interne de rotor.., etc.
- Stator : deux parties deux parties à renseigner
    - La forme d'encoche : encoche fermée, ouverte, semi fermée…
    - Général : le nombre d'encoche, le rayon externe du stator…
- Bobinage : il s'agit de donner des informations sur les bobines maillées.

III.3. Overlay 3D

Pour créer l'Overlay 3D, on est parti de l'Overlay 2D, ce qui permet d'éviter de refaire des tâches telles que la construction de la géométrie en 2D.

Dans la première partie de ce projet, nous nous sommes intéressés aux machines synchrones à aimants permanents (aimants montés en surface, aimants insérés, aimants enterrés…). Dans l'Overlay 3D nous conservons la même architecture que l'Overlay 2D avec quelques modifications pour la prise en compte de la boite infinie en 3D, la troisième dimension et le bobinage, l'architecture est la suivante :

- Général
- Entrefer
- Rotor
- Stator
- Extrusion suivant l'axe z (z dimensions)
- Bobinage

Dans un premier temps nous cherchons à spécifier les étapes permettant une définition métier de l'Overlay 3D. Les trois étapes essentielles sont :

- Mise en place du diagramme UML
- Génération de l'interface homme machine (IHM)
- Implémentation dans le langage de commande Flux des commandes

III.3.1. Mise en place du diagramme UML

Dans le monde du génie logiciel, le formalisme UML (Unified Modeling Language) est reconnu et très utilisé. Notamment grâce au diagramme de classes, il permet de définir un modèle de données structuré, maintenable et évolutif. C'est donc sur la base de ce formalisme que l'on peut décrire et visualiser un modèle de données dans FluxBuilder qui est un logiciel permettant de décrire le modèle d'un logiciel et de le sauvegarder sous forme de fichier.

Dans FluxBuilder, une entité décrit un objet ou une partie d'un objet que l'utilisateur pourra manipuler dans le logiciel de simulation. Elle contient un nom et des attributs. Ces attributs décrivent la structure des paramètres de l'objet. Sur chaque objet on peut également définir des méthodes qui modélisent des comportements de l'objet et qui sont des commandes que l'on peut lui associer.



Figure 23 : L'entité boite infinie et ses attributs

III.3.2. Génération de l'interface homme machine (IHM)

Une fois le modèle décrit, FluxBuilder propose également d'organiser l'IHM de son application. La description du modèle de données et des commandes n'est pas suffisante, il faut maintenant décrire comment ces boîtes seront accessibles dans l'interface du logiciel. Le logiciel propose un certain nombre de composants qui constitueront l'interface de son logiciel et dans lesquels il pourra placer son modèle de données.

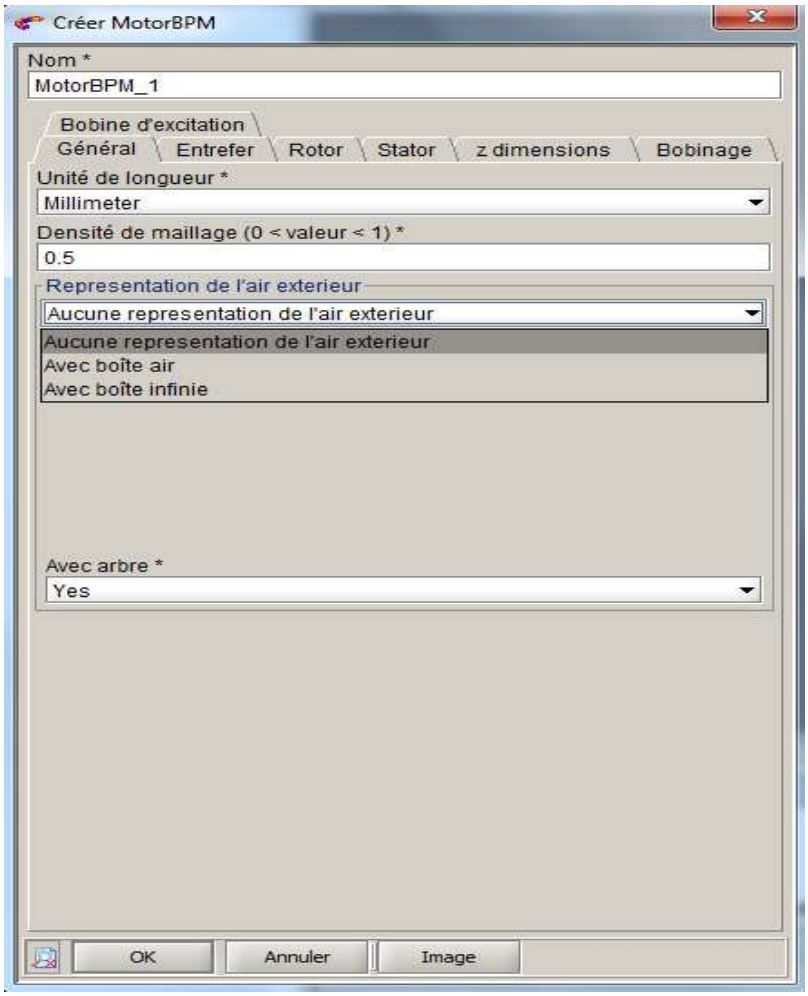


Figure 24 : Exemple d'interface homme machine (IHM)

III.3.3. Implémentation dans le langage de commande Flux

Une fois le modèle décrit (aucune ligne de code n'a encore été écrite), il est interprété par la machine virtuelle qui le traduit dans le logiciel de simulation. Nous devons également prévoir une implémentation de leur modèle. Cette implémentation sera chargée dans Flux en même temps que le modèle métier. Elle contiendra la description de l'enchainement des commandes Flux à réaliser à partir des valeurs des objets métiers créés pour automatiser la construction du Bobinage. Une implémentation des objets métiers dans le langage de commande de Flux, le Python.
En effet, toute action dans le logiciel est reproductible à l'aide d'une ligne de script.

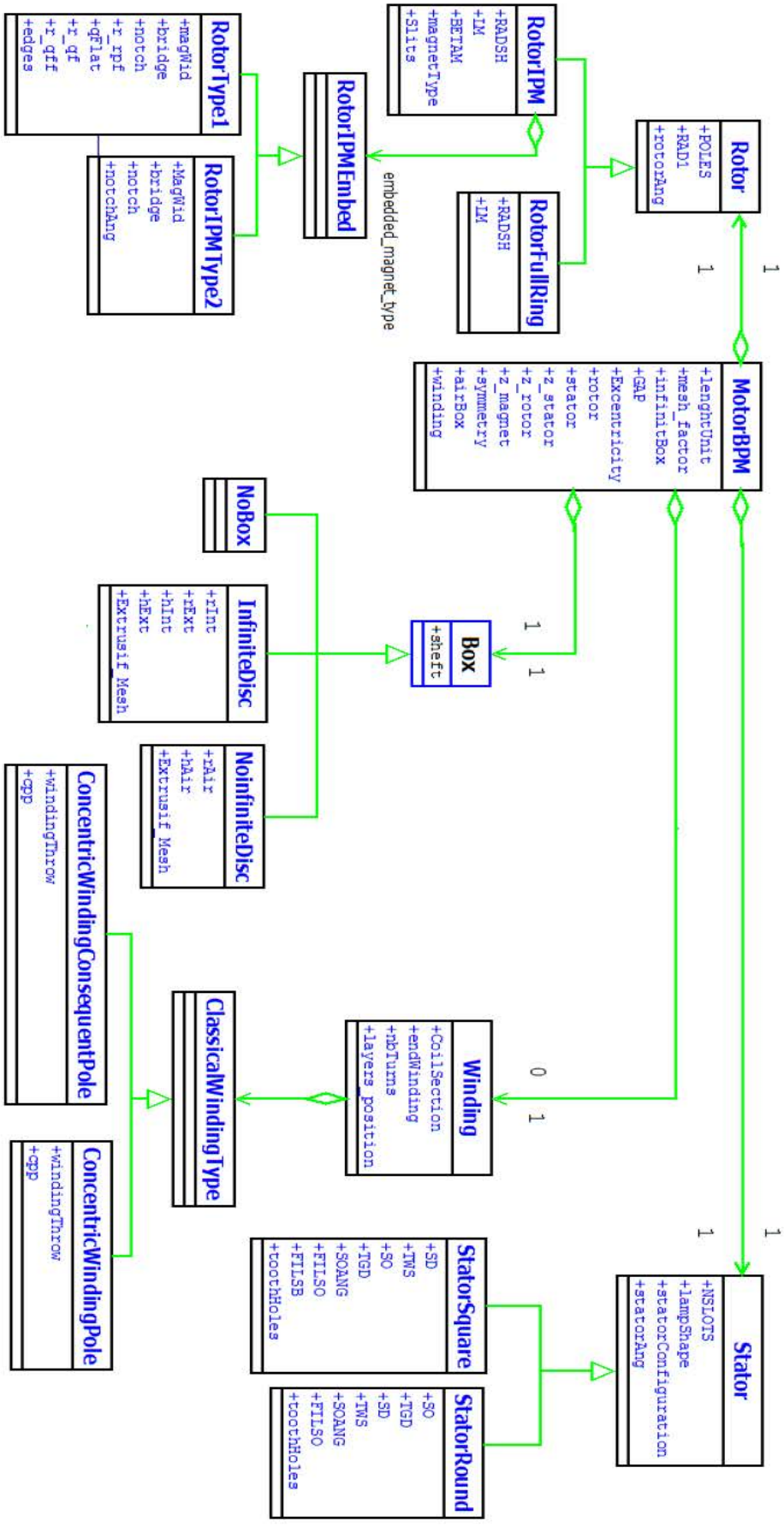


Figure 25 : Visualisation du modèle UML de la machine synchrone à aimants

III.3.4. Construction de la machine en 3D

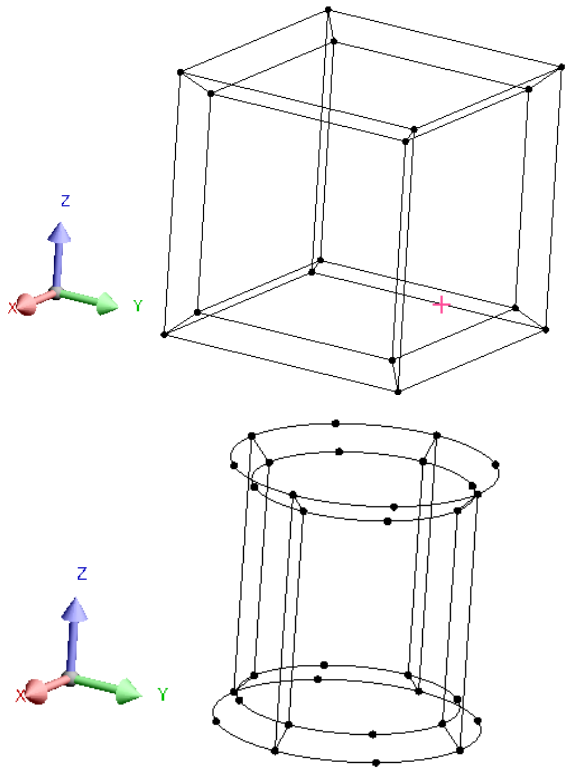
Pour créer la machine en 3D nous avons créé des extrusions selon l'axe 'Z' pour une hauteur donnée pour les trois parties : stator, aimants et rotor. La hauteur de l'extrusion est choisie par l'utilisateur. Cette extrusion permet de créer des volumes en conservant le même nom que les régions surfaciques (plus de détails sur les notions de régions sont fournis en annexe A). i.e. l'utilisateur ne va pas créer des régions volumiques. Exemple d'un script python pour une extrusion en 3D selon un axe :

```
TransfTranslationVector(name='TRANS_Z',
                        coordSys=CoordSys['XYZ1'],
                        vector=['0',
                                '0',
                                'LF/2'])
```

Le script contient: le nom de la transformation, le repère, la hauteur et l'axe de la transformation.

III.3.5. Boîte infinie

Le domaine d'étude dans Flux2D/3D est défini comme étant la zone modélisée dans laquelle seront résolues les équations. Elle est délimitée par des frontières sur lesquelles seront appliquées les conditions aux limites. Nous trouvons différents types de frontières telles que la limitation du domaine d'étude par boite infinie. Les différents types de boites infinies que l'on peut utiliser dans le domaine d'étude 3D et les caractéristiques utiles pour leurs descriptions sont données dans la figure ci-dessous :



Parallélépipède:

- Centré en (0,0,0) dans le système de coordonnées mondial
- Comprend 16 points, 32 lignes
- Dimensions définies par l'utilisateur

Cylindre:

- Le long de l'axe X, Y ou Z
- Centré en (0,0,0) dans le système de coordonnées mondial
- Comprend 32 points, 32 lignes
- Dimensions définies par l'utilisateur

Figure 26 : Types de boite infinie pour un domaine d'étude 3D

Dans le projet Overlay 3D le type de la boite infinie utilisée est cylindrique selon l'axe Z. Ce choix se justifie par la forme des machines électriques, généralement cylindrique.

La première étape avant d'intégrer la boite infinie dans l'Overlay 3D, est de créer une entité "infiniteBox" dans XBuilder. Une fois l'entité créée, on génère dans Flux le script python qui permet la création de la boite infinie et on le greffe dans le script principal de la machine synchrone à aimants.

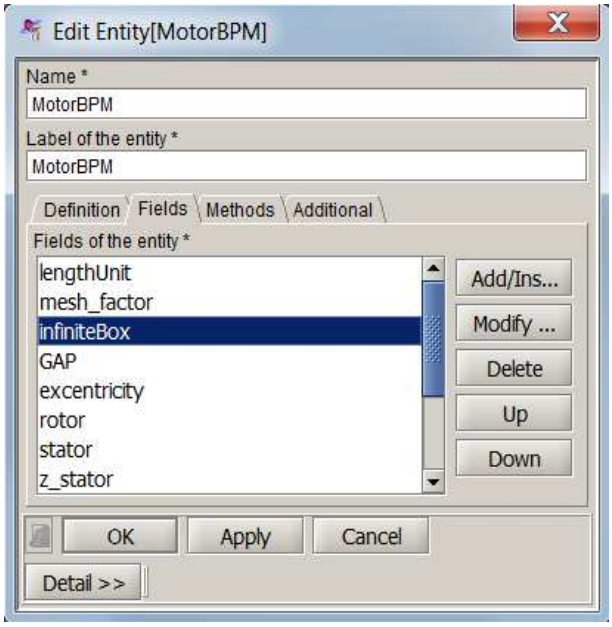


Figure 27 : Création de l'entité "infiniteBox" dans Xbuilder

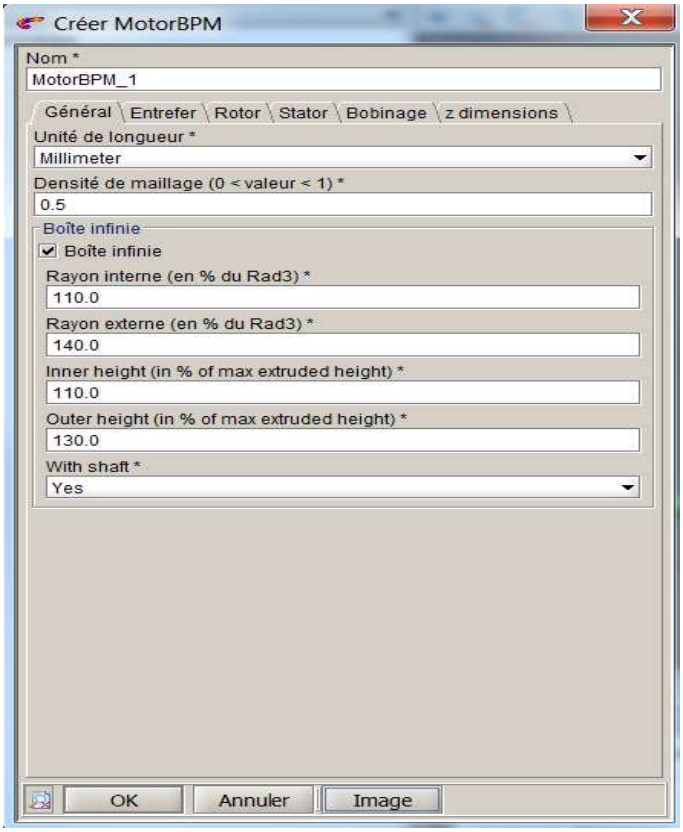


Figure 28 : Définition de la Boîte infinie dans l'interface de l'Overlay 3D

Le script python pour créer une boîte infinie est le suivant :

```
LastInstance = InfiniteBoxCylinderZ (size= ['110',
                                            '140',
                                            '110',
                                            '140'])
```

Ce script contient :

- La direction de la boîte infinie
- Les dimensions de la boîte infinie

La figure 28 montre l'interface de l'Overlay 3D une fois l'entité créée et le script python est greffé dans le script principal.

III.3.6. Boîte air

Le deuxième type de frontière que l'on peut utiliser pour limiter le domaine d'étude est la boîte d'air. La différence par rapport à la boite infinie se situe au niveau de la façon avec laquelle on crée la boite. Les étapes à suivre pour créer la boîte d'air sont :

- Création des points
- Création des lignes
- Création des faces
- Création des transformations de type translation pour chaque partie de la machine. Les transformations sont utilisées pour créer le dispositif en 3D
- Création d'une autre transformation de type translation avec une hauteur différente de celle de la première transformation. Cette dernière est utilisée pour créer de l'air autour du dispositif de la partie haute de la machine (voir figure 29).

Pour chaque étape, Flux génère une commande python. Cette commande est utilisée dans le script principal de l'Overlay 3D. Toutes ces étapes seront intégrées dans le script principal de l'Overlay 3D.

Pour l'intégration de la boîte d'air dans l'Overlay 3D, nous suivons les mêmes étapes que la boîte infinie :

- Mise en place du diagramme UML
- Génération de l'interface homme machine (IHM)
- Implémentation dans le langage de commande Flux

Les figures 29, 30, et 31 montrent un récapitulatif des étapes à suivre pour créer la boîte d'air, création de l'entité "airBox" et la comparaison entre la boîte infinie et la boîte air successivement. Le choix de la boîte air se justifie par le fait que quand on crée la boîte infinie avec un mailleur automatique Flux peut rencontrer des difficultés à mailler le dispositif.

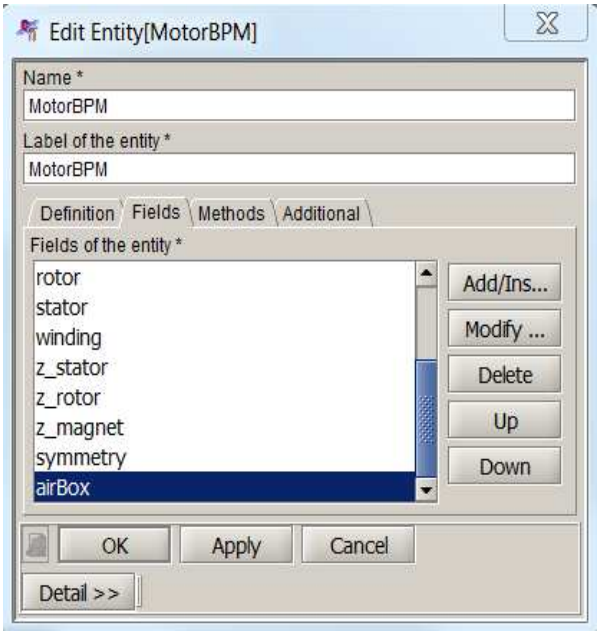
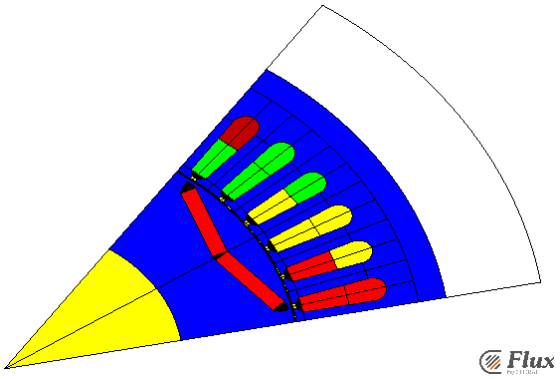
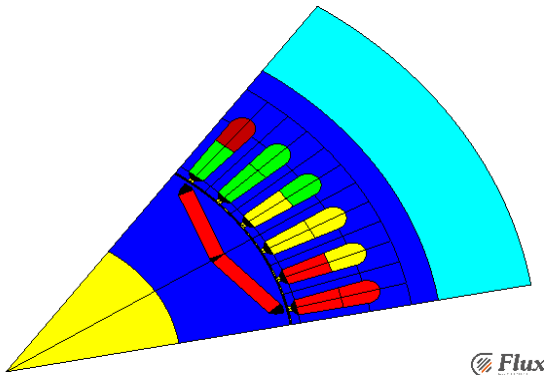


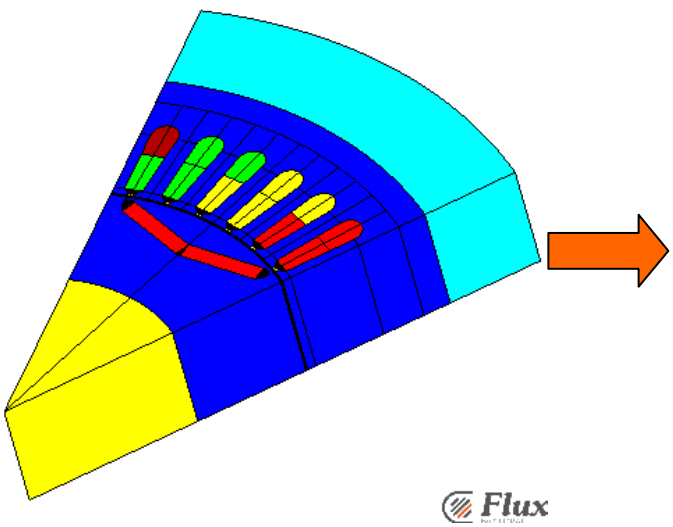
Figure 29 : Création de l'entité "airBox" dans XBuilder

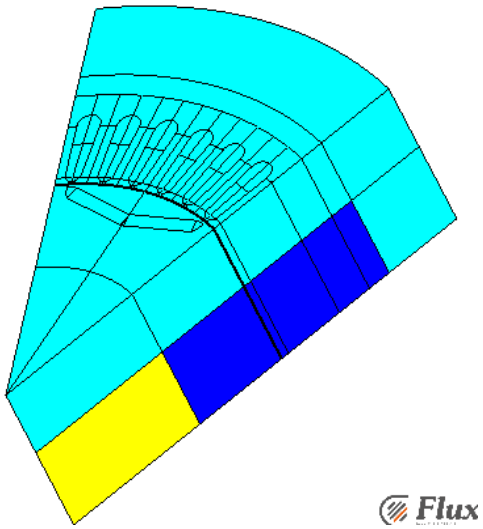


Création des points et lignes



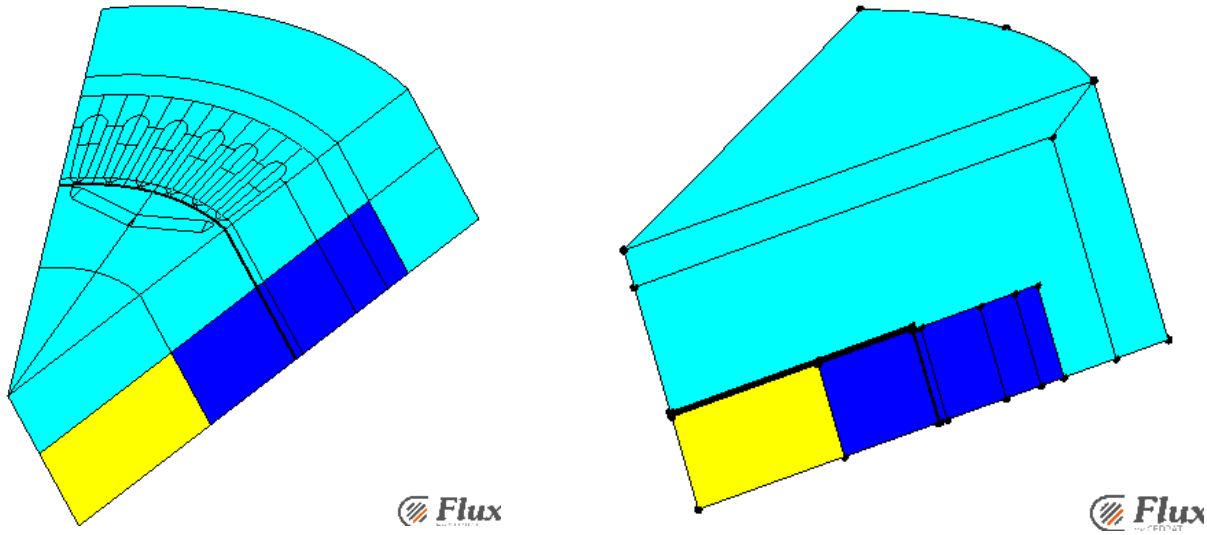
Création de la face





Création de volume d'air autour de la machine

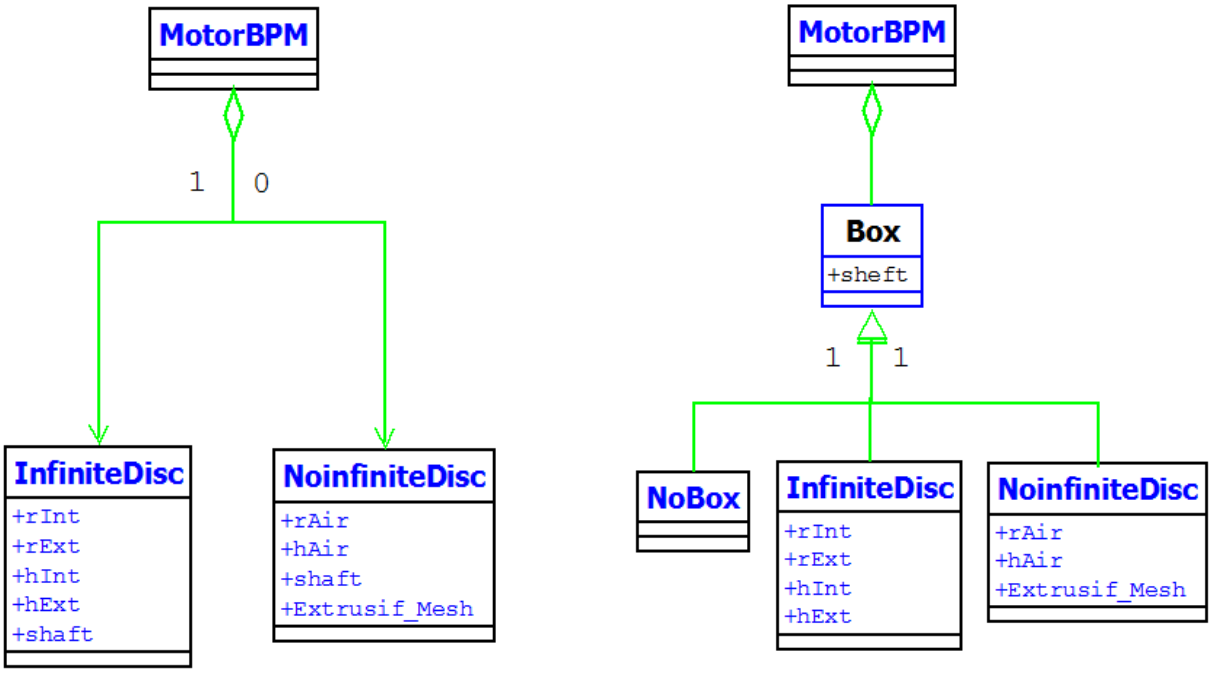
Figure 30 : Création de l'entité "airBox" dans Flux



Boîte air (créée avec la machine comme une région) **Boîte infinie (créée de façon automatique)**

Figure 31 : Comparaison entre boîte infinie et boîte air

Après réflexion, nous avons structuré d'une autre manière la création de la boite infinie et la boite air (voir figure 32), ce qui permet d'avoir une interface moins chargée. La deuxième proposition est retenue pour la suite du projet Overlay 3D.



1ère proposition **2ème proposition**

Figure 32 : Structuration de la création de la boite infinie et de la boite air

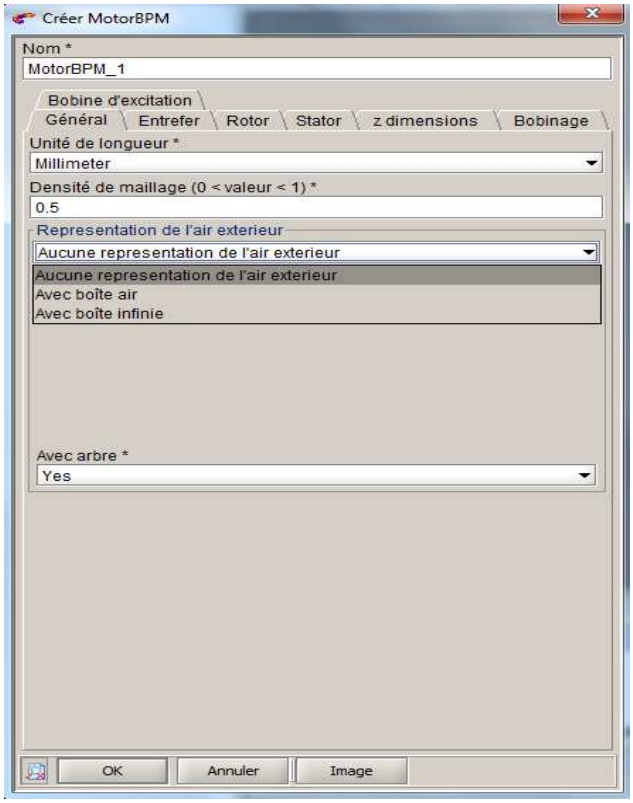
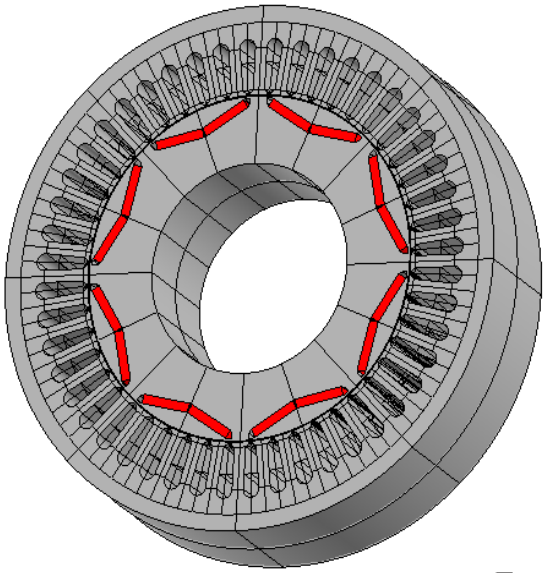
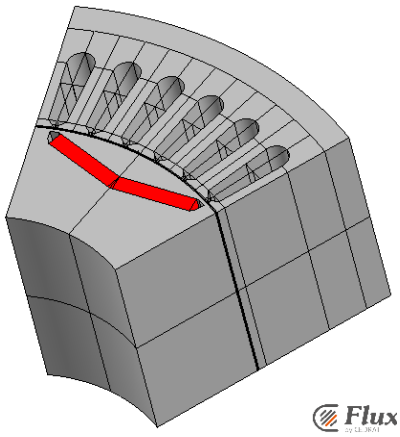


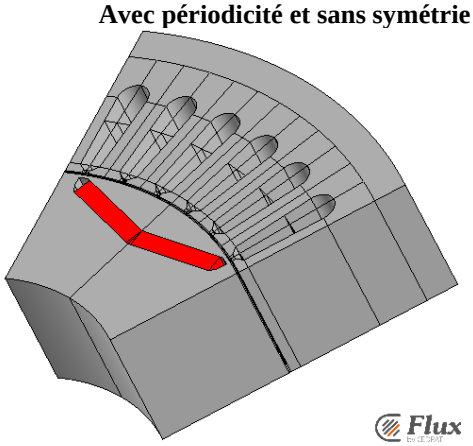
Figure 33 : Nouvelle interface de l'Overlay 3D



Sans périodicité et symétrie



Avec périodicité et sans symétrie



Avec périodicité et symétrie

Figure 34 : Prise en compte de la périodicité et de la symétrie

III.3.7. La symétrie et périodicité

Le deuxième type de frontières disponible dans Flux, est la réduction du domaine d'étude en présence de :

- Motifs répétitifs (périodicités)
- Plans de symétrie

Dans Flux il y a deux types de symétrie :

- Champ magnétique tangent / Champ électrique Normal
- Champ magnétique normal / Champ électrique Tangent

La prise en compte de la symétrie et de la périodicité permet de ne représenter qu'une portion du dispositif électromagnétique (1/2, 1/4,1/8…, etc) (voir figure 34). Cela veut dire que le nombre de nœuds du maillage est réduit. Par conséquent le temps de calcul est lui aussi réduit. Les mêmes démarches que pour la création de la boîte infinie et la boîte air sont utilisées pour intégrer la symétrie dans l'Overlay 3D ; c'est-à-dire :

- Mise en place du diagramme UML
- Génération de l'interface homme machine (IHM)
- Implémentation dans le langage de commande Flux

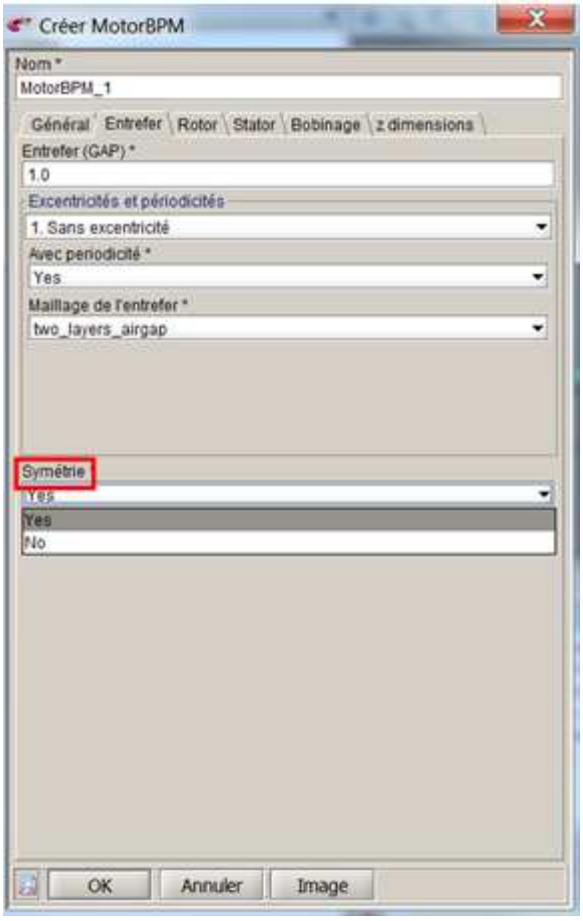


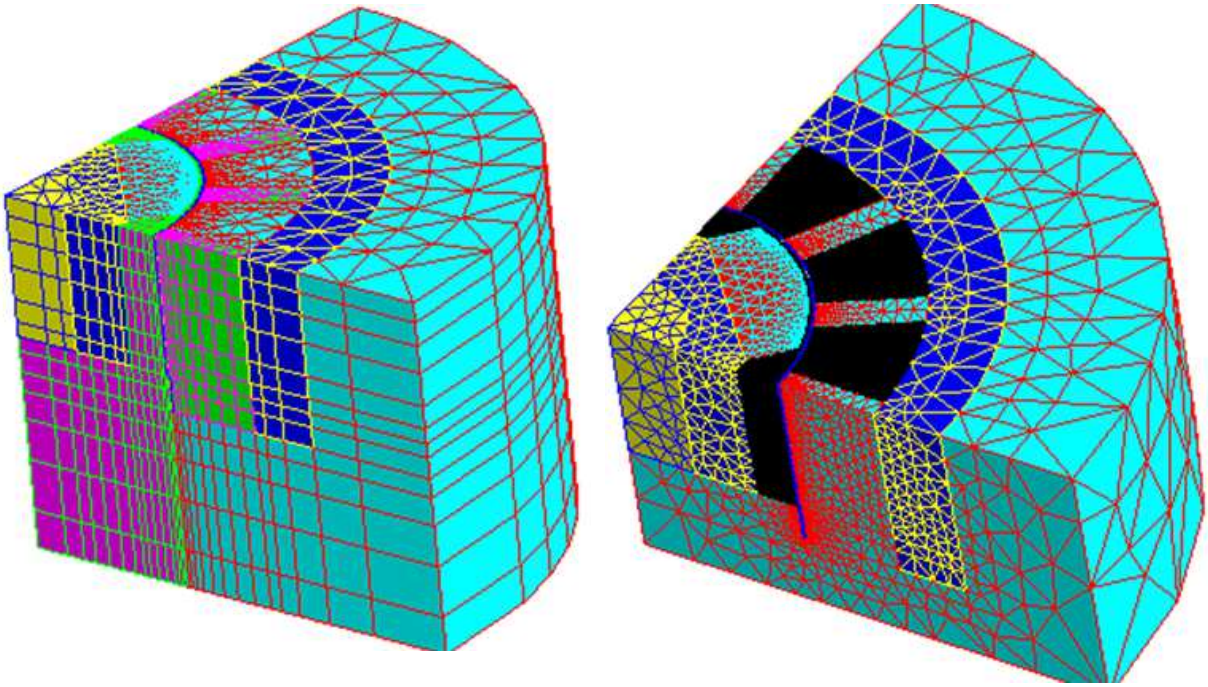
Figure 35 : Symétrie dans l'Overlay 3D

III.3.8. Le maillage

Dans Flux 3D, nous avons quatre types de mailleurs (voir la partie II):

Selon le type des conditions aux limites (boite infinie ou boite air), on choisit le type de mailleur. Ce choix de mailleur se justifie par la faisabilité du maillage. En effet, dans le cas de la boite infini on ne peut pas utiliser un maillage extrusif (problème de connexion des nœuds de la boite infinie avec le dispositif). Pour contourner ce problème nous avons proposé d'utiliser la boite air avec un maillage extrusif ou d'utiliser la boite infinie avec un maillage automatique, selon les besoins de l'utilisateur.

La figure 36 montre une comparaison entre les deux types de maillages : extrusif dans le cas de la boite air et automatique dans le cas de la boite infinie.



Maillage extrusive : boîte air **Maillage automatique : boîte infinie**

Figure 36 : Maillage de la boite infinie et la boite air dans l'Overlay 3D

Pour intégrer ces options de maillage dans l'Overlay 3D nous suivons la démarche décrite ci-dessus :

- Mise en place du diagramme UML
- Génération de l'interface homme machine (IHM)
- Implémentation dans le langage de commande Flux

Lors de test de l'Overlay 3D nous avons constaté des temps de calcul importants avec le maillage automatique. C'est pour cette raison que nous avons proposé de créer une région « AIR » entre le stator et la boite infinie avec le maillage automatique tout en gardant le maillage extrusif de notre dispositif (voir figure 39). Nous avons créé le cylindre de glissement pour pouvoir étudier les machines tournantes avec différentes positions du rotor, sans avoir à modifier la géométrie et le maillage.

La figure 38 montre les étapes à suivre pour créer la région « AIR »

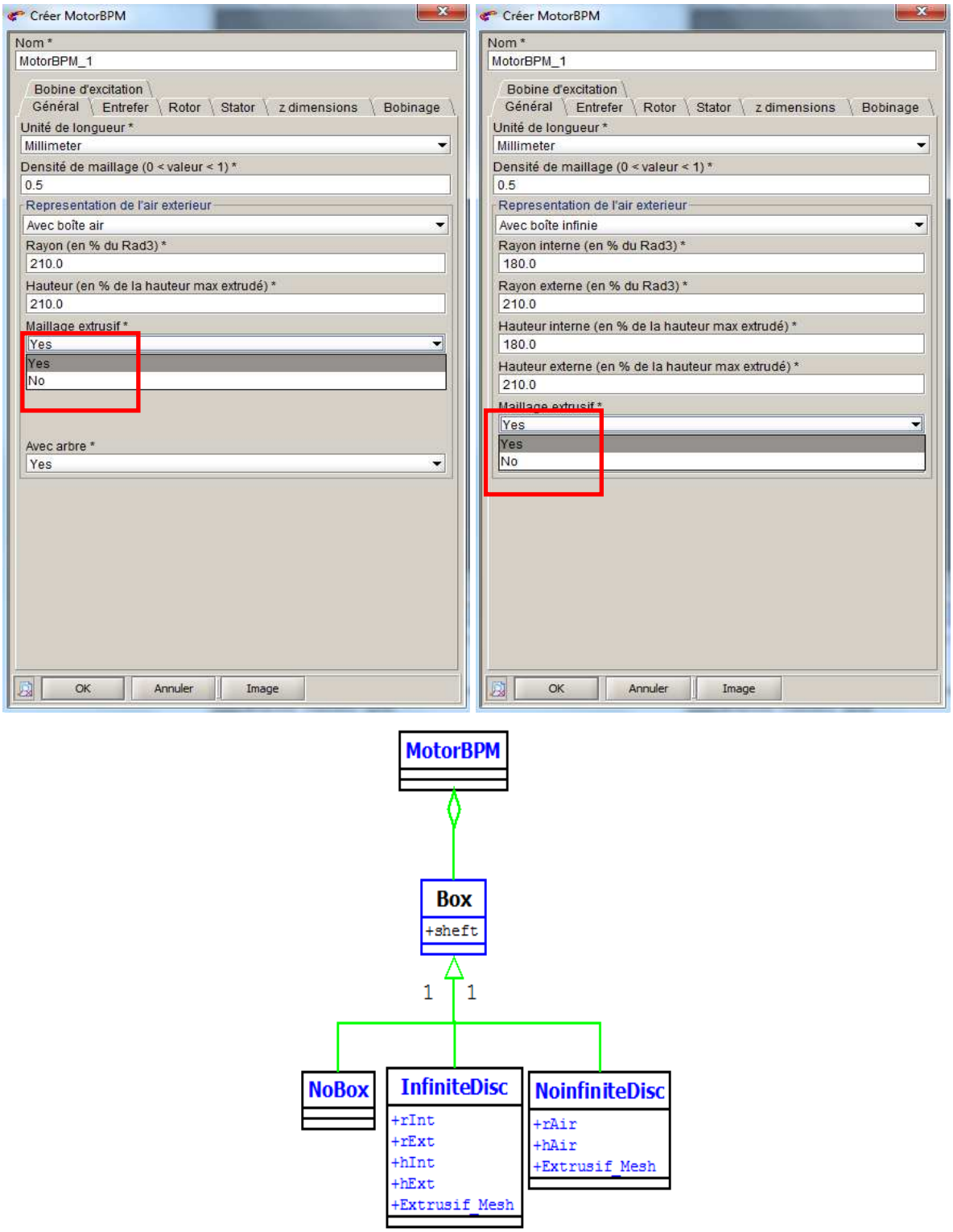
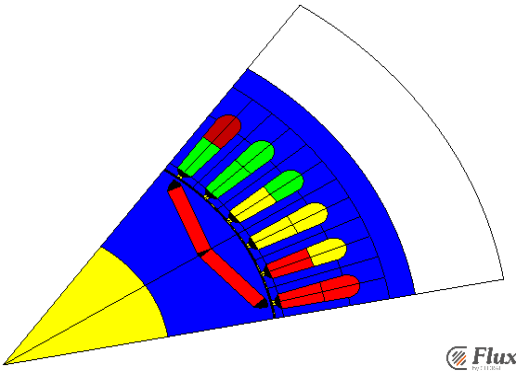
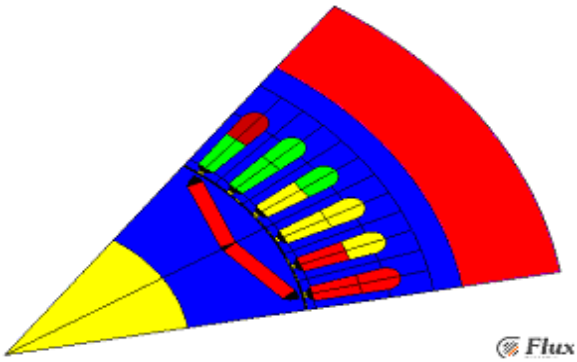


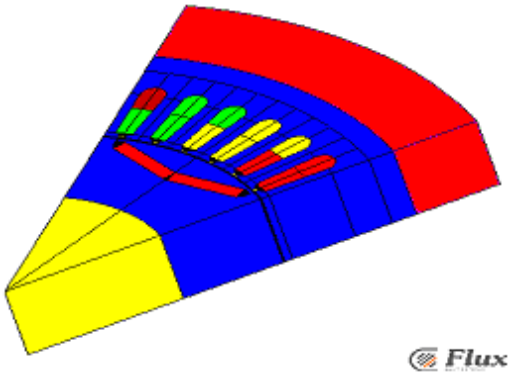
Figure 37 : Options du maillage dans l'Overlay 3D



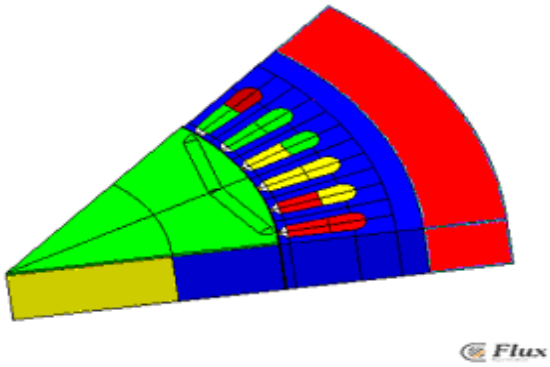
Création des points et lignes



Création de la face



Création du volume entre le stator et la boite infinie

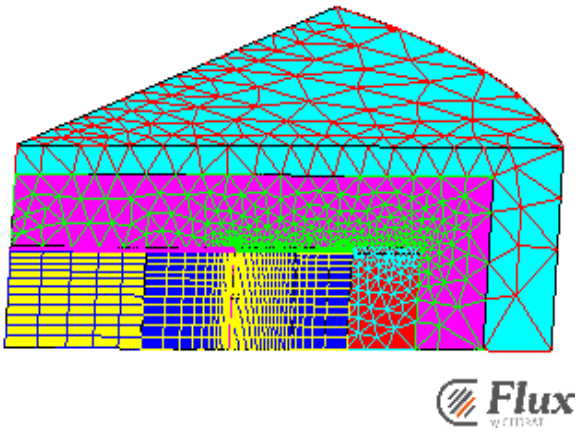


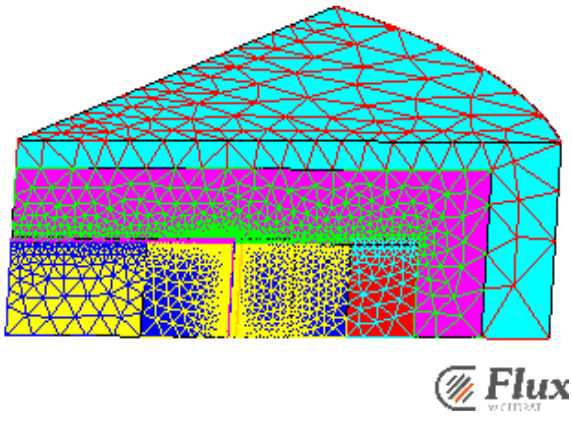
Création du cylindre de glissement

Figure 38: Création de la région « AIR » et « cylindre de glissement »

Quelque soit la méthode de maillage utilisée, automatique ou mixte, le maillage peut être plus ou moins rapide.

La figure 39 montre une comparaison entre le maillage automatique et le maillage mixte dans le cas de la boite infinie. On constate que le nombre de nœuds du maillage automatique est beaucoup plus important par rapport au maillage mixte par conséquent le temps de calcul est lui aussi réduit.





Nombre de noeuds : 85308
Nombre d'elements lineiques : 8564
Nombre d'elements surfaciques : 60145
Nombre d'elements volumiques : 202405
Ordre de maillage : 1er ordre

Nombre de noeuds : 176080
Nombre d'elements lineiques : 15198
Nombre d'elements surfaciques : 214672
Nombre d'elements volumiques : 999692
Ordre de maillage : 1er ordre

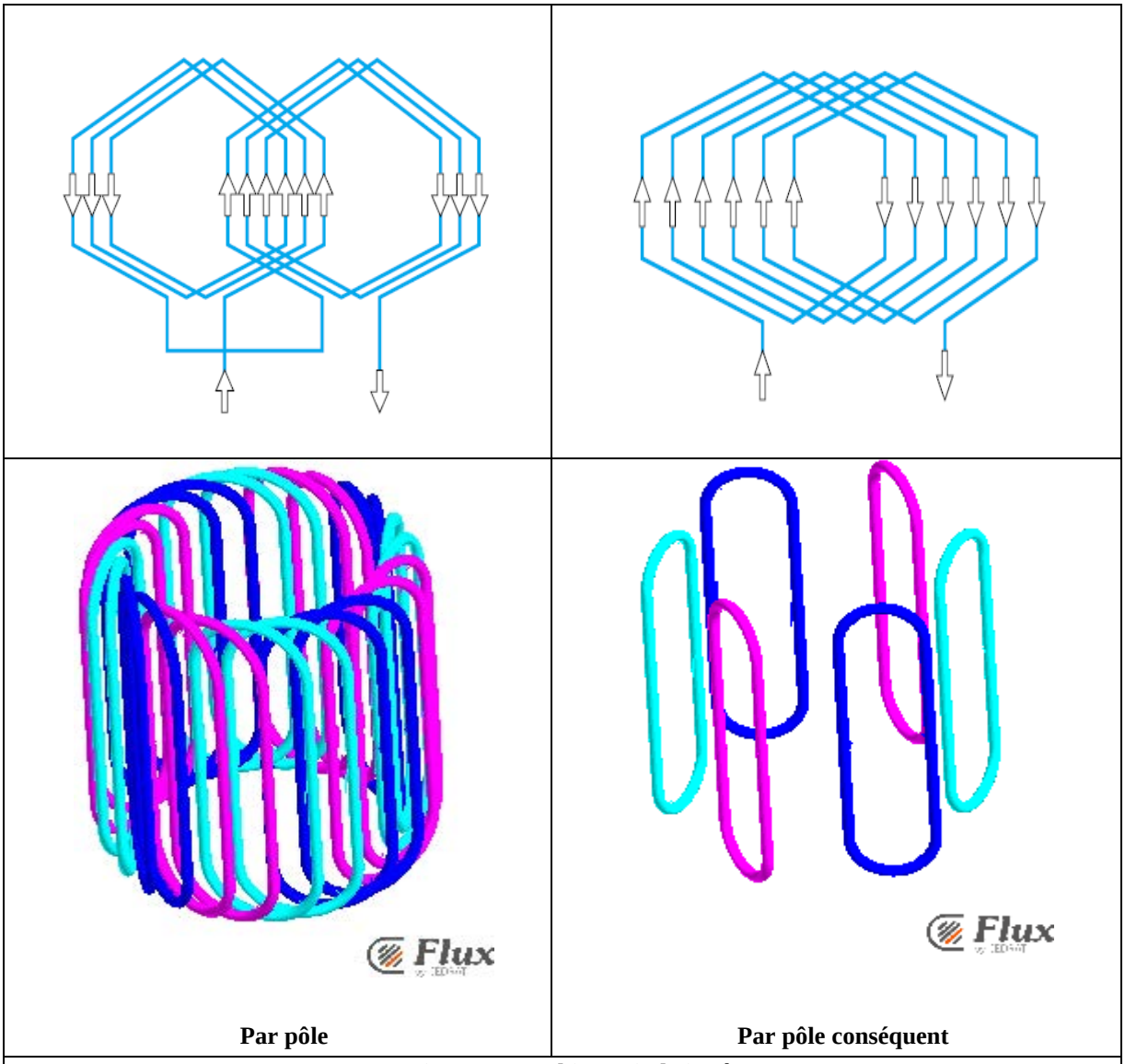
Figure 39 : Comparaison entre le maillage automatique et le maillage mixte

III.3.9. Le bobinage

Le dernier point traité dans ce rapport de stage est le bobinage statorique des machines électriques exécuté sous plusieurs formes. Dans l'Overlay 3D nous avons proposé deux types de bobinages : imbriqué et concentrique.
Les principaux paramètres caractérisant un enroulement sont :

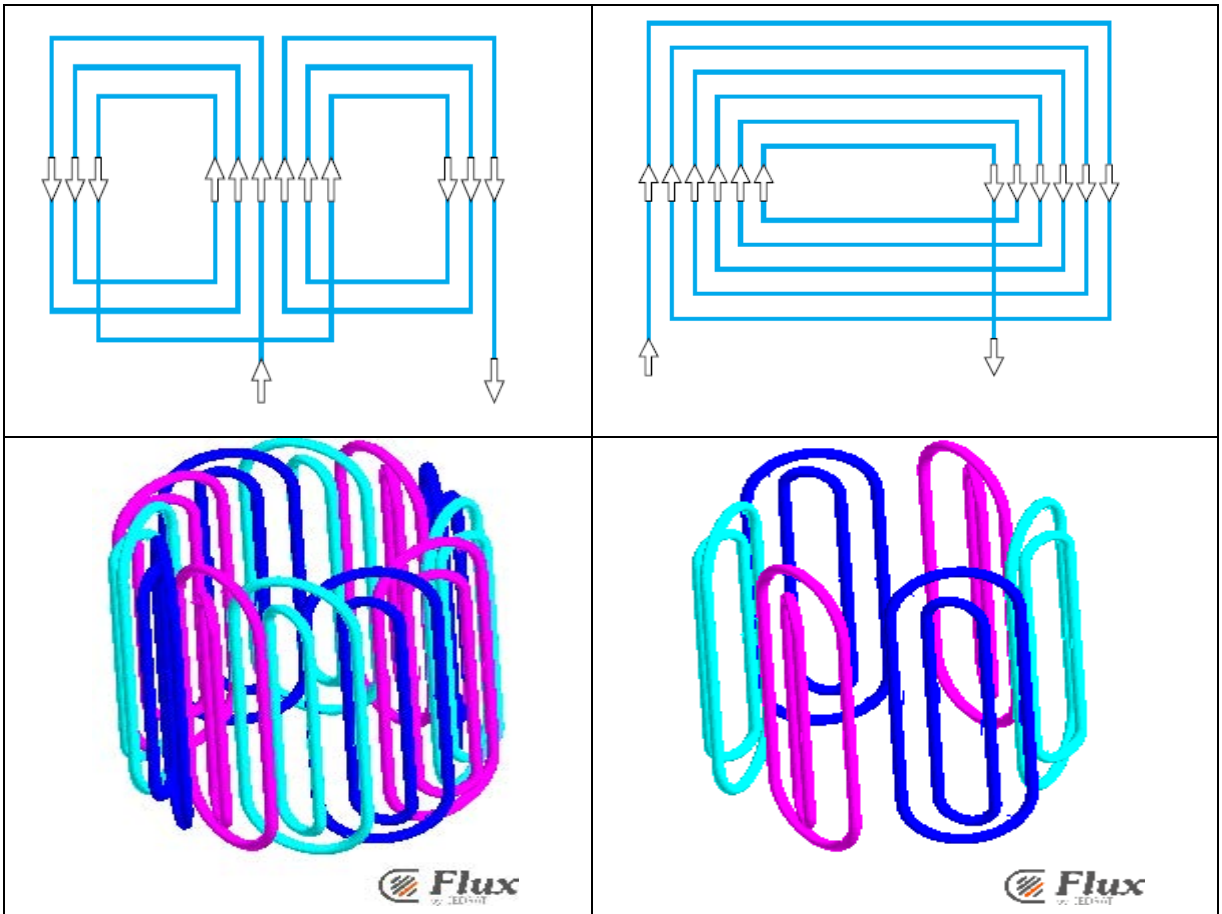
- Le nombre d'encoches par pole et par phase
- Le pas polaire qui est défini comme l'intervalle entre un pôle Nord et un pôle Sud consécutifs
- Le pas dentaire qui est l'intervalle entre deux encoches successives dans le langage de commande Flux

Les figures 40 et 41 montrent les deux types de bobinages utilisés :



Par pôle | **Par pôle conséquent**

Figure 40 : Bobinage imbriqué



Par pôle **Par pôle conséquent**

Figure 41 : Bobinage concentrique

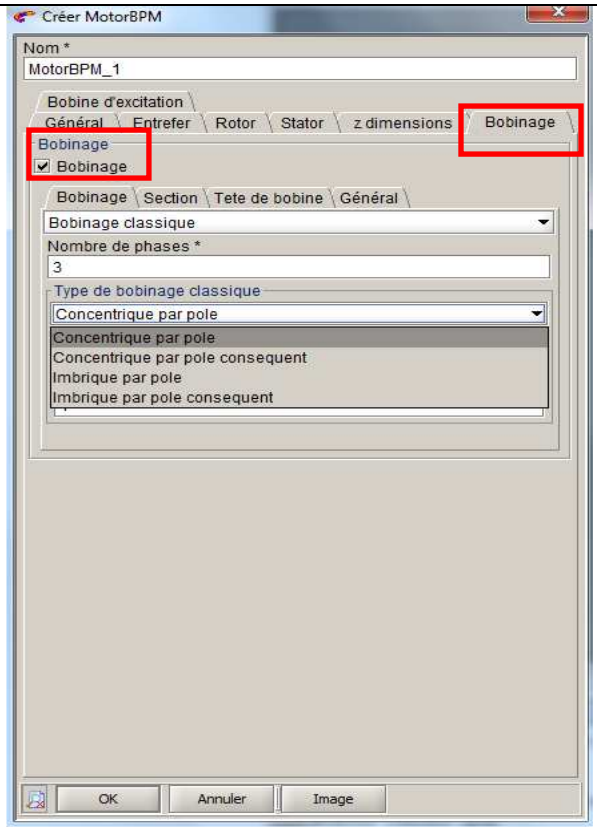


Figure 42 : Bobinage dans l'Overlay 3D

III.4. Overlay 3D : tests

La figure 43 montre les différents types de machine à aimants permanents que nous pouvons créer dans Overlay 3D.

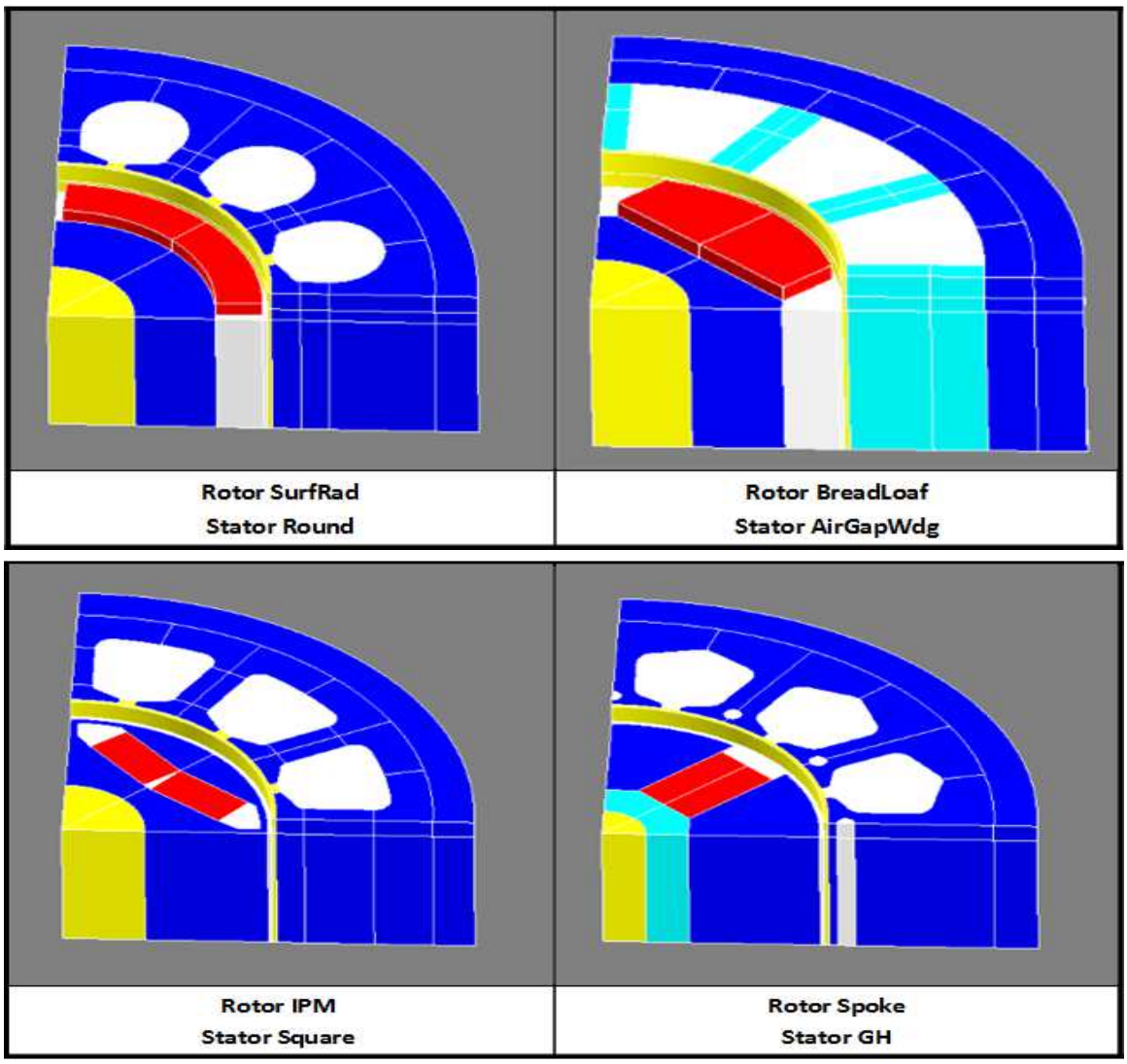


Figure 43 : Différents types de machine à aimants permanents

III.5. Conclusion

Un Overlay 3D d'une machine synchrone à aimants permanents a été créé, son objectif est de faciliter la création de la machine avec peu de paramètres géometriques et un gain de temps dans la construction de dispositif électromagnétique. Les différents points traités sont :

- Les conditions aux limites : boîte infinie et boîte air
- La symétrie
- Le maillage
- Le bobinage

Une série de tests (créer les différents type de machine à aimants : montés en surface, insérés, enterrés…) a été effectuée afin de voir la robustesse et la fiabilité de l'Overlay 3D.

Conclusion et perspectives

Dans ce projet nous nous sommes intéressés à la modélisation 3D à base de la méthode des éléments finis des machines électriques.

Dans la première partie, nous avons commencé par la présentation de l'entreprise, de ses activités, de ses différents services et de son environnement de travail.

Dans la seconde partie, nous avons traité un exemple de modélisation 3D d'une machine synchrone à aimants enterrés, depuis la création du dispositif (construction de la géométrie, maillage, physique, résolution) jusqu'au post traitement. L'objectif était de voir les différentes étapes nécessaires pour créer et modéliser une machine électrique dans Flux 3D. En d'autres termes il s'agissait de déterminer la démarche la plus efficace à suivre pour créer et modéliser en 3D le plus efficacement possible une machine électrique.

En se basant sur les différentes étapes évoquées dans la deuxième partie et sur l'Overlay 2D, nous avons créé notre Overlay 3D qui permet de créer la machine à partir de peu de paramètres géométriques. Les différentes modifications que nous avons intégrées dans l'Overlay 3D sont :

- La boîte infinie en 3D
- La boîte air
- Le maillage (extrusif, automatique et mixte)
- La symétrie
- Le bobinage (bobinage imbriqué, concentrique…)

A la fin de la troisième partie, plusieurs essais on été effectués sur L'Overlay 3D pour tester sa robustesse et sa fiabilité.

Le travail réalisé dans ce projet sera intégré dans les nouvelles versions de Flux3D.

Les perspectives de ce travail sont multiples. Elles sont à situer sur deux niveaux :

- Appliquer la même démarche sur d'autres types de machines : machine asynchrone, machine à reluctance variable…
- Améliorer l'Overlay 3D en intégrant :
    - La physique
    - La résolution (essai à vide, essai en charge, essai en court circuit, démarrage…)
    - Le post-traitement (calcul des performances de la machine : couple électromagnétique, pertes, rendement, inductance…)

Bibliographie

CEDRAT, Groupe. Geometry and mesh. [Tutorial]

CEDRAT, Groupe. Induction heating. [Tutorial]

CEDRAT, Groupe. Translating motor. [Tutorial]

CEDRAT, Groupe. Induction motor. [Tutorial]

CEDRAT, Groupe. Brushless SPM motor. [Tutorial]

CEDRAT, Groupe. Brushless IPM motor. [Tutorial]

Jacques SAINT-MICHEL. Bobinage des machines tournantes à courant alternatif. Technique de l'ingénieur.

Marcel JUFER. Electromagnétisme pour le génie électrique : Circuit magnétiques. p.137.

Technique de l'ingénieur. Réf. Internet : 42252

Daniel CHOÏ. Méthode des éléments finis par l'exemple. Université de Caen, version Avril 2010.

Gérard SWINNEN. Apprendre à programmer avec Python. Institut S^{t} Jean Berchmans – S^{te} Marie, Belgique.

XAVIER Dupé. Programmation avec le langage Python. Mai 2010

Annexes

Annexe A : notions de régions dans Flux

Une région se définit par groupe d'entités géométriques de même type (volumes, faces, lignes, points), qui ont les mêmes caractéristiques physiques

Exemples d'utilisation :

Région (2D)	**Région (3D)**	**Utilisation pour la description …**	**Exemple**
Face	Volume	● Partie de toute forme	
Ligne	Face	● Parties planes de faible épaisseur ● Courants linéiques ● Conditions aux limites spécifiques	Entrefer, feuilles magnétiques
Point	Ligne	● Parties filiformes pour petites sections	Barres magnétiques minces

Région volumique pour application (3D)

Une région …	**permet la modélisation …**
Air ou vide	de l'air ou du vide (perméabilité μr=1)
Magnétique (non conducteur)	d'un milieu magnétique (perméabilité μr) non conducteur
De type conducteur bobiné	de conducteur **avec source** non magnétique (perméabilité μr=1)

Une région …	**permet la modélisation …**
De type conducteur massif	d'un milieu magnétique (perméabilité μr) conducteur (résistivitéρ)

Annexe B: exemple d'un script python pour créer une bobine non maillée dans Flux 3D

```
#! Flux3D 12 SP1
# HISTORY :
# 150615 : version initiale
# goal : Create non meshed coil

"""
@param POLES                    R08  1  1       8        Nombre de pôle
@param BOB_RADIUS_A1            R08  1  1       59       Rayon de la bobine
@param BOB_LENGTH               R08  1  1       34       Longueur de la bobine
@param TOOTH_SLAP               R08  1  1       5        Ouverture polaire
@param NB_TOOTH_PER_POLE        R08  1  1       6        Nombre de dent/pôle
@param NB_TURN                  R08  1  1       6        Nobre de spires
@param RS                       R08  1  1       3       Résistance statorique
"""
def CreateNonMeshedCoil(POLES,BOB_RADIUS_A1,BOB_LENGTH,TOOTH_SLAP,NB_TOOTH_PER_POLE,NB_TURN,RS):
## Mechanical set
        MechanicalSetFixed(name='STATOR')

## Non mesh coil geometrical parameter
        ParameterGeom(name='POLES',
                      expression='8')

        ParameterGeom(name='BOB_RADIUS_A1',
                      expression='59')

        ParameterGeom(name='BOB_LENGTH',
                      expression='34')

        ParameterGeom(name='TOOTH_SLAP',
                      expression='5')

        ParameterGeom(name='NB_TOOTH_PER_POLE',
                      expression='6')

    ParameterGeom(name='NB_TURN',
                  expression='6')

## Non mesh coil E/S parameter
    VariationParameterFormula(name='RS',
                          formula='3')

    if len(Application[ALL])==1:
## Define transient application
        lastInstance = ApplicationMagneticTransient3D(formulationModel=MagneticTransient3DFormulationModelAutomatic(approximation=VectorPotentialApproximationEdge()),
                                scalarVariableOrder=ScalarVariableAutomaticOrder(),
                                vectorNodalVariableOrder=VectorNodalVariableAutomaticOrder(),
                                coilCoefficient=CoilCoefficientAutomatic(),
                                transientInitialization=TransientInitializationStaticComputation())
```

```
## Circuit stranded coil creation
        Equipotential(name='PT_1')

        Equipotential(name='PT_2')

        CircuitStrandedCoil(name='I_A1',
                            terminals=[Equipotential['PT_1'],
                                       Equipotential['PT_2']],
                            resistanceFormula='RS')

        CoordSysCylindrical(name='BOB_STATOR_A1',
                            parentCoordSys=Local(coordSys=CoordSys['XYZ1']),
                            origin=['0',
                                    '0',
                                    '0'],
                            rotationAngles=RotationAngles(angleX='0',
                                                          angleY='0',
                                                          angleZ='0'),
                            visibility=Visibility['VISIBLE'])

##########################################################

        pointList = [] # liste de tous les CoilPoints
        POLES = 8
        y = 0;
        i = 0;
        Rad = 'BOB_RADIUS_A1'
        TETA = str(y)+'*NB_TOOTH_PER_POLE*TOOTH_SLAP'
        z_pos = 'BOB_LENGTH'
        z_neg = '-BOB_LENGTH'
    for i in range(0,POLES):
        pointList.append(CoilPoint(coordinates=[Rad,TETA,z_pos],radius='0'))
        pointList.append(CoilPoint(coordinates=[Rad,TETA,z_neg],radius='0'))
        y+= 1
        TETA = str(y)+'*NB_TOOTH_PER_POLE*TOOTH_SLAP'
        pointList.append(CoilPoint(coordinates=[Rad,TETA,z_neg],radius='0'))
        pointList.append(CoilPoint(coordinates=[Rad,TETA,z_pos],radius='0'))
        y+= 1
        TETA = str(y)+'*NB_TOOTH_PER_POLE*TOOTH_SLAP'

    sublist = pointList

# créer des OpenPathCoil à partir des morceaux

    ComposedCoil(strandedCoil=CoilConductor['I_A1'],
                 turnNumber='NB_TURN',
                 seriesOrParallel=AllInSeries(),
                 coilDuplicationBySymmetriesPeriodicities=CoilNoDuplication(),
                 coordSys=CoordSys['BOB_STATOR_A1'],
                 coilPath=OpenPathCoil(coilPoints=sublist),
                 section=DiscCoilSection(sectionRadius='1'),
                 mechanicalSet=MechanicalSet['STATOR'],
                 aspect=ASPECT['OPAQUE_MAT'],
                 color=Color['Turquoise'],
                 visibility=Visibility['VISIBLE'])
### Non-mesh coil definition
    ComposedCoil(strandedCoil=CoilConductor['I_A1'],

                 turnNumber='NB_TURN',

                 seriesOrParallel=AllInSeries(),

                 coilDuplicationBySymmetriesPeriodicities=CoilDuplication(),
```

```
                coordSys=CoordSys['BOB_STATOR_A1'],

                coilPath=OpenPathCoil(coilPoints=[CoilPoint(coordinates=['BOB_RADIUS_A1',
                                                                         '0',
                                                                         '-BOB_LENGTH'],
                                                            radius='0'),
                                                  CoilPoint(coordinates=['BOB_RADIUS_A1',
                                                                         '0',
                                                                         'BOB_LENGTH'],
                                                            radius='0'),
                                                  CoilPoint(coordinates=['BOB_RADIUS_A1',
                                                                         'NB_TOOTH_PER_POLE*TOOTH_SLAP',
                                                                         'BOB_LENGTH'],
                                                            radius='0'),
                                                  CoilPoint(coordinates=['BOB_RADIUS_A1',
                                                                         'NB_TOOTH_PER_POLE*TOOTH_SLAP',
                                                                         '-BOB_LENGTH'],
                                                            radius='0'),
                                                  CoilPoint(coordinates=['BOB_RADIUS_A1',
                                                                         '2*NB_TOOTH_PER_POLE*TOOTH_SLAP',
                                                                         '-BOB_LENGTH'],
                                                            radius='0'),
                                                  CoilPoint(coordinates=['BOB_RADIUS_A1',
                                                                         '2*NB_TOOTH_PER_POLE*TOOTH_SLAP',
                                                                         'BOB_LENGTH'],
                                                            radius='0'),
                                                  CoilPoint(coordinates=['BOB_RADIUS_A1',
                                                                         '3*NB_TOOTH_PER_POLE*TOOTH_SLAP',
                                                                         'BOB_LENGTH'],
                                                            radius='0'),
                                                  CoilPoint(coordinates=['BOB_RADIUS_A1',
                                                                         '3*NB_TOOTH_PER_POLE*TOOTH_SLAP',
                                                                         '-BOB_LENGTH'],
                                                            radius='0'),
```

```
                CoilPoint(coordinates=['BOB_RADIUS_A1',

                                       '4*NB_TOOTH_PER_POLE*TOOTH_SLAP',

                                       '-BOB_LENGTH'],

                          radius='0'),

                CoilPoint(coordinates=['BOB_RADIUS_A1',

                                       '4*NB_TOOTH_PER_POLE*TOOTH_SLAP',

                                       'BOB_LENGTH'],

                          radius='0'),

                CoilPoint(coordinates=['BOB_RADIUS_A1',

                                       '5*NB_TOOTH_PER_POLE*TOOTH_SLAP',

                                       'BOB_LENGTH'],

                          radius='0'),

                CoilPoint(coordinates=['BOB_RADIUS_A1',

                                       '5*NB_TOOTH_PER_POLE*TOOTH_SLAP',

                                       '-BOB_LENGTH'],

                          radius='0'),

                CoilPoint(coordinates=['BOB_RADIUS_A1',

                                       '6*NB_TOOTH_PER_POLE*TOOTH_SLAP',

                                       '-BOB_LENGTH'],

                          radius='0'),

                CoilPoint(coordinates=['BOB_RADIUS_A1',

                                       '6*NB_TOOTH_PER_POLE*TOOTH_SLAP',

                                       'BOB_LENGTH'],

                          radius='0'),

                CoilPoint(coordinates=['BOB_RADIUS_A1',

                                       '7*NB_TOOTH_PER_POLE*TOOTH_SLAP',

                                       'BOB_LENGTH'],

                          radius='0'),

                CoilPoint(coordinates=['BOB_RADIUS_A1',

                                       '7*NB_TOOTH_PER_POLE*TOOTH_SLAP',

                                       '-BOB_LENGTH'],

                          radius='0'),

                CoilPoint(coordinates=['BOB_RADIUS_A1',

                                       '8*NB_TOOTH_PER_POLE*TOOTH_SLAP',
```

```
                                        '-BOB_LENGTH'],

                                radius='0'),

                            CoilPoint(coordinates=['BOB_RADIUS_A1',

                                        '8*NB_TOOTH_PER_POLE*TOOTH_SLAP',

                                        'BOB_LENGTH'],

                                radius='0'),

                            CoilPoint(coordinates=['BOB_RADIUS_A1',

                                        '9*NB_TOOTH_PER_POLE*TOOTH_SLAP',

                                        'BOB_LENGTH'],

                                radius='0'),

                            CoilPoint(coordinates=['BOB_RADIUS_A1',

                                        '9*NB_TOOTH_PER_POLE*TOOTH_SLAP',

                                        '-BOB_LENGTH'],

                                radius='0'),

                            CoilPoint(coordinates=['BOB_RADIUS_A1',

                                        '10*NB_TOOTH_PER_POLE*TOOTH_SLAP',

                                        '-BOB_LENGTH'],

                                radius='0'),

                            CoilPoint(coordinates=['BOB_RADIUS_A1',

                                        '10*NB_TOOTH_PER_POLE*TOOTH_SLAP',

                                        'BOB_LENGTH'],

                                radius='0'),

                            CoilPoint(coordinates=['BOB_RADIUS_A1',

                                        '11*NB_TOOTH_PER_POLE*TOOTH_SLAP',

                                        'BOB_LENGTH'],

                                radius='0'),

                            CoilPoint(coordinates=['BOB_RADIUS_A1',

                                        '11*NB_TOOTH_PER_POLE*TOOTH_SLAP',

                                        '-BOB_LENGTH'],

                                radius='0'),

                            CoilPoint(coordinates=['BOB_RADIUS_A1+3',

                                        '12*NB_TOOTH_PER_POLE*TOOTH_SLAP',

                                        '-BOB_LENGTH'],

                                radius='0'),
```

```
                                                CoilPoint(coordinates=['BOB_RADIUS_A1+3',

                                                                       '12*NB_TOOTH_PER_POLE*TOOTH_SLAP',

                                                                       'BOB_LENGTH'],

                                                                  radius='0')]),

                        section=DiscCoilSection(sectionRadius='1'),

                        mechanicalSet=MechanicalSet['STATOR'],

                        aspect=ASPECT['OPAQUE_MAT'],

                        color=Color['Turquoise'],

                        visibility=Visibility['VISIBLE'])
```

UNIVERSITÉ DE LORRAINE

Master I2E2I

Faculté des sciences et technologies
BP70239

54506 VANDOEUVRE LES NANCY

RESUME

La simulation occupe actuellement une place importante dans le domaine du génie électrique, en particulier dans les machines électriques.

Dans ce rapport nous nous sommes intéressés à la modélisation des machines électriques à aimants permanents avec la méthode des éléments finis en utilisant le logiciel Flux 3D, qui est un logiciel complet, capable de résoudre des problèmes magnétiques, électriques et thermiques, nous avons abordé en détail les différentes étapes pour construire la géométrie, définir la physique, le maillage, la résolution et le post-traitement.

Afin de faciliter à l'utilisateur de créer sa géométrie pour modéliser les dispositifs électromagnétiques à partir de peu de paramètres géométriques nous avons créé une interface de l'Overlay 3D à partir de l'Overlay 2D.

Cette expérience professionnelle m'a permis d'approfondir mes connaissances théoriques, mes connaissances en simulation numérique et m'a énormément apporté sur le plan personnel.

Pendant cette période j'ai pu assister à des nombreuses réunions et démonstrations ce qui m'a permis de mieux m'intégrer sur le plan professionnel, et de faire un point sur l'état d'avancement de mon stage.

ABSTRACT

Nowadays simulation of industrial systems becomes inevitable, especially in the field of electrical engineering. In this report, we present a 3D finite element modeling of an electric machine including permanents magnets using Flux 3D. This last is a complete software which has the ability to solve magnetic, electrical and thermal problems. We introduce step by step how to build the geometry, define physical properties, meshing of the system, resolution and the post-processing. In order to facilitate the building of geometries, we created a 3D Overlay from the Overlay 2D to help the users to create there own geometry using just few parameters.

This experience was very fruitful, I took part in different meetings and talks, that allows me to enhance my knowledge in the field of numerical simulation and made better my integration in my professional team.

MOTS-CLES

Flux 2D/3D, Overlay 2D/3D, Machine synchrone, Aimants permanents, Maillage.