Réalisation de cas de démo Flux accessibles depuis le superviseur

Islam Bouray

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 électrique »
Année universitaire 2013/2014

Réalisation de cas de démo Flux accessibles depuis le superviseur

Mémoire présenté par « M. Bouray Islam »
Soutenu le : 12/09/2014

Stage effectué à l'entreprise **Cedrat SA**
15 chemin de malacher Inovallée 38246 Meylan, France

Tuteur industriel : Dr. Soualmi Abdessamed

Sommaire

Remerciements **2**
Introduction **3**
PRESENTATION GENERALE DU GROUPE CEDRAT **4**
Introduction 5
Historique 5
Généralités 6
Ses activités 6
Son chiffre d'affaire 7
Ses solutions logicielles 7
Présentation du logiciel FLUX 7
Présentation du stage : 8
MODELISATION PAR ELEMENTS FINIS **9**
Historique 10
Principe général 10
Choix d'un maillage 11
Conception assisté par ordinateur 11
Principe 11
Historique 11
La CAO en électrotechnique 12
GENERATION DES EXEMPLES SUR FLUX **13**
Contenu des exemples Flux 14
Structure des fichiers Python 15
Structure du fichier 'BuildGeomesh.Py' 15
Structure du fichier 'BuildPhys.Py' 17
Structure du fichier 'Solving.Py' 19
Structure du fichier 'PostProcessing.Py' 19
Structure des fichiers 'main.Py' et 'mainEX.Py' 21
Contenu du fichier XML 23
PRESENTATION DES RESULTATS OBTENUS **24**
Projet réalisés 25
Projets restants 25
Présentation du projet « Machine Synchrone à Aimants Permanents Enterrés en 3D » 25
Présentation du dispositif 25
Caractéristiques 26
Maillage 26
Exploitation des résultats 27
Conclusion et perspectives **36**
Bibliographies **37**
Annexes **38**

Remerciements

Au terme de ce stage de fin d'étude, il ya des personnes qui me restent en mémoire et que je tiens à remercier pour le rôle constructif qu'elles ont joué, et pour avoir ainsi contribué à rendre cette expérience professionnelle riche en enseignements.

En premier lieu, j'aimerai remercier Dr. Abdessamed Soualmi ingénieur application à l'entreprise CEDRAT et tuteur de stage, pour m'avoir accompagné tout au long de ma mission et pour m'avoir permis d'acquérir rapidement une méthode de travail appliquée et centrée sur l'analyse.

Remerciement à Dr. Patrick Lombard, responsable du service Application, pour s'être montré toujours disponible, aidant et à l'écoute et pour m'avoir permis d'effectuer ces six mois de stage au sein de l'entreprise.

Je remercie M. Benjamin Valet, pour le temps qu'il m'a consacré mon stage, pour l'aide qu'il m'a apporté, sa disponibilité et ses conseils riches en informations.

Je tiens à remercier également Dr. Farid Zidat, Leïla Nguimpi-Langue, Diana Mavrudieva et Sylvain Perez pour les conseils précieux qu'ils m'ont apportés.

Enfin, je remercie tout l'ensemble des employés de CEDRAT pour leur accueil et pour m'avoir permis de réaliser ce stage dans d'excellentes conditions.

Je réserve également des remerciements très spéciaux à l'ensemble des enseignants de la faculté des sciences et technologies de l'université de Lorraine.

Introduction

Actuellement CEDRAT dispose de tutoriels permettant de mettre en avant les possibilités du logiciel FLUX 2D/3D pour modéliser différents dispositifs (Capteurs, moteurs, actionneurs, …). Ils sont très détaillés et parfois longs. Leur compréhension n'est pas toujours intuitive. Il manque un niveau d'exemples simplifiés qui permettrait de mettre rapidement en avant une ou plusieurs fonctionnalités. Ceci permettrait également d'enrichir abondamment la base d'exemples.

L'objectif du stage se résume en :

- Définir les exemples à ajouter dans le superviseur du Flux
- Définir la structure des différents exemples
- Intégrer les nouveaux exemples dans le superviseur de Flux

Le travail de se stage est structuré dans ce rapport en trois parties :
Partie I :
Présentation du groupe CEDRAT

Partie II :
Principe de modélisation par la méthode des éléments finis
Démarche pour création et réalisation d'un projet sous Flux (2D et 3D)

Partie III :
Exemple de modélisation 3D sous Flux : machine synchrone à aimants permanents enterrés.

PRESENTATION GENERALE DU GROUPE CEDRAT

Introduction

Dans le cadre de ma formation de master deuxième année Energie électrique, à la faculté des sciences et technologies Henry Poincaré, j'ai été amené à effectuer mon stage de fin d'étude au sein de la société CEDRAT.SA dont voici l'historique.

Historique

1971 : Création de la société CEDRAT (Communauté d'Etude et de Développement Régional et Aménagement du Territoire), par un groupe d'agronomes, d'ingénieurs et d'économistes.
1977 : Création de l'entreprise CEDRAT recherche, société d'ingénierie en électrotechnique
(Label SRC : société de recherche sous contrat).
1986 : Fondation de la société américaine MAGSOFT.
1992 : Séparation des activités de la société et création d'une entité supplémentaire CEDRAT
Développement.
CEDRAT est alors un groupe dont les 3 entreprises liées sont :
CEDRAT (Développement et commercialisation de logiciels, gestion administrative du groupe).
CEDRAT Développement (Aménagement du territoire).
CEDRAT Recherche (Bureau d'études en électrotechnique).
2001 : CEDRAT Recherche change de nom et devient CEDRAT Technologie.
2004 : CEDRAT se sépare de CEDRAT Développement consacré à l'environnement et au développement rural afin de pouvoir réunir un même domaine de compétence sous la même enseigne.
2006 : Achat de 33.6% de la société Adapted Solutions par CEDRAT.
2011 : Séparation de la société CEDRAT TECHNOLOGIE de la société CEDRAT.SA

Aujourd'hui, la société s'appelle CEDRAT SA et elle est spécialisée dans les solutions logicielles, le groupe a pris une dimension internationale grâce à sa filiale américaine Magsoft, crée en 1986, et à la société allemande Adapted Solutions, détentrice du logiciel Portunus**,** qui a rejoint le groupe en 2006 ainsi qu'à son réseau de distributeurs dont voici la carte. (Figure 1.1)

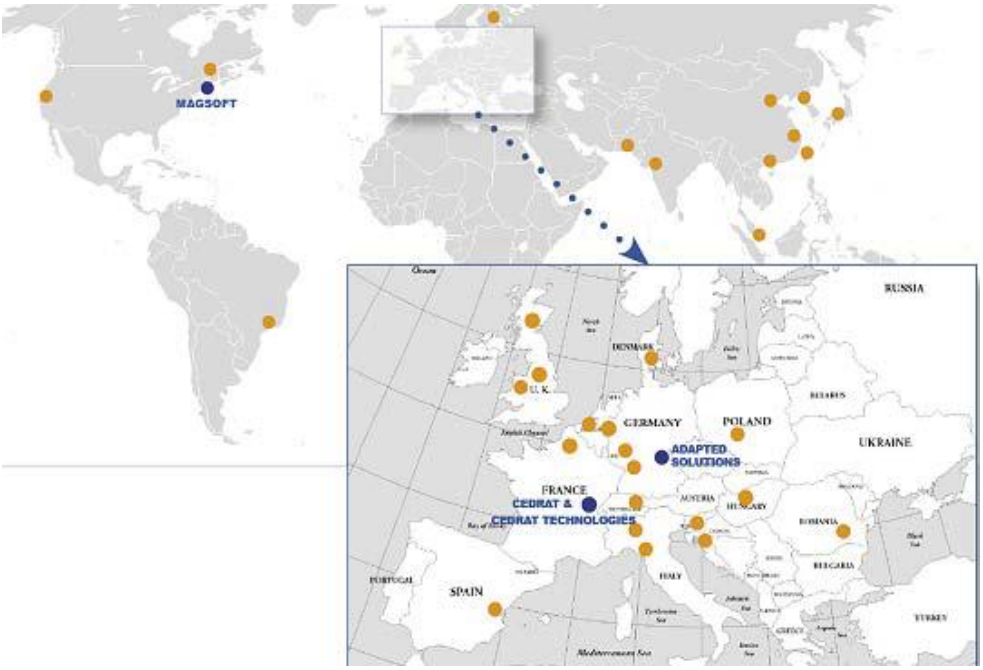


Figure 1.1 réseau de distributeurs

Généralités

Le groupe Cedrat est composé de trois sociétés qui sont les suivantes :

Ses activités

Le groupe CEDRAT fournit des solutions innovantes dans les domaines du génie électrique et de la mécatronique en proposant à ses clients des solutions logicielles, la possibilité de réaliser des études où bien de concevoir et de fabriquer leurs systèmes.

Son chiffre d'affaire

Le chiffre d'affaire 2013 du groupe CEDRAT est de 5,5 millions d'euros et aujourd'hui, celui-ci emploie environ 56 employés dont une quinzaine de docteurs ingénieurs.

Ses solutions logicielles

Les différents logiciels proposés par le groupe CEDRAT sont les suivants :

Flux	logiciel de simulation 2D et 3D des dispositifs électromagnétiques et thermiques
InCa3D	simulation avancée pour les connexions électriques
Portunus	simulation des systèmes mécatroniques
Got-It	Optimisation pour les systèmes et dispositifs électromagnétiques
Motor-CAD	optimisation thermique des moteurs
SPEED	dimensionnement des moteurs électriques
PSCAD	Solution pour les professionnels de la simulation des systèmes de puissances

Tableau 1.1 Logiciels proposés par Cedrat

Présentation du logiciel FLUX

Flux 2D /3D est un logiciel de conception assisté par ordinateur utilisant la méthode des éléments finis, il permet le calcul des états magnétiques, électriques ou thermiques des dispositifs en régimes permanents, transitoires et harmoniques, avec des fonctionnalités d'analyse multiparamétrique étendues, les couplages circuit et cinématique. Ceci nécessite la résolution d'équations diverses: équations de Maxwell, de la chaleur, loi de comportement des matériaux.

Flux est développé (en collaboration avec le G2ELab, fusion du Laboratoire d'Electrotechnique de Grenoble, du Laboratoire d'Electrostatique et Matériaux Diélectriques et du Laboratoire de Magnétisme du Navire) et distribué en France par la société CEDRAT S.A.

Applications physiques disponibles dans Flux	**2D**	**3D**	**Skew**
Magnéto Statique	**X**	**X**	**X**
Magnétique Transitoire	**X**	**X**	**X**
Magnéto Harmonique	**X**	**X**	**X**
Electro Statique		X	X
Conduction Electrique		X	X

Electro Harmonique	X	X
Thermique Permanent	X	X
Thermique Transitoire	X	X
Magnéto Harmonique couplée Thermique Transitoire	X	X
Conduction Electrique couplée Thermique Transitoire	O	O
Electro Harmonique couplé Thermique Transitoire	O	O

Tableau 1.2 : application physique dans Flux (2D/3D)

Présentation du stage :

Ce projet consiste à réaliser des exemples de simulation de dispositifs électriques accessibles depuis le superviseur du logiciel Flux 2D/3D (voir figure dessous).
Les missions principales du stage sont :

- Produire des prototypes d'exemples simplifiés dans un premier temps et dans un second temps, une première base d'exemples simplifiés couvrant la majorité des fonctionnalités du logiciel et des métiers utilisateur
- Produire des tutoriaux avec le niveau technique requis sur des métiers bien ciblés (transformateurs, actionneurs, moteurs …),
- Définir la stratégie de la documentation à réaliser en étant la plus proche de l'utilisateur, fait également partie du périmètre du stage.

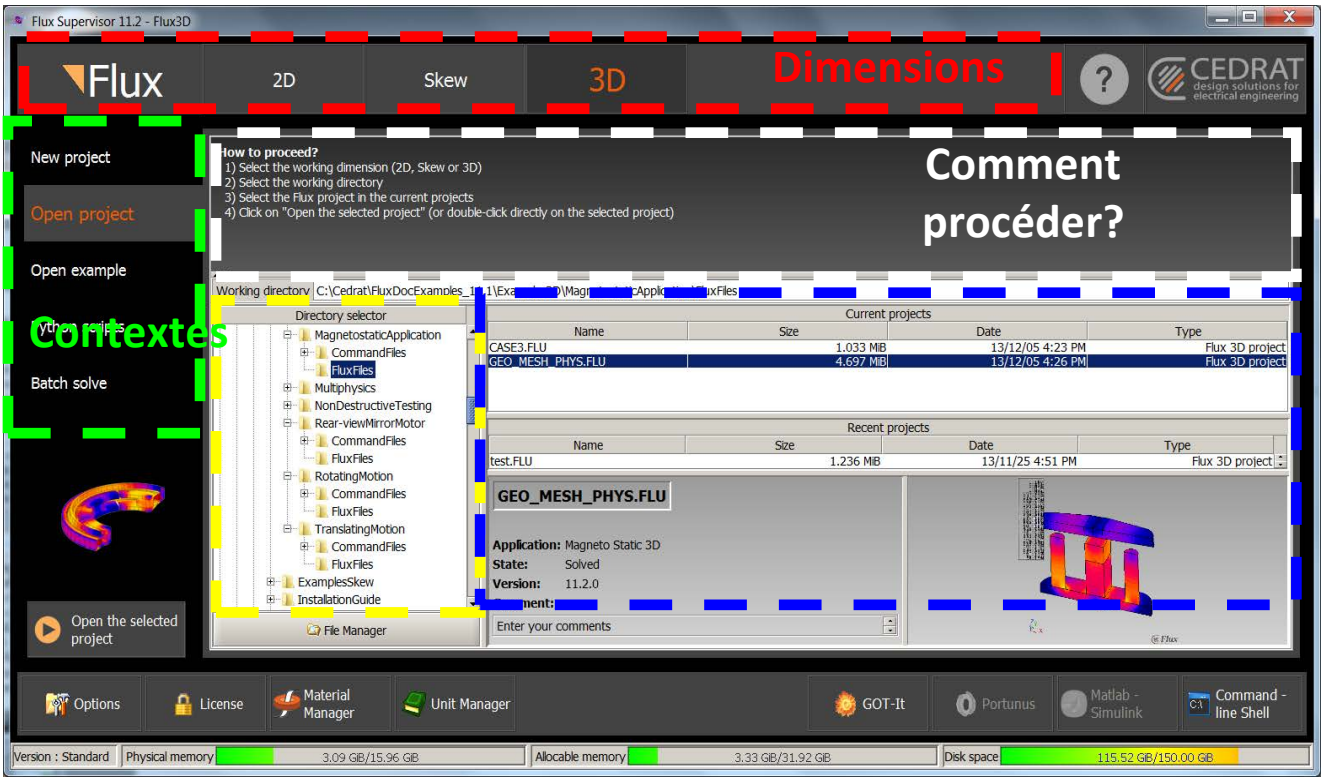


Fig1.2 Interface Flux 2D/3D

MODELISATION PAR ELEMENTS FINIS

En analyse numérique, la méthode des éléments finis est utilisée pour résoudre numériquement des équations aux dérivées partielles. Celles-ci peuvent par exemple représenter analytiquement le comportement dynamique de certains systèmes physiques (mécaniques, thermodynamiques, acoustiques, etc.)[1]
Concrètement, cela permet par exemple de calculer numériquement le comportement d'objets même très complexes, à condition qu'ils soient continus et décrits par une équation aux dérivées partielles linéaire : mouvement d'une corde secouée par l'un de ses bouts, comportement d'un fluide arrivant à grande vitesse sur un obstacle, déformation d'une structure métallique, etc.

Historique

La méthode des éléments finis est apparue avec l'analyse des structures, née vers 1850. Les premières études menées sur la résistance des matériaux dans des conditions de petites déformations, ce qui a permis d'obtenir des systèmes simples résolus « manuellement », notamment par Maxwell, Castigliano, Mohr. La formalisation, et le concept mathématique d'élément fini est apparu bien plus tard, vers 1940 et la définition est posée par Newmark, Hrenikoff, Mc Henry et Courant.
L'arrivée du calcul numérique et de méthodes de résolution performantes par ordinateur a permis de populariser la méthode.

Principe général

La méthode des éléments finis permet donc de résoudre de manière discrète une EDP dont on cherche une solution approchée « suffisamment » fiable. De manière générale, cette EDP porte sur une fonction u, définie sur un domaine. Elle comporte des conditions aux bords permettant d'assurer existence et unicité d'une solution.[2][3]
Sauf cas particuliers, la discrétisation passe par une redéfinition et une approximation de la géométrie, on considère donc le problème posé sur la géométrie approchée par un domaine polygonal ou polyédrique par morceaux. Une fois la géométrie approchée, il faut choisir un espace d'approximation de la solution du problème, dans la MEF, cet espace est défini à l'aide du maillage du domaine (ce qui explique aussi pourquoi il est nécessaire d'approcher la géométrie). Le maillage du domaine permet d'en définir un pavage dont les pavés sont les éléments finis. Un élément fini est la donnée d'une cellule élémentaire et de fonctions de base de l'espace d'approximation dont le support est l'élément, et définies de manière à être interpolantes (voir Fonctions de base).[2]
Bien qu'il existe de nombreux logiciels exploitant cette méthode et permettant de « résoudre » des problèmes dans divers domaines, il est important que l'utilisateur ait une bonne idée de ce qu'il fait, notamment quant au choix du maillage et du type d'éléments qui doivent être adaptés au problème posé : aucun logiciel ne fait tout pour l'utilisateur, et

il faut toujours garder un œil critique vis-à-vis de solutions approchées. Pour cela il existe des indicateurs d'erreur et des estimateurs d'erreur qui permettent d'ajuster les différents paramètres.[3]
La solution trouvée, il reste cependant à déterminer les caractéristiques de la méthode ainsi développée, notamment l'unicité de l'éventuelle solution ou encore la stabilité numérique du schéma de résolution. Il est essentiel de trouver une estimation juste de l'erreur liée à la discrétisation et montrer que la méthode ainsi écrite converge, c'est-à-dire que l'erreur tend vers 0 si la finesse du maillage tend elle aussi vers 0.
Dans le cas d'une EDP linéaire avec opérateur symétrique (comme l'est l'opérateur laplacien), il s'agit finalement de résoudre une équation algébrique linéaire, inversible dans le meilleur des cas.

Choix d'un maillage

La méthode des éléments finis repose sur un découpage de l'espace selon un maillage. D'habitude l'on choisit un maillage carré ou triangulaire mais rien n'interdit de choisir des maillages plus complexes. Il n'est pas non plus nécessaire que le maillage soit régulier et l'on a tendance à resserrer le maillage près des endroits d'intérêt (par exemple aux endroits où l'on pense que la solution va beaucoup varier) ; cependant, il faut veiller à avoir des éléments faiblement distordus (se rapprocher d'un polygone régulier). Plus ce maillage est resserré, plus la solution que l'on obtient par la méthode des éléments finis sera précise et proche de la « vraie » solution de l'équation aux dérivés partielles.

Conception assisté par ordinateur

Principe

La conception assistée par ordinateur (CAO) comprend l'ensemble des logiciels et des techniques de modélisation géométrique permettant de concevoir, de tester virtuellement – à l'aide d'un ordinateur et des techniques de simulation numérique – et de réaliser des produits manufacturés et les outils pour les fabriquer.

Historique

La CAO décolla dans les années 75-90, lorsque le coût de mise en place d'un poste se rapprocha du coût annuel d'un dessinateur. La mise en place fut un peu pénible au début en raison d'une nécessité de reprendre les plans existants. On s'aperçut à cette occasion que statistiquement près de 10 % des cotations sur les plans existants étaient inexactes, que des références de plans existaient en double, qu'une référence unique pouvait correspondre à plusieurs plans légèrement différents, etc. Au bout du compte, le gain de fiabilité de l'information se révéla constituer un argument supplémentaire important décidant à généraliser la CAO.[4]

La CAO en électrotechnique

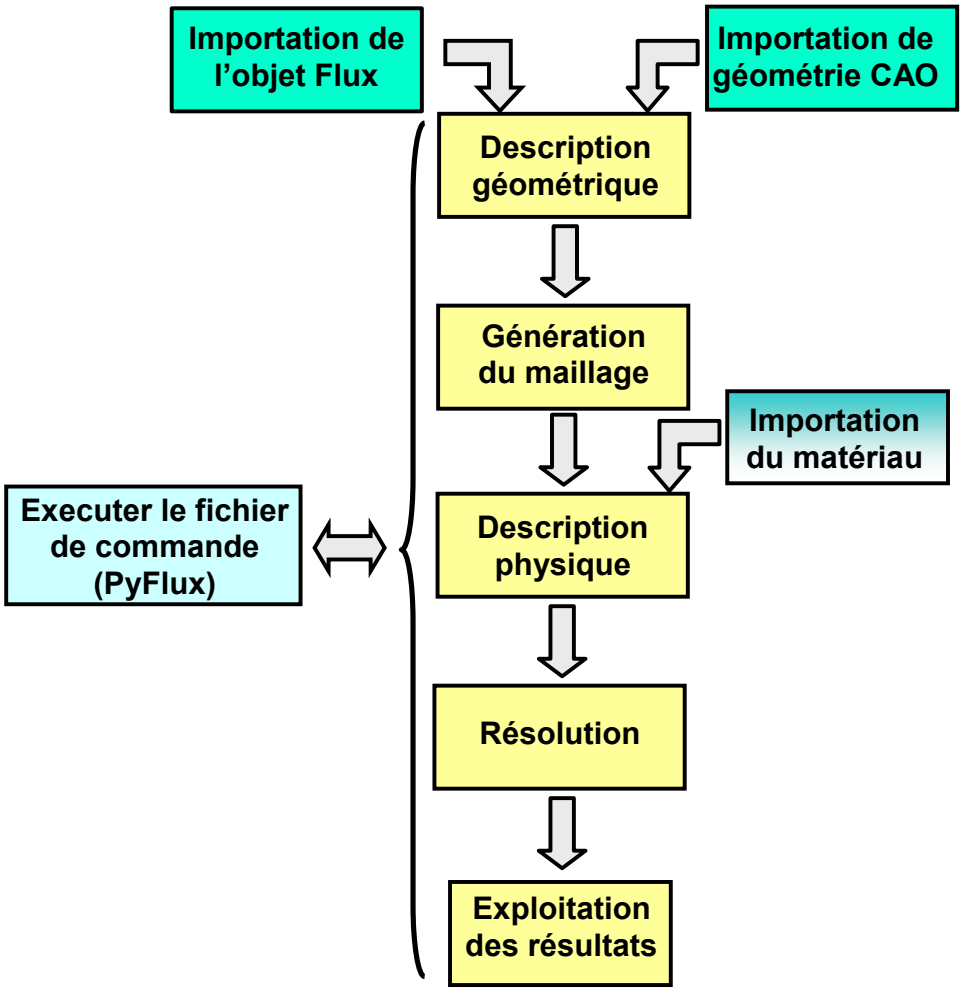
Les logiciels de conception permettent la réalisation de plans de câblage électrique pour les domaines de l'industrie, distribution d'énergie, automobile, aéronautique, …
Le logiciel de CAO permet au concepteur une prise en charge globale du projet par un même outil (réalisation des plans, des liens entre composants et plans, des borniers et connecteurs, des nomenclatures, des implantations composants, des faisceaux de câblage…).
Les logiciels de CAO électriques facilitent également les échanges entre les corps de métier amenés à collaborer sur certains projets tel que le bâtiment. Les plans d'architectes réalisés dans les formats standards sont ensuite importés et utilisés comme base par des logiciels spécialisés notamment dans les schémas d'implantation électrique. Ce type de logiciels ne vise pas la réalisation de schémas de fonctionnement mais permet à l'utilisateur la création d'installations électriques domestiques ou tertiaires et de visualiser les dépendances entre appareils (interrupteurs ↔ lampes...), le matériel nécessaire au projet (appareils, filerie, gaines...) ainsi que des contenus détaillés de chaque gaine ou des boites de dérivation.[4][5]
Dans ce domaine deux modes de conception existent :

- Mode symbolique : utilisé depuis le début de la CAO électrotechnique (1984 en France avec SAFIRS), il consiste à prendre les symboles des plans comme éléments principaux contenant les informations de la CAO.
- Mode objet : créé en France depuis 1995, il consiste à prendre les objets (composants de nomenclature) comme éléments principaux contenant les informations de la CAO. Il permet ainsi de faire les créations et modifications depuis tout type de représentation (ou non) en assurant une mise à jour en temps réel sur le projet entier. (Il est possible de commencer par la nomenclature et finir par le schéma, de modifier un appareil ou des câbles sans avoir à régénérer les nomenclatures, borniers, carnet câble, etc.)

GENERATION DES EXEMPLES SUR FLUX

Pour analyser des dispositifs électriques, magnétiques et thermiques par la méthode dés élément finis dans le logiciel Flux (2D/3D), Cinq étapes essentiels pour la création et l'analyse des résultats du projet :

- Construction géométrique
- Génération du maillage
- Propriétés physique
- Résolution
- Exploitation des résultats



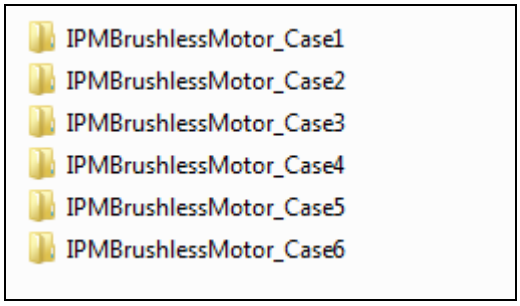
Construction d'un projet Flux (2D/3D)

Créer des exemples dans le superviseur de Flux nous permet d'automatiser les cinq étapes essentielles pour la création d'un projet sous forme d'un script à base du logiciel Python.

Contenu des exemples Flux

Les exemples Flux sont des projets Flux tous faits qui ont pour but de modéliser un dispositif électrique (Machine, transformateur…).

Ces projets contiennent plusieurs cas de simulation qui se trouvent chacun dans un répertoire contenant plusieurs fichiers qui permettent de lancer une simulation à partir du superviseur du logiciel Flux.



Ces fichiers sont organisés comme suit :

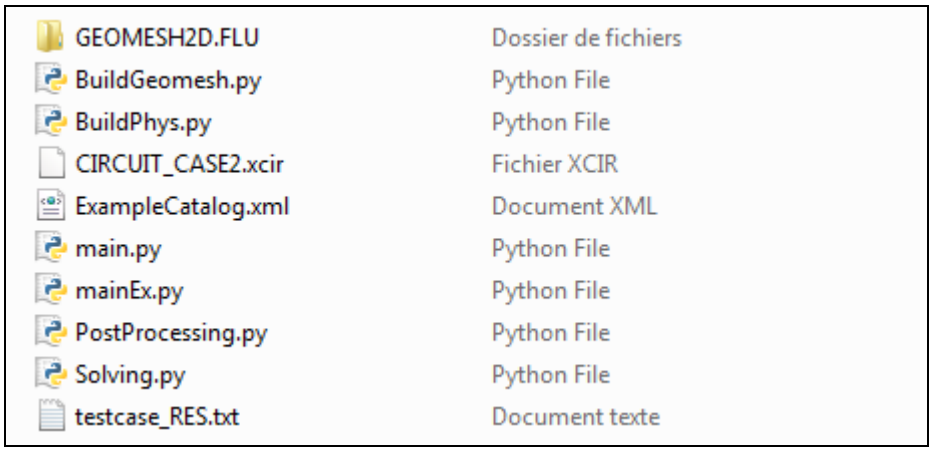
6 fichiers '.Py' : BuildGeomesh.Py, BuildPhys.Py, Solving.Py, PostProcessing.Py, main.Py et main.EX.

1 fichier '.xml' : ExampleCatalog.xml

1 fichier '.xcir' : Le circuit externe (Dans quelques simulations).

1 répertoire '.FLU' : Qui contient la géométrie 2D dans le cas d'un import.

1 fichier '.txt' : Contient les résultats de l'exploitation de chaque cas.



Structure des fichiers Python

Structure du fichier 'BuildGeomesh.Py'

Ce fichier contient la géométrie et le maillage du dispositif, Il peut être initialisé par un import d'une géométrie 2D (Si on travaille sur Flux 3D) ou la géométrie est construite d'une façon classique (Points, lignes, faces...)

```
newProject()

importObjectFromFlux(projectName='GEOMESH2D.FLU',
                     filter='GEOMETRY')
```

Dans le cas d'un import d'une géométrie 3D, une extrusion des faces est nécessaire pour avoir la géométrie 3D.

```
lastInstance = TransfTranslationVector(name='EXTRUDE',
                        coordSys=CoordSys['XYZ1'],
                        vector=['0',
                                '0',
                                'Lfer/2'])

result = FaceAutomatic[ALL].extrude(transformation=Transf['EXTRUDE'],
         repetitionNumber=1,
         extrusionType='STANDARD',
         buildingOption='ADD VOLUMES AND ASSOCIATED EXTRUSION MESH GENERATOR')
```

Ensuite, le fichier doit contenir les transformations nécessaires pour la construction de la géométrie (Symétrie, périodicité..). Cela nous permet de dessiner une part de la machine au lieu de la dessiner complètement, ce qui permet un gain au niveau du temps de simulation.

```
lastInstance = SymmetryXYplane(physicalType=SymmetryTangentMagFields(),
               position='0')
```

Une fois la géométrie construite, il faut assigner les informations du maillage aux entités géométriques (Points, lignes…)

```
lastInstance = MeshPoint(name='STATOR_MIDDLE_SLOT_3',
               lengthUnit=LengthUnit['MILLIMETER'],
               value='1.5*STATOR_MESH_MIDDLE_SLOT/(1+MESH_FACTOR)*10**3*LENGTH_UNIT/2',
               color=Color['Yellow'])

Point[30].mesh=MeshPoint['STATOR_MIDDLE_SLOT_3']

lastInstance = MeshLineArithmetic(name='A_8',
                        color=Color['White'],
                        number=8)

Line[285].mesh=MeshLine['A_8']
```

L'étape suivante est la création de la boite infinie (conditions aux limites), cette boite doit englober toute la géométrie du dispositif. Plusieurs formes sont proposées pour cette boite (Cube, Cylindre...)

```
lastInstance = InfiniteBoxCylinderZ(size=['160',
                                          '180',
                                          '90',
                                          '105'])
```

Dans le cas d'une géométrie 3D cette boite doit être complétée par une commande avec laquelle on peut associer aussi les informations du maillage de la boite.

```
InfiniteBoxCylinderZ['InfiniteBoxCylinderZ'].complete3D(buildingOption='ADD VOLUMES, FACES, LINES AND POINTS',
        coordSys=CoordSys['AIRGAP_COORD'],
        linkMesh='ADD LINK MESH ASSOCIATED')
```

La dernière étape est le lancement du maillage et la sauvegarde du projet sous le nom « GeomeshBuilt.FLU »

```
meshDomain()

saveProjectAs('GeomeshBuilt.FLU')
```

Structure du fichier 'BuildPhys.Py'

Ce fichier contient toute la physique du dispositif c.à.d. les matériaux, les ensembles mécaniques, le type du bobinage …etc.
La première étape est de choisir l'application qui correspond au scénario qu'on veut résoudre (Magnétostatique, magnétique transitoire…)

```
LastInstance = ApplicationMagneticTransient3D(formulationModel=MagneticTransient3DFormulationModelAutomatic(approximation=VectorPotentialApproximationEdge()),
                                              scalarVariableOrder=ScalarVariableAutomaticOrder(),
                                              vectorNodalVariableOrder=VectorNodalVariableAutomaticOrder(),
                                              coilCoefficient=CoilCoefficientAutomatic(),
                                              transientInitialization=TransientInitializationStaticComputation())
```

Après ça, il faut créer et/ou importer les matériaux utilisés pour la modélisation de la machine. Pour l'import, on utilise la banque des matériaux de Flux.

```
lastInstance = Material(name='NDFEB',
                        propertyBH=PropertyBhMagnetOneDirection(br='1.2',
                                                                mur='1.05'))

import os
INSTALL = os._getEnvironment()["INSTALLFLUX"]+ "//.."
macroFile = INSTALL+"/Materials/FLUX_111_MATERI.DAT"
importMaterial(fileName=macroFile, materialNames=['FLU_M270_35A          :'])
```

Puis, il faut créer les ensembles mécaniques de la machine (Parties mobiles, parties fixes)

```
lastInstance = MechanicalSetRotation1Axis(name='ROTOR',
                                          kinematics=RotatingImposedSpeed(velocity='1/6',
                                                                          initialPosition='0'),
                                          rotationAxis=RotationZAxis(coordSys=CoordSys['XYZ1'],
                                                                     pivot=['0',
                                                                            '0',
                                                                            '0']))

lastInstance = MechanicalSetFixed(name='STATOR')
```

Si la simulation nécessite une connexion à un circuit externe, il faut importer le circuit qui serait déjà fait à partir de l'interface du logiciel.

```
importCircuit(filename='CIRCUIT_CASE3.xcir',
              importValue=1,
              ImportFELink=0)
```

L'étape suivante est la création des régions volumiques (ou surfaciques sur Flux 2D) et on associe les volumes de la machine à ces régions.

```
lastInstance = RegionVolume(name='WEDGE',
                            magneticTransient3D=MagneticTransient3DVolumeVacuum(),
                            color=Color['White'],
                            visibility=Visibility['VISIBLE'],
                            mechanicalSet=MechanicalSet['STATOR'])

Volume[20,13,35,70,36,71,37,72,38,73,39,74].region=RegionVolume['WEDGE']
```

Dans le cas des aimants permanents, une orientation de l'induction rémanente est nécessaire.

Pour la prise en compte des courants induits dans les aimants permanents, on les représente par des régions de type conducteur massif avec une résistivité.

```
orientRegVolMaterial(region=RegionVolume['MAGNET1_1_POLE1'],coordSys=CoordSys['ROTOR_COORD'],orientation='Direction',angle='10')
orientRegVolMaterial(region=RegionVolume['MAGNET2_1_POLE1'],coordSys=CoordSys['ROTOR_COORD'],orientation='Direction',angle='-10')
```

Au niveau des bobines, selon le besoin de l'utilisateur. C'est-à-dire : si on utilise des bobines non maillées, ces dernières doivent être crées point par point, pour cela, il faut créer les paramètres et les repères avec lesquels on crée la géométrie des bobines. (Annexe1).

A la fin le projet est enregistré sous le nom « PhysBuilt.FLU »

```
saveProjectAs('PhysBuilt.FLU')
```

Structure du fichier 'Solving.Py'

Ce fichier permet la création du scénario de résolution ainsi que le lancement de la résolution.
Dans la première étape, il faut choisir le paramètre qui varie, l'intervalle et sa discrétisation. C'est-à-dire : choisir le nombre de points par période et les bornes de l'intervalle (valeur min et max).

```
Scenario(name='SHORT_CIRCUIT')

startMacroTransaction()

Scenario['SHORT_CIRCUIT'].addPilot(pilot=MultiValues(parameter=VariationParameter['TIME'],
                                                     intervals=[IntervalStepNumber(minValue=0.0,
                                                                                   maxValue=0.6,
                                                                                   stepNumber=121)]))

endMacroTransaction()
```

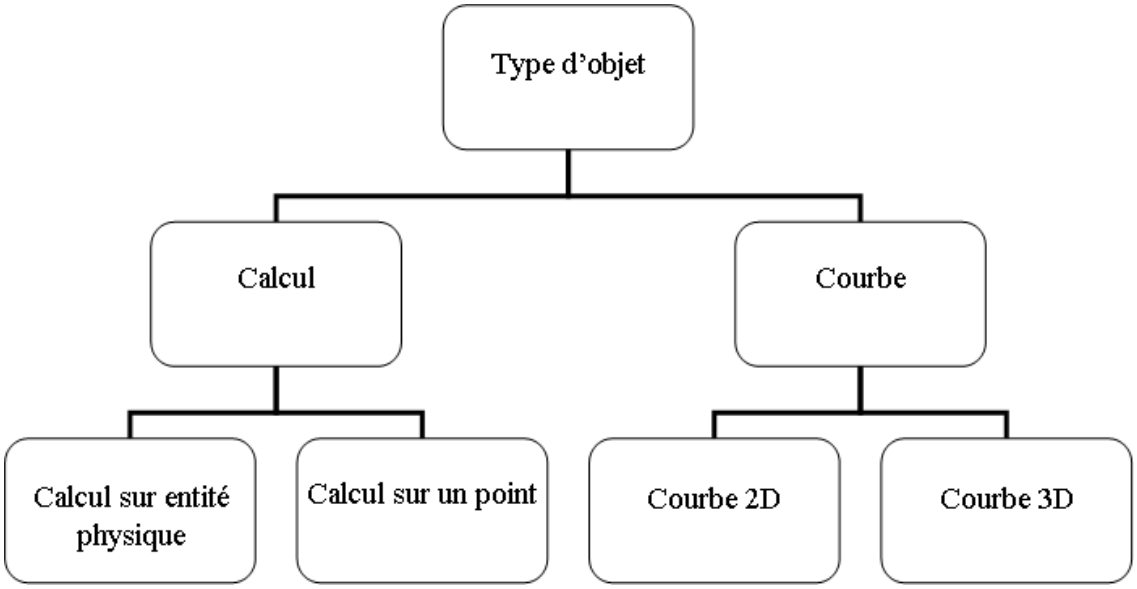
Une fois le scénario est créer, on lance la simulation dans un projet qui s'appelle 'Solved.FLU'

```
Scenario['SHORT_CIRCUIT'].solve(projectName='Solved.FLU')
```

Structure du fichier 'PostProcessing.Py'

Ce fichier permet l'exploitation des résultats de simulation.

Cet organigramme illustre les types d'exploitation existant sur FLUX



Différents types d'exploitation des résultats

Dans le cas d'une courbe 2D, il faut choisir l'intervalle et la grandeur à afficher.

```
EvolutiveCurve2D(name='CURRENTS',
                 evolutivePath=EvolutivePath(parameterSet=[SetParameterXVariable(paramEvol=VariationParameter['TIME'],
                                                                                 limitMin=0.0,
                                                                                 limitMax=0.6)]),
                 formula=['I(BPA)',
                          'I(BPB)',
                          'I(BMC)'])
```

Pour un calcul sur une entité physique, il faut choisir l'expression de la grandeur qu'on veut calculer.

```
ComputePhysicTransient['ComputePhysic'].formula=['Mod(I(COILCONDUCTOR_1))/sqrt(2)',
                                                 'PjouleCC(COILCONDUCTOR_1)',
                                                 'EMag(1_DOMAIN)',
                                                 'ModV(ModC(FluxCoil(COILCONDUCTOR_1)))/Sqrt(2)']
```

Une autre forme d'exploiter les résultats est l'affichage des flèches ou des isovaleurs d'une grandeur magnétique, pour cela, il faut préciser le pas de simulation, la grandeur et les régions volumiques où on veut l'afficher.

```
lastInstance = IsovalueSpatialGroup(name='ISOVAL_1',
                    formula='B',
                    forceVisibility='YES',
                    smoothValues='YES',
                    regionInternalExternal='AUTOMATIC',
                    internalComputation='NO',
                    regionType='VOLUME REGION',
                    group=[Groupspatial['2_NO_VACUUM']])
```

La dernière étape est l'exportation des résultats dans un fichier texte, dont l'utilité est de stocker ces résultats et les comparer après lors de la sortie d'une autre version du logiciel

```
CurveVariation2D[ALL].exportTXT(txtFile='testcase_RES.txt',
            mode='ADD VALUES')
```

Structure des fichiers 'main.Py' et 'mainEX.Py'

- Le fichier mainEX.py :

Une autre technique pour lancer un projet Flux via le superviseur est d'utiliser le fichier : mainEX.py.

Il sert à exécuter les 4 fichiers '.Py' précédents dans l'ordre. A la fin, on aura comme résultat le projet PostProcessed.FLU avec toutes les exploitations faites pour le cas de simulation lancé.

```
try :

    executeBatchSpy('BuildGeomesh.py')


except :

    resetError()

try :


    executeBatchSpy('BuildPhys.py')

except :

    resetError()

try :


    executeBatchSpy('Solving.py')

except :

    resetError()


try :


    executeBatchSpy('PostProcessing.py')

except :

    resetError()
```

- **Le fichier 'main.py'**

Ce fichier exécute le fichier mainEX.Py et ferme le projet ensuite, il sert à vérifier l'absence d'erreurs lors du lancement du projet.

```
try :

    executeBatchSpy('mainEx.py')


except :

    resetError()

closeProject()

exit()
```

Contenu du fichier XML

Le fichier XML est un fichier de commande, il sert à :

- Nommer chaque cas sur l'interface du logiciel (fr et an)
- Donner la page correspondante sur le fichier explicatif
- Citer les mots clefs correspondant à chaque cas (fr et an)
- Commander l'interface de l'exemple en exécutant les fichiers '.Py' qui correspondent à chaque action sur l'interface du superviseur.

```
<ExampleCatalog>
   <Label language="fr" value="Calcul du couple de détente"/>
   <Label language="en" value="Cogging torque computation"/>
   <Summary page="2"/>
   <Description>
      <Text language="fr"></Text>
      <Text language="en"></Text>
   </Description>
   <Description>
      <HtmlFile language="fr" file="Description.fr.html"/>
      <HtmlFile language="en" file="Description.en.html"/>
   </Description>
   <Keywords language="fr">Moteur brushless IPM, Rotation, Magnétique Transitoire, Circuit, Couple de détente, Champ magnétiaque</Keywords>
   <Keywords language="en">IPM Brushless motor, Rotation, Transient Magnetic, Circuit, Non meshed coils, Cogging torque, Magnetic field</Keywords>
   <Actions>
      <Action identifier="Action1" type="PythonScript" arguments="mainEx.py">
         <Label language="fr" value="Exploitation"/>
         <Label language="en" value="Postprocessing"/>
      </Action>
   </Actions>
</ExampleCatalog>
```

PRESENTATION DES RESULTATS OBTENUS

Dans cette partie, nous allons nous intéresser à la description géométrique et à la modélisation d'une machine synchrone à aimants permanents enterrés. Il s'agit de décrire les différentes caractéristiques, performances et la physique de la machine.

Projet réalisés

Nom du projet	2D/3D/Skew	Projet de base	Nouveauté	
			Non	oui
Machine asynchrone	2D	/	x	
Brushless IPM motor	3D	Cahier technique 2D		x
Brushless SPM motor	3D	Cahier technique 2D		x
Tore magnétique	3D	/		x
Brushless IPM motor	Skew	Cahier technique 2D		x
Machine linéaire	3D	/		x

Tableau 4.1 : Projets réalisés pendant le stage

Projets restants

Nom du projet	2D/3D/Skew	Projet de base
Alternateur synchrone	3D	Cahier technique 2D
Turbo-alternateur	3D	/

Tableau 4.2 : Projets restants

Présentation du projet « Machine Synchrone à Aimants Permanents Enterrés en 3D »

Présentation du dispositif

Le « Machine Synchrone à Aimants Permanents Enterrés en 3D » est un moteur de traction/génération conçu pour un véhicule électrique hybride (TOYOTA Prius), c'est un moteur brushless (sans balais) AC avec aimants permanents enterrés, il est représenté sur la figure ci-dessous et comprend les éléments suivants:

- Une partie fixe (le stator), comprenant la culasse, les encoches et les bobines.
- Un entrefer.
- Une partie mobile (rotor) comprenant les aimants enterrés.

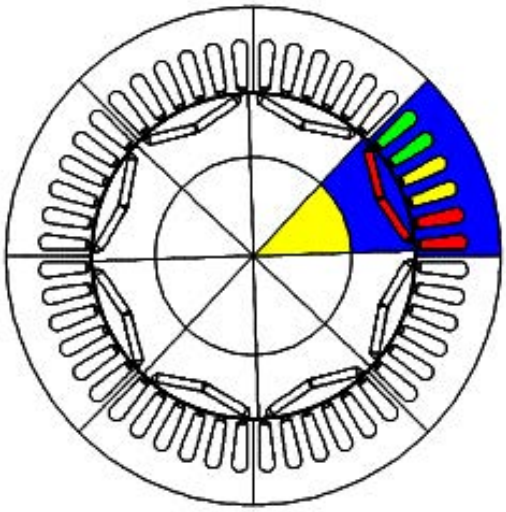


Fig4.1 Géométrie de la machine

Seule la partie colorée sera modélisée sur Flux. En utilisant une périodicité, les calculs seront faits pour la machine complète.

Caractéristiques

Electriques

- Tension max : 500 V
- Couple nominale : 400 N.m
- Vitesse max : 6000 rpm
- Puissance nominale : 50 kW pour 1200-1500 rpm

Physiques

- 48 encoches statoriques
- 4 paires de pôles
- Aimants permanents NdFeB
- Type de laminage M270-35A
- Diamètre externe 242 mm
- Longueur utile 75 mm

Maillage

Pour construire un projet dans Flux, la première étape c'est de rentrer les données géométriques du dispositif (machine électrique, câble électrique,…)
La deuxième étape c'est le maillage.

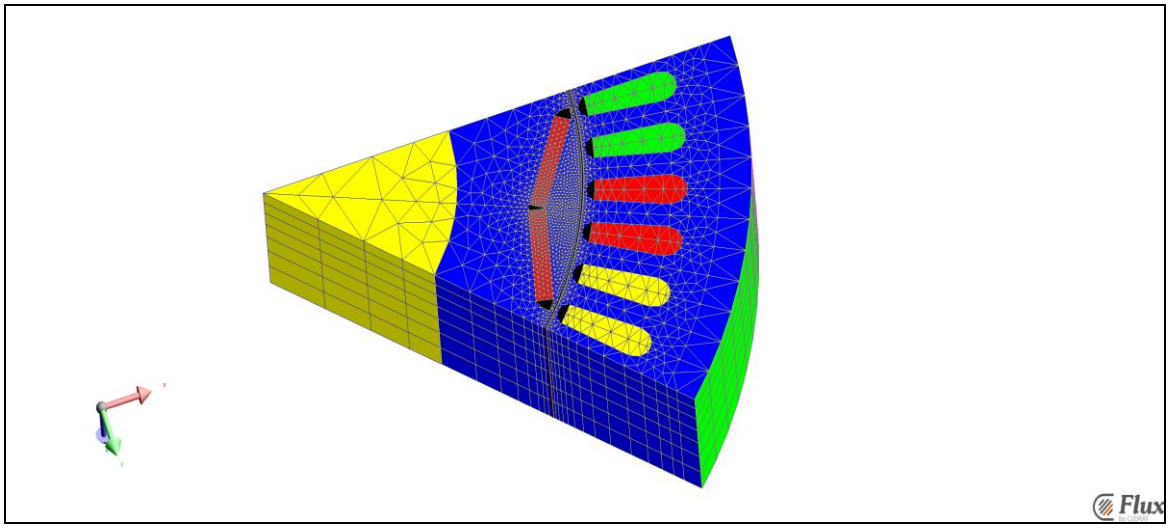


Fig4.2 Maillage de la machine

Différents types de discrétisations sont utilisés pour mailler le dispositif, cela dépend de la précision qu'on veut sur la région volumique. On remarque bien que le maillage est plus concentré dans les parties : Entrefer, aimants permanents et les parties de la culasse qui sont proches de l'entrefer.

Bobines non maillées

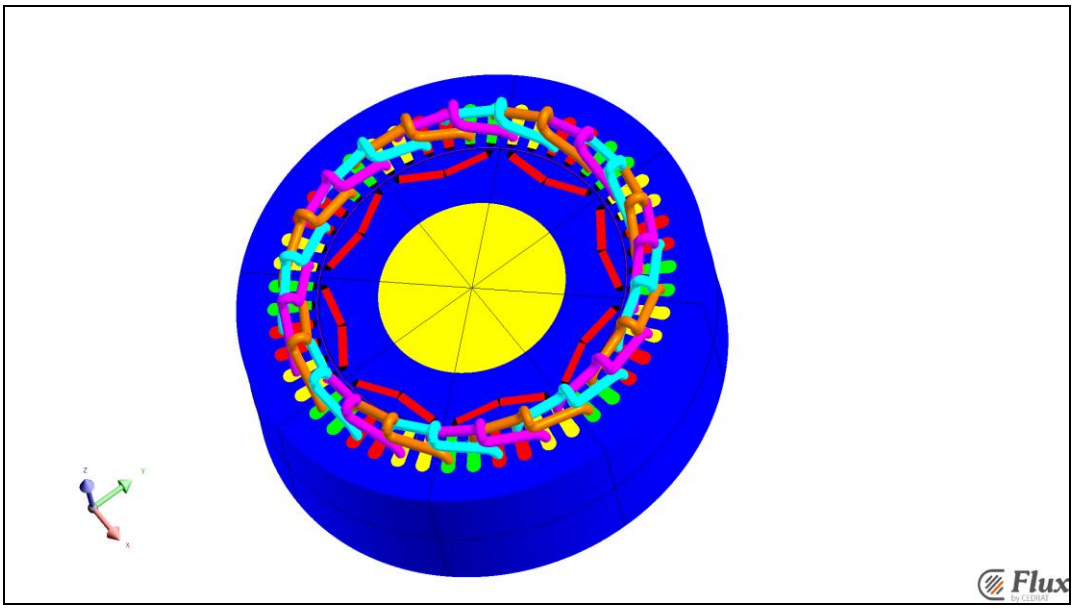


Fig4.3 Schéma de la machine avec les bobines non maillées

Exploitation des résultats

Cas 1 : Calcul du couple de détente

Le couple de détente est calculé à l'aide d'une simulation multi-position et hors tension. Elle est simulée à l'aide d'une application transitoire à vitesse constante. La vitesse choisie est de 1/6 tr/mn, ce qui correspond à 1 degré mécanique par seconde.
Il est demandé de tracer la courbe du couple électromagnétique ainsi que d'afficher le dégradé d'induction sur la machine.

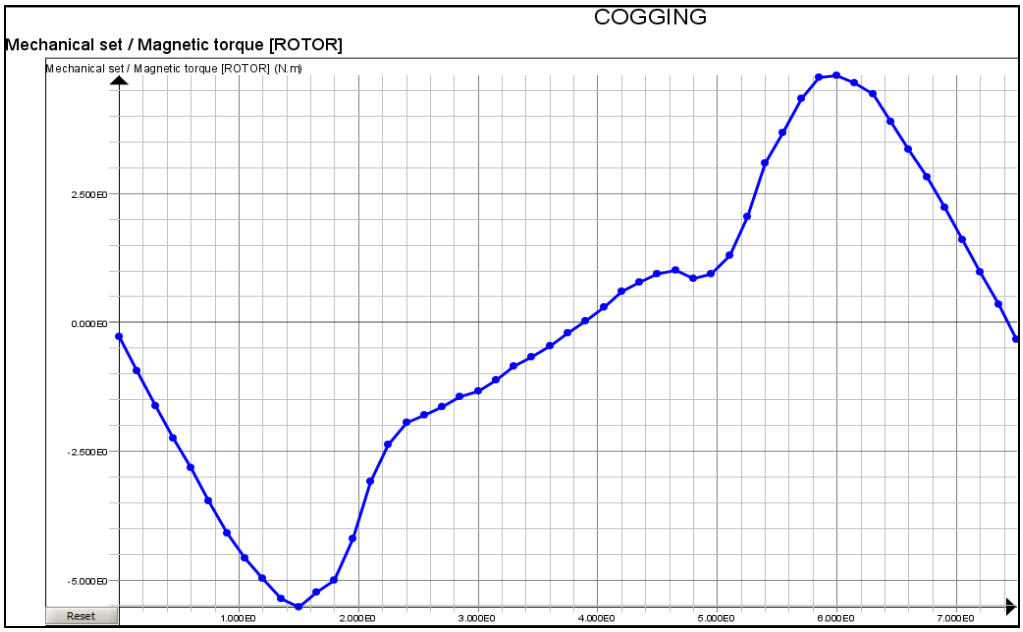


Fig4.4 Couple détente

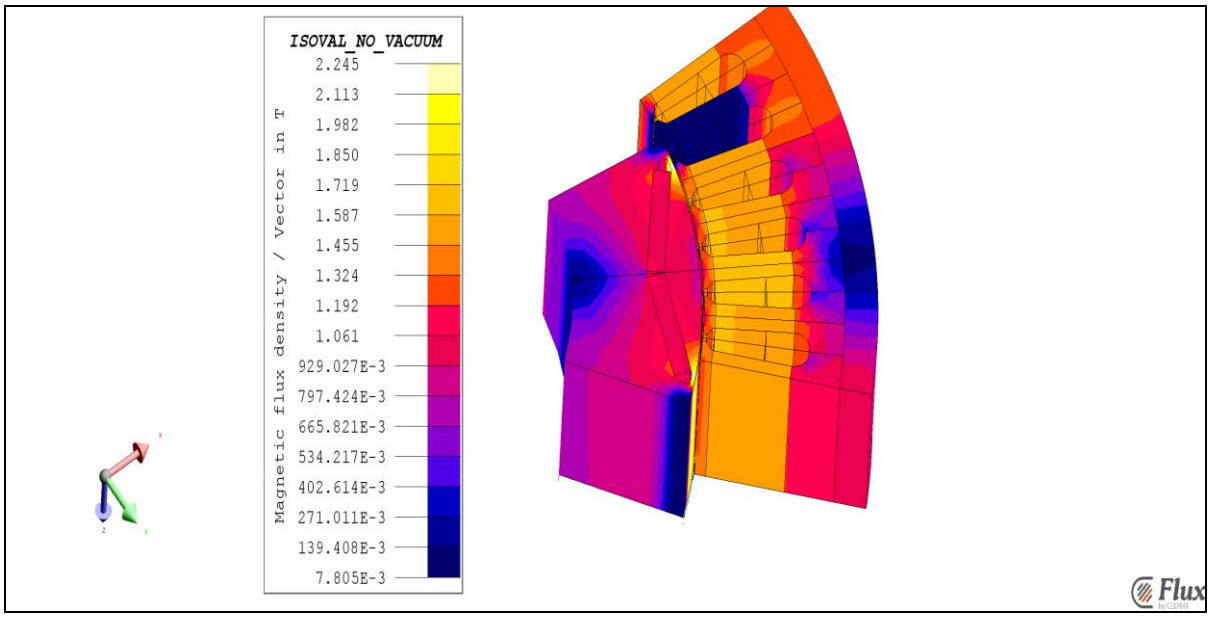


Fig4.5 Dégradé d'induction

Cas 2 : Calcul de la force électromotrice

Par définition la force électromotrice aux bornes d'un circuit est la dérivée par rapport au temps du flux qui le traverse pour chaque spire. Elle est calculée à une vitesse de 1000 tr/mn en fonctionnement à vide de la machine. Avec cette simulation on peut afficher :

- Isovaleurs de champ magnétique
- Evolution de la force électromotrice en fonction de la position
- Calcul de la FFT de la force électromotrice

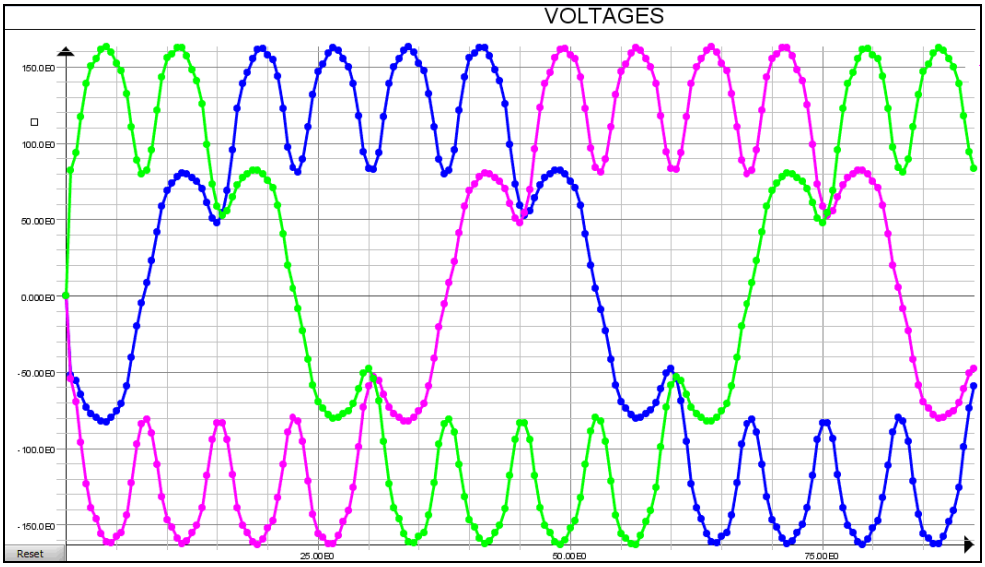


Fig4.6 Tensions à vide

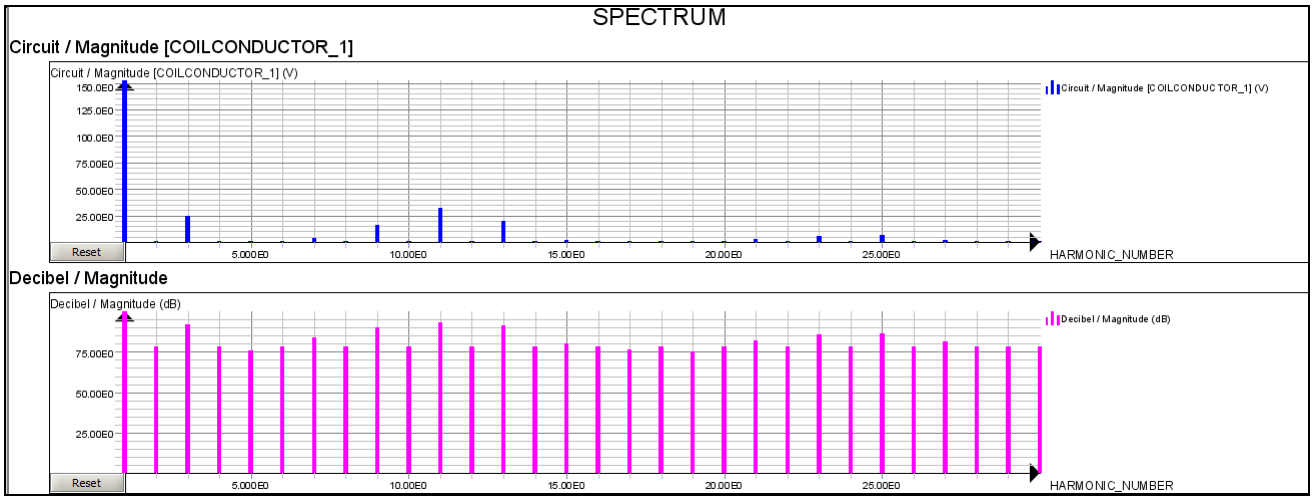


Fig4.7 Le spectre de U(BPA)

Cas 3 : Performances à vitesse imposée

Le moteur est alimenté par un courant sinusoïdal triphasé et fonctionne à vitesse constante. L'objectif de la simulation est de calculer les performances de la machine synchrone à aimants enterrés à savoir :

- Couple électromagnétique
- Pertes fer (Bertotti et modèle LS)
- Efficacité.
- Puissance absorbée
- Puissance mécanique

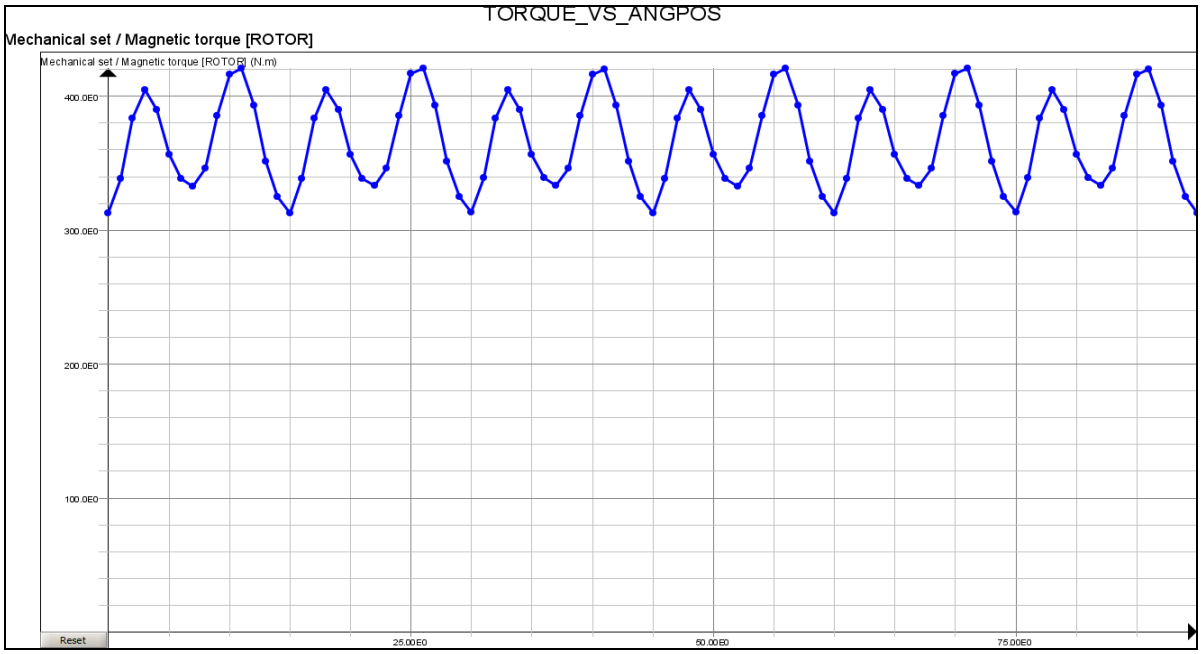


Fig4.8 Le couple

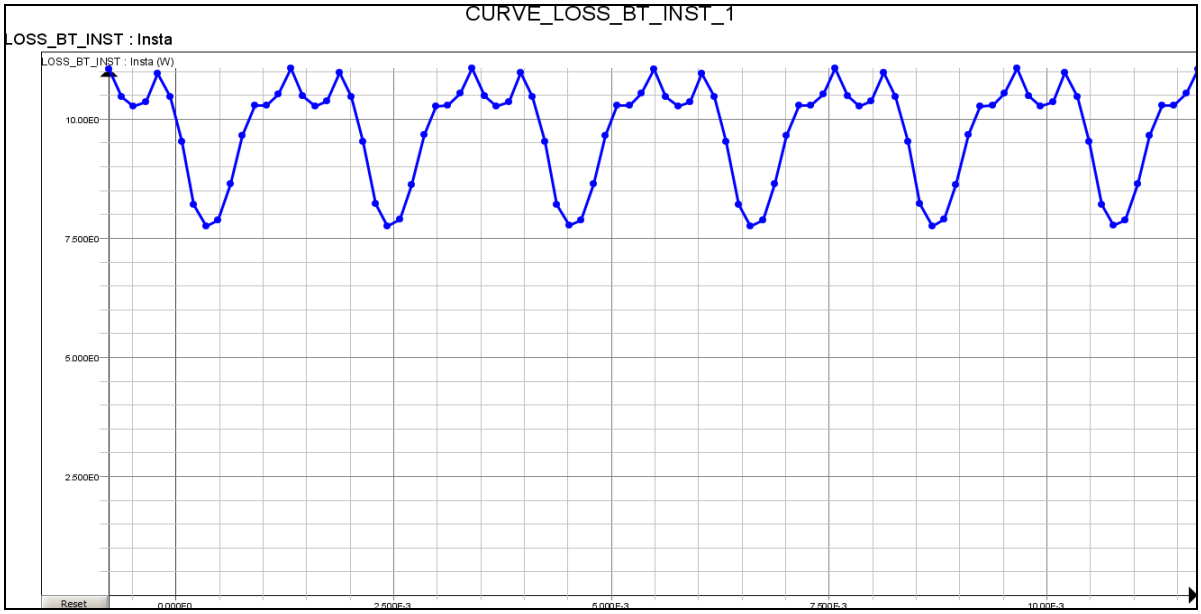


Fig4.9 Les pertes fer « modèle BT »

Cas 4 : Performances au démarrage

Le comportement dynamique du moteur au démarrage est simulé à l'aide d'une stratégie de contrôle du courant. Le bobinage est alimenté en fonction de la position du rotor.
On trace l'évolution de la position, de la vitesse et du couple en fonction du temps et de tracer des courbes 2D des courants sans les 3 phases en fonction du temps aussi.

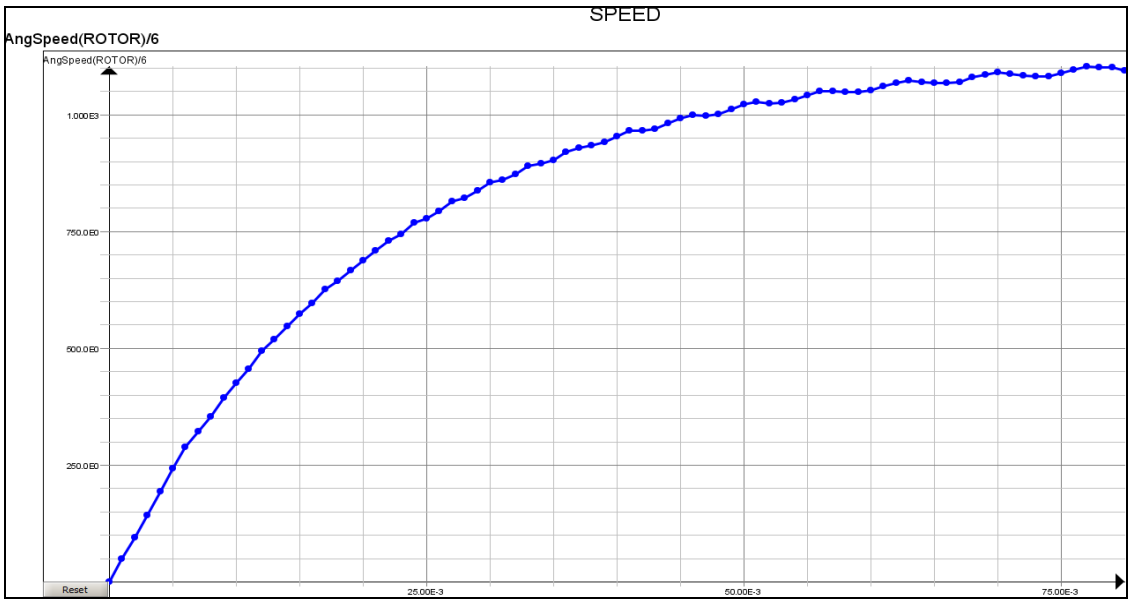


Fig4.10 Evolution de la vitesse en fonction de temps

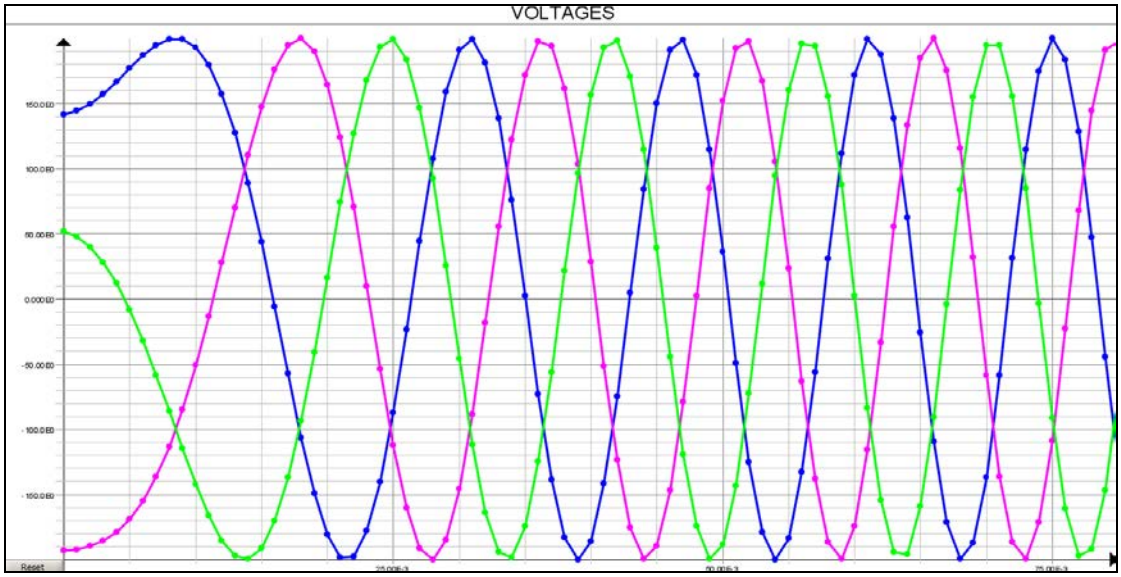


Fig4.11 Evolution de la tension en fonction de temps

Cas 5 : Etude multistatique

Cette simulation consiste à calculer les inductances et le couple en fonction du courant et de la position du rotor.
Il est demandé d'afficher les courbes de couple en fonction de la position et du courant et d'afficher les courbes de flux en fonction de la position et du courant. A la fin on calcul les inductances Ld et Lq en fonction du courant.

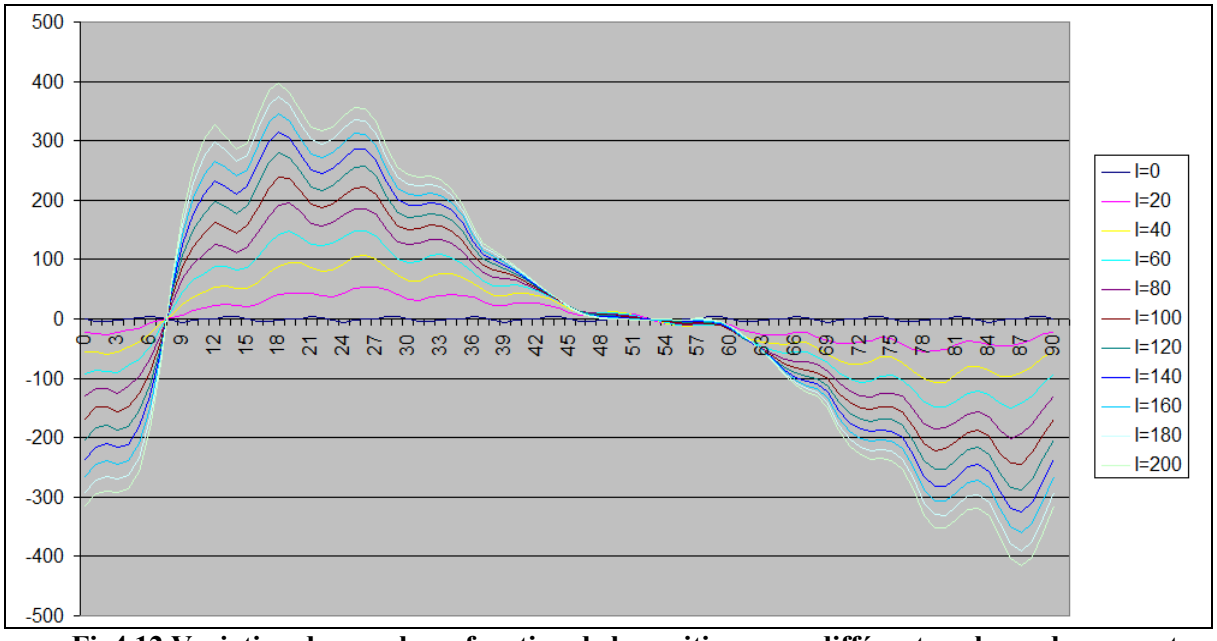


Fig4.12 Variation du couple en fonction de la position pour différente valeurs de courant

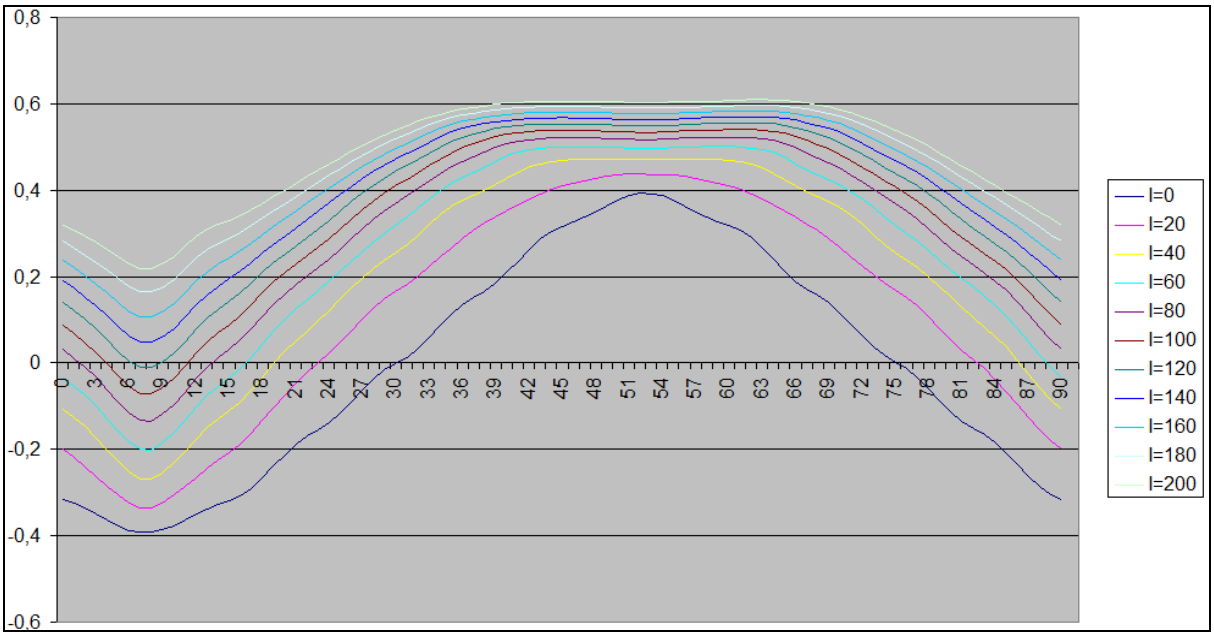


Fig4.13 Variation du flux en fonction de la position pour différents valeurs de courant

Pour I=0A, on peut tirer les angles correspondant aux positions d et q du rotor, le flux s'annule pour un angle de 30° (Position q) et il est maximal pour un angle de 52° (Position d). On trace maintenant pour ces deux positions la dérivée du flux par rapport au courant pour trouver les inductances Ld et Lq.

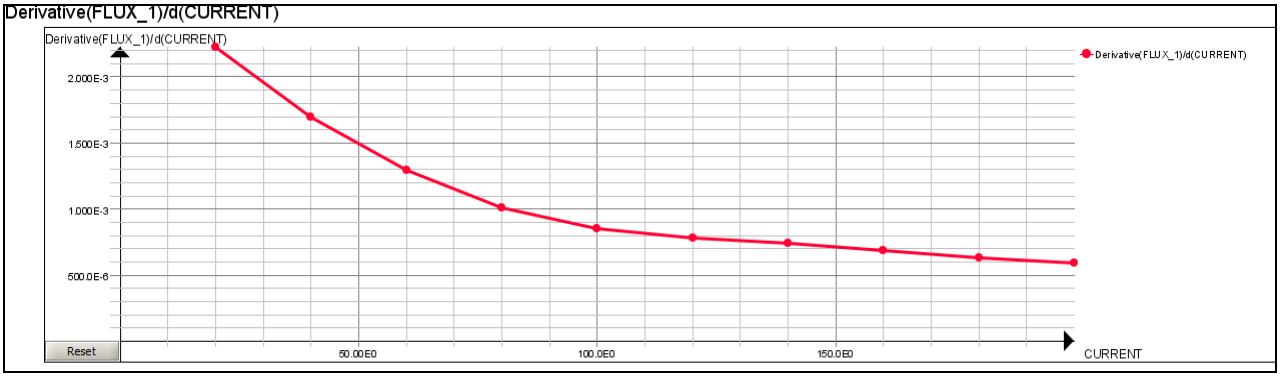


Fig4.14 Evolution de l'inductance dans l'axe d en fonction de courant pour une position donnée

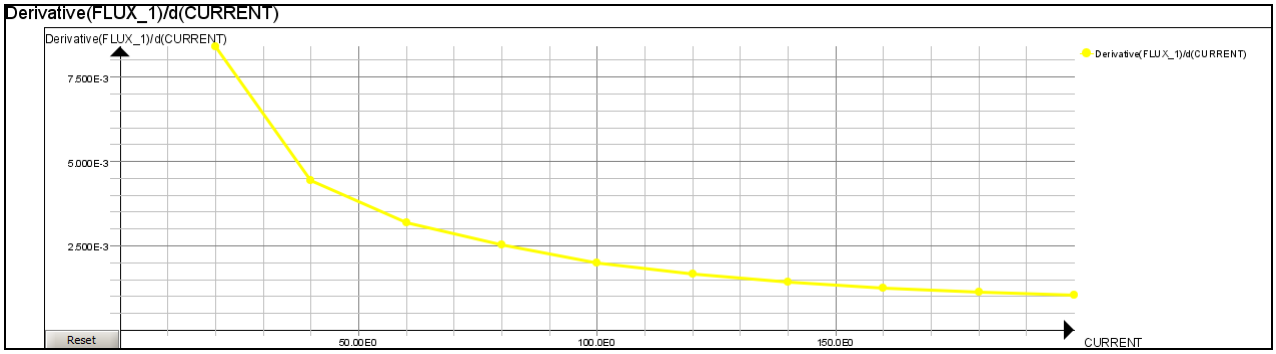


Fig4.15.Evolution de l'inductance dans l'axe q en fonction de courant pour une position donnée

Cas 6 : Essai en court circuit

Dans ce cas, un essai en court circuit est simulé à partir d'un circuit externe contenant des switchs. Il est demandé d'afficher les courbes des tensions et des courants dans les bobines.

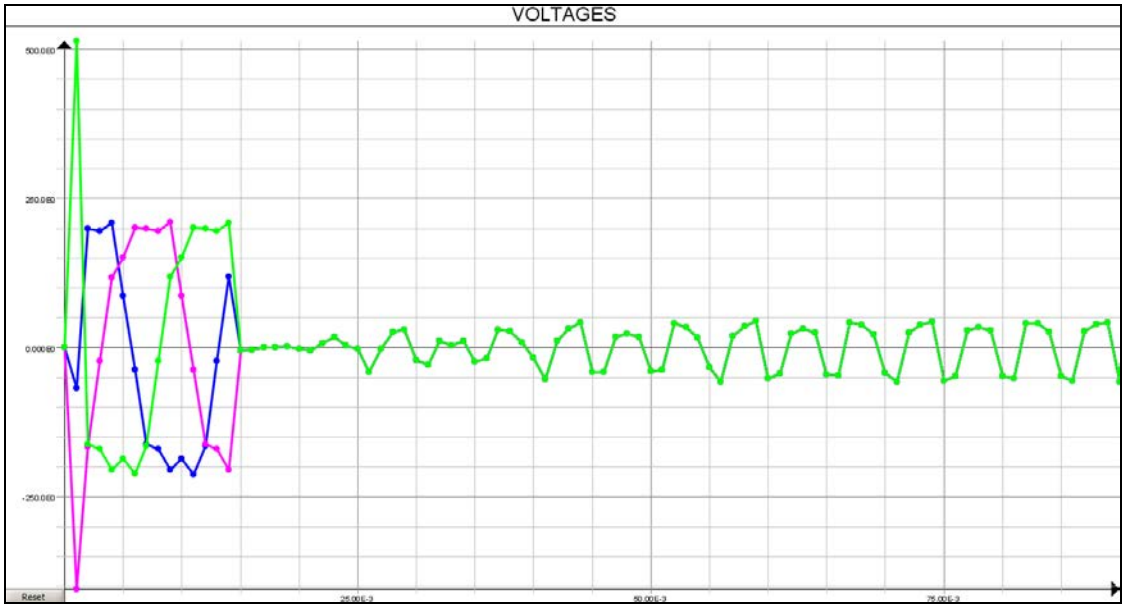


Fig4.16 Tensions de court circuit

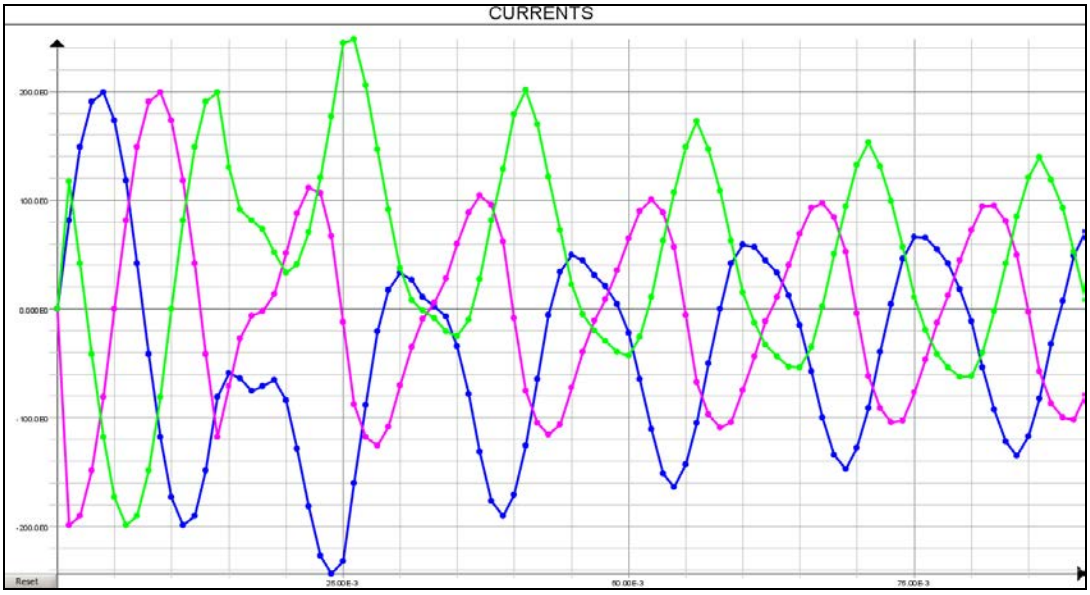


Fig4.17 Courants de court circuit

Les courbes montrent qu'après le court circuit il y a un régime transitoire qui est représenté par des perturbations sur les courants et sur les tensions, ce régime dure plusieurs périodes avant que le fonctionnement se stabilise avec une tension nulle.

Conclusion et perspectives

Le travail exposé dans ce rapport a pour but de présenter les différentes tâches effectuées pendant le stage de fin d'études que j'ai fait à l'entreprise Cedrat SA. Plusieurs scriptes à base du logiciel Python ont été réalisés pour différentes types de machines électriques.
Ce manuscrit commence par une présentation du groupe, sa composition, son chiffre d'affaire et ses activités.
En deuxième partie représente une étude bibliographique sur la modélisation par éléments finis où on montre son principe et ses multiples domaines d'utilisations.
Le troisième chapitre a été dédié à la structure des projets Flux. Les différentes étapes pour la création d'un exemple via des scripts Python dans le logiciel Flux (2D/3D) sont détaillées dans la partie III du manuscrit ainsi que la structure des scriptes et la structure des fichiers de commande (commande qui permet l'exécution des scriptes Python).
La dernière partie de ce manuscrit est consacrée pour montrer les différents exemples réalisés. Un exemple d'une machine synchrone à aimants permanents enterrés est détaillé, avec plusieurs simulations à savoir :

- Simulation à vide
- Simulation en charge
- Simulation en court-circuit..Etc.

Ces simulation nous permettent de calculer les performances de la machines à savoir : le flux de phase, les inductions magnétiques dans différentes régions de la machine, l'inductance, le couple de détente, couple électromagnétique, le rendement…Etc.
Ces exemples vont être ajoutés aux exemples déjà existants sur le superviseur et cela va faciliter la découverte du logiciel pour les nouveaux utilisateurs ainsi que l'accès très rapide et simple à plusieurs cas de simulation de différents dispositifs électriques et permet aussi d'enrichir la base de données de logiciel.
Différents exemples n'ont pas été abordés dans mon sujet et qui manquent actuellement à la liste des exemples que Cedrat dispose notamment au niveau des transformateurs dont la modélisation est très complexe mais très utile aussi vu la nécessité de simuler le comportement du transformateur avant de l'utiliser.
Ce travail m'a été très bénéfique à plusieurs niveaux, d'un coté ça m'a permis de découvrir un nouveau domaine qui est la modélisation par éléments finis ainsi que la métrise d'un des logiciels les plus connus dans ce domaine qui est « Flux 2D/3D » et d'un autre coté, intégrer une équipe comme celle de Cedrat m'a permis de développer mes capacités mentales et aussi de me familiariser avec le monde du travail.

Bibliographies

[1] G. Allaire and A. Craig: Numerical Analysis and Optimization: An Introduction to Mathematical Modelling and Numerical Simulation

[2] K. J. Bathe: Numerical methods in finite element analysis, Prentice-Hall (1976) (ISBN 0136271901)

[3] J. Chaskalovic, Méthode des éléments finis pour les sciences de l'ingénieur (2004), Éd. Lavoisier. (ISBN 2-7430-0708-7)

[4] Cours : Fabrication Assistée par Ordinateur Ecole de Technologie Supérieure – Université du Québec.

[5] Bézier P., Mathématiques et CAO/Courbes et surfaces. Hermès Science Publications, 1987.

Annexes

Script pour la création des bobines non maillées :

```
## Creation of coils
        ## Creation of parameters

   ParameterGeom(name='HCAL',
                 expression='2.0')
   ParameterGeom(name='NES',
                 expression='48.0')
  ParameterGeom(name='HEN',
                 expression='27.0')
   ParameterGeom(name='HACT',
                 expression='40.0')
    ParameterGeom(name='RS',
                 expression='90.6')
   ParameterGeom(name='HFER',
                 expression='3.0')
   ParameterGeom(name='HTB',
                 expression='7.0')
   ParameterGeom(name='HTB1',
                 expression='4.0')
   ParameterGeom(name='OTB',
                 expression='14.0')
   ParameterGeom(name='TETAOD',
                 expression='360/NES')
   ParameterGeom(name='YB',
                 expression='5*TETAOD')
   ParameterGeom(name='X',
                 expression='5.0')
   ParameterGeom(name='BEC',
                 expression='1.0')

## Creation of coordinate system for the coil
      lastInstance = CoordSysCylindrical(name='COIL1',
                    parentCoordSys=Local(coordSys=CoordSys['XYZ1']),
                    origin=['0',
                            '0',
                            '0'],
```

```
                rotationAngles=RotationAngles(angleX='0',
                                              angleY='0',
                                              angleZ='0'),
                visibility=Visibility['VISIBLE'])

## Create the coil point by point
                ComposedCoil(strandedCoil=CoilConductor['COILCONDUCTOR_1'],
            turnNumber='13',  ## Number of turns
            seriesOrParallel=AllInSeries(),
            coilDuplicationBySymmetriesPeriodicities=CoilDuplication(),
            coordSys=CoordSys['COIL1'],  ## The coordinate system
            coilPath=OpenPathCoil(coilPoints=[CoilPoint(coordinates=['RS+BEC+HCAL+(HEN/4)',
                                                     'TETAOD+(TETAOD/2)',
                                                     '0'],
                                              radius='8'),
                                             CoilPoint(coordinates=['RS+BEC+HCAL+(HEN/4)',
                                                     'TETAOD+(TETAOD/2)',
                                                     'HACT+HFER'],
                                              radius='8'),
                                              CoilPoint(coordinates=['RS+BEC+HCAL+(HEN/4)+2',
                                                     'TETAOD+(TETAOD/2)+(YB/2)',
                                                     'HACT+HFER+X'],
                                              radius='8'),
                                              CoilPoint(coordinates=['RS+BEC+HCAL+(HEN/4)+2',
                                                     'TETAOD+(TETAOD/2)+(YB/2)',
                                                     'HACT+HFER+X+HTB'],
                                              radius='5'),
                                              CoilPoint(coordinates=['RS+BEC+HCAL+(HEN/4)+(OTB/2)+2',
                                                     'TETAOD+(TETAOD/2)+(YB/2)',
                                                     'HACT+HFER+X+HTB+HTB1'],
                                              radius='5'),
                                              CoilPoint(coordinates=['RS+BEC+HCAL+(HEN/4)+(OTB/2)+4',
                                                     'TETAOD+(TETAOD/2)+(YB/2)',
                                                     'HACT+HFER+X+HTB'],
                                              radius='5'),
                                              CoilPoint(coordinates=['RS+BEC+HCAL+(HEN/4)+(OTB/2)+4',
                                                     'TETAOD+(TETAOD/2)+(YB/2)',
                                                     'HACT+HFER+X'],
                                              radius='5'),
                                              CoilPoint(coordinates=['RS+BEC+HCAL+(HEN/4)+(HEN/2)',
                                                     'TETAOD+(TETAOD/2)+(YB)',
                                                     'HACT+HFER'],
```

```
                    radius='8'),
                CoilPoint(coordinates=['RS+BEC+HCAL+(HEN/4)+(HEN/2)',
                    'TETAOD+(TETAOD/2)+(YB)',
                    '0'],
                    radius='8')]),
section=DiscCoilSection(sectionRadius='3'),
mechanicalSet=MechanicalSet['STATOR'],  ## Choice of the mechanical set
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 :

Ce manuscrit a pour but de montrer le travail fait pendant le stage de fin d'étude effectué à l'entreprise Cedrat SA, une présentation de cette dernière est faite dans le premier chapitre suivie par une partie bibliographique sur la modélisation par éléments finis, ensuite les différentes étapes effectuées pour la création des exemples Flux sont détaillées dans le troisième chapitre et enfin le dernier chapitre contient les projets réalisés pendant le stage dont on a choisi un pour le bien détaillé dans le rapport.

MOTS-CLES :

Cedrat, modélisation, simulation, éléments finis, machine, performance, python,

ABSTRACT :

In this manuscript we show the work done during the internship made at the company Cedrat SA, we start by a presentation of the company followed by a literature review of the finite element method, after that, the different steps taken to create the exampels Flux are detailed in the third chapter and finally, the last chapter contains the projects achieved during the internship and one of them is choosed to be detailed in the report.

KEYWORDS :

Cedrat, modelisation, simulation, finite element, machine, performance Python