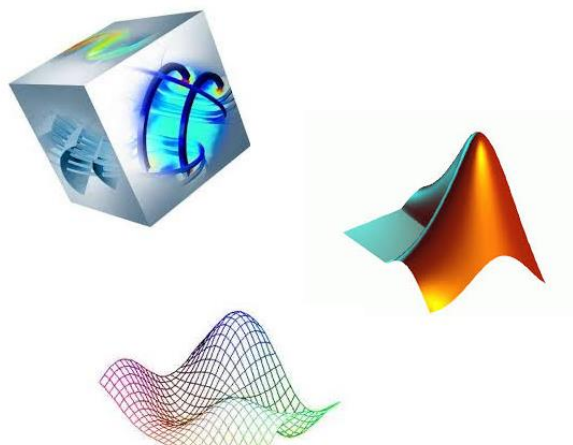


P 1 3 A 0 8

NOTE D'APPLICATION DE PROJET

« MODELISATION, SIMULATION ET OPTIMISATION D'UN MOTEUR ROUE »

INTERFACE COMSOL/MATLAB ET
ALGORITHME GENETIQUE



MOHAMED OUAMEUR
LE 01/01/2014

TABLES DES MATIERES

Introduction.....	5
Le logiciel COMSOL.....	6
1.1. Introduction	6
1.2. Introduction à l'interface utilisateur de COMSOL 3.5a	8
Le logiciel MATLAB.....	9
2.1. Introduction	9
2.2. Présentation de l'interface du logiciel.....	10
Intégrer COMSOL MULTIPHISICS avec MATLAB.....	15
3.1 Présentation.....	15
3.2. La combinaison de COMSOL Multiphysics et MATLAB.....	15
3.3 Comment combiner COMSOL et MATLAB	16
Algorithme génétique	18
4.1. Présentation	18
4.2. Principe de fonctionnement.....	20
Conclusion.....	22
Bibliographie.....	22

TABLES DES ILLUSTRATIONS ET TABLEAUX

<u>Figure 1. Logo de Logiciel COMSOL.....</u>	6
<u>Figure 2. Les modules complémentaires pour COMSOL Multiphysics.....</u>	7
<u>Figure 3. Les différents éléments de l'interface utilisateur de COMSOL 3.5a.....</u>	8
<u>Figure 4. Logo de logiciel MATLAB.....</u>	9
<u>Figure 5. Les différents éléments de l'interface utilisateur de MATLAB.....</u>	10
<u>Figure 6. Fenêtre de commande de l'interface utilisateur de MATLAB.....</u>	11
<u>Figure 7. Espace de travail de l'interface utilisateur de MATLAB.....</u>	12
<u>Figure 8. Répertoire en cours de l'interface utilisateur de MATLAB.....</u>	13
<u>Figure 9. Historique des commandes de l'interface utilisateur de MATLAB.....</u>	14
<u>Figure 10. Intégration COMSOL Multiphysics avec MATLAB.....</u>	15
<u>Figure 11. Interface utilisateur COMSOL Multiphysics.....</u>	16
<u>Figure 12. Comment combiner COMSOL avec MATLAB.....</u>	17
<u>Figure 13. La forme de l'algorithme génétique générique.....</u>	19
<u>Figure 14. Le principe de fonctionnement d'un algorithme génétique.....</u>	20

INTRODUCTION

Dans le cadre du notre projet de fin d'études, sous le titre de « modélisation, simulation et optimisation d'un moteur roue »

On a eu l'occasion d'utiliser un logiciel de modélisation informatique cette méthode est méthode principale aujourd'hui pour réaliser la conception d'une machine électrique. Dans notre cas, la modélisation consiste à produire géométrie optimale en régime statique permettant d'obtenir le meilleur rendement possible de la puissance en régime dynamique. La simulation permettant d'étudier les caractéristiques données que de simuler le fonctionnement de moteur électrique réel et montrer sa performance. Par opposition aux moyennes traditionnelles, l'utilisation de COMSOL seulement pour optimiser la géométrie rend cette partie plus complexe, et avec l'existence d'une interface COMSOL/MATLAB, nous avons utilisés le logiciel MATLAB pour la partie d'optimisation.

Ce document a pour l'objectif d'écrire les deux logiciels COMSOL et MATLAB, ainsi l'interface existant entre les deux, et aussi le principe de l'algorithme génétique utilisé pour optimiser la géométrie.

1.1. Introduction

COMSOL Multiphysics est un logiciel de simulation numérique basé sur la méthode des éléments finis. Ce logiciel permet de simuler de nombreuses physiques et applications en ingénierie, et tout particulièrement les phénomènes couplés ou simulation multi-physiques.



Figure 2. Logo de Logiciel COMSOL

Il a été développé par des étudiants de Germund Dahlquist (1925-2005) à la Royal Institute of Technology à Stockholm

Il possède aussi l'interface MATLAB, pour faire une association entre les deux logiciels COMSOL/MATLAB.

C'est logiciel multiplateforme: Windows, Mac, GNU-Linux, il contient la plupart des équations aux dérivées partielles (EDP), soit sous forme différentielle, soit sous formulation faible. Les couplages avec des équations aux dérivées ordinaires (EDO) et des équations algèbro-différentiels (EAD) sont également possibles. Et il utilise une interface graphique, comme on peut utiliser un peu de programmation directe.

L'utilisateur définit ses couplages ou sélectionne les interfaces prédéfinies. Les différentes étapes du processus de modélisation :

- ✓ définir la géométrie,
- ✓ les propriétés matériaux,
- ✓ le maillage,
- ✓ choisir la ou les physiques,
- ✓ résoudre et afficher les résultats.

Toutes ces étapes sont intégrées dans une seule interface. Des modules d'applications optionnels offre des interfaces spécialisées notamment en mécanique linéaire et non-linéaire, acoustique, écoulement, transfert de chaleur, génie chimique, géo-physique, électromagnétisme basse et haute fréquence, corrosion, plasma, suivi de particules, optimisation, MEMS, ainsi qu'avec les logiciels de CAO et Matlab.

Liste des modules complémentaires pour COMSOL Multiphysics :

- ☐ AC/DC Module
- ☐ Acoustics Module
- ☐ Batteries & Fuel Cells Module
- ☐ CAD Import Module
- ☐ CFD Module
- ☐ Chemical Reaction Engineering Module
- ☐ Corrosion Module
- ☐ ECAD Import Module
- ☐ Electrochemistry Module
- ☐ Electrodeposition Module
- ☐ Fatigue Module
- ☐ File Import for CATIA V5
- ☐ Geomechanics Module
- ☐ Heat Transfer Module
- ☐ LiveLink for AutoCAD
- ☐ LiveLink for Creo Parametric
- ☐ LiveLink for Excel
- ☐ LiveLink for Inventor
- ☐ LiveLink for MATLAB
- ☐ LiveLink for Pro/ENGINEER
- ☐ LiveLink for Solid Edge
- ☐ LiveLink for SolidWorks
- ☐ LiveLink for SpaceClaim
- ☐ Material Library
- ☐ MEMS Module
- ☐ Microfluidics Module
- ☐ Molecular Flow Module
- ☐ Multibody Dynamics Module
- ☐ Nonlinear Structural Materials Module
- ☐ Optimization Module
- ☐ Particle Tracing Module
- ☐ Pipe Flow Module
- ☐ Plasma Module
- ☐ RF Module
- ☐ Semiconductor Module
- ☐ Structural Mechanics Module
- ☐ Subsurface Flow Module
- ☐ Wave Optics Module

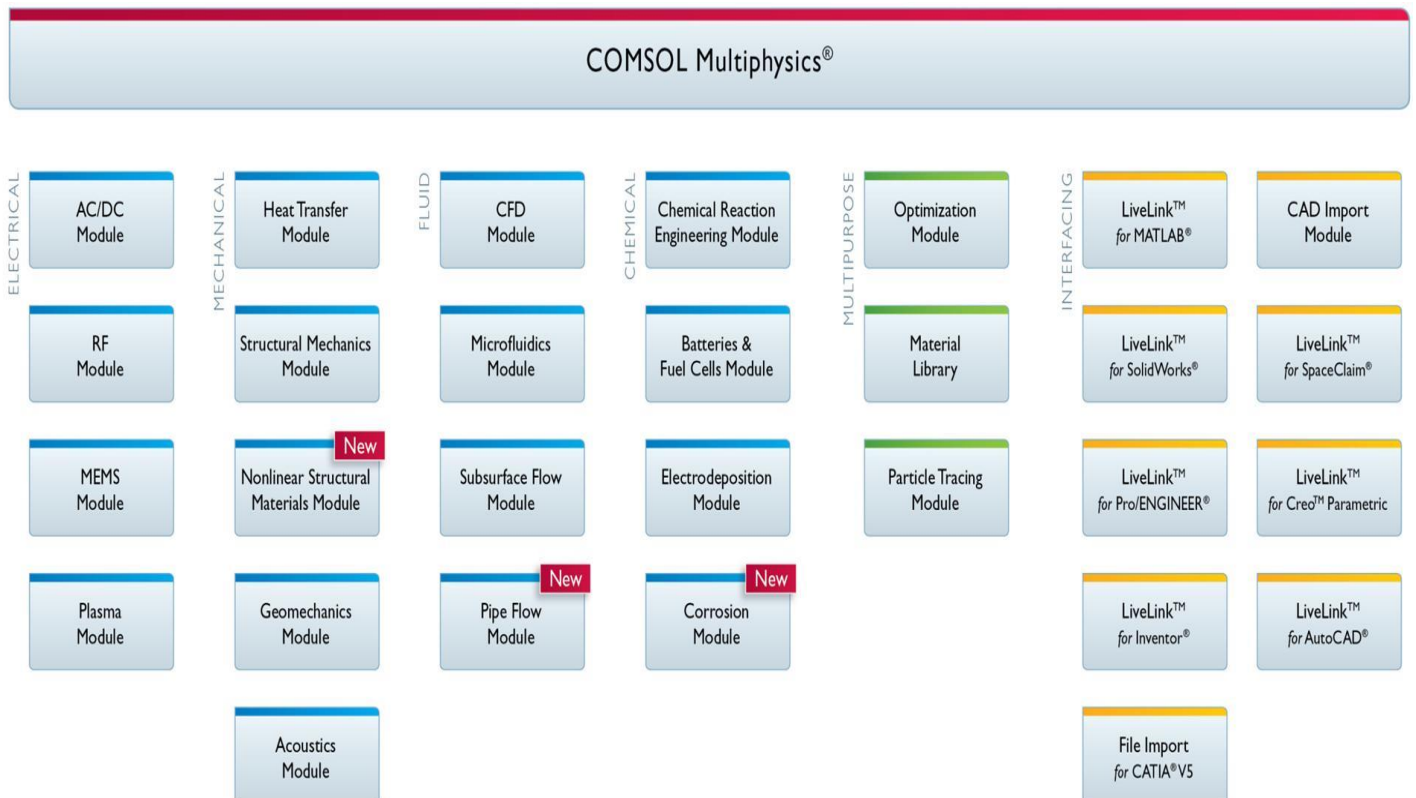


Figure 2. Les modules complémentaires pour COMSOL Multiphysics

1.2. Introduction à l'interface utilisateur de COMSOL 3.5a

L'interface de COMSOL version 3.5a constituée de quatre grandes parties. On retrouve comme la première partie c'est la barre de menu (en noir) où il est possible de définir des variables et autres paramètres du problème (Parameters), le modèle (Model) où se retrouvent la géométrie (Geometry), les propriétés des matériaux formant la géométrie, le ou les modèles de physique s'appliquant au problème étudié et les paramètres de maillage (Mesh). Le Model Builder comporte aussi le type de problèmes et les paramètres de solveur (Study) et les options d'affichage et de post traitement des données (Results). La colonne (en bleu) à droit comporte des options relatives à ce qui est sélectionné dans la barre de menu, par exemple, les dimensions d'un objet qui vient d'être créé dans Geometry. C'est aussi à cet endroit que les paramètres initiaux de la simulation et les modèles physiques nécessaires sont choisis.

La troisième partie (en rouge) est la fenêtre d'affichage graphique (Graphics) permet de visualiser la géométrie, le maillage ou les résultats. Juste à droit de cette fenêtre se retrouve diverses options permettant de changer le grossissement de l'affichage, l'orientation d'un objet tridimensionnel, etc. Les options permettant de sélectionner des objets, des domaines, des frontières ou des points se retrouvent aussi à droit de cette fenêtre.

Finalement, directement au-dessous de la fenêtre d'affichage graphique, il y a une fenêtre (en vert) permettant de visualiser les messages d'erreurs, le progrès des simulations, la liste des opérations effectuées lors du calcul de la solution ainsi que des résultats numériques calculés une fois la simulation terminée.

Les différents éléments de l'interface utilisateur de COMSOL 3.5a sont présentés à la figure suivante :



Figure 3. Les différents éléments de l'interface utilisateur de COMSOL 3.5a

2.1. Introduction

MATLAB est un logiciel de haut niveau et un environnement interactif pour les calculs numériques, la visualisation et la programmation. Grâce à MATLAB, nous pouvons analyser des données, développer des algorithmes et créer des modèles et des applications. Le langage, les outils et les fonctions mathématiques intégrées nous permettent d'explorer diverses approches et d'arriver à une solution plus rapidement qu'en utilisant des feuilles de calcul ou des langages de programmation traditionnels.



Figure 4. Logo de logiciel MATLAB

Nous pouvons aussi utiliser MATLAB pour une vaste gamme d'applications, en particulier le traitement du signal et des communications, le traitement des images et des vidéos, les systèmes de contrôle, le test et les mesures, la finance et la biologie. MATLAB est le langage de calcul scientifique pour plus d'un million d'ingénieurs et de scientifiques dans l'industrie et le monde académique.

Un script Matlab est composé d'une suite d'instructions, toutes séparées par une virgule (ou de manière équivalente, un passage à la ligne) ou un point-virgule. La différence entre ces deux types de séparation est liée à l'affichage ou non du résultat à l'écran (seulement effectué dans le premier cas). Comme tout langage, Matlab possède aussi un certain nombre d'instructions syntaxiques (boucles simples, conditionnelles, etc...) et de commandes élémentaires (lecture, écriture, etc...).

Dès que le calcul à effectuer implique un enchaînement de commandes un peu compliqué, il vaut mieux écrire ces dernières dans un fichier. Par convention un fichier contenant des commandes Matlab porte un nom avec le suffixe « .m » et s'appelle pour cette raison un M-file ou encore script. On utilisera toujours l'éditeur intégré au logiciel qui se lance à partir de la fenêtre de commande en cliquant sur les icônes new M-file ou open-file dans la barre de menu. Une fois le fichier enregistré sous un nom valide, on peut exécuter les commandes qu'il contient en tapant son nom - sans le suffixe .m - dans la fenêtre de commande. Si vous avez ouvert l'éditeur comme indiqué, à partir de la fenêtre de commande, les M-file seront créés dans le répertoire courant, accessible depuis cette fenêtre, et vous n'aurez pas de problème d'accès. Si vous voulez exécuter des scripts qui se trouvent ailleurs dans l'arborescence des fichiers, vous aurez éventuellement à modifier le Path en cliquant sur le menu file => SetPath ou bien en changeant de répertoire de travail (cliquer sur l'onglet current directory).

2.2. Présentation de l'interface du logiciel

L'interface de Matlab est divisée en quatre parties principales soit :

- ✓ La fenêtre de commandes (*Command Window*) en rouge;
- ✓ L'espace de travail (*Workspace*) en vert;
- ✓ Le répertoire en cours (*Current Directory*) en noir;
- ✓ L'historique des commandes (*Command History*) en bleu.

Ces parties sont représentées sur l'image ci-dessous.

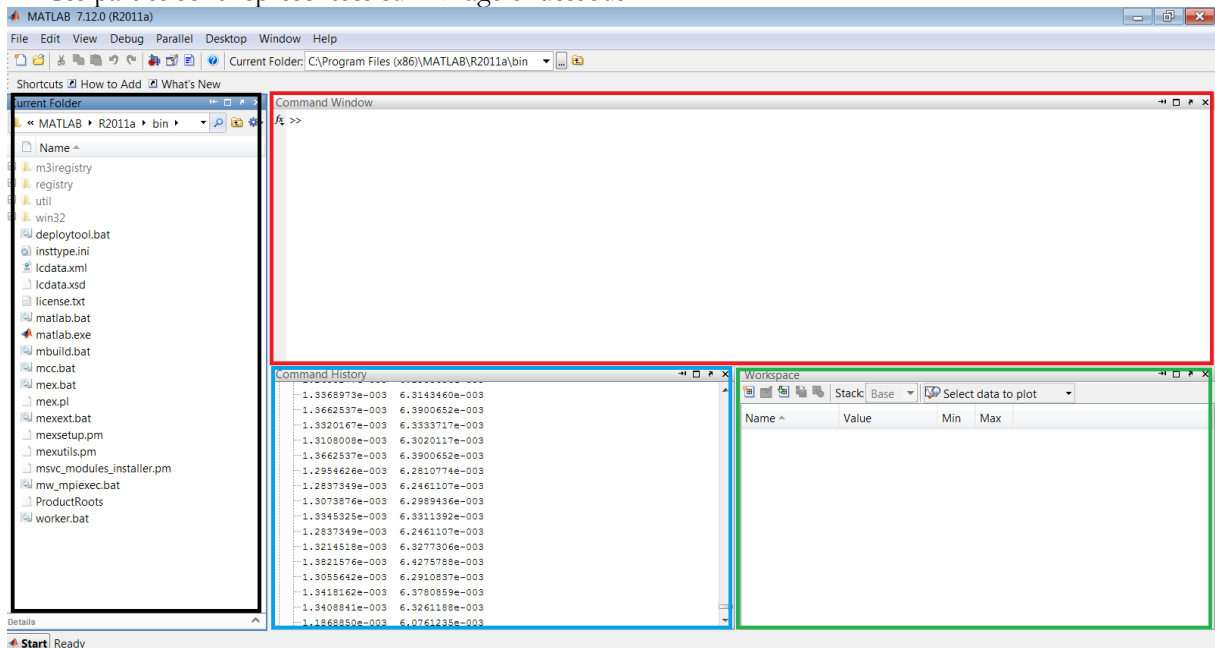



Figure 5. Les différents éléments de l'interface utilisateur de MATLAB

Nous allons maintenant voir l'utilité ainsi que les principales fonctionnalités de chacune de ces fenêtres d'affichage.

 **Fenêtre de commandes** : C'est la fenêtre dans laquelle on dicte les opérations à effectuer un peu comme l'ancien DOS où l'on exécutait des programmes ou des opérations à partir de lignes commandes. En utilisant la syntaxe appropriée, on peut se servir de la fenêtre de commandes pour effectuer des opérations de calcul ou pour appeler des fonctions provenant des bibliothèques de Matlab ou des fonctions que l'on a nous-même construites comme on peut voir sur l'image ci-dessous. On peut également demander de l'aide en utilisant la commande *help* soit seul ou suivi du nom de la fonction dont on veut de l'information (exemple : *help mean*) et on peut accéder à certaines fonctionnalités de Matlab telles que Simulink et GUIDE. En bref, la fenêtre de commandes c'est ce qui fait le lien entre l'utilisateur et le programme.



```

>>
>> % Calcule la valeur de a modulo n en prenant pour systeme de residus
% 1, ... , n au lieu de 0, ... , n-1.
%
% appel : [r,q] = modulo(a,n)
%
% Arguments de sortie :
% r : le residu
% q : le quotient

q = floor(a./n);
r = a - n*q;


% si le reste de la division entiere vaut 0, le residu vaut par convention n
if r == 0, r = n; end
>> % Calcule la valeur de a modulo n en prenant pour systeme de residus
% 1, ... , n au lieu de 0, ... , n-1.
%
% appel : [r,q] = modulo(a,n)
%
% Arguments de sortie :
% r : le residu
% q : le quotient

q = floor(a./n);
r = a - n*q;

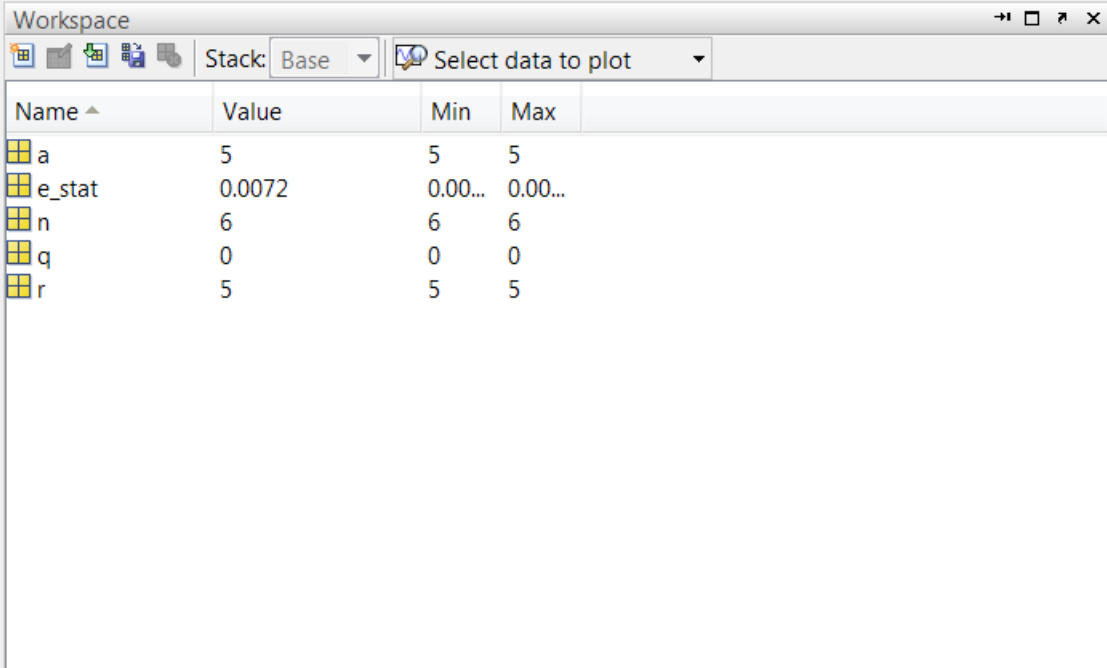
% si le reste de la division entiere vaut 0, le residu vaut par convention n
if r == 0, r = n; end
>>

```

Figure 6. Fenêtre de commande de l'interface utilisateur de MATLAB

 **Espace de travail** : On trouve dans cette fenêtre toutes les variables qui ont été créés et qui sont utilisables dans la fenêtre de commandes pour des opérations de calcul et pour faire de la visualisation. Ces variables sont regroupées sous forme de classes qui indiquent la nature de la variable. Il existe différents types de variables qui peuvent être définies. Dans ce cours, vous aurez à utiliser seulement des variables de type *double* (i.e. une variable qui a une précision double par rapport à une variable de type *float* mais qui occupe deux fois plus de mémoire soit 64 bits). D'autre part, on retrouve également la valeur de la variable sous la colonne *value* si sa matrice contient seulement un élément, sinon on retrouve inscrite dans cette colonne la taille de la matrice représentative de la variable.

La fenêtre d'affichage de l'espace de travail contient une barre d'outils vous permettant d'effectuer plusieurs opérations sur vos données très rapidement. Les principales opérations sont montrées sur l'image ci-dessous.




The screenshot shows the MATLAB Workspace window. At the top, there are icons for workspace, command window, and other tools. Below the icons, there is a 'Stack' dropdown menu set to 'Base' and a 'Select data to plot' button. The main area is a table with the following data:

Name	Value	Min	Max
a	5	5	5
e_stat	0.0072	0.00...	0.00...
n	6	6	6
q	0	0	0
r	5	5	5

Figure 7. Espace de travail de l'interface utilisateur de MATLAB

Dans un premier temps, on peut créer de nouvelles variables que l'on peut définir par la suite à l'aide de la fenêtre de commandes. On peut également ouvrir l'éditeur de matrices qui nous permet de visualiser les données sous forme de tableau. Des données provenant de Microsoft Excel peuvent être facilement importées dans l'éditeur de matrices en utilisant les fonctionnalités de Windows copier et coller. On peut aussi effacer ou ajouter manuellement des données dans l'éditeur. D'autre part, on peut charger ou sauvegarder des données à partir de la barre d'outils de la fenêtre d'affichage de l'espace de travail. Ces données sont compilées dans un fichier ayant une extension *.mat et peuvent être placées dans le répertoire de votre choix. Par contre, pour charger ces données, vous devez absolument spécifier le répertoire dans lequel elles se trouvent. Le répertoire en cours peut être spécifié dans la barre d'outils principale de Matlab à l'endroit marqué *Current Directory*.

 **Répertoire en cours** : Le répertoire en cours permet de visualiser des fichiers se trouvant dans le même répertoire. Tous les fichiers peu importe leur extension sont affichées. On peut ouvrir un fichier en cliquant directement dessus. Les principaux fichiers qui peuvent être lus par Matlab sont les fichiers de fonctions ou de programmes (*.m), les fichiers de données (*.mat) et les fichiers des graphiques (*.fig). De plus, dans la barre d'outils de la fenêtre d'affichage du répertoire en cours, on peut créer des rapports qui nous documentent sur les fichiers se trouvant dans le répertoire. Parmi les plus importants, il y a le *M-Lint Code Check Report* qui va répertorier les erreurs et les avertissements dans le code d'un programme ou d'une fonction. Il y a également le *File Comparison Report* qui repère les différences au niveau du code entre deux documents.

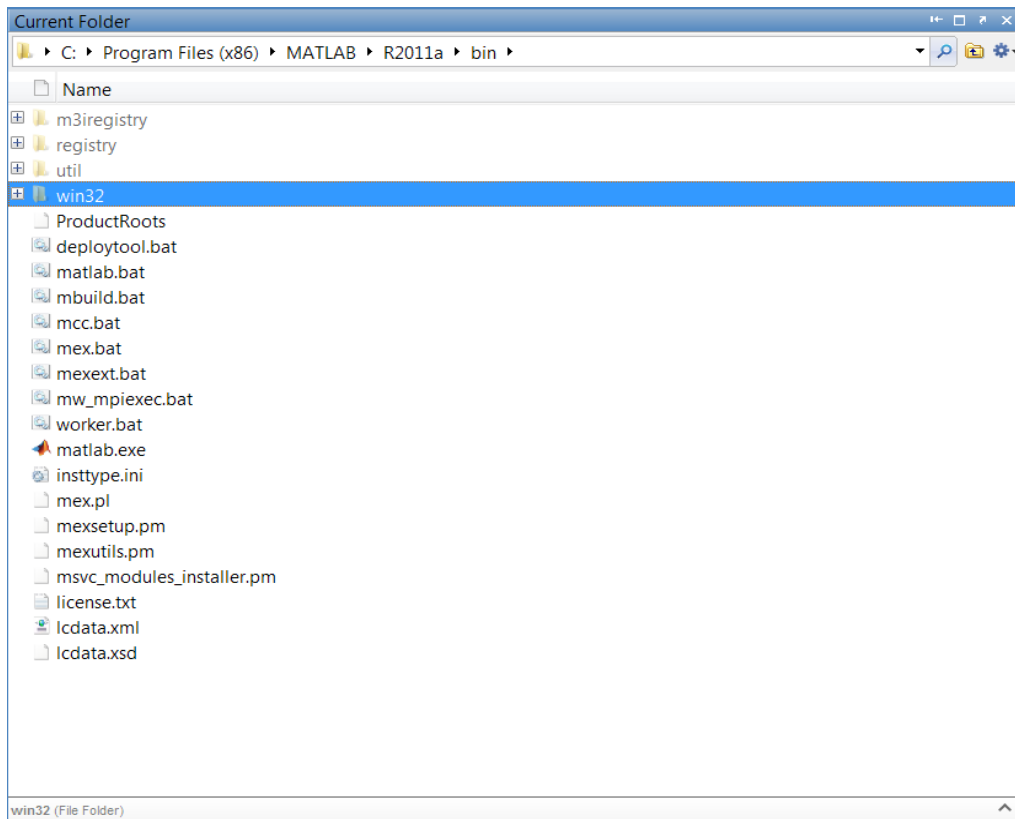

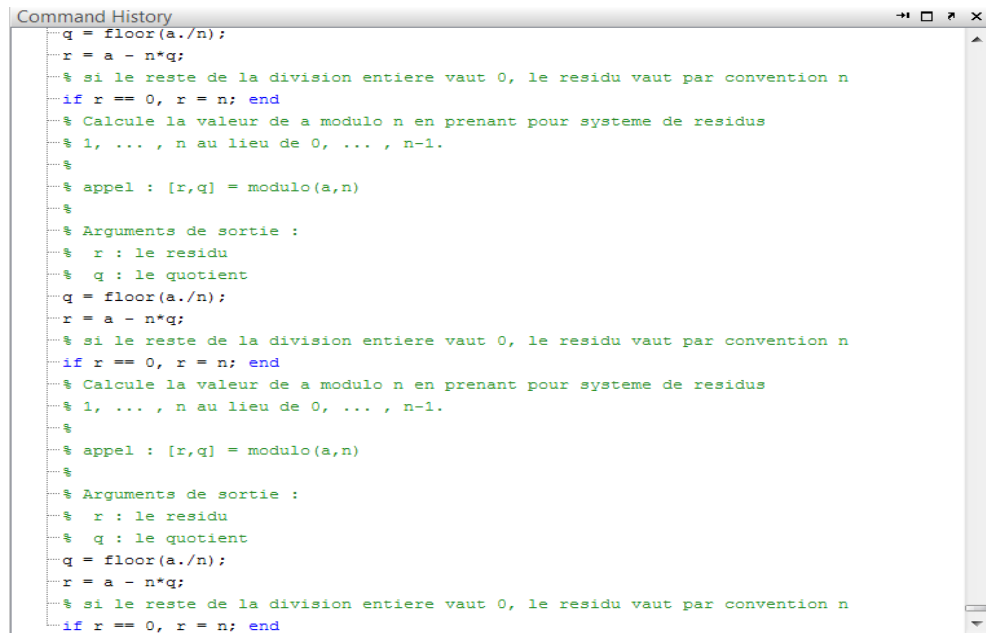


Figure 8. Répertoire en cours de l'interface utilisateur de MATLAB

 **Historique des commandes** : Cette fenêtre contient les commandes utilisées précédemment dans la fenêtre de commandes. L'historique des commandes peut compiler plusieurs centaines de lignes de commandes. À partir de cette fenêtre, on peut copier et coller des lignes de commandes pour les réutiliser dans la fenêtre de commandes. D'autre part, à partir de la fenêtre de commandes, on peut retrouver des lignes de commandes utilisées précédemment en appuyant sur la flèche du haut se trouvant sur votre clavier. L'image ci-dessous montre cette fenêtre d'affichage.

The image shows a screenshot of the MATLAB Command History window. The window has a title bar with standard OS controls (minimize, maximize, close) and a scroll bar on the right. The text inside is MATLAB code for a function named 'modulo'. The code is as follows:

```
q = floor(a./n);  
r = a - n*q;  
% si le reste de la division entiere vaut 0, le residu vaut par convention n  
if r == 0, r = n; end  
% Calcule la valeur de a modulo n en prenant pour systeme de residus  
% 1, ... , n au lieu de 0, ... , n-1.  
%  
% appel : [r,q] = modulo(a,n)  
%  
% Arguments de sortie :  
% r : le residu  
% q : le quotient  
q = floor(a./n);  
r = a - n*q;  
% si le reste de la division entiere vaut 0, le residu vaut par convention n  
if r == 0, r = n; end
```

Figure 9. Historique des commandes de l'interface utilisateur de MATLAB

3.1. Présentation

Intégrez COMSOL Multiphysics avec MATLAB pour étendre votre modélisation avec la programmation de scripts dans l'environnement MATLAB. MATLAB nous permet d'utiliser la pleine puissance de ce logiciel et ses boîtes à outils de prétraitement, de manipulation de modèle, et le post-traitement :

- ✓ D'améliorer aussi notre maison en code MATLAB avec de puissantes simulations multiphysiques
- ✓ Baser notre modélisation de la géométrie sur les données probabiliste ou l'image
- ✓ Effectuer une analyse statistique arbitraire sur les résultats de simulation
- ✓ Exporter des modèles COMSOL sur l'état de l'espace sous forme de matrice pour l'intégration dans les systèmes de contrôle
- ✓ Appeler des fonctions MATLAB de la COMSOL Desktop

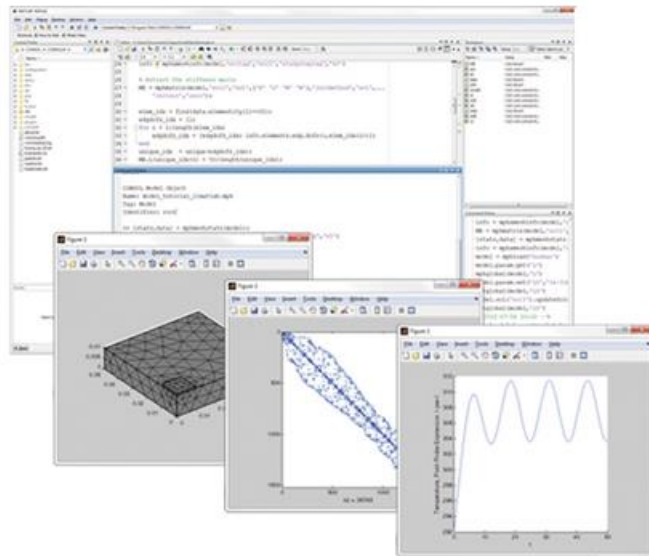


Figure 10. Intégration COMSOL Multiphysics avec MATLAB

3.2. La combinaison de COMSOL Multiphysics et MATLAB

Nous pouvons combiner les forces de COMSOL Multiphysics et MATLAB en temps réel pour résoudre des simulations d'ingénierie, et c'est la solution adoptée dans la partie d'optimisation de notre projet.

Pourquoi utiliser COMSOL Multiphysics et MATLAB ensemble

MATLAB nous fournit un environnement de développement interactif, où nous pouvons utiliser un langage de haut niveau ainsi que des fonctionnalités intégrées pour l'analyse de données et la visualisation, le calcul numérique, le développement d'algorithmes, et le développement d'applications. COMSOL Multiphysics, d'autre part, vous permet de personnaliser et de combiner un nombre quelconque de la physique, de la façon dont nous voulons. Avec ouverte technique informatique environnement MATLAB et multiphysique la plate-forme de modélisation de COMSOL, vous pourriez prendre sur presque n'importe quel défi scientifique.

La plupart des problèmes d'ingénierie tournent autour de trouver des paramètres de conception optimales ou des conditions de fonctionnement d'un appareil. Le dispositif lui-même les fonctions sur la base de certains principes de la physique (ou même physique multiple). Prenons un exemple. Dites, nous souhaitons concevoir un chauffage électrique qui a une couche métallique mince déposée sur le dessus du verre. Les travaux de chauffage basés sur le principe de chauffage par effet Joule, dans lequel un courant passant à travers la couche métallique produit de la chaleur dans la couche métallique, qui à son tour crée un profil de température variant dans l'espace à la fois dans les couches de verre et de métal. En tant qu'ingénieur de conception, nous voudrions peut-être faire en sorte que le stress généré par dilatation thermique à l'interface de verre et de métal n'est pas trop élevée pour provoquer une défaillance mécanique de l'appareil. Nous savons aussi que d'un stand point pratique, nous pouvons contrôler l'épaisseur des couches de verre et métalliques dans les limites de dimensions prévues par le cahier des charges de fabrication.

3.3. Comment combiner COMSOL et MATLAB

Nous pouvons configurer la simulation multiphysique qui implique Joule et la dilatation thermique dans COMSOL Multiphysics, et effectuer la conception des expériences (DOE) sur le modèle en utilisant la fonctionnalité qui est disponible dans le cadre de la Statistics Toolbox dans MATLAB.

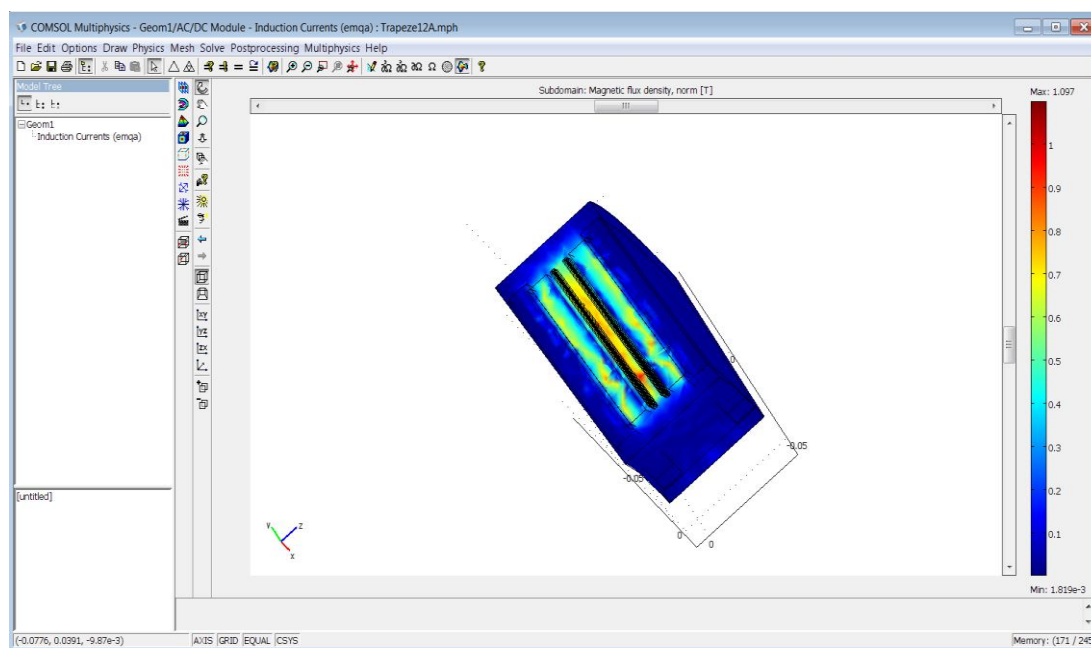


Figure 11. Interface utilisateur COMSOL Multiphysics

Vous pouvez certainement écrire un script MATLAB qui crée réponse modèles de surface (une technique DOE) pour votre modèle de COMSOL. Ou si vous êtes intéressé par la répétition des interactions faciles avec votre modèle, ou si vous souhaitez partager votre travail avec d'autres, vous pourriez écrire une coutume MATLAB comme celle ci-dessous. L'application présentée ici peut résoudre le modèle COMSOL récursive pour un ensemble de points de conception, et de créer une surface de réponse que vous ou un décideur pouvez ensuite utiliser pour prendre une décision d'ingénierie instruits.

Pour rendre la simulation plus réaliste, vous pouvez même utiliser des données expérimentales qui décrivent la variation des propriétés des matériaux en fonction de la température. Ces données numériques peuvent être prétraitées en utilisant les fonctions de base de MATLAB ou via des fonctions / applications spécialisées disponibles dans les boîtes à outils tels

que l'ajustement de courbe Toolbox, et vous pouvez facilement appeler ces fonctions à partir du modèle de COMSOL.

Et vous pouvez lancés cette interface en lançant la connexion avec MATLAB à partir de COMSOL, en appuyant sur : File -> Client/Server/MATLAB -> Connect To MATLAB...

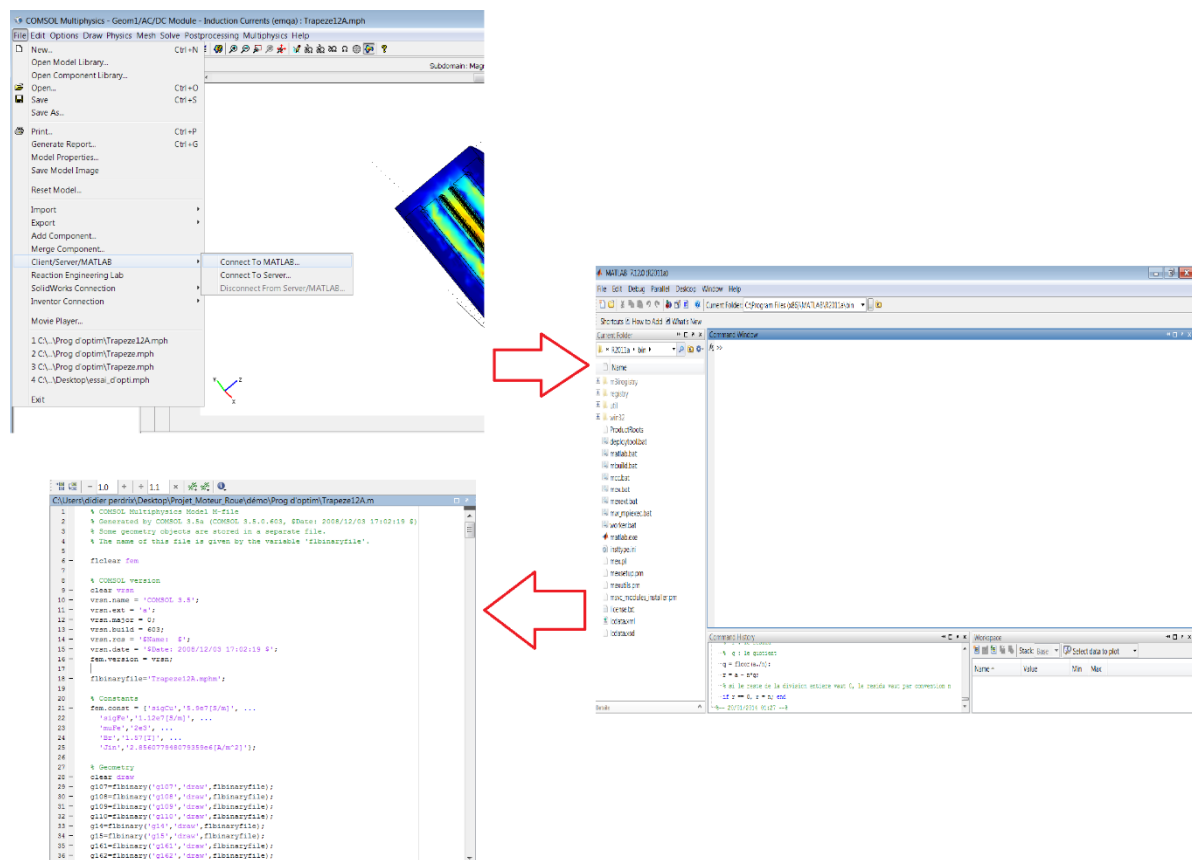


Figure 12. Comment combiner COMSOL avec MATLAB

Maintenant, vous avez une fonction MATLAB décrivant le comportement du matériau qui est appelée par un modèle COMSOL Multiphysics, qui à son tour est appelée par un design MATLAB d'expériences application.

4.1. Présentation

Les algorithmes génétiques, initiés dans les années 1970 par John Holland, sont des algorithmes d'optimisation s'appuyant sur des techniques dérivées de la génétique et des mécanismes d'évolution de la nature : croisement, mutation, sélection.

Ces algorithmes génétiques peuvent être particulièrement utiles dans les domaines suivants:

- ✓ Optimisation : optimisation de fonctions, planification.

Et c'est le cas dans notre projet « modélisation, simulation et optimisation d'un moteur roue »

- ✓ Apprentissage : classification, prédiction, robotique.
- ✓ Programmation automatique : programmes LISP, automates cellulaires.
- ✓ Etude du vivant, du monde réel : marchés économiques, comportements sociaux, systèmes immunitaires.

Un algorithme génétique à la forme suivante :

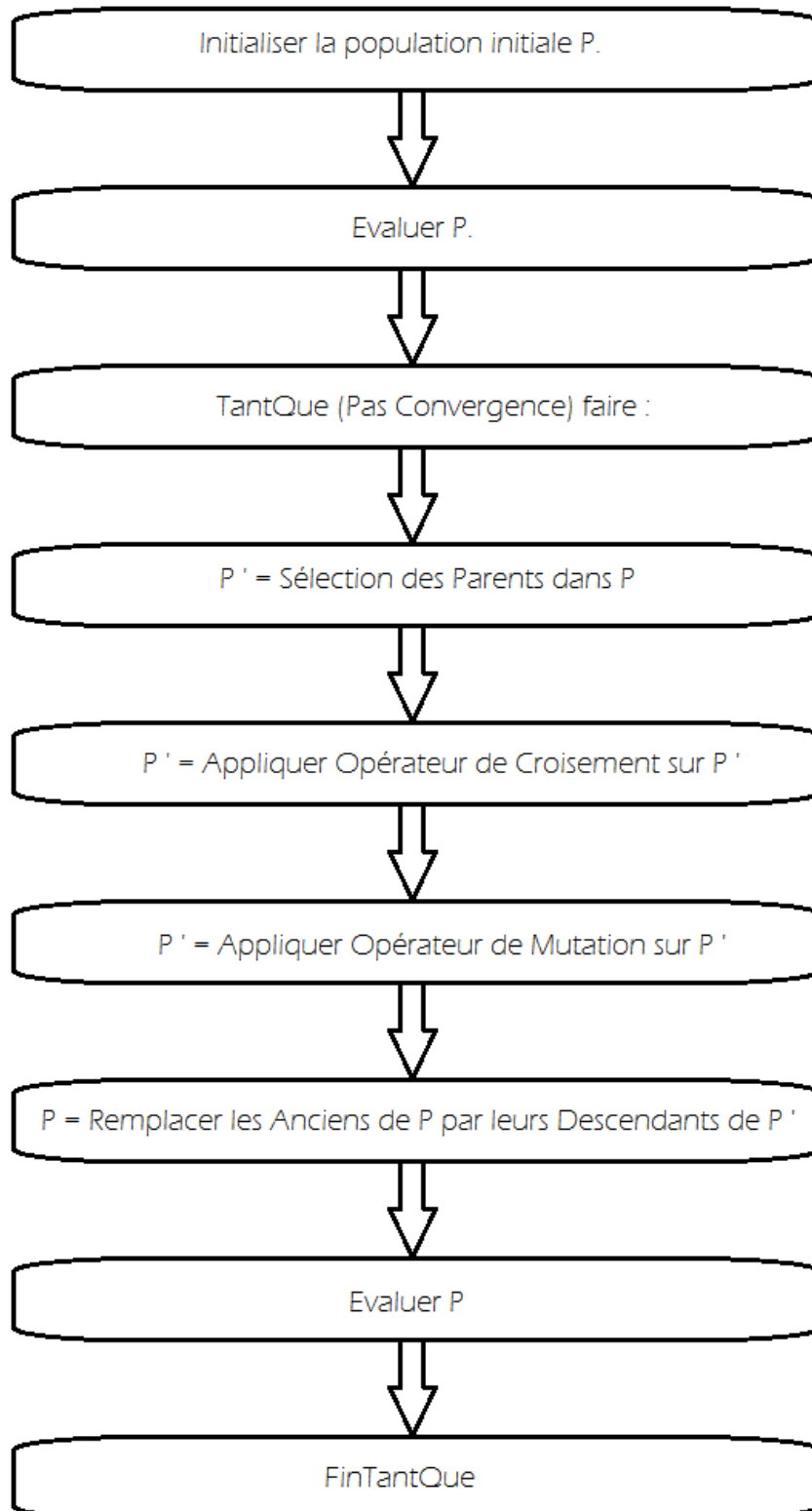


Figure 13. La forme de l'algorithme génétique

4.2. Principe de fonctionnement

Donc le but d'utiliser ce genre d'algorithme est d'obtenir une solution approchée à un problème d'optimisation, lorsqu'il n'existe pas de méthode exacte (ou que la solution est inconnue) pour le résoudre en un temps raisonnable. Les algorithmes génétiques utilisent la notion de sélection naturelle et l'appliquent à une population de solutions potentielles au problème donné. La solution est approchée par « bonds » successifs, comme dans une procédure de séparation et évaluation, à ceci près que ce sont des formules qui sont recherchées et non plus directement des valeurs.

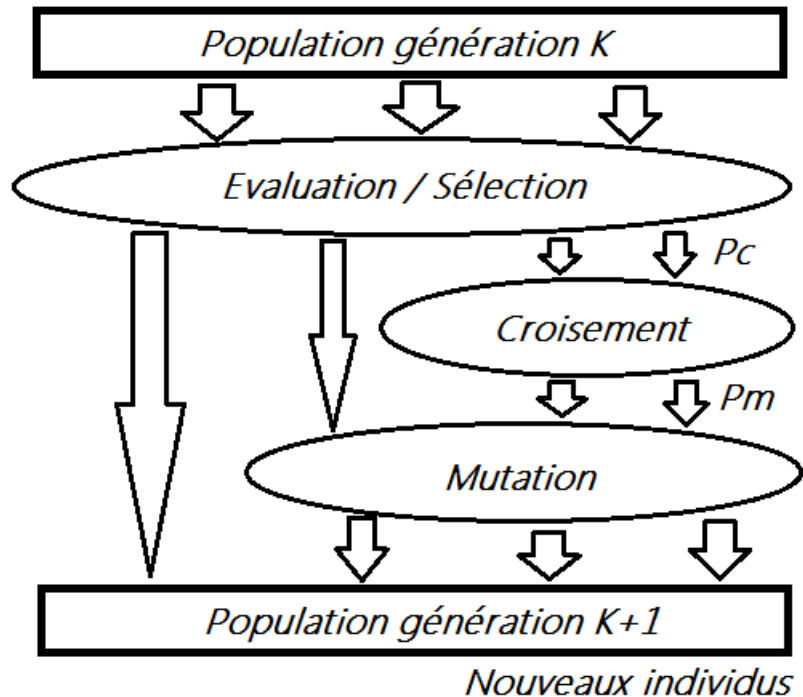


Figure 14. Le principe de fonctionnement d'un algorithme génétique

Et afin de permettre la résolution de problèmes, se basent sur les différents principes décrits ci-dessus. Le problème théorique de la convergence a été résolu par Raphael Cerf, en se basant sur la théorie de Friedlin Weizel des perturbations stochastiques des systèmes dynamiques. La démonstration de R. Cerf montre d'ailleurs que le processus de convergence dépend essentiellement de la mutation, le croisement pouvant être éliminé en théorie. Cependant, la preuve théorique de convergence n'a que peu d'utilité dans la pratique, où l'opérateur de croisement fait bien souvent toute la richesse de l'algorithme génétique par rapport à des méthodes de type recuit simulé.

De manière globale, on commence avec une population de base qui se compose le plus souvent de chaînes de caractères correspondant chacune à un chromosome. Nous reviendrons par la suite sur les différentes structures de données possibles (voir Codage) mais nous retiendrons pour le moment l'utilisation du codage binaire (ex. : 0100110).

Le contenu de cette population initiale est généré aléatoirement. On attribue à chacune des solutions une note qui correspond à son adaptation au problème. Ensuite, on effectue une sélection au sein de cette population.

Il existe plusieurs techniques de sélection. Voici les principales utilisées :

Sélection par rang

Cette technique de sélection choisit toujours les individus possédant les meilleurs scores d'adaptation, le hasard n'entre donc pas dans ce mode de sélection. En fait, si n individus constituent la population, la sélection appliquée consiste à conserver les k meilleurs individus (au sens de la fonction d'évaluation) suivant une probabilité qui dépend du rang (et pas de la fonction d'évaluation).

Probabilité de sélection proportionnelle à l'adaptation

Appelé aussi roulette ou roue de la fortune, pour chaque individu, la probabilité d'être sélectionné est proportionnelle à son adaptation au problème. Afin de sélectionner un individu, on utilise le principe de la roue de la fortune biaisée. Cette roue est une roue de la fortune classique sur laquelle chaque individu est représenté par une portion proportionnelle à son adaptation. On effectue ensuite un tirage au sort homogène sur cette roue.

Sélection par tournoi

Cette technique utilise la sélection proportionnelle sur des paires d'individus, puis choisit parmi ces paires l'individu qui a le meilleur score d'adaptation.

Sélection uniforme

La sélection se fait aléatoirement, uniformément et sans intervention de la valeur d'adaptation. Chaque individu a donc une probabilité $1/P$ d'être sélectionné, où P est le nombre total d'individus dans la population.

Lorsque deux chromosomes ont été sélectionnés, on réalise un croisement. On effectue ensuite des mutations sur une faible proportion d'individus, choisis aléatoirement. Ce processus nous fournit une nouvelle population. On réitère le processus un grand nombre de fois de manière à imiter le principe d'évolution, qui ne prend son sens que sur un nombre important de générations. On peut arrêter le processus au bout d'un nombre arbitraire de générations ou lorsqu'une solution possède une note suffisamment satisfaisante.

Exemple d'explications :

Considérons par exemple les deux individus suivants dans une population où chaque individu correspond à une chaîne de 8 bits : $A = 00110010$ et $B = 01010100$. On ajuste la probabilité d'enjambement à 0,7 ($8 \times 0,7 = 5,6$ alors on va croiser 6 bits sur les 8 bits des deux mots).

Supposons ici que l'enjambement ait lieu, on choisit alors aléatoirement la place de cet enjambement (toutes les places ayant la même probabilité d'être choisies). En supposant que l'enjambement ait lieu après le deuxième allèle, on obtient A' et B' (« : » marquant l'enjambement sur A et B). Ensuite, chacun des gènes des fils (ici, chacun des bits des chaînes) est sujet à la mutation. De la même manière que pour les combinaisons, on définit un taux de mutation (très bas, de l'ordre de 0,001 - ici on peut s'attendre à ce que A' et B' restent identiques).

Chromosome	Contenu
A	00:110010
B	01:010100
A'	00 010100
B'	01 110010

En effectuant ces opérations (sélection de deux individus, enjambement, mutation), un nombre de fois correspondant à la taille de la population divisée par deux, on se retrouve alors avec une nouvelle population (la première génération) ayant la même taille que la population initiale, et qui contient globalement des solutions plus proches de l'optimum. Le principe des algorithmes génétiques est d'effectuer ces opérations un maximum de fois de façon à augmenter la justesse du résultat.

Il existe plusieurs techniques qui permettent éventuellement d'optimiser ces algorithmes, on trouve par exemple des techniques dans lesquelles on insère à chaque génération quelques

individus non issus de la descendance de la génération précédente mais générés aléatoirement. Ainsi, on peut espérer éviter une convergence vers un optimum local.

CONCLUSION

Le projet (Modélisation, simulation et optimisation d'un moteur roue) nous a permis de pratiquer et d'appliquer les connaissances qui nous ont été inculquées au cours de formation à Polytech' Clermont-Ferrand ainsi que sur les deux logiciels utilisés (COMSOL et MATLAB), et de découvrir l'interface qui existe entre les deux, ainsi que de nous initier à la recherche dans une optique éventuelle de poursuite d'études dans ce domaine. Nous avons été confrontés à de nombreux problèmes et dans la plupart des cas nous avons pu trouver une solution alternative afin de les résoudre partiellement.

D'un point de vue humain, le travail en groupe est une bonne chose, cela m'a permis de travailler en équipe, de respecter les choix et idées de tous les membres afin de servir une cause commune. Également, le fait de se découper le travail m'a permis de se concentrer sur une partie précise où l'on va s'investir à fond dans l'intérêt du groupe.

Enfin ce projet et notamment cette partie concernant l'optimisation aura été l'occasion de découvrir et d'utiliser des outils dont nous n'avions pas la moindre idée de leurs existences.

BIBLIOGRAPHIE

- La documentation de COMSOL MULTIPHYSICS®
- www.comsol.com
- <http://www.recherche.enac.fr/opti/papers/thesis/HABIT/main002.html>
- <http://www.mathworks.fr/>
- <http://deptinfo.unice.fr/twiki/pub/Linfo/PlanningDesSoutenances20032004/Radet-Souquet.pdf>
- <http://www-gremaq.univ-tlse1.fr/perso/cbontemps/Cours/AG/CoursOpti2.pdf>
- <http://www.ann.jussieu.fr/~postel/matlab/>