ISIMA
Institut Supérieur d'Informatique de
modélisation et de leurs Applications

Campus des Cézeaux
24 avenue des Landais
BP 10125
63173 AUBIERE cedex.

Valeo

Valeo Systèmes d'Essuyage
Rue Marie Curie,
63500, Issoire.

Rapport d'ingénieur
projet de deuxième année ISIMA
F4

POSTRAITEMENT D'UN PLAN D'EXPERIENCE DE COURBES CARACTERISTIQUES DE SPLINES D'ESSUIE-VITRE

Présenté par :
Malek Ben Salem
Safae Faraji

Responsable Valeo :
M. Sebastien Jallet

Responsable ISIMA :
M. Gilles Leborgne

8 mars 2013

Remerciements

Nous tenons tout d'abord à remercier notre responsable de projet chez Valeo, Sébastien Jallet, pour nous avoir accordé son temps et nous avoir guidé tout au long de ce projet. Nous le remercions vivement pour son aide et sa disponibilité.
Nous sommes aussi reconnaissants à Gilles Leborgne et à Christophe Tilmant pour leurs précieux conseils.

Résumé

Les travaux présentés dans ce rapport ont été réalisés dans le cadre d'un projet avec Valeo. Ils sont basés sur un ensemble de résultats d'expériences, effectuées au sein de la direction de recherche du site Valeo Issoire, sur les splines des essuie-vitres (partie métallique de part et d'autre ou à l'intérieur des nouveaux balais plats).

Une spline est définie par de nombreux paramètres qui influencent la forme de sa courbe caractéristique et ses *extrema* locaux tels que la section, le rayon cible et la force de définition.

le projet, réalisé sur Matlab, consiste ainsi à extraire, à partir d'un plan d'expériences réalisées sur plusieurs splines de différents couples (force, rayon) pour obtenir leurs courbes caractéristiques correspondantes, les *extremums* locaux tout en lissant les courbes (suppression des variations locales dues à la non linéarité : dé-bruitage). L'objectif, ensuite, est de lier les *extrema* locaux identifiés aux paramètres d'entrée (dans le cadre de ce projet les couples (Rayon,Force)) pour avoir une base de données (Force,Rayon,Extremum) et déterminer à travers cette base une méthode d'interpolation qui permettrait de déterminer les extremums de la courbe caractéristique d'une spline sans réaliser l'expérience correspondante.

La méthode d'interpolation la plus efficace qui a été utilisée est l'interpolation par le Krigeage. L'erreur moyenne obtenue est de l'ordre de 10^{-3} pour l'exemple traité plus bas.

Mots-clé : Matlab, Interpolation, Krigeage, extremum, dé-bruitage

Abstract

The work presented in this report was established within the framework of a project with Valeo. It is based upon an exeperiments set of results made within the directionof research of Valeo Issoire on splines metal portion on both sides or inside the new flat blades.

A spline is defined through numerous parameters which influence the shape of its characteristic curve as well as its local extrema such as the section, the target radius and the strength. The project, realized on Matlab, and based on experiments realized on several splines of various couples (strength and radius) to obtain their corresponding characteristic curves, and to extract the local extrema while smoothing curves (removal of the local variations due to the non linearity : denoising) . The purpose, then, is to link the local extrema identified with the parameters of entry (in this project the couples (radius, Strength)) to have a base of connected data (Strength, radius, Extrema) and to determine through this base an interpolation method (Kriging) which would allow to determine the extrema of the characteristic curve of a spline without realizing the corresponding experiment.

Keywords : Matlab, Interpolation, Krigeage, extremum ,denoising

Table des matières

Table des figures

Introduction

Afin de prévoir le comportement des essuie-vitres soumis à des situations mécaniques variées, Valeo effectue des expériences sur les splines qui composent ces essuie-vitres. Quelques résultats obtenus de ces expériences nous ont été communiqués dans le cadre de ce projet.

Le but de notre projet était de déduire, à partir de quelques données d'expériences, les caractéristiques de splines non testées. Afin de résoudre ce problème, nous avons procédé en plusieurs étapes. Il a fallu en premier déterminer les points intéressants sur les courbes caractéristiques. Nous avons donc lissé ces courbes et déterminé les *extrema* qui permettent de caractériser la forme de la courbe. Ensuite, nous avons utilisé la méthode d'interpolation par Krigeage, que nous avons enfin comparée à autre méthode d'interpolation, la IDW, afin de vérifier son efficacité.

Le rapport est structuré comme suit :
Dans la première partie, nous faisons une description plus détaillée du problème. Ensuite, nous expliquerons comment nous avons procédé pour extraire les données des fichiers fournis et le traitement que nous avons effectué afin de déterminer les points intéressants. Nous expliquerons ensuite la méthode d'interpolation par Krigeage avant de la comparer à la méthode d'interpolation de pondération par l'inverse de la distance.

I Description du problème

Valeo est un groupe spécialisé dans la production et la vente d'équipements pour l'industrie automobile. Il est présent dans 28 pays et dispose de 124 sites de production, 21 centres de recherche, 40 centres de développement et 12 plateformes de distribution. Le groupe se divise en quatre grands pôles d'activités dont le Pôle Systèmes de Visibilité, qui se divise quant à lui en deux familles de produits : Éclairage Signalisation (VLS) et Systèmes d'Essuyage (VWS).

Le site d'Issoire, qui a proposé ce projet, est spécialisé dans les Systèmes d'Essuyage. Dans ce cadre, nous avons été en contact avec Sébastien Jallet qui y occupe le poste d'expert.

Dans le cadre de ses activités, Valeo Issoire effectue des expériences sur les splines composants les essuie-vitres. Les splines d'essuie-vitre sont en effet les parties métalliques de part et d'autre des balais plats, elles sont caractérisées par leurs longueurs et par leurs rayons de courbure.

Ces tests effectués sur ces splines visent à déterminer leur comportement quand elles sont soumises à une force de pression. Une roulette de contrôle se déplace sous la spline soumise à un effort d'appui en son milieu. Le comportement des deux moitiés de la spline étant le même, le test sur une seule moitié suffit à déterminer le comportement général de la spline.

L'expérience peut être modélisée ainsi :

FIGURE 1 – modélisation de l'expérience

Les résultats de ces expériences sont regroupés dans des fichiers structurés comme suit :

```
File created on : Wed Jun 09 11:19:29 2010
by Flat Blade Calculations Procedures Version 2 Release 0 for Marc 2007-2008 r1

  Blade type                          :      FB1    [033.153]
  Spline length                 (mm):                605.00
  Base radius                   (mm):                2000.0
  Spline Definition pressure (N/mm):                0.017000      (and not blade definition pressure)

  Test force                    (N):     5.100
  Gravity                           :   Without

HISTORY PLOT
Analysis

Curve 1
X : Pos X sensor [0]
Y : Force sensor [0]

       X              Y
-------------------------------
     -302.50       0.893669
     -300.00       0.914886
     -297.50       0.928657
     -295.00       0.958478
     -292.50       0.972986
     -290.00       1.003942
```

FIGURE 2 – Exemple de fichiers de données

Exemple des courbes caractéristiques obtenues à l'issue de ces expériences :

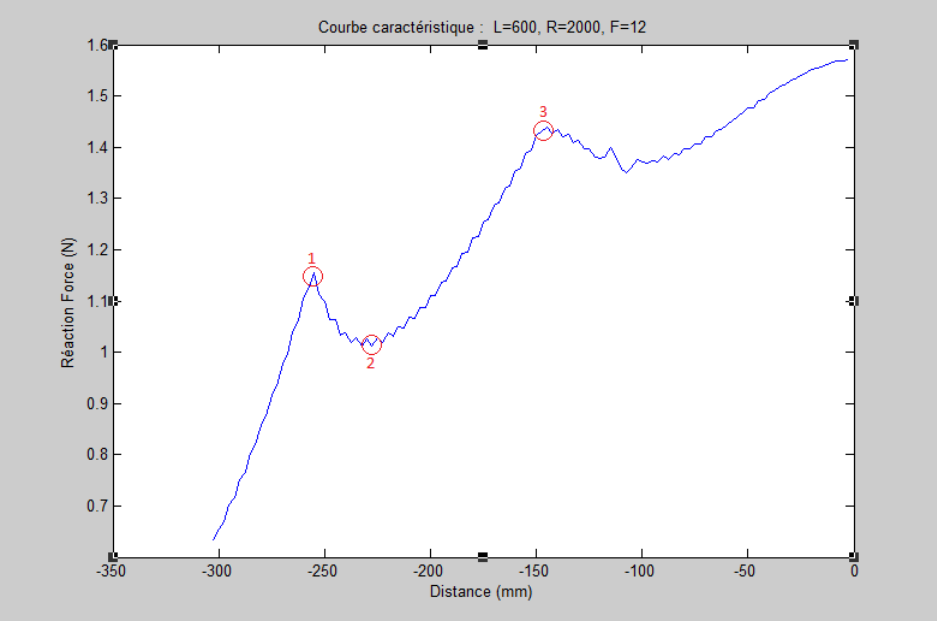

FIGURE 3 – Courbe caractéristique : L=600 mm, R=2000 mm, F=12 N

Les points entourés en rouge sur la figure sont les points qui caractérisent chaque spline. En effet, la roulette de contrôle se déplace du bord de la spline à son milieu. Le pic 1 caractérise en fait l'instant où le bord de la spline, relevé par la roulette, retouche le plan, le point 2 est l'instant où la réaction de la spline est minimale, le point 3 caractérise le moment où la réaction reste presque constante du fait de l'approche de la roulette du

centre de la spline. Ce sont les points qui nous intéressent dans la suite de l'étude.

Le projet consiste à déduire, pour chaque spline, sans effectuer l'expérience et en se basant seulement sur les données obtenues par d'autres expériences, les points caractéristiques qui dépendent de la force appliquée, du rayon de courbure de la spline et de sa longueur.

Pour résoudre le problème, nous avons établit les étapes suivantes :

- Lisser les courbes caractéristiques
- Déterminer les coordonnées des trois points caractéristiques
- Estimer par une méthode d'interpolation les caractéristiques de splines non testées

II Détermination des points caractéristiques

II.1 Extraction et organisation des données

II.1.1 L'organisation existante

On dispose des résultats des expériences effectuées sur les splines de longueur L=600 (mm), de Rayon cible R et force F variables dans le dossier $L600FdefPdef$ qui nous a été communiqué dans le cadre du projet. Ce dossier contient à son tour des fichiers avec des noms significatifs, par exemple $fb1_L600_R5000_F18$ est le fichier contenant les coordonnées de la courbe caractéristique résultant de l'expérience sur la spline de longueur L=600 (mm), de rayon cible R=5000 (mm) et de force de définition F=18 N/m.

On a 91 dossiers qui correspondent à tous les couples possibles suite à la combinaisons d'une force F qui varie entre 12 F/m et 18 F/m et de pas 0.5 F/m (13 valeurs possibles) et d'un rayon cible R qui varie entre 2000 mm et 5000 mm d'un pas de 500 mm (7 valeurs possibles).

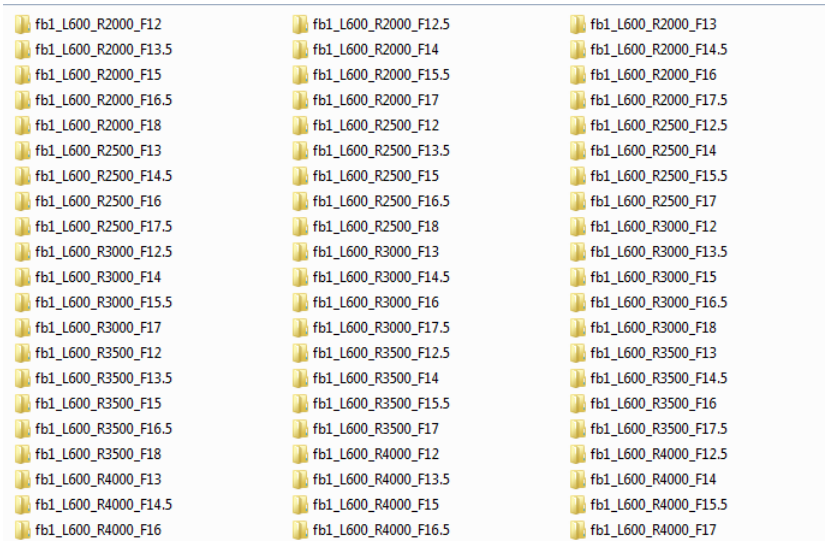

FIGURE 4 – L'organisation existante des fichiers

Chaque dossier contient un fichier qui comporte les données résultant des expériences effectuées sur la spline de rayon et force bien déterminés.

Les données sont deux vecteurs qui représentent la distance en millimètre et la réaction force en N/m correspondantes mesurées à l'expérience.

Nous avons remarqué que les dossiers $fb1_L600_R2500_F13$ et $fb1_L600_R2500_F12$ ne contiennent pas les résultats des expériences correspondantes.

II.1.2 Extraction des données

L'organisation des données de cette manière pratique et efficace pour la reconnaissance des entrées correspondantes, ne nous a pas néanmoins aidé pour l'extraction de ces données sur Matlab, la navigation dans les dossiers étant différente selon le système d'exploitation utilisé, Unix ou Windows. C'est ainsi qu'on a décidé d'écrire deux fonction d'extraction appelées « lecture », la première correspond au système Unix et la deuxième correspond aux système Windows.

Différences Majeures entre Unix et Windows :

Matlab assure la navigation dans les dossiers utilisés en système des commandes ls, $cd..$, $cd\ \ 'FolderName', etc.$ Pour pouvoir l'exploiter dans notre cas, nous avons eu besoin de lire le contenu du dossier $L600FdefPdef$ en utilisant la commande 'ls'. Cette commande retourne sur Windows une matrice ou chaque ligne contient le nom d'un sous-dossier ou d'un fichier du $L600FdefPdef$, alors que sur Unix, 'ls' retourne un vecteur qui contient tous ces noms l'un à coté de l'autre.

exemple : si on est dans le dossier $L600FdefPdef$

```
>>  cd 'L600FdefPdef' ;
>> A=ls;
>> size(A)

ans =

    94    40

>>
```

FIGURE 5 – Accès aux fichiers sur Windows

Remarque : sur Windows les deux premières lignes correspondant à '.' et à '..' représentent l'adresse du dossier parent.

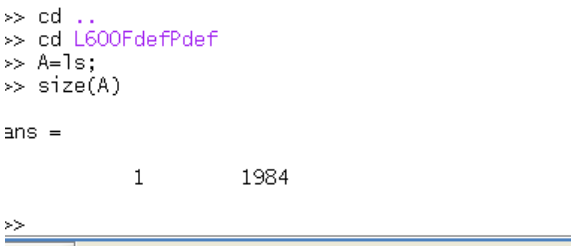

FIGURE 6 – Accès aux fichiers sur Linux

Nous avons donc découpé le vecteur 'A' pour pouvoir accéder à chaque sous-dossier en utilisant le fait que tous les noms de ces sous-dossiers commençant par le même préfixe. Nous avons pu alors chercher l'indice du début de chaque nom dans le vecteur. Alors que sur Windows, on exploite le fait que le nom d'un dossier est une ligne propre. Ensuite, on boucle le même travail sur tous les sous-dossiers en accédant chaque fois à l'un, puis on effectue la lecture du fichier qui contient les données de l'expérience correspondante.

II.1.3 Structuration des données

Étant donné que nous avons besoin de connaître le rayon cible et la force de définition qui correspondent à chaque expérience, nous avons conçu la fonction *lecture* pour qu'elle retourne deux matrices notées comme suit :

• rf : une matrice qui contient les valeurs de rayon cible et de la force de définition des expériences, *ie* rf(1,i), respectivement rf(2,i) contiennent les valeurs respectives du rayon cible et de la force de définition de la i-ème expérience.

• mat : une matrice qui contient pour chaque expérience, les deux vecteurs extraits du fichier correspondant, *ie* si X=mati, X(:,1) respectivement X(:,2) contiennent les valeurs respectives de la distance et de la réaction de la ieme expérience, c'est à dire l'expérience effectuée ou le rayon cible R=rf(1,i) et la Force de définition F=rf(2,i).

Nous avons pu ainsi assuré la liaison entre les données de chaque expérience et les paramètres correspondants d'une manière simple et même intuitive pour des non-spécialistes (cf les fonctions lecture, lectureUnix).

Pour effectuer cette structuration tout en assurant la correspondance entre 'mat' et 'rf' qui contiennent des données exploitables, nous avons ajouté la condition qui vérifie l'existence des données de l'expérience dans tous les fichiers.

II.2 Lissage des courbes et détermination des points caractéristiques

Afin de déterminer plus exactement les *extrema* qui nous intéressent pour la caractérisation des splines, nous avons lissé les courbes avec plusieurs méthodes :

II.2.1 Filtre moyenneur

Pour lisser la courbe, l'idée qui a été la plus triviale est de calculer pour chaque point x_o la valeur moyenne sur N points successifs centrés en x_o.
A titre d'exemple, pour $N = 7$, pour chaque point X (une distance) où l'intervalle $[X-3, X+3]$ appartient à l'ensemble des distances, nous calculons la valeur moyenne des valeurs de la réaction force sur l'intervalle des distances $[X-3, X+3]$ [1] Nous nous sommes rendus compte que le filtre moyenneur qui lisse la courbe, admet un erreur importante par rapport à la courbe initiale au niveau du premier pic. Cette erreur atteint 0.08 pour la courbe de l'expérience où F=13.5, et R=2500, elle s'explique par le fait que tous les points du voisinage du premier point caractéristique ont des valeurs inférieures à celle de l'*extremum*.

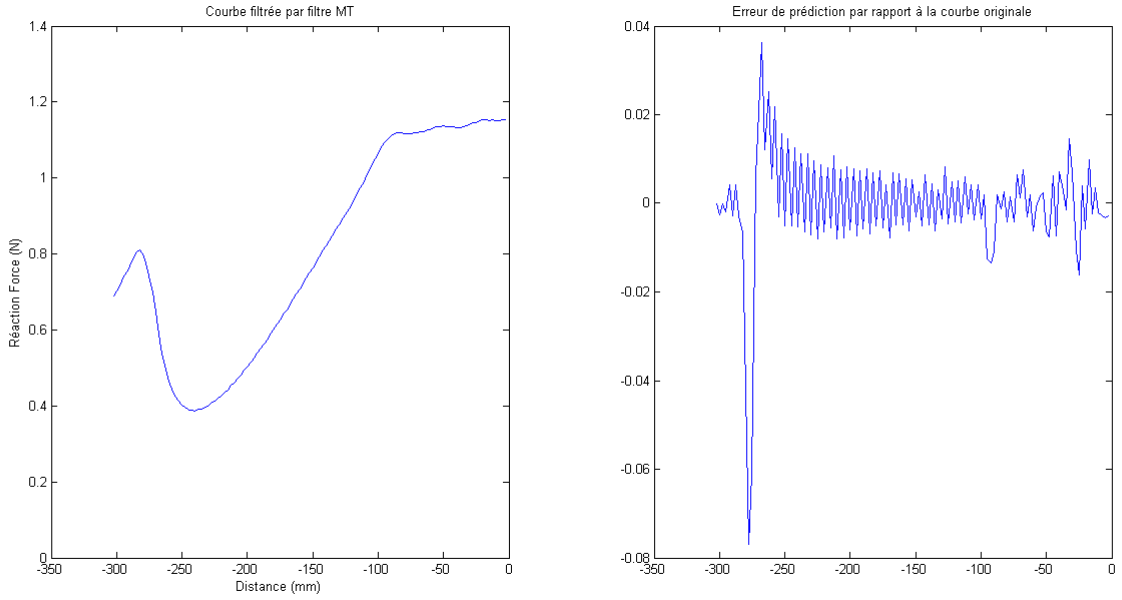

FIGURE 7 – Courbe filtrée et erreur après application du filtre MT

Nous avons donc décidé de découper la courbe au niveau du premier pic. Cette première partie a été modélisée par deux fonctions affines, qui correspondent aux voisinages du pic. Nous avons ensuite utilisé le filtre moyenneur pour lisser le reste de la courbe [2].

1. cf. MT.m Annexe page II
2. cf. moyenneur.m Annexe page III

Nous avons obtenu des résultats meilleurs. L'erreur maximale atteint 0.025 pour la courbe de l'expérience où F=13.5, et R=2500, alors que l'erreur maximale était de 0.08 pour cette même expérience, sans le découpage.

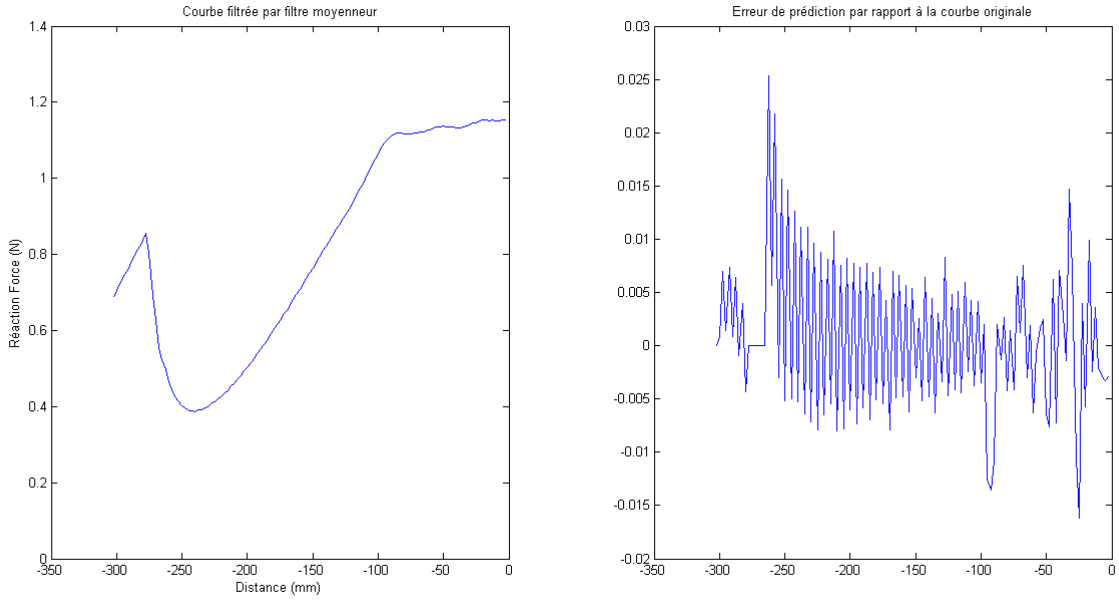

FIGURE 8 – Courbe filtrée et erreur après application du filtre moyenneur

A ce stade, nous avons réfléchi à une manière d'améliorer encore ce résultat en minimisant davantage l'erreur.

II.2.2 Débruitage

Les filtres passe-bas ont pour rôle d'atténuer les parties d'un signal se trouvant hors la bande qui nous intéresse. Nous avons utilisé un filtre de cette catégorie afin d'enlever le bruit constaté sur les courbes.
Nous avons procédé de la même manière que précédemment pour le filtre moyenneur : découper la courbe et modéliser la première partie par une fonction affine, filtrer ensuite le reste de la courbe par un filtre passe-bas qui supprime les hautes fréquences [3].

La fonction *denoise* que nous avons écrite dans ce but accepte en entrée un vecteur contenant les valeurs de la fonction bruitée et renvoie un vecteur contenant les valeurs lissées.
Elle consiste à filtrer la fonction par un filtre passe-bas.

L'application de la fonction *denoise* sur la courbe caractéristique correspondant à F=13.5 N et R=2500 mm donne le graphe suivant :

3. cf. denoise.m Annexe page III

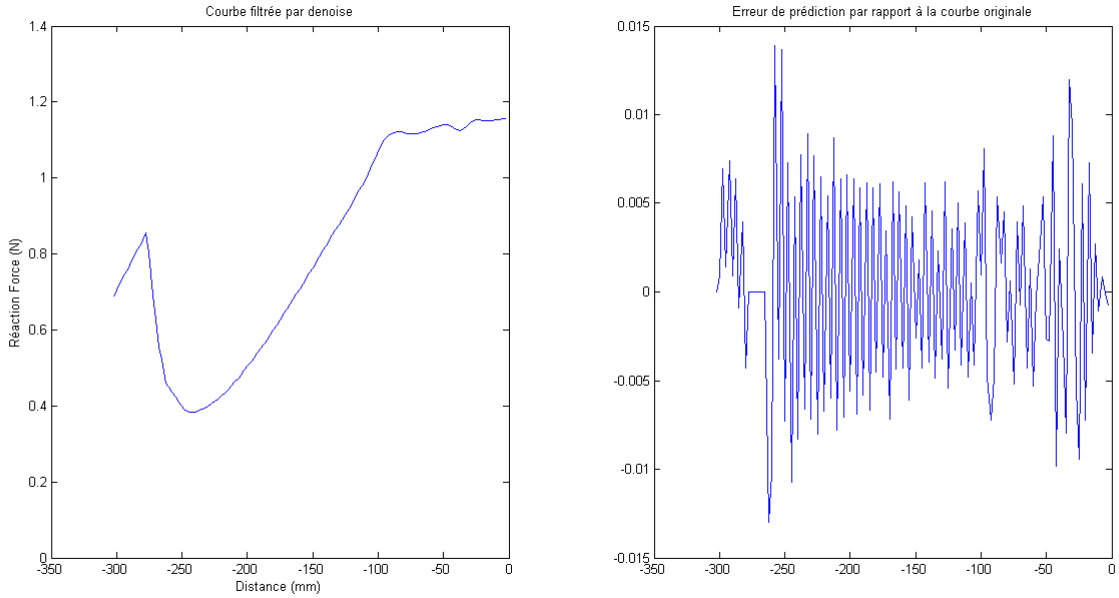

FIGURE 9 – Courbe filtrée et erreur après application du filtre denoise

L'erreur maximale obtenue pour la même expérience (F=13.5, R=2500) est inférieure à 0.015.

II.2.3 Processus auto-régressif

Un signal $u(n)$ est la réalisation d'un processus auto-régressif d'ordre n. Si $u(n)$ s'écrit :

$$u(p) = \sum_{k=1}^{n} \omega_k \cdot u(p-k) + v(p) \tag{1}$$

où les ω_k, $k = 1, \cdots, n$ sont les coefficients du modèle et $v(n)$ est un bruit blanc.

On construit par (1) une prédiction linéaire qui permet de prédire chacun des termes $u(p)$ par une combinaison de ses n prédécesseurs, multipliés par des poids $\phi_1, \phi_2, \cdots, \phi_k$, et ce, à une erreur de prédiction près $v(n)$.
(1) peut s'écrire sous la forme :

$$u(p) + \sum_{k=1}^{n} \phi_k \cdot u(p-k) = v(p) \tag{2}$$

Avec $\phi_k = -\omega_k$, pour tout $k = 1, \cdots, n$.

En multipliant l'équation précédente par $u(p-j), j > 0$, et en calculant l'espérance mathématique :

$$E[\sum_{k=0}^{n} \phi_k \cdot u(p-k)u(p-j)] = E[v(p)u(p-j)],\ p = 1, \cdots, n \tag{3}$$

Avec $\phi_0 = 1$.

Puisque $u(p-j)$ ne dépend pas du bruit $v(p)$, les intercorrélations entre les deux sont nulles sauf pour $j = 0$.
On a donc :

$$E[v(p)u(p-j)] = 0 \ \ si \ \ j > 0, \ \ \sigma^2 \ \ si \ \ j = 0. \tag{4}$$

Les termes de gauche dans (3) sont les autocorrélations de u. Revenons maintenant à la notation en $\omega_k = -\phi_k$, on peut écrire :

$$\sum_{k=1}^{n} \omega_k \ E[u(p-k)u(p-j)] = E[u(p)u(p-j)],\ p = 1, \cdots, n \tag{5}$$

Et on pose :

$$E[u(p-k)u(p-j)] = \nu_{j-k} \tag{6}$$

Déterminer les poids ϕ_k de l'équation (1) revient donc à résoudre le système linéaire suivant [1] :

$$\begin{pmatrix} \nu_{-1} & \nu_{-2} & \nu_{-3} & \cdots & 1 \\ \nu_0 & \nu_{-1} & \nu_{-2} & \cdots & 0 \\ \nu_1 & \nu_0 & \nu_{-1} & \cdots & 0 \\ \nu_2 & \nu_1 & \nu_0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \end{pmatrix} \begin{pmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \vdots \\ \sigma^2 \end{pmatrix} = \begin{pmatrix} \nu_0 \\ \nu_1 \\ \nu_2 \\ \nu_3 \\ \vdots \end{pmatrix} \tag{7}$$

Nous avons écrit la fonction modelAR [4], qui applique le processus auto-régressif pour déterminer la valeur de chaque point en fonction des valeurs de ses prédécesseurs. Cette fonction nous servira à la détection des points qui nous intéressent. En effet, un pic dans l'erreur entre la fonction originale et celle obtenue après application du processus auto-régressif, est dû à un changement brusque de pente. Il nous est donc facile de détecter le premier et le troisième *extrema*, alors que le deuxième est moins facile à voir par cette méthode car l'allure générale de la courbe dans cette partie est "plutôt dérivable".

Nous obtenons pour différents ordres, les courbes suivantes :

4. cf. Annexe page IV

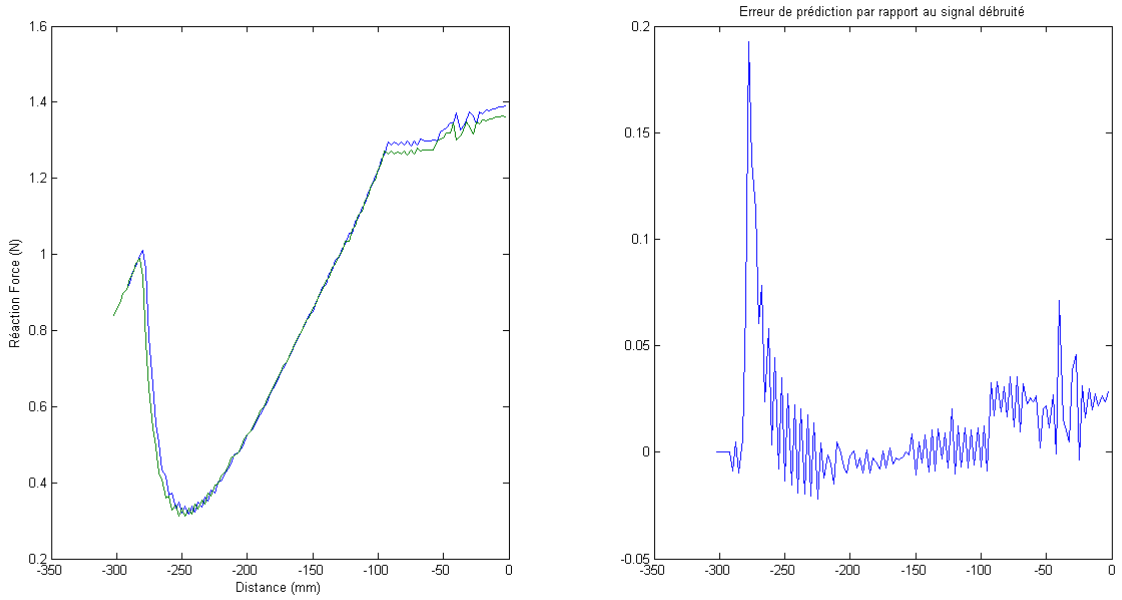

FIGURE 10 – Comparaison de la fonction caractéristique au modelAr à l'ordre 3

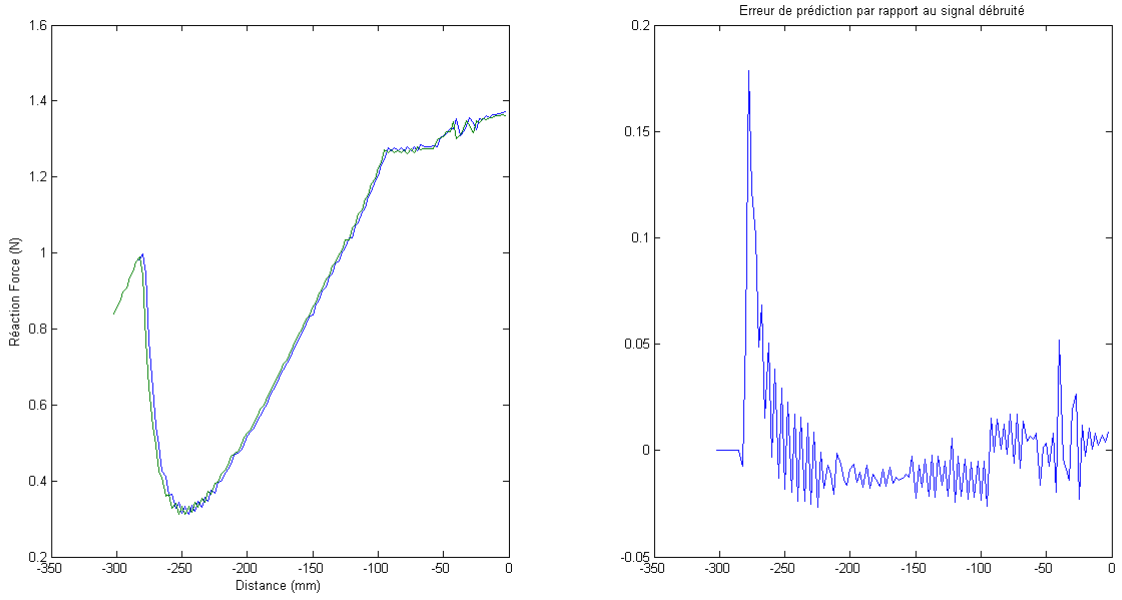

FIGURE 11 – Comparaison de la fonction caractéristique au modelAr à l'ordre 6

Cette méthode pose néanmoins quelques difficultés ; notamment, l'ordre du filtre convenant à chaque exemple est une variable du problème *a priori* inconnue. L'objectif étant de trouver le meilleur jeu de paramètres permettant la prédiction avec une erreur minimale, il faut faire plusieurs tests pour chaque jeu de données, sans pour autant assurer la meilleure solution.

II.3 Synthèse

Pour rassembler les résultats de traitement des courbes obtenues plus haut, nous avons écrit une fonction appelée *exec.m*[5].

La fonction *exec.m* accepte en entée :

- le nom de dossier (folderName) contenant l'ensemble des enregistrement. Si cet argument n'est pas précisé, elle accède par défaut au dossier 'L600FdefPdef'.

Elle renvoie en sortie :

- m : matrice qui contient tout les données de tout les fichiers
- rf : tableau qui contient le rayon cible et la force correspondante à chaque ensemble de valeurs
- Xopt, Iopt : les coordonnées des points intéressants pour chaque enregistrement
- pi : un vecteur contenant les valeurs du ième point intéressant

Elle enregistre aussi automatiquement pour l'ensemble des tests les courbes suivantes :

- La courbe initiale pour avoir une idée sur l'allure
- La courbe lissée par *denoise.m*
- La courbe des différences entre la courbe l'initiale et la courbe lissée par *denoise.m* pour avoir une idée sur l'erreur
- La courbe lissée par *moyenneur.m*
- La courbe qui résulte de *ModelAr.m*
- La courbe de l'erreur relative au modèle AR

Exemple d'exécution sur la courbe caractéristique correspondant à R = 2000 mm et à F= 12.5 N :

5. cf. exec.m Annexe page V

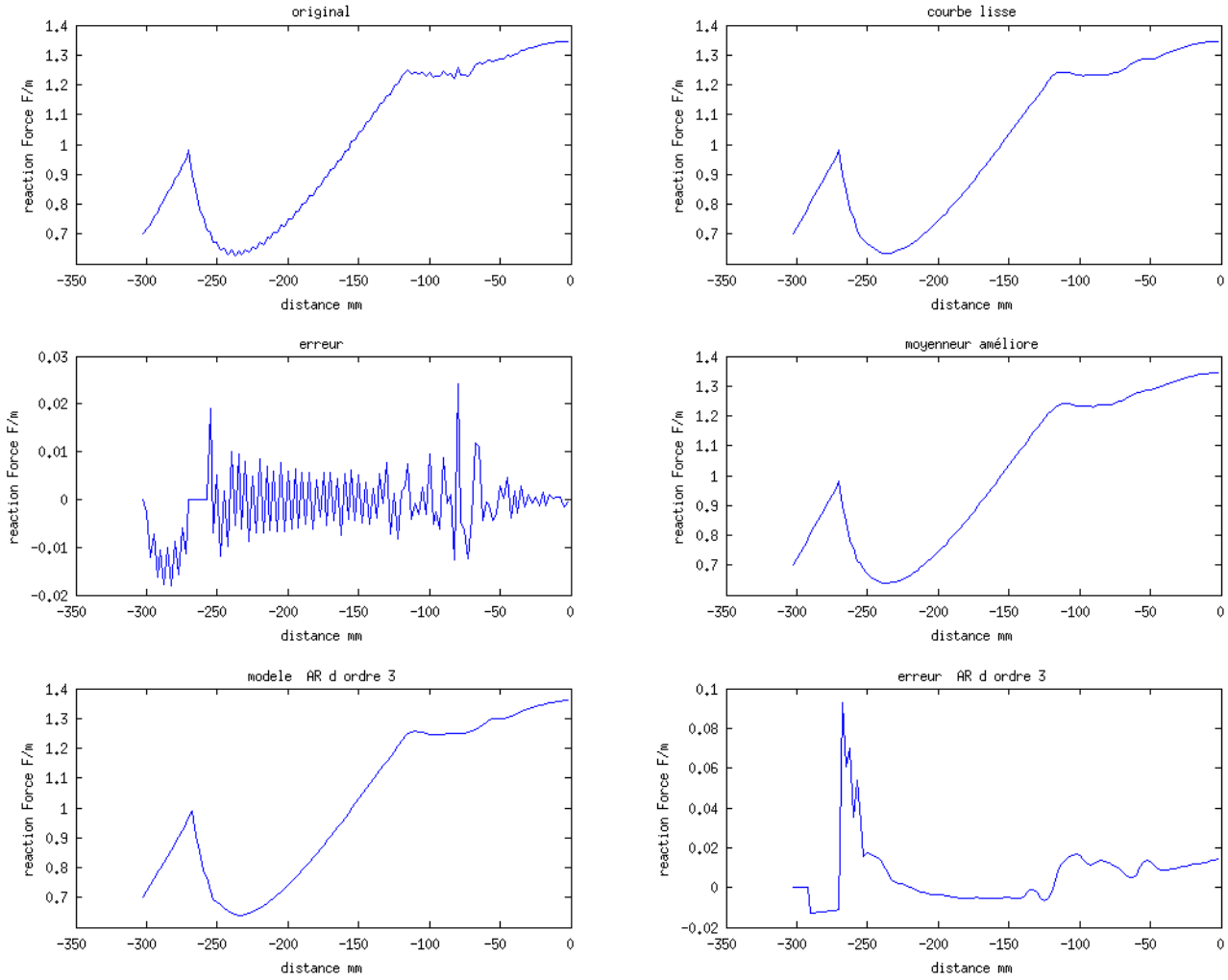

FIGURE 12 – Exemple d'exécution de *exec.m*

III Krigeage

III.1 Itroduction au krigeage

Le Krigeage est une méthode d'interpolation, minimisant la variance d'estimation telle que calculée à l'aide du variogramme[2].

L'interpolation spatiale est un problème d'estimation linéaire d'une fonction $F(x)$, où x dans notre cas un point du plan, à partir de valeurs connues de F en un certain nombre de points environnants x_i :

$$F(x) = \sum_{i=1}^{n} \lambda_i \cdot F_(x_i) \tag{8}$$

Le problème consiste à déterminer la pondération, les λ_i, de chacun des points environnants.

III.2 Méthode

Le Krigeage consiste ensuite à calculer les λ_i de l'équation (2) à l'aide des valeurs de la fonction :

$$W(d) = \frac{1}{2n(d)} \sum_{i=1}^{n(d)} (X_i - Y_i)^2 \tag{9}$$

Où :

- d : une distance entre deux points parmi les X_i
- $n(d)$: nombre de point séparés par la distance d
- X_i et Y_i : deux points séparés par la distance d

Cette fonction nous donne un ensemble de points et la fonction $W(d)$ est à la suite la fonction qui résulte de l'ajustement de ces points par une méthode comme la des moindres carrés par exemple.

Pour estimer la valeur du vecteur $\lambda = (\lambda_i)$ pour un point X_p la méthode de krigeage avec n nombre connu de points consiste à résoudre le problème linéaire suivant :

$$A\lambda = b \tag{10}$$

Où $b = (W(d_{ip}))_{1 \leq i \leq n}$, et si on note $A = (a_{ij})_{1 \leq i \leq n+1, 1 \leq j \leq n+1}$, alors

$$\begin{cases} a_{ij} = d_{ij}, \quad 1 \leq i, j \leq n \\ a_{ij} = 1, \quad 1 \leq j \leq n \quad et \quad i = n+1 \\ a_{ij} = 1, \quad 1 \leq i \leq n \quad et \quad j = n+1 \\ a_{n+1,n+1} = 0 \end{cases} \tag{11}$$

La dernière ligne de la matrice A est ajoutée pour que la somme des λ_i soit égale à 1 pour

que Pour que la solution soit non-biaisée et le dernier élément du vecteur λ de l'équation (4) est un multiplicateur de Lagrange qui vise à minimiser l'erreur d'estimation.

Remarque : La matrice A est la matrice de covariance.

III.3 Programme

A chaque couple de rayon et force (r, f), nous voulons calculer l'estimation des *extrema* sans faire un test pour chacun de ces points. Donc, connaissant le rayon et la force, le calcul des points qui nous intéressent se fait par in terpolation par le krigeage.

III.3.1 Algorithme

L'algorithme Kriging est le suivant :

1. Vérifier la compatibilité des entrées en termes de dimensions.
2. Combiner les valeurs à interpoler et les valeurs connues.
3. Calculer le vecteur b du problème (10).
4. Calculer la matrice A du problème (10).
5. Résolution du problème (10).

Cf kriging.m Annexe page IX.

III.3.2 Application

Nous avons écrit une fonction pour tester et exécuter le Krigage et visualiser la surface des points en fonction de R et de F en 3D.

Expérience : En entrée le vecteur des rayons, le vecteur des forces et l'ensemble des valeurs du premier extremum.

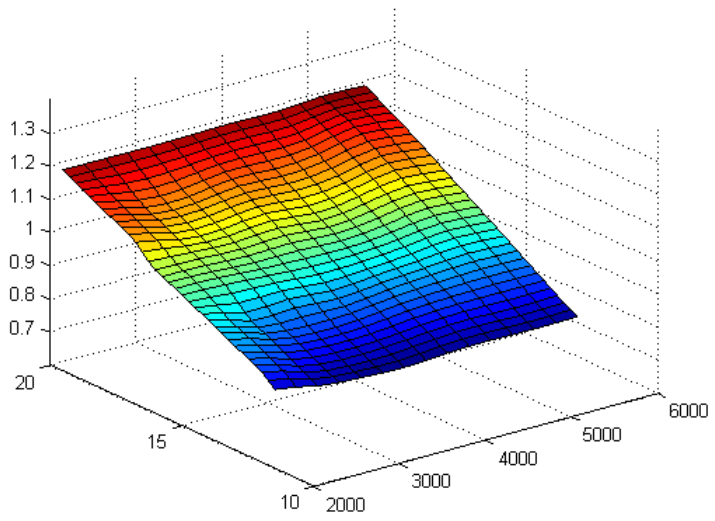

FIGURE 13 – Krigeage : Premiers *extremums*.

Expérience : En entrée le vecteur des rayons, le vecteur des forces et l'ensemble des valeurs de deuxième extremum (le minimum).

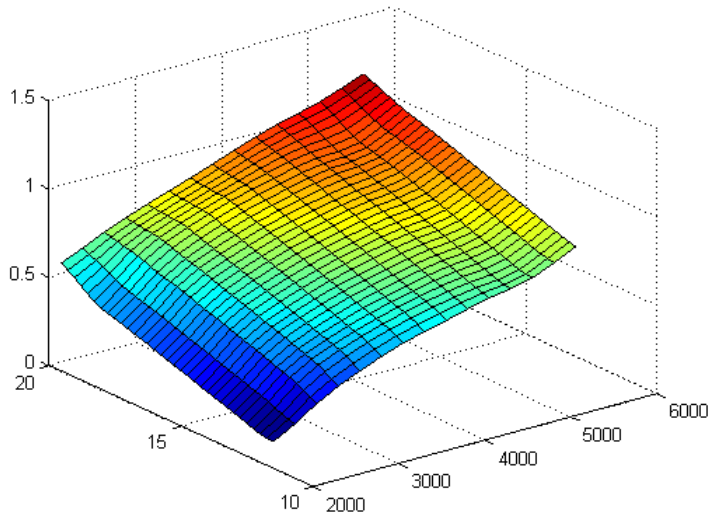

FIGURE 14 – Krigeage : Seconds *extremums*.

Afin de vérifier l'intérêt et la précision de la méthode d'interpolation par le krigeage, nous l'avons testée en subdivisant l'ensemble de valeurs que nous avons en deux parties : un ensemble de base avec lequel nous implémentons la fonction de krigeage et un ensemble de test sur lequel nous testons la méthode. Le krigeage étant une méthode d'interpolation

exacte, le résultat sur l'ensemble de base est connu.
La fonction *Krigtest*[6] accepte en paramètre un nombre n qui représente le nombre des valeurs de test. La fonction choisit ces valeurs de test aléatoirement et calcule ensuite l'erreur maximale et l'erreur moyenne pour l'ensemble de test.
Il faut noter que l'erreur d'estimation obtenue n'est pas une constante pour un entier n fixe car le choix des valeurs de test par *Krigtest* est aléatoire, elle reste cependant de même ordre. A titre d'exemple, l'erreur est de l'ordre de 10^-3 pour $n = 10$.

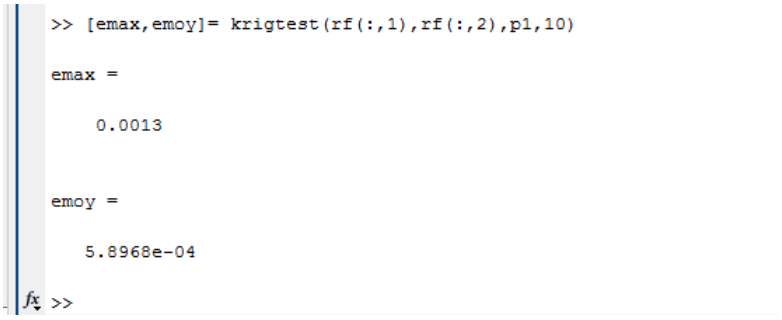

FIGURE 15 – Ordre d'erreur du krigeage

Enfin, pour avoir une idée sur la qualité de la méthode, et malgré le critère aléatoire des valeurs testées, nous avons remarqué que l'allure générale de la fonction de l'erreur maximale résultante en fonction du nombre des valeurs testées (supprimés de la base de krigeage) est croissante.

6. cf. krigtest.m Annexe page VII

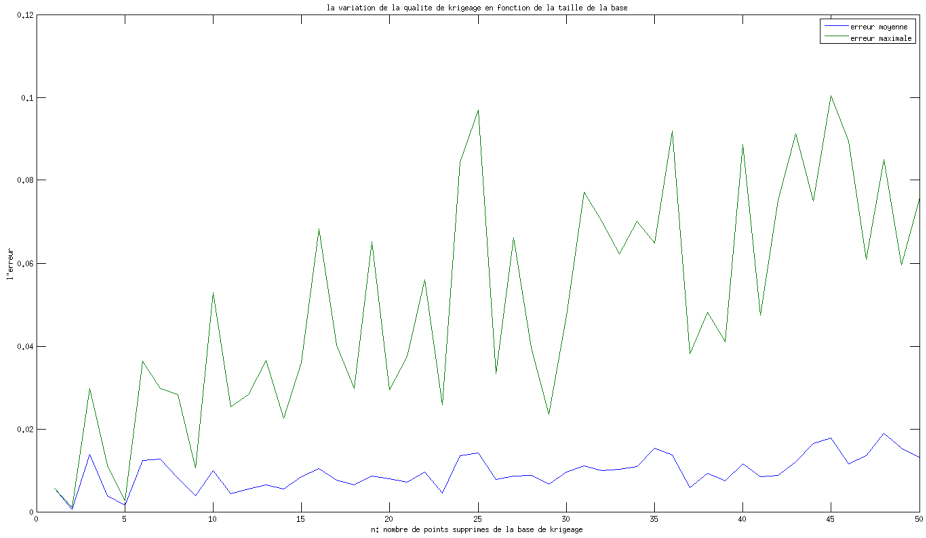

FIGURE 16 – Erreur maximale du krigeage en fonction du nombre de points supprimés

IV Autres méthodes d'interpolation : comparaison

Toujours dans un souci de vérification de l'efficacité de la méthode du krigeage, nous l'avons comparée à la méthode de l'interpolation de pondération par l'inverse de la distance (IDW) "Inverse Distance Weighted" [3].
L'IDW consiste à déterminer les valeurs des points à estimer par la combinaison pondérée de manière linéaire de l'ensemble des points d'une base composée d'échantillons. La fonction de pondération de ces points est une fonction de l'inverse de la distance :
Dans le cas général, pour un point de coordonnées (x, y), et un ensemble de base $(x_1, x_2, \cdots, x_n)$ qui représente l'échantillon sur lequel nous nous baserons pour estimer la valeur correspondante à (x, y) via la combinaison pondérée de manière linéaire de la base $(x_1, x_2, \cdots, x_n)$.

Ainsi, pour chaque point $x = (R, F)$ on calcule la distance d_i par rapport à chaque point $i = (Ri, Fi)$ de l'ensemble de base d'échantillons et on estime la valeur en x par la fonction [7] :

$$f(x) = \frac{\sum_{i=1}^{n} \frac{f(i)}{d_i}}{\sum_{i=1}^{n} d_i} \tag{12}$$

Nous avons aussi comparé ces deux méthodes en variant le nombre de points supprimés de la base, et nous nous sommes aperçus que la méthode du krigeage donne toujours une erreur maximale inférieure à celle de la méthode de pondération par l'inverse de la distance IDW.
Cette remarque est illustrée par le graphe suivant :

7. cf. invDist.m Annexe page VIII

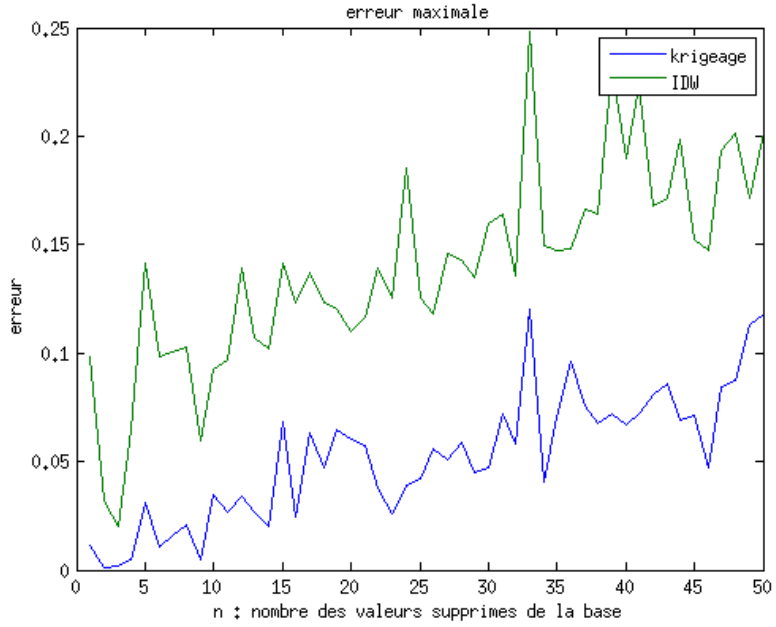

FIGURE 17 – Courbes des erreurs maximales

Ci-dessous, le graphe de la différence entre l'erreur moyenne obtenue par la méthode IDW et l'erreur moyenne obtenue par le krigeage. La courbe est positive donc la valeur moyenne de l'erreur de la fonction de krigeage est toujours inférieure à celle obtenue par la méthode IDW.

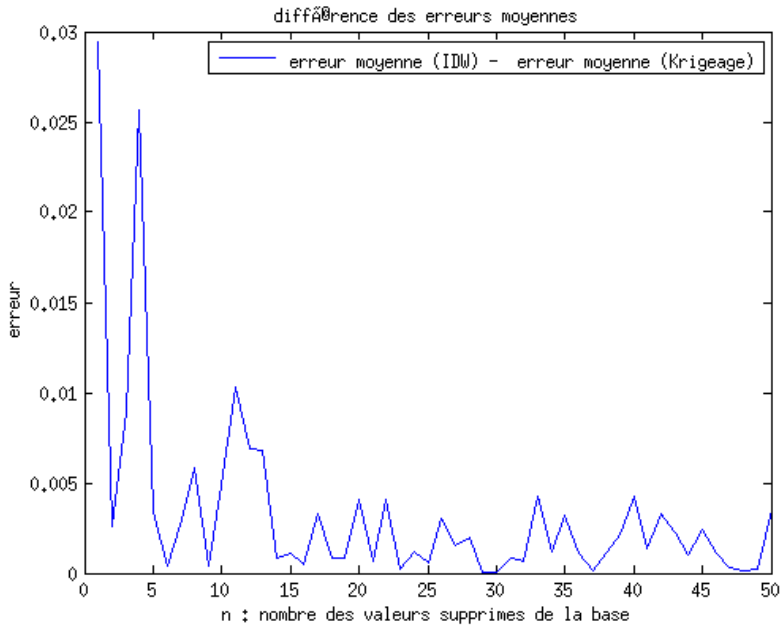

FIGURE 18 – Différence entre les erreurs moyennes

En comparant l'erreur maximale et l'erreur moyenne, nous trouvons que la méthode de krigeage est plus précise que la méthode IDW.

```
>> [emaxKrig,emoyKrig,emaxIDW,emoyIDW]= krigtest(rf(:,1),rf(:,2),p1,10)

emaxKrig =

    0.0098


emoyKrig =

    0.0042


emaxIDW =

    0.1309


emoyIDW =

    0.1007

fx >>
```

FIGURE 19 – Krigeage vs IDW : erreurs

Conclusion

En exploitant les résultats obtenus des expériences effectuées sur les splines d'essuie-vitres, nous avons pu, par une méthode d'interpolation, estimer le comportement d'autres splines sans les tester. Il nous a fallu d'abord écrire des fonctions pour traiter les données afin de mieux les utiliser : le lissage des courbe. Nous avons ensuite déterminé les points caractéristiques de chaque courbe en exploitant l'erreur du filtre auto-régressif. Enfin, nous avons appliqué la méthode d'interpolation par le krigeage et nous avons testé son efficacité en la comparant à une méthode d'interpolation linéaire, l'IDW.

Les principales fonctions exigées ont été ainsi réalisées malgré les difficultés rencontrées. En effet, une grande partie de notre travail a été consacrée à la documentation, notamment sur la méthode d'interpolation par le krigeage. le développement des fonctions en Matlab nous a également posé une difficulté, nous ne pouvions utiliser Matlab que dans les locaux de l'ISIMA.

Le résultat de notre projet aide dans la détermination des paramètres susceptibles d'influencer le comportement des essuie-vitres et améliorer certaines fonctionnalités. Il peut servir pour l'élaboration d'un outil plus développé intégrant une interface graphique et prenant, en plus de la force de réaction et le rayon de courbure, la longueur des splines pour une estimation en 3 dimensions.

Annexes

Codes source

```
function [Mat, RF] = lecture( FolderName )

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% fonction de lecture des fichiers sur linux (matlab R2007b)
% BEN SALEM Malek  FARAJI Safae
% Projet ZZ2 ISIMA  Valeo
% en entree le nom du dossier
% en sortie la matrice Mat: contenant les matrices du fichier de chaque sous dossier
%                          RF: La matrice contenant le rayon R et la force F de chaque enregistremen
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



cd (FolderName)
A=ls;
temp=[];
%trouver l'emplacement des separations entre les noms des dossiers
%debut
        vect=strfind(A,'f');

%fin
Mat=[];
v= size(vect);
k=v(2)+1-5;
for i=1:k
        c1= A(vect(i):vect(i+1)-1);

        r=strfind(c1,'R');
    F=strfind(c1,'F');
    RF(i,1)=str2num( c1(r+1:r+4));
        bb=2;
     if (numel(c1)> F+3 & c1(F+4)=='5')
                bb=4;
          end
                        RF(i,2)=str2num( c1(F+1:F+bb));

        if (size(strfind(c1,'j'))==1);
                ch=strfind(c1,'j');
                c=c1(1:ch(1)-1);

                else c=c1;
                end


        dossier = deblank(strtrim(string(c)));
        cd (dossier)
        nomfic=deblank(strtrim(string(ls)));

        %lecture d'un fichier
        temp=[];
        pfin = fopen( nomfic );
        if (pfin > 0)
                Ligne = fgetl(pfin);
                        for l=1:20
                        Ligne = fgetl(pfin); % Sauter la ligne vide
                        end
                while ~feof(pfin)
                        Ligne = strrep(Ligne,',','.');
                        temp=vertcat(temp,str2num(Ligne));
                        Ligne = fgetl(pfin);
                end
                Mat{i} = temp;
                        fclose(pfin);
```

```
        end

        %fin lecture du fichier
        cd ..
end
cd ..
```

```
function Mat = lecture( FolderName )

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% fonction de lecture des fichiers sur windows (matlab R2012b)
% BEN SALEM Malek FARAJI Safae
% Projet ZZ2 ISIMA Valeo
% en entree le nom du dossier
% en sortie la matrice contenant les matrices du fichier de chaque sous dossier
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



cd (FolderName);

A=ls;
temp=[];

Mat=[];
v= size(A);
k=v(1)-1;
for i=3:k
        c1= A(i,:);

        dossier = deblank(strtrim(string(c1)));
        cd (dossier)
    B1=ls;
    B=B1(3,:);
        nomfic=deblank(strtrim(string(B)));

        %lecture d'un fichier
        temp=[];
        pfin = fopen( nomfic,'r' );
        if (pfin > 0)
                Ligne = fgetl(pfin);
                        for l=1:20
                        Ligne = fgetl(pfin); % Sauter la ligne vide
                        end
                while ~feof(pfin)
                        Ligne = strrep(Ligne,',','.');
                        temp=vertcat(temp,str2num(Ligne));
                        Ligne = fgetl(pfin);
                end
                Mat{i} = temp;
                        fclose(pfin);
        end

        %fin lecture du fichier
        cd ..
end
cd ..
cd ..
```

```
function V=MT(X)


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% fonction pour lisser les courbes
% BEN SALEM Malek et FARAJI Safae
% Projet ZZ2 ISIMA  Valeo
% en entree: X: vecteur des valeurs de la fonction
% en sortie: V: vecteur contenant les valeurs lissees
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


A=size(X,1);
for i=4:A(1)-3
V(i)=(X(i)+X(i+1)+X(i+2)+X(i+3)+X(i-1)+X(i-2)+X(i-3))/7;
end
for i=1:4
V(i)=X(1);
end

for i = A(1)-2:A(1)
V(i)= (X(A(1))+X(A(1)-1)+X(A(1)-2))/3;
end
```

```
function b=moyenneur(X)


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% fonction pour lisser les courbes
% BEN SALEM Malek et FARAJI Safae
% Projet ZZ2 ISIMA Valeo
% en entree: X: vecteur des valeurs de la fonction
% en sortie: V: vecteur contenant les valeurs lissees
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


b= MT(X);
temp=X(1:35);
[m,i]= max(temp);
d=X(1);
for j=1:i
b(j)=d+(j-1)*(m-d)/(i-1);
end
for j=1:6
b(i+j)=X(i+j);
end
```

```
function b=denoise(X)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% fonction pour lisser les courbes
% BEN SALEM Malek et FARAJI Safae
% Projet ZZ2 ISIMA Valeo
% en entree: X: vecteur des valeurs de la fonction
% en sortie: V: vecteur contenant les valeurs lissees
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

temp=X(1:35);
[m,i]= max(temp);
b1= wden(X(i+6:end),'heursure','s','one',2,'sym8');
d=X(1);
for j=1:i
b(j)=d+(j-1)*(m-d)/(i-1);
end
for j=1:5
b(i+j)=X(i+j);
end
```

```
21
22 b=[b b1'];
```

```
1  function Z=modelAR(X,n)
2
3  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
4  %            fonction modelAR
5  % BEN SALEM Malek   FARAJI Safae
6  % Projet ZZ2 ISIMA   Valeo
7  % X   :signal d'entree
8  % n   : ordre
9  % Z   :signal de sortie prevu par le modele autoregressif
10 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
11
12
13  k = n;
14 % Calcul des coefficients de l'equation
15                % vecteur d'autocovarience
16                autocov= xcorr(X,k+1,'biased');
17                %la Matrice hermitienne de autocov
18                R =  toeplitz(autocov(k+2:end));
19                %  la partie qui nous interresse de la matrice hermitienne
20                A =  R(1:k,1:k);
21                % le vecteur B
22                B = -R(2:k+1,1);
23
24 %resolution
25                coeff = inv(A)* B;
26 %tenir compte du premier indice
27                coeff = [1 coeff'];
28
29
30 %determination du signal de sortie
31 for i=1:n+2
32 Z(i)=X(i);
33 end
34 for i=n+3:numel(X)
35 X1=X(i-n-2:i-1);
36 Z(i)=0;
37   for k= 2:n
38     Z(i)=Z(i)-X(i-k+1)*coeff(k);
39   end
40 end
```

```
function [m,rf,Xopt,Iopt,p1,p2,p3]=exec( FolderName )


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% fonction de lectures des fichiers et de pretraitement des donnees
% en entee : FolderName le nom de dossier
%          : sinon par default FolderName =  'L600FdefPdef'
% en sortie:
%             m=Matrice qui contient tout les donnees de tout les fichiers
%              rf= tableau qui contient le rayon cible et la force
%              correspondante a chaque ensemble de valeurs
%              Xopt Iopt: les points interressants et leurs indices
%              pi : un vecteur contenant les valeurs de ieme point interressant
%
% BEN SALEM Malek et FARAJI Safae
% Projet ZZ2 ISIMA Valeo
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


if nargin==0, FolderName='L600FdefPdef';end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[m,RF] = lecture( FolderName );
i=0;
for j=1:size(m,2)
    temp=m{j};
    if(size(temp,1)>0)
    i=i+1;
    rf(i,1)=RF(j,1);
    rf(i,2)=RF(j,2);
    Y=temp(:,1);
    matMT{i}=MT(temp(:,2));
    [Xo, Io]=extrem(temp);
    Xopt{i}=Xo;
    Iopt{i}=Io;
    p1(i)=Xo(1);
    p2(i)=Xo(2);
    p3(i)=Xo(3);
    if (i==4)
        figure(i)
     subplot(3,2,1);
    plot(Y,temp(:,2));
    title('original')
        xlabel('distance mm');
        ylabel('reaction Force F/m');
    lisse=denoise(temp(:,2));
    subplot(3,2,2)
    plot(Y,lisse);
        xlabel('distance mm');
        ylabel('reaction Force F/m');
    title ('courbe lisse')
    subplot(3,2,3)
    plot(Y,temp(:,2)-lisse');
    title ('erreur')
        xlabel('distance mm');
        ylabel('reaction Force F/m');
    subplot(3,2,4)
    plot(Y,moyenneur(temp(:,2)));
    title ('moyenneur ameliore')
        xlabel('distance mm');
        ylabel('reaction Force F/m');
        subplot(3,2,5)
    plot(Y, modelAR(denoise(temp(:,2)),3));
    title ('modele AR d ordre 3')
```

```
64          xlabel('distance mm');
65          ylabel('reaction Force F/m');
66      subplot(3,2,6)
67          plot(Y,modelAR(denoise(temp(:,2)),3)- denoise(temp(:,2)));
68          title ('erreur  AR d ordre 3')
69          xlabel('distance mm');
70          ylabel('reaction Force F/m');
71      saveName = (['R', num2str(RF(i,1)),'F',num2str(RF(i,2)),'_graphe',num2str(i),'_.jpg']);
72      saveas(gcf, saveName);
73      close
74          end
75      else
76          disp(['le fichier ne contient pas les resultats pour R =: ' num2str(RF(j,1)),' et F=', nu
77      end
78
79  end
```

```
function [emax,emoy,emaxi,emoyi]= krigtest1(Re,Fe,pe,n)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% fonction pour comparer le Krigage a la methode IDW
% BEN SALEM Malek FARAJI Safae
% Projet ZZ2 ISIMA Valeo
% en entree Re,Fe: deux vecteur: les coordonnees des points de la base
%             pe:un vecteur contenant les valeurs des points interressants
%             n :le nombre des points qu'on supprimer de la base et avec
%                 lesquels on teste les deux methodes
% en sortie: emoy: erreur moyenne (Krigeage)
%             emax: erreur maximale (Krigeage)
%             emoyi: erreur moyenne (IDW)
%             emaxi: ereur maximale (IDW)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if (nargin==0)
[m,rf,Xopt,Iopt,p1,p2,p3]=exec('L600FdefPdef');
Re=rf(:,1);
Fe=rf(:,2);
Pe=p1;
end
taille=size(Fe,1);
if (nargin==4)
r = floor (1+ (taille -1).*rand(n,1));

R=Re;
F=Fe;
p=pe;
for i=1:n
    k=max(r(i)-i+1,1);
    Rtest(i)=R(k);
    Ftest(i)=F(k);
    ptest(i)=p(k);
    R(k)=[];
    F(k)=[];
    p(k)=[];
end

emax=0;
emoy=0;
emaxi=0;
emoyi=0;
for i=1:n
    [rt,rf,pkt] = kriging(R/1000,F,p,Rtest(i)/1000,Ftest(i));
    emax=max(abs(pkt-ptest(i)),emax);
    emoy=emoy+abs(pkt-ptest(i));
     pkt = invDist(R,F,p,Rtest(i),Ftest(i));
    emaxi=max(abs(pkt-ptest(i)),emaxi);
    emoyi=emoy+abs(pkt-ptest(i));
end
emoy=emoy/n;
disp(['l erreur maximale par la methode IDW ' ,num2str(emaxi),' et par le Krigeage', num2str(emax
disp(['l erreur maximale par la methode de Krigeage', num2str(emoyi),' et par le Krigeage', num2s
end

if (nargin==3)
R=Re;
F=Fe;
p=pe;
r = floor (1+ (taille -1).*rand(n,1));
n=50;
for i=1:n
```

```
        k=max(r(i)-i+1,1);
        Rtest(i)=R(k);
        Ftest(i)=F(k);
        ptest(i)=p(k);
        R(k)=[];
        F(k)=[];
        p(k)=[];
end

emax(n)=0;
emoy(n)=0;
emaxi(n)=0;
emoyi(n)=0;
for i=1:n
        [rt,rf,pkt] = kriging(R,F,p,Rtest(i),Ftest(i));
        emax(n)=max(abs(pkt-ptest(i)),emax(n));
        emoy(n)=emoy(n)+abs(pkt-ptest(i));
          pkt = invDist(R,F,p,Rtest(i),Ftest(i));
        emaxi(n)=max(abs(pkt-ptest(i)),emaxi(n));
        emoyi(n)=emoy(n)+abs(pkt-ptest(i));
end
emoy(n)=emoy(n)/n;
emoyi(n)=emoyi(n)/n;

subplot(1,2,1)
plot(1:50,- emoy +emoyi);
legend(' erreur moyenne (IDW) -  erreur moyenne (Krigeage)');
xlabel('n : nombre des valeurs supprimes de la base');
ylabel('erreur ');
title('difference des erreurs moyennes');

 subplot(1,2,2)
 plot(1:50,emax,1:50,emaxi);
 legend(' erreur maximale  (Krigeage)',  'erreur maximale (IDW) ');
 xlabel('n : nombre des valeurs supprimes de la base');
 ylabel('erreur ');
 title('comparaison en erreur maximale');
end
```

```
function pk=invDist(r,f,p,R1,F1)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% fonction de la methode d'interpolation  IDW
%en entree:R,F: deux vecteur: les coordonnees des points de la base
%               p: un vecteur; Les valeurs de la fonction dans les points de
%               la base
%               R1,F1: Les coordonnees du point auquel on cherche la valeur
%en sortie pk: la valeur estimee au  point  (R1,F1)
% Projet ZZ2 Ben Salem Malek et Safae Faraji
%                ISIMA   Valeo
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
pk=0;
s=0;
for i=1:numel(r)
        dist=((R1-r(i))^2+(F1-f(i))^2)^.5;
        if (dist~= 0)
        s=s+1/dist;
        pk=pk+p(i)/dist;
        end
end
pk=pk/s;
```

```
function [RC,F,pkt] = kriging(Rbase,Fbase,pbase,RC,F)

 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% fonction pour estimer la valeur d'un point par le krigeage
% BEN SALEM Malek FARAJI Safae
% Projet ZZ2 ISIMA Valeo
% en entree Rbase,Fbase: deux vecteur: les coordonnees des points de la base
%           pbase:un vecteur contenant les valeurs des points interressants
%           RC,F : les coordonnees des points a interpoler
% en sortie RC F:les coordonnees des points a interpoler
%            pkt: La valeur estimee du point (RC,F)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
nmax = 500;
pblock = 1/3;
 n = [8 1];
 err = 0;
  sc =0;
 len = 2;
 RR = [Rbase(:); RC(:)];
 FF = [Fbase(:); F(:)];
 ss = [ones(prod(size(Rbase)),1); zeros(1,1)];
 Lr = [min(RR) max(RR)];
 Lf = [min(FF) max(FF)];
 ind= ones(size(RR));
 np = length(ind);
 Sp = sparse(ind,1:np,1,1,np);
 npb = full(sum(Sp'));
 nb = prod(size(Rbase));
 part_blk = part_blk/sqrt(2*nb/n(1));
 sc = max(sc,[mean(diff(Lr')), mean(diff(Lf'))]*part_blk);

    pbase = pbase(:);
        is_vec = 1;
        szb = size(pbase);
        oz = ones(1,szb(2));;
        z0 = mean(pbase); % la moyenne de l'ensemble des valeurs des points
        var = mean((pbase-z0(:,oz)).^2);
        sz = size(pbase);
        Rx = Rbase(:);
        Fx = Fbase(:);
    pkt = pbase(:);
        r0(1) = mean(Rx);
        r0(2) = mean(Fx);
        z0 = mean(pkt);
        Rx = Rx-r0(1);
        Fx = Fx-r0(2);
        zo = pkt-z0(1);
        pblock = [Rx Fx]\zo;
        pblock = [z0 pblock'];
        zo = zo-Rx*pblock(2)-Fx*pblock(3);
        zo = reshape(zo,sz(1),sz(2));
        pbase(:)=zo;
        g=pblock;
        pkt = zeros(prod(size(RC)),szb(2));
        wi = zeros(prod(size(RC)),1);
        er1 = 1-err;

  Rx = diff(Lr);
  Fx = diff(Lf);
  ibp= (1:nb)';
  iip=nb+1;
  ob = ones(size(ibp));
  oi = ones(size(iip));
```

```
64    % Matrice de covariance pour les points de base
65       r = exp(-(Rx/sc(1)).^2-(Fx/sc(2)).^2);
66     r = (1-(er1*r));
67     A = r;
68     v = r\pbase(ibp,:);
69     %  points d interpolation
70     len = length(iip);
71       w_ch = 1;
72       i1 = iip(1);
73       och = ones(size(i1));
74       Rx = RR(i1,ob)-RR(ibp,och)';
75       Fx = FF(i1,ob)-FF(ibp,och)';
76       r = exp(-(Rx/sc(1)).^2-(Fx/sc(2)).^2);
77       r = (1-(er1*r));
78       i1 = i1-nb;
79       pkt(i1,:) = pkt(i1,:)+w_ch(:,oz).*(r*v);
80       wi(i1) = wi(i1)+w_ch;
81
82  pkt = pkt./wi(:,oz);      % Diviser par les poids
83  sz = size(RC);
84
85    pkt(:,1) = pkt(:,1)+g(1,1)+(RC(:)-r0(1,1))*g(1,2);
86    pkt(:,1) = pkt(:,1)+(F(:)-r0(1,2))*g(1,3);
87
88   % si on n'a pas besoin de Xi et Yi
89  if nargout<=1,  RC = pkt; end
90
91  end
```

Bibliographie

[1] C. Jutten, *Filtrage linéaire optimal*, http://www.gipsa-lab.grenoble-inp.fr/~christian.jutten/mescours/Cours_FLO.pdf

[2] Y. Gratton, *Le krigeage : La méthode optimale d'interpolation spatiale*, ftp://oceane.obs-vlfr.fr/pub/roullier/BIBLIO/STAT/R/krigeage_juillet2002.pdf

[3] *ArcGIS Resource Center*, http://help.arcgis.com/fr/arcgisdesktop/10.0/help/index.html#/na/009z00000075000000/